**#Capstone2- Ecommerce Dataset- Customer Segmentation**

**Client Introduction**

Our client is an online retailer based in the UK. They sell all-occasion gifts, and many of their customers are wholesalers.

* Most of their customers are from the UK, but they have a small percent of customers from other countries.
* They want to create groups of these international customers based on their previous purchase patterns.
* Their goal is to provide more tailored services and improve the way they market to these international customers.



**Data Acquisition-**

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers

**What does currently Retailer do?**

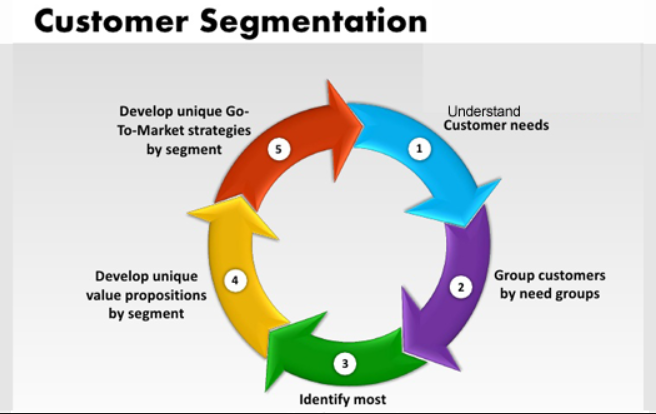
Currently, the retailer simply groups their international customers by country. As you'll see in the project, this is quite inefficient because:

1. There's a large number of countries (which kind defeats the purpose of creating groups).
2. Some countries have very few customers.
3. This approach treats large and small customers the same, regardless of their purchase patterns.

**Our Project Objective**

The retailer has hired us to help them create customer clusters, a.k.a "customer segments," through a data-driven approach.

* They've provided us a dataset of past purchase data at the transaction level.
* Our task is to build a clustering model using that dataset.
* Our clustering model should factor in both **aggregate sales patterns** and **specific items purchased**.



**Steps to achieve the client requirement-**

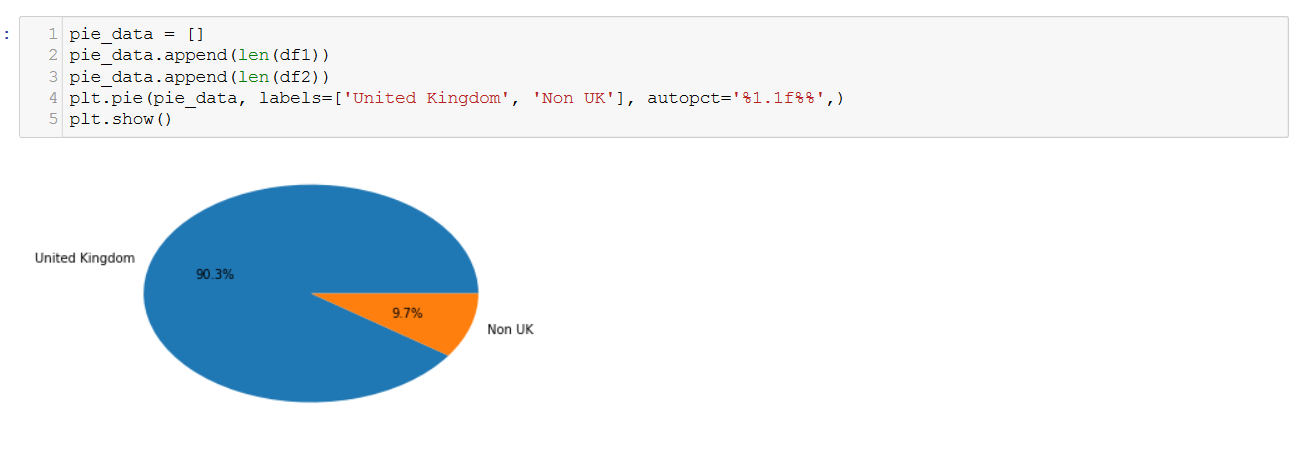
* Basic data preparation
* Exploring the data content- Basic exploratory data analysis
* Data Wrangling
* Dimensionality Reduction
* Principal Component Analysis
* Cluster Analysis
* Feature Comparison
* Conclusion Summary

**Data Description**

1. **InvoiceNo**: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
2. **StockCode**: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
3. **Description**: Product (item) name. Nominal.
4. **Quantity**: The quantities of each product (item) per transaction. Numeric.
5. **InvoiceDate**: Invoice Date and time. Numeric, the day and time when each transaction was generated.
6. **UnitPrice**: Unit price. Numeric, Product price per unit in sterling.
7. **CustomerID**: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
8. **Country**: Country name. Nominal, the name of the country where each customer resides

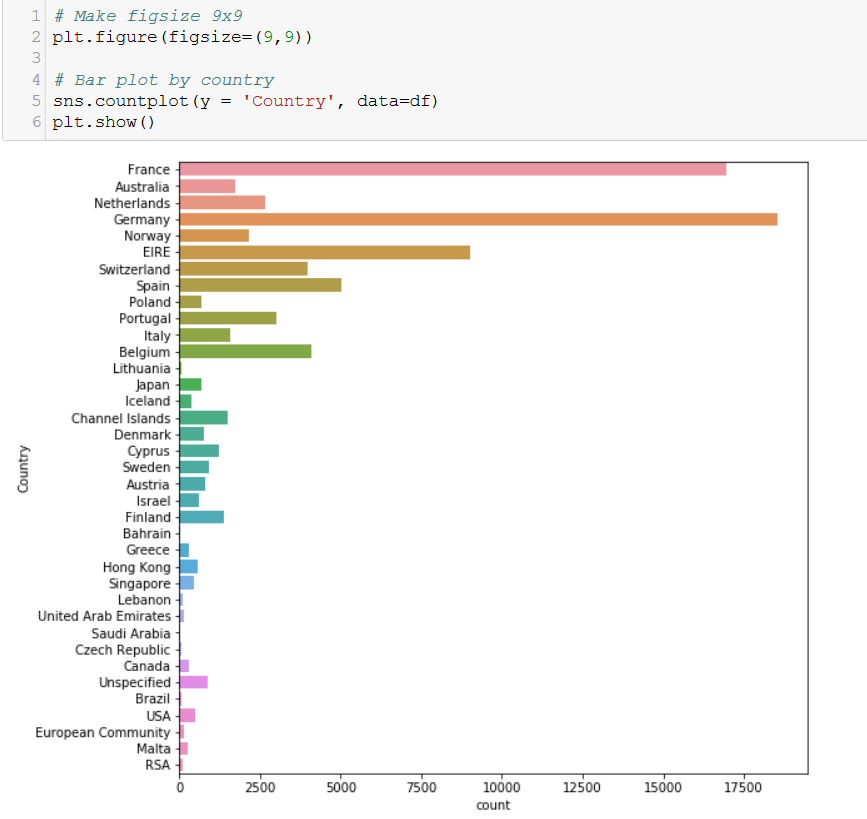
**Data Preparation**

Acquired data was loaded and was prepared for analysis. As, we can see the dataset contains most of the transactional record from UK, so we will create a dataframe for international transaction i.e other than United Kingdom for our analysis and solution



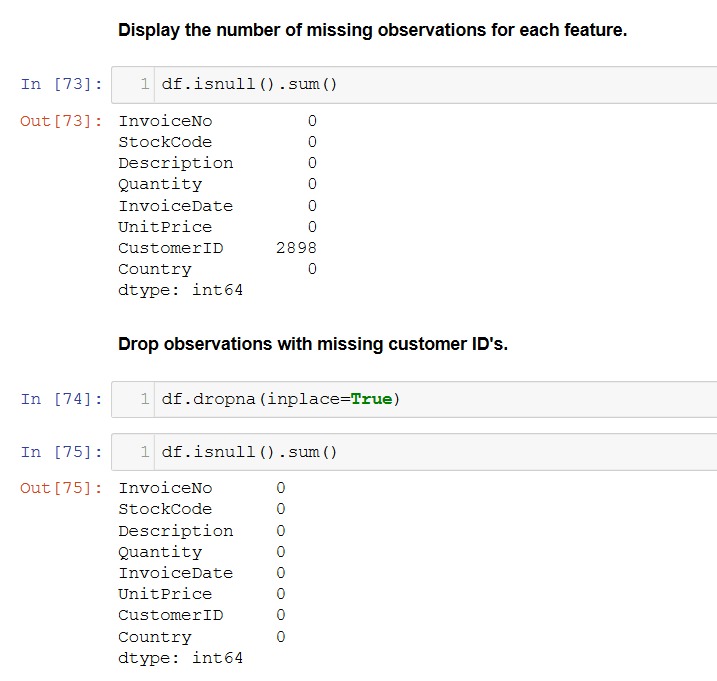
**Data Wrangling and Basic EDA**

Even though we eventually want customer-level data, it's still helpful to do some basic exploratory analysis at the transaction level.



**Data Cleaning-**

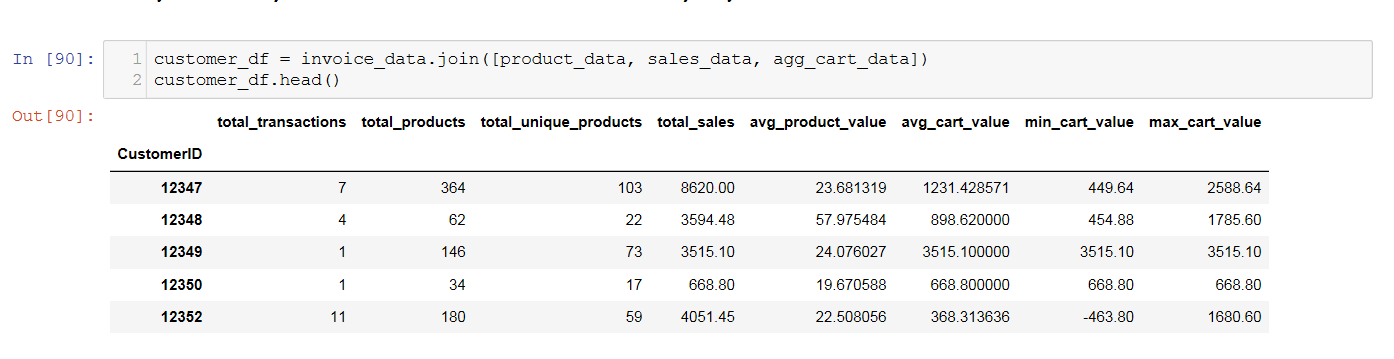
* **Drop observations with missing customer ID's**
* **Convert the CustomerID's and Invoice IDs from floats into strings**
* **Saved the cleaned dataset into CSV**



**Customer-level data Aggregation-**

For the better analysis purpose all the possible data combination were done and data was rolled up at customer level following below steps

* **Aggregating product data by customer**
* **Aggregate sales data by customer**
* Aggregating Cart data to Customer level
* Joining All data frames created above and converted into a csv ‘***Customer\_df\_analysis.csv’***



**Dimensionality Reduction**

As, our client wishes to incorporate information about **specific item purchases** into the clusters. For example, our model should be more likely to group together customers who buy similar items.

* Now, we prepared individual item features for our clustering algorithms.
* We'll introduce a simple way to reduce the number of dimensions by applying thresholds.

Below are the steps used as a part of dimensionality reduction -

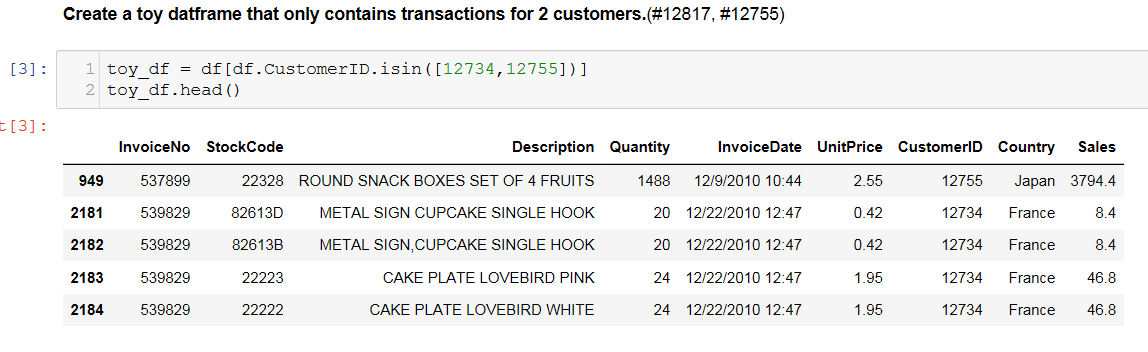
* Import Cleaned dataset
* Applied Toy example on sample data for reduce dimensionality
* Applied Toy example to entire dataset- High Dimensionality
* Created a Threshold to reduce dimensionality of entire dataframe

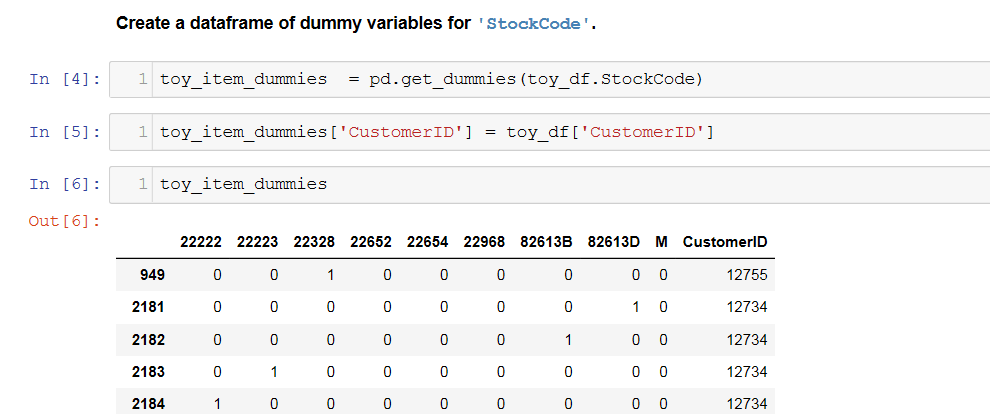
**Applied Toy example on sample data for reduce dimensionality-**

To illustrate how we'll roll up item information to the customer level, let's use another toy example.

First, create a toy\_df that only contains transactions for 2 customers.

* Include transactions for these 2 CustomerID's: 12755 and 12734
* By the way, there's nothing special about these customers. We just chose them because they have relatively few purchases, making the toy example more manageable.

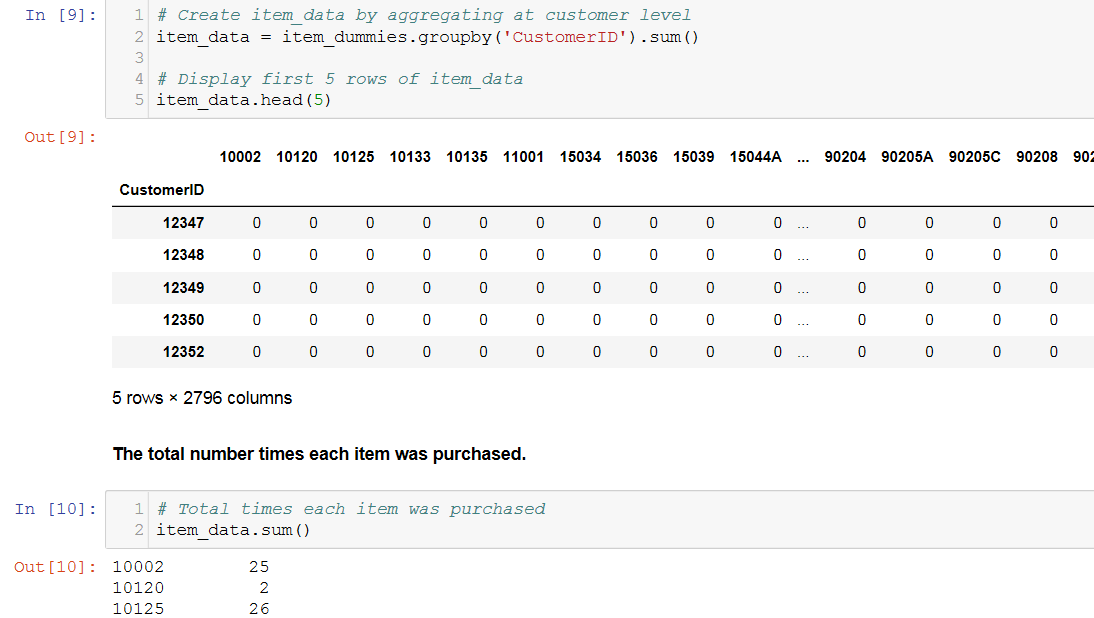




**Applied Toy example to entire dataset- High Dimensionality**

Create a dataframe of dummy variables for 'StockCode', this time for the full dataset.

* Name it item\_dummies.
* Then, add 'CustomerID' to this new dataframe so that we can roll up by customer later.
* **Saved this customer-level item dataframe as 'item\_data.csv'**

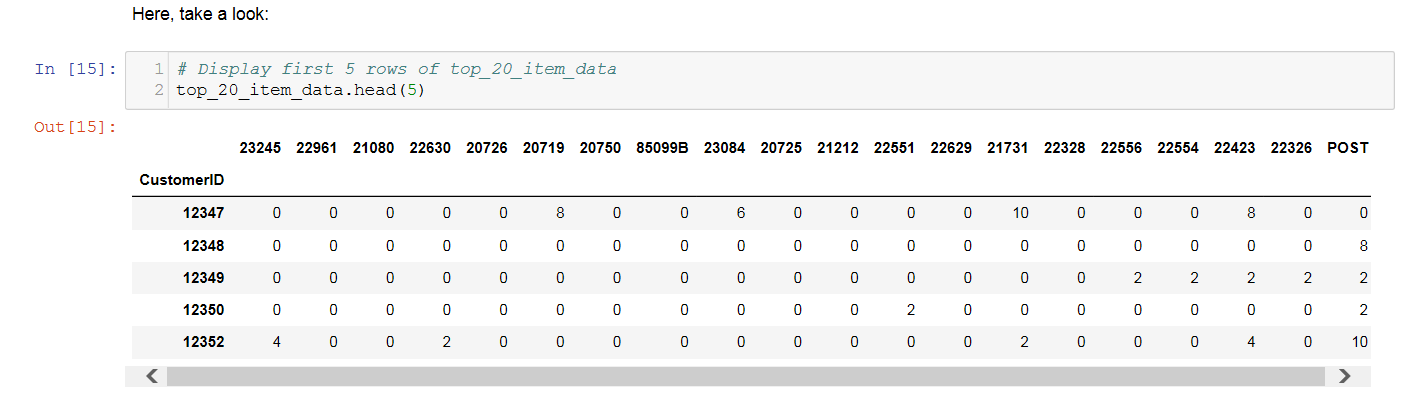


**Created a Threshold to reduce dimensionality of entire dataframe**

Simple and straightforward way to reduce the dimensionality of this item data is to set a threshold for keeping features.

We can see which items those are and the number of times they were purchased.

1. Take the sum by column.
2. Sort the values.
3. Looking at the 20 sample records for threshold in ascending order and save it to **threshold\_item\_data.csv**



**Principle Component Analysis**

Principal Component Analysis, or PCA, is a popular dimensionality reduction technique.

PCA seeks to create new features by finding linear combinations of your original ones. These new features, called **principal components**, are meant to maximize the **"explained variance,"** which we'll explain further in the module.

* Here, we'll prepare individual item features for our clustering algorithms, except this time we'll use PCA instead of thresholding.
* PCA is especially effective when you have many correlated features.

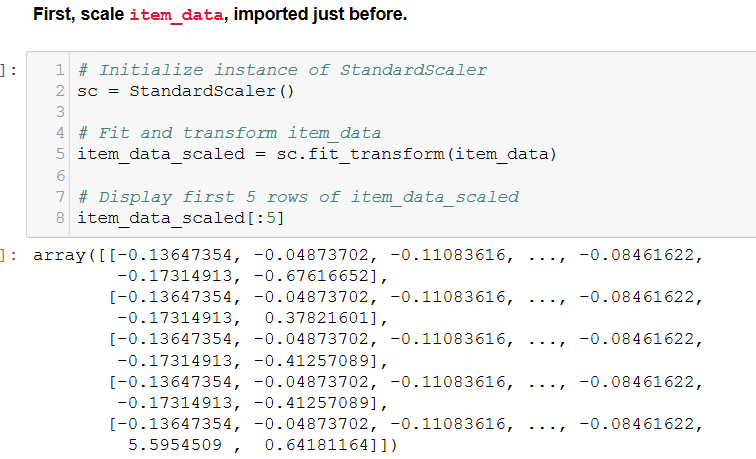
Note- PCA creates new features that replace the original ones.

Below are the steps performed

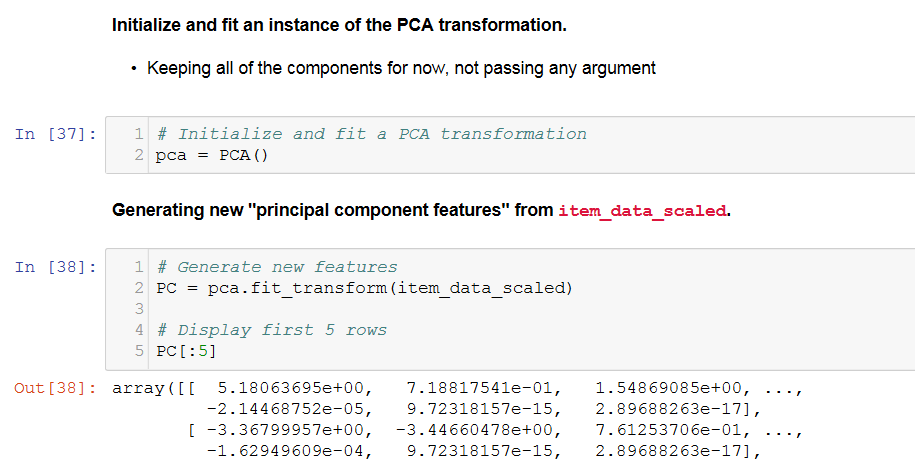
* Item data - Principal Component Analysis
* Explained Variance
* Dimensionality Reduction

**Item data - Principal Component Analysis**

Scaled item\_data, which we saved before and named it ‘item\_data\_scaled’

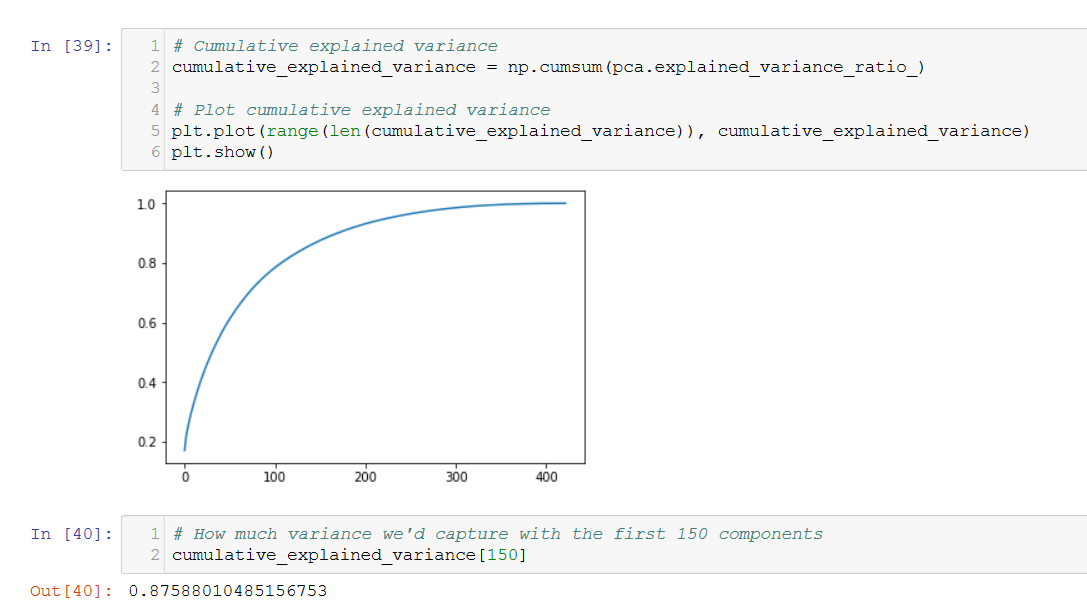


**Next, initialize and fit an instance of the PCA transformation and generate new "principal component features" from item\_data\_scaled.**

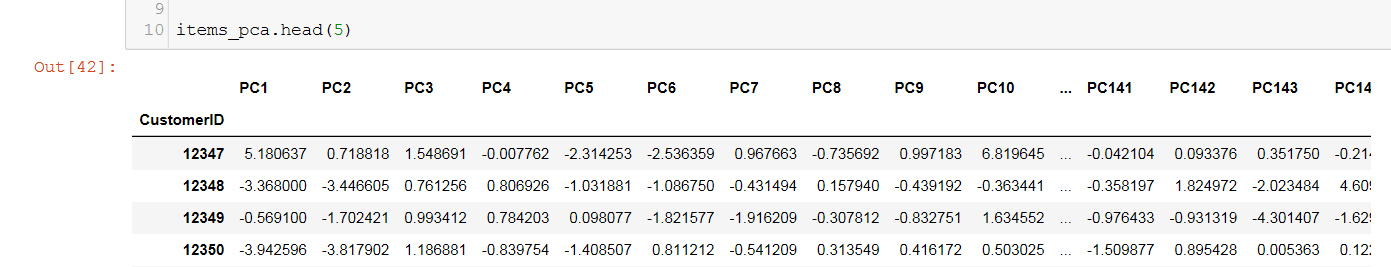


**Explained Variance**

**It's very helpful to calculate and plot the explained variance.**

* This will tell us the total amount of variance we'd capture if we kept up to the n-th component.
* First, we'll use np.cumsum() to calculate the cumulative explained variance.
* Then, we'll plot it so we can see how many features we'd need to keep in order to capture most of the original variance.

**Initialize and fit another PCA transformation with 150 components and save it as ‘pca\_item\_data.csv’**



**Cluster Analysis**

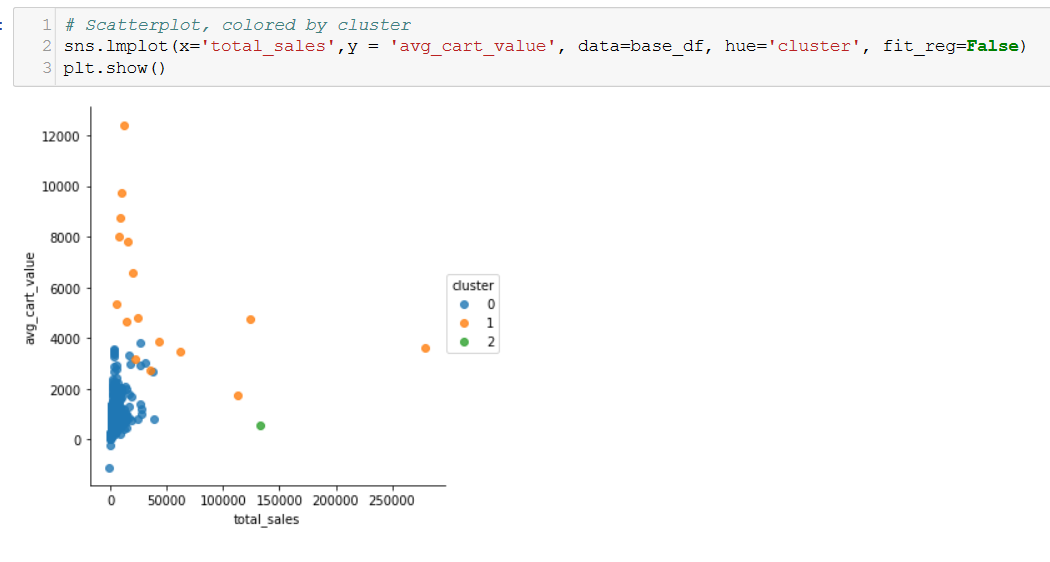
For clustering problems, the chosen input features are usually more important than which algorithm you use.

* Here, we'll apply the K-Means algorithm to 3 different feature sets for comparison

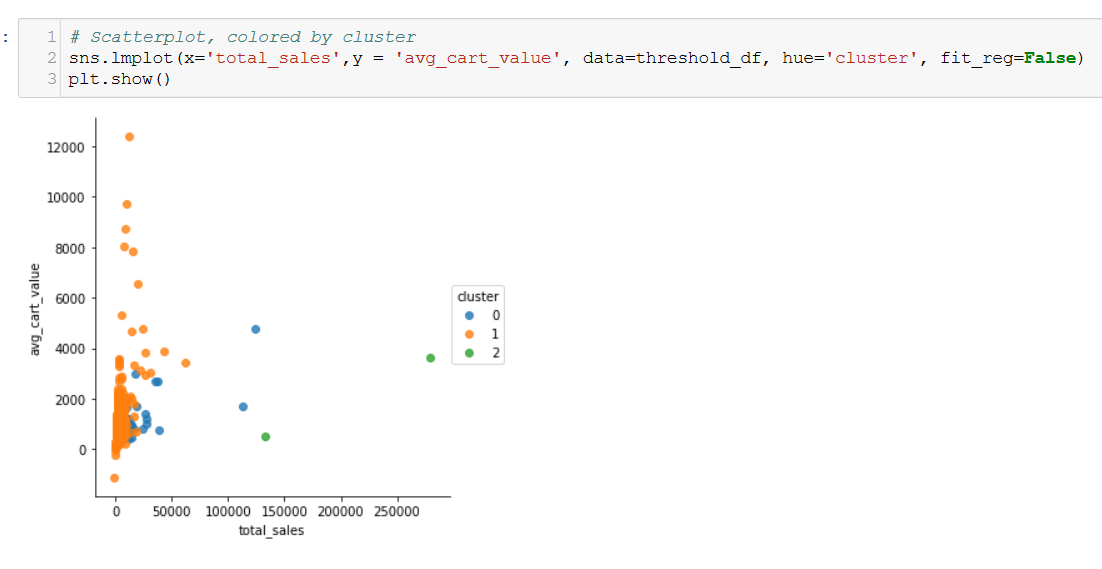
**Import 3 CSV files we've saved before**

* 'Customer\_df\_analysis.csv' as base\_df.
* 'threshold\_item\_data.csv' as threshold\_item\_data.
* 'pca\_item\_data.csv' as pca\_item\_data.

**Feature1- Result after applying K-Means algorithm to base\_df with random\_seed=126 and three cluster**



**Feature2- Result after applying K-Means algorithm to threshold\_item\_data with random\_seed=126 and three cluster**



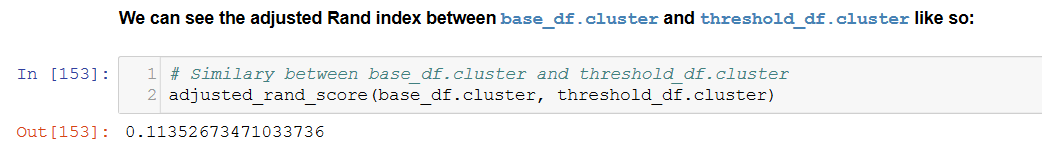
**Feature3- Result after applying K-Means algorithm to pca\_item\_data with random\_seed=126 and three cluster**



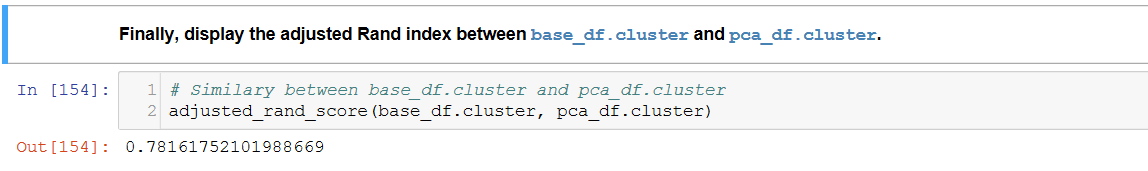
**Model comparison-**

Comparing features developed before using scikit learn- adjusted randon score as below-

**1.Compare base\_df.cluster and threshold\_df.cluster**



**2. Compare base\_df.cluster and pca\_df.cluster**



**Summary-**

* The first stage of this work consisted in dimensionality reduction to incorporate specific item purchases and build a platform to apply clustering algorithm.
* The second stage of this work was aimed to perform principle component analysis which is a unique way of dimensionality reduction and prepare individual item features for our clustering algorithms
* In the final stage, we applied K-Means algorithm to the different features built and performed the model comparison.
* Based on the adjusted random score while comparing base and threshold cluster with that of PCA, and it was found that base data frame was much closer to PCA data frame.
* The performance of the classifier therefore seems correct given the potential shortcomings of the current model