Project 2: Online Retail

Elvessa Tatum

UHID: 2064084

Clustering Analysis

Background info

- Today, I plan on using the UCI online retail dataset to group customers by their spending behavior. This dataset has a list of transactions between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. This is important to see certain aspects of online shopping, such as people who return items, how much they spend, etc etc..
- This dataset was chosen due to the small number of features that make it easy to focus on, along with the high amount of recorded observations. The 6 features are listed to the side.

InvoiceNo	ID	Categorical	a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation		no
StockCode	ID	Categorical	a 5-digit integral number uniquely assigned to each distinct product		no
Description	Feature	Categorical	product name		no
Quantity	Feature	Integer	the quantities of each product (item) per transaction		no
InvoiceDate	Feature	Date	the day and time when each transaction was generated		no
UnitPrice	Feature	Continuous	product price per unit	sterling	no
CustomerID	Feature	Categorical	a 5-digit integral number uniquely assigned to each customer		no
Country	Feature	Categorical	the name of the country where each customer resides		no

Background info continued

- K means clustering assigns records to each cluster to find the mutually exclusive cluster of spherical shape based on distance. It does this by using a pre-specified number of clusters.
- Hierarchical clustering is a set of nested clusters that are arranged as a tree. It can use any number of clusters, and have divisive or agglomerative methods. Agglomerative methods begin with ‘n’ clusters and sequentially combine similar clusters until only one cluster is obtained. Divisive methods work in the opposite direction, beginning with one cluster that includes all the records.
- Some advantages of K means clustering are that convergence is guaranteed, and it is specialized to clusters of different sizes and shapes. Some disadvantages of it are that K-Values are difficult to predict and it doesn't work well with global cluster.
- One advantage of hierarchical clustering are the ease of handling of any forms of similarity or distance. Consequently, another advantage is the applicability to any type of attribute. A big disadvantage of hierarchical clustering is that it requires the computation and storage of an $n \times n$ distance matrix. For very large datasets, this can be expensive, and make the model very slow.
- Hierarchical clustering is non-parametric. It doesn't assume a specific number of clusters or a specific shape of the clusters. It is flexible and can adapt to various types of cluster shapes.
- K means is parametric. It assumes that the data forms clusters that are spherical and uses the Euclidean distance to form the clusters. It requires you to predefine the number of clusters (k), which makes it a parametric model.
- We’ll be using K means clustering and hierarchical clustering on the dataset, as we can estimate the right amount of clusters needed for both algorithms (K means through elbow method and hierarchical through the dendrogram.

Importing data

Let's import the data and see what we have

```
retail_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[3]:
```
retail_data = pd.read_csv('online_retail.csv')
retail_data.head()
```

[3]:

	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

Importing data continued

Let's use describe

```
retail_data.describe()
```

	Quantity	UnitPrice	CustomerID	TotalPrice
count	406829.000000	406829.000000	406829.000000	406829.000000
mean	12.061303	3.460471	15287.690570	20.401854
std	248.693370	69.315162	1713.600303	427.591718
min	-80995.000000	0.000000	12346.000000	-168469.600000
25%	2.000000	1.250000	13953.000000	4.200000
50%	5.000000	1.950000	15152.000000	11.100000
75%	12.000000	3.750000	16791.000000	19.500000
max	80995.000000	38970.000000	18287.000000	168469.600000

Importing data continued

- From here, we can see that the categorical variables are: InvoiceNo, StockCode, Country, Description, InvoiceDate, and customerID.

- On the other hand, the numerical values are quantity, UnitPrice, and customerID.

- CustomerID is listed as categorical on the dataset website, but numerical in the actual dataset. We will count it as “both” here.

Importing Data continued

- Let’s check for missing values and handle them. We’ll also get rid of description, since it isn’t important.

[5]:
```
# checking missing values
missing = retail_data.isnull().sum()
missing = missing[missing > 0].sort_values(ascending=False)
print(missing)
```

```
CustomerID     135080
Description      1454
dtype: int64
```

[6]:
```
# pre processing data for clustering customers based on return behavior

# getting rid of rows without a customer ID
retail_data = retail_data.dropna(subset=['CustomerID'])

# getting rid of description
retail_data = retail_data.drop(columns=['Description'])
```

Pre processing data

- Let’s add a variable. We’ll add a variable called “IsReturn,” which labels if a customer returned or canceled an item.
- We’ll also start to add more features based on customer behavior.

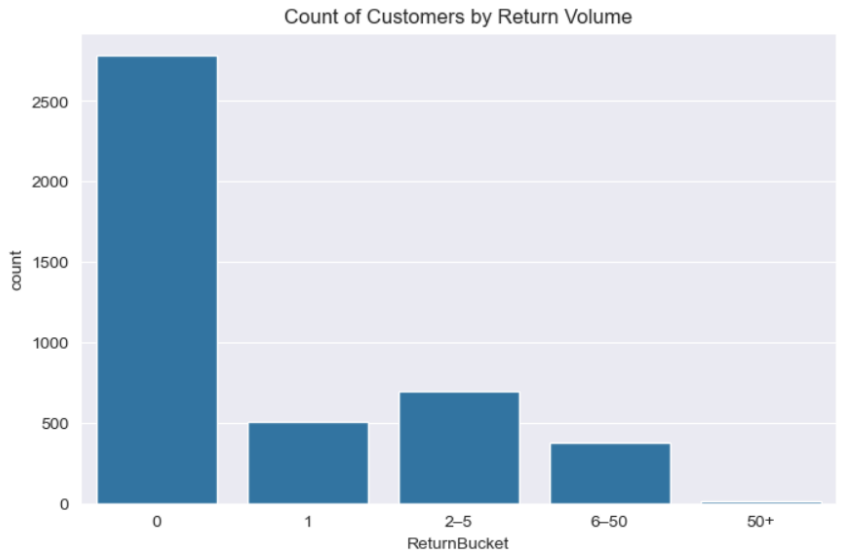
```
# create a "Return" flag per row
retail_data['IsReturn'] = retail_data['InvoiceNo'].astype(str).str.startswith('C') | (retail_data['Quantity'] < 0)
```

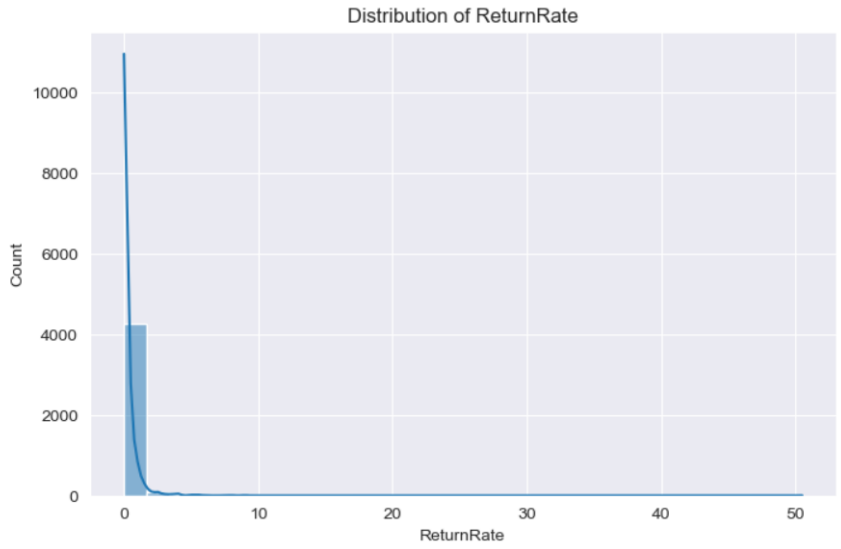
Pre-Processing data continued

- Let’s break this down. The “TotalPrice” variable is used to see how much money was involved in each transaction
- We then make a variable named “customer_returns,” which is a variable that holds information on the spending behavior on each customer. For this, we calculate the following:
- TotalTransactions: Number of unique invoices (how many separate purchases they made).
- TotalItems: Total quantity of items bought (net of purchases and returns).
- TotalReturns: How many of those transactions were returns.
- TotalQuantityReturned: Sum of quantities that were returned (only negative quantities).
- TotalMoneySpent: Sum of total prices where the transaction was positive (they actually bought something).
- TotalMoneyReturned: Sum of total prices where the transaction was negative (they returned something).
- Finally, we calculate ReturnRate and NetSpending. ReturnRate denotes what proportion of a customers purchases involved returns. Meanwhile, NetSpending represents true spending after accounting for refunds.

Pre-Processing data continued

[7]:
```python
# more pre processing. adding features per customer for clustering

# create total price per row (Quantity x UnitPrice)
retail_data['TotalPrice'] = retail_data['Quantity'] * retail_data['UnitPrice']

# grouping by CustomerID
customer_returns = retail_data.groupby('CustomerID').agg(
    TotalTransactions=('InvoiceNo', 'nunique'),
    TotalItems=('Quantity', 'sum'),
    TotalReturns=('IsReturn', 'sum'),
    TotalQuantityReturned=('Quantity', lambda x: x[x < 0].sum()),
    TotalMoneySpent=('TotalPrice', lambda x: x[x > 0].sum()),
    TotalMoneyReturned=('TotalPrice', lambda x: x[x < 0].sum())
)

# derived metrics
customer_returns['ReturnRate'] = customer_returns['TotalReturns'] / customer_returns['TotalTransactions']
customer_returns['NetSpending'] = customer_returns['TotalMoneySpent'] + customer_returns['TotalMoneyReturned']  # since returns are negative
```



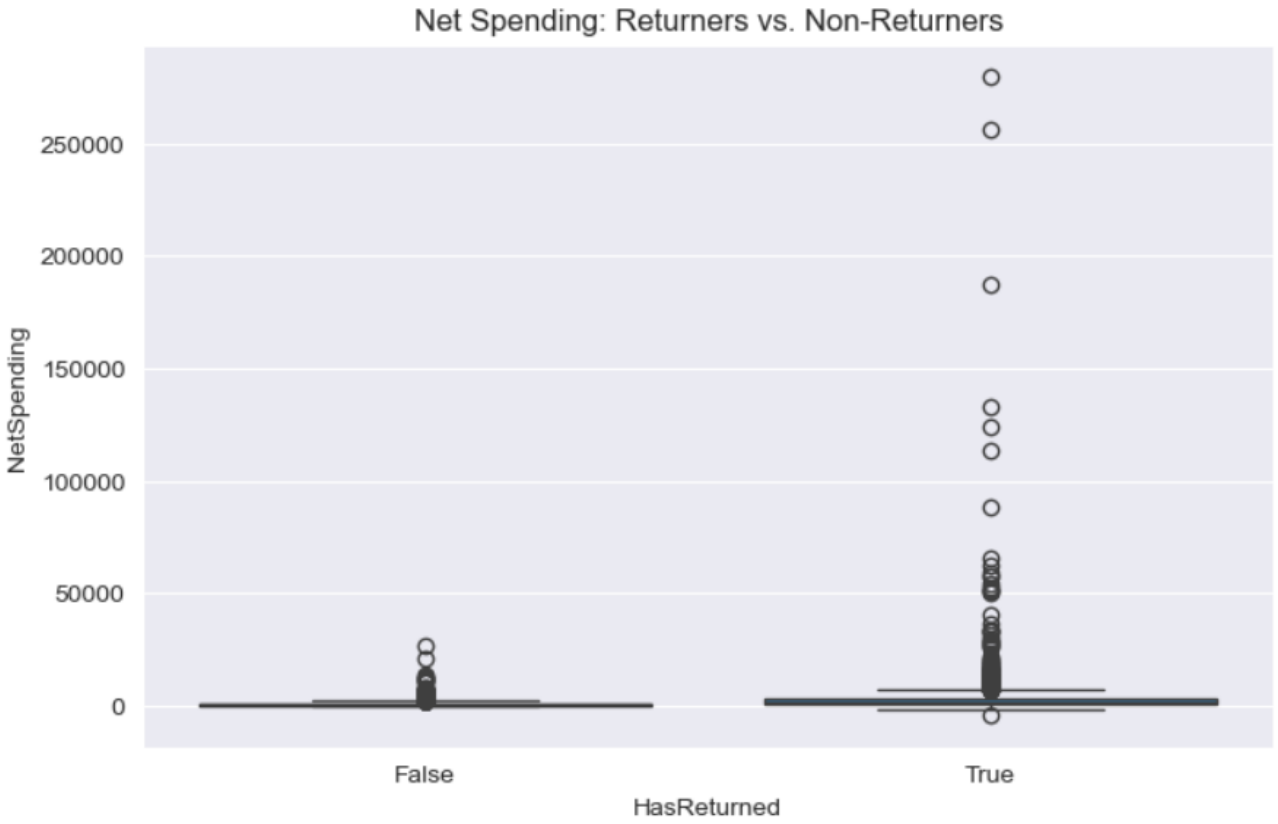


Data Exploration and visualization

- Now that we have our new variables to account for customer behavior, let's look at some graphs surrounding them. Here is a distribution of ReturnRate, and a overall count of customers by return volume

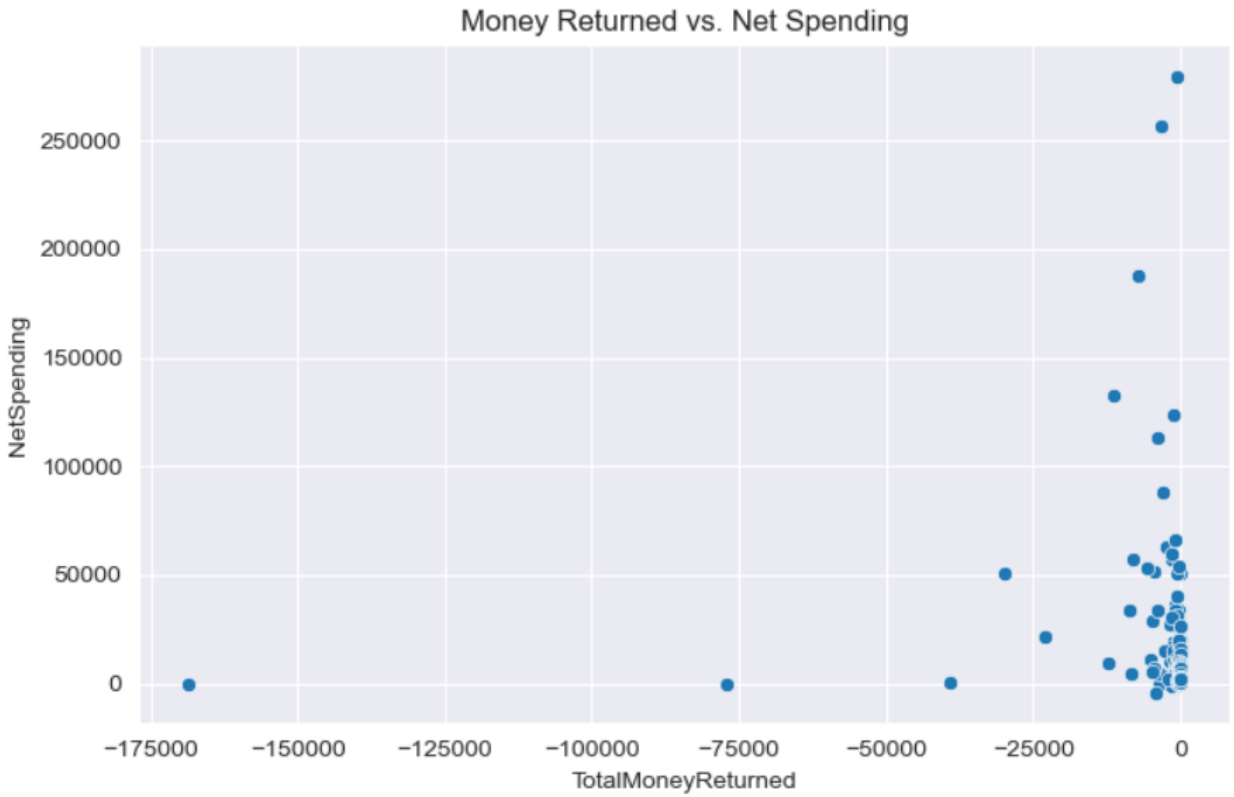
- From here, we can see that the amount of people who return things isn't very high. Most customers haven't returned any items. Those who have, haven't returned that many.

Data exploration and visualization continued



- Now, let's look at the NetSpending of people who returned items vs didn't return items
- Clearly, people who have returned items spend muchmore money than those who haven't

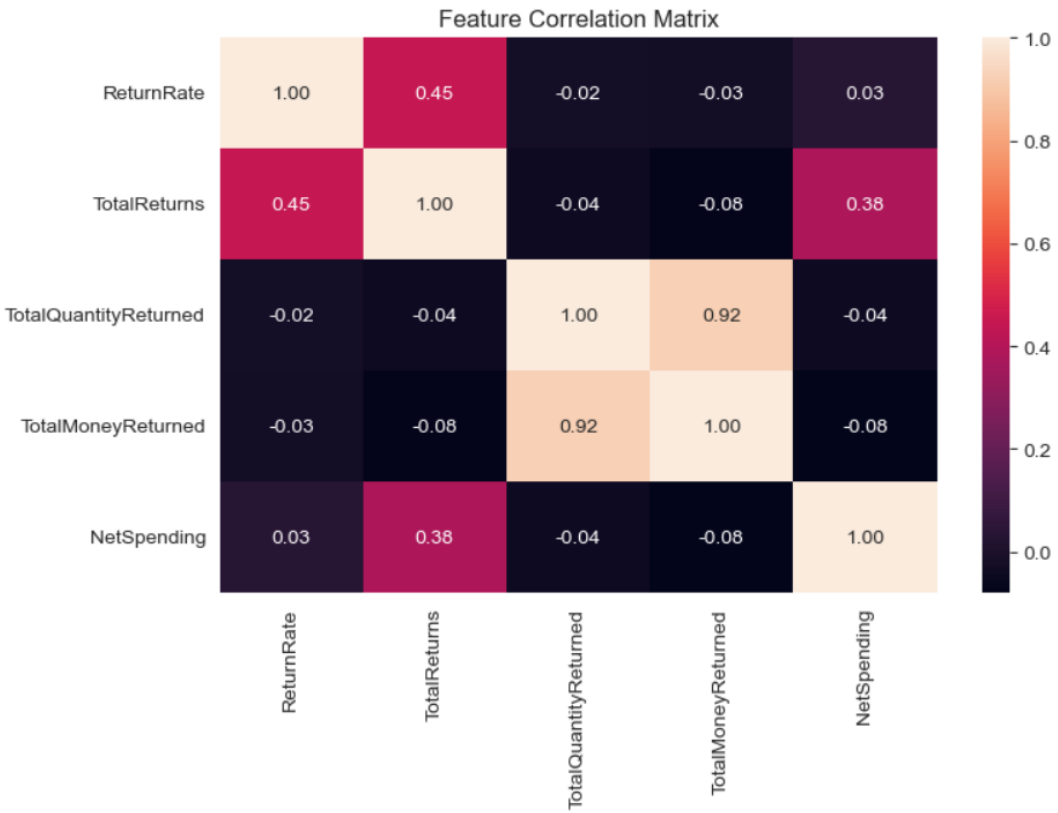
Data exploration and visualization continued



- Next, let's look at a graph of the money returned vs net spending.

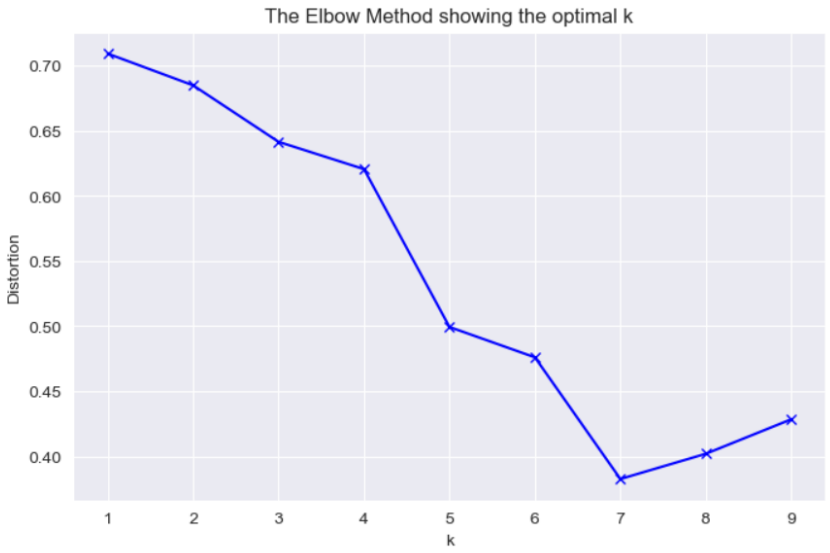
- Outside of some outliers, it is clear that people didn't receive much money back, but did spent a lot of money. This lines up with the idea that not many returns were done. Since there weren't many returns, people didn't get much money back.

Data exploration and visualization continued



- Finally, let's look at a heatmap of our new variables.

- From this, we can see that the total money returned is highly correlated with the total quantity returned. Some other variables that are correlated well with each other are net spending and total returns, and total returns and return rate.

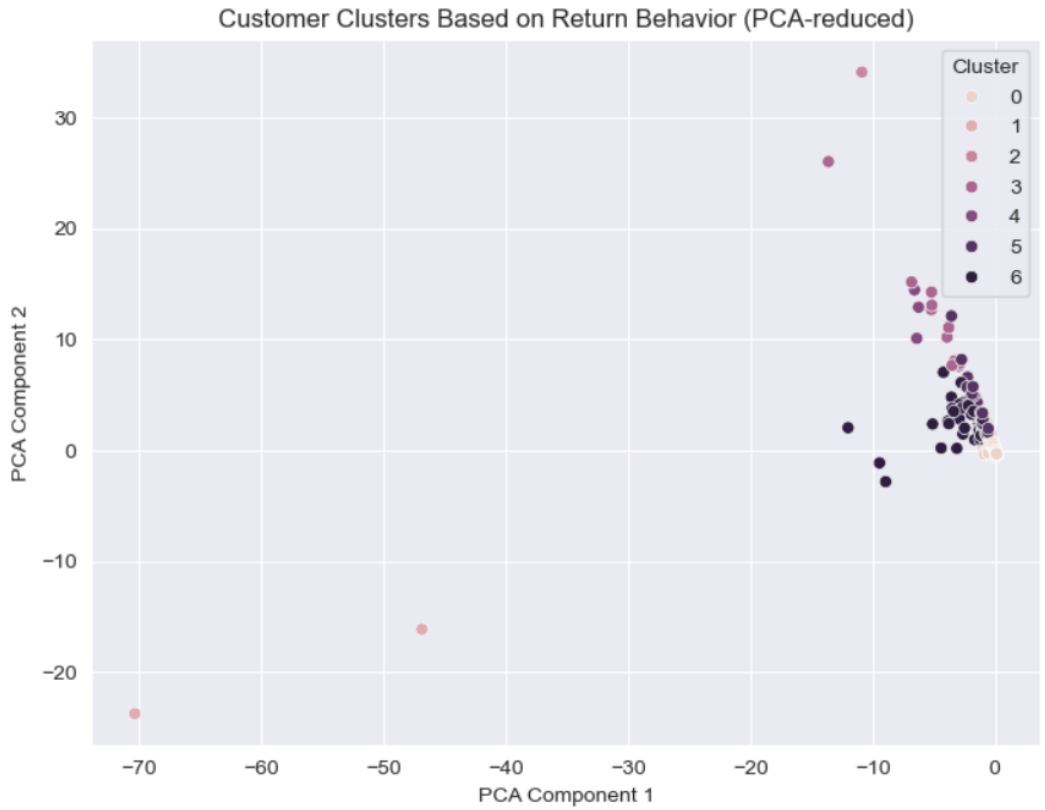



```
# scaling the data

from sklearn.preprocessing import StandardScaler

X = customer_returns[features].fillna(0)  # Fill NA if needed

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Scaling and initial model creation

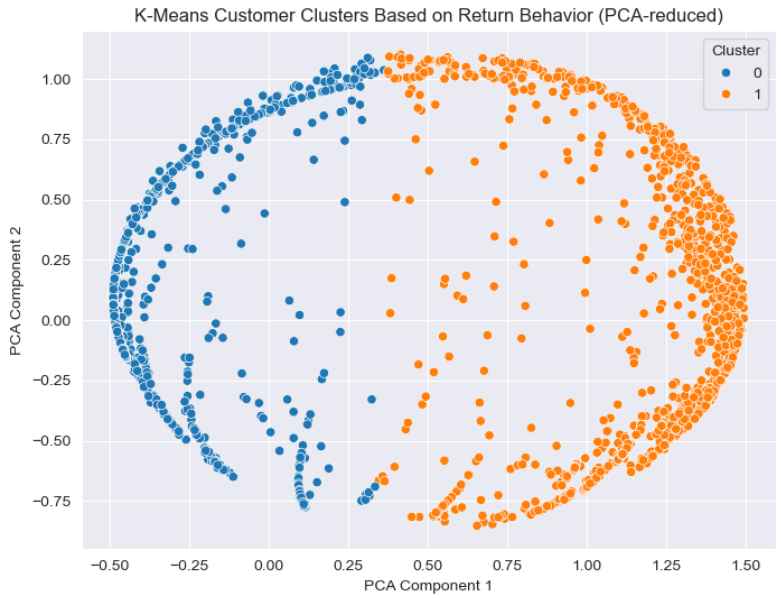
- Let’s use the standard scaler, as scaling the data will help balance it. Thus, our model will be more accurate.
- We’ll start with K means, finding the optimal k with the elbow method.
- The optimal K seems to be around 7, so we’ll use that for our model

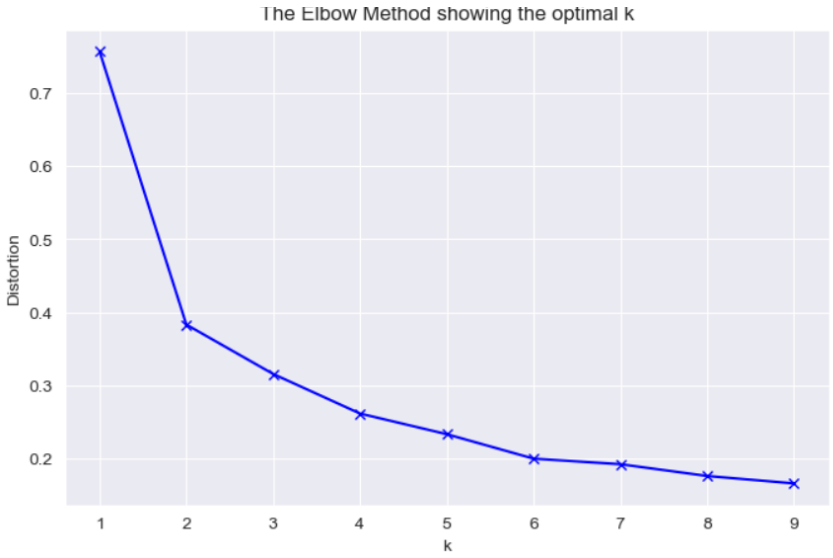
Initial model creation



- We will now use PCA to make the model good for plotting, then plot the K means model

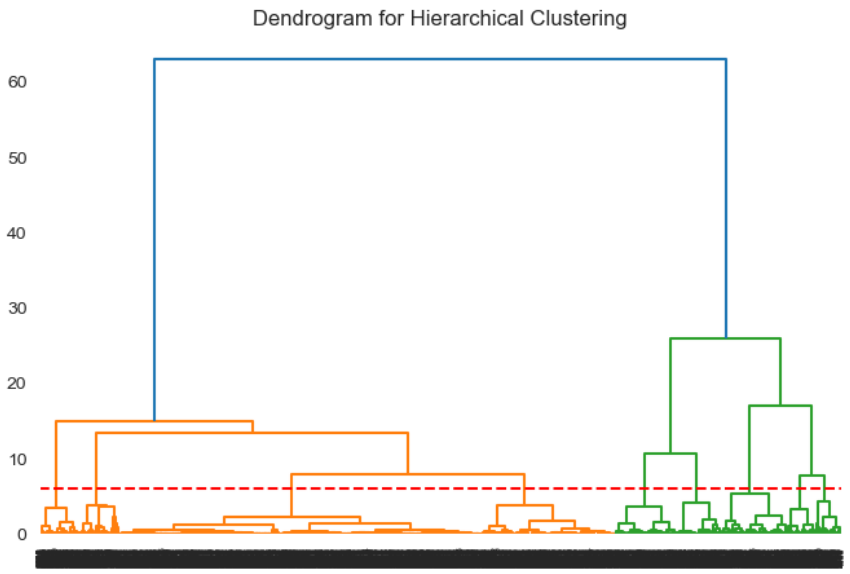
- As we can see, our model is heavily skewed. Let's normalize it then try again.

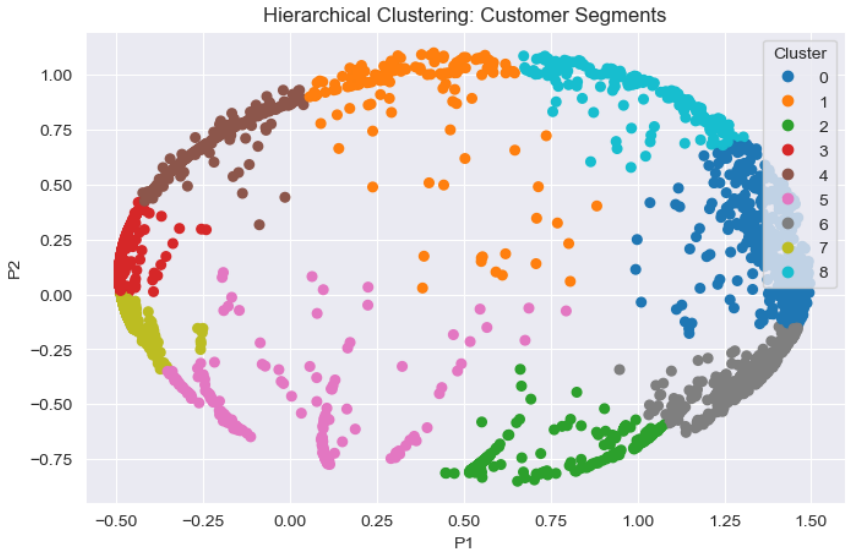




Model creation continued

- After normalizing, the optimal K seems to be 2. let's run the model again with this in mind.
- Now we have a better model visually, and the silhouette score for this model is 0.68, which is very good
- This model shows a clear distinction of the 2 clusters based on the customer's spending habits

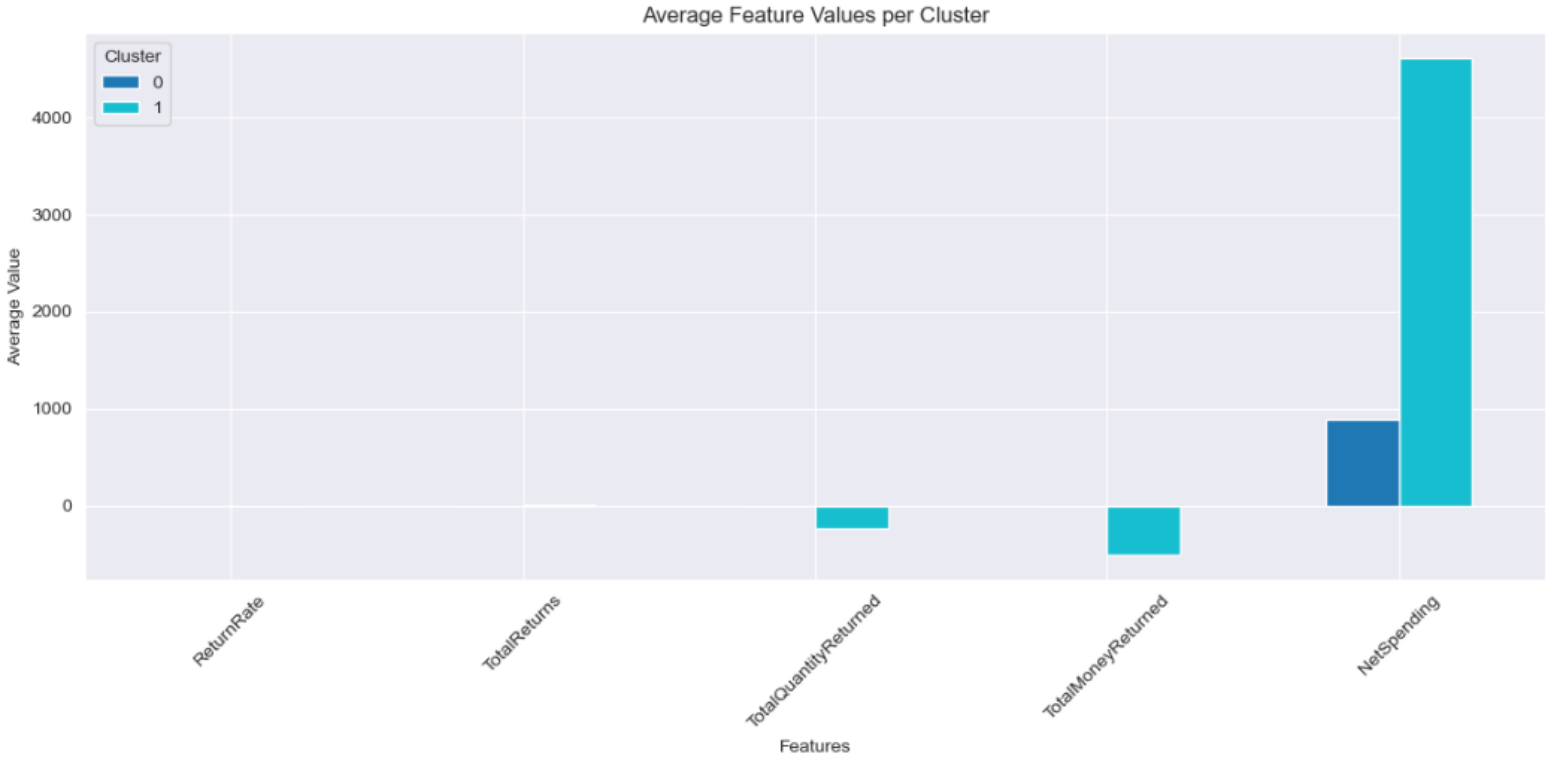




Model creation continued

- Now, let's use hierarchical clustering. First we'll use a dendrogram to find the right number of clusters then run it.
- The right number of clusters seems to be 9.
- The silhouette score is 0.58. Definitely worse than the K means model, but does show the clusters better than before without the normalizing

Model creation continued



- Finally, let's look at the average values of key return-related features for each customer cluster.
- From here, we can tell that Cluster 1 customers spend much more on average than Cluster 0. This indicates that Cluster 1 likely includes high-value customers.
- Cluster 1 has higher negative values for TotalMoneyReturned and TotalQuantityReturned, meaning they return more items and return higher-value products. Despite that, their high NetSpending suggests they still generate significant revenue.
- Both clusters have relatively low ReturnRates and TotalReturns on average. Differences here are minor compared to NetSpending.
- From this, we can tell that Cluster 0 represents lower-spending, lower-returning customers (possibly occasional or new buyers). Meanwhile, Cluster 1 represents high-spending, high-returning customers (possibly loyal or bulk buyers with more complex purchasing behavior).

Conclusion

- Clearly, the K means model did far better than the hierarchical clustering model. This can be seen by the K means model having a silhouette score over 0.1 higher than the hierarchical clustering model. Normalizing the data also helped make the models work far better than just scaling them.

- We have seen that despite there being a multitude of spending behaviors in the clusters, a majority of customers don’t return items. Even if they do, those who do spend far more money than those who do not. Thus, businesses shouldn’t worry too much about people returning items. If anything, it may be a good thing for them, since those customers will spend more money on the business anyways.

- If I was to do this again in the future, I’d use different scalers to see if they work better with the data.