#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?

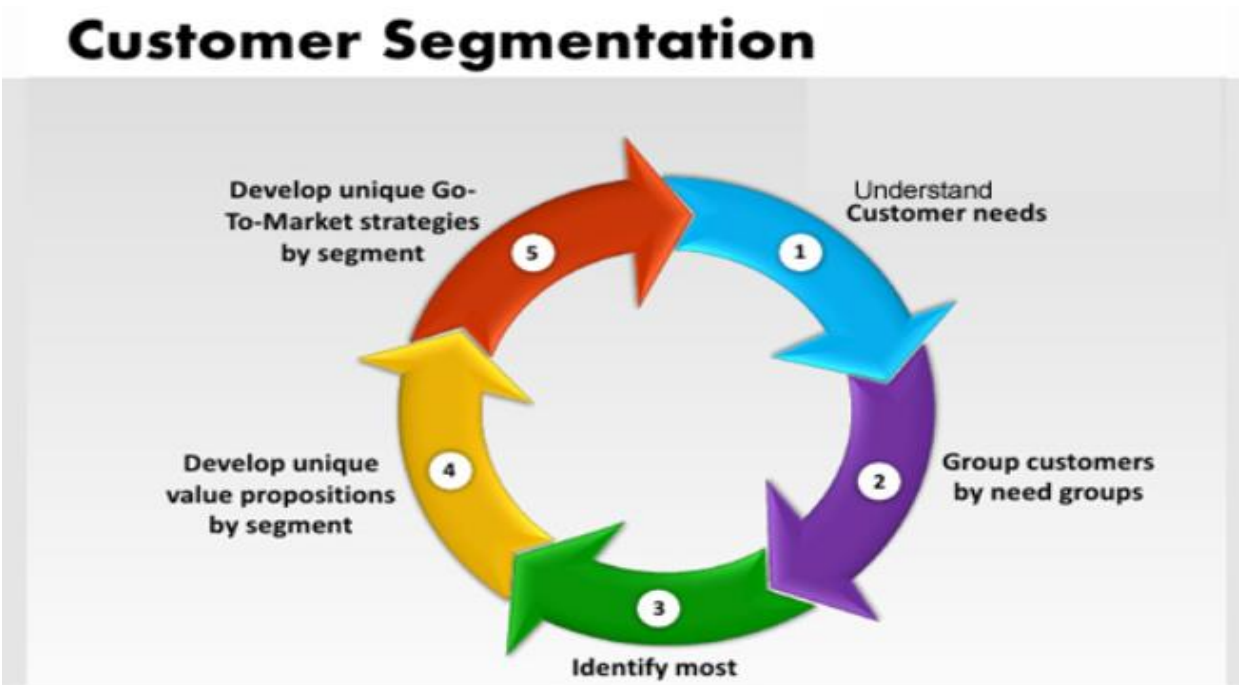
Currently, the retailer simply groups their international customers by country. As you'll see in the project, this is quite inefficient because:

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

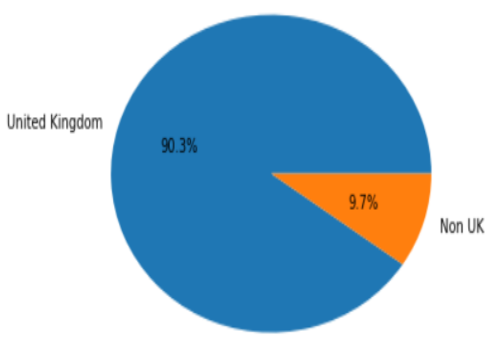
Data Description

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

```
pie_data = []
pie_data.append(len(df1))
pie_data.append(len(df2))
plt.pie(pie_data, labels=['United Kingdom', 'Non UK'], autopct='%1.1f%%',)
plt.show()
```

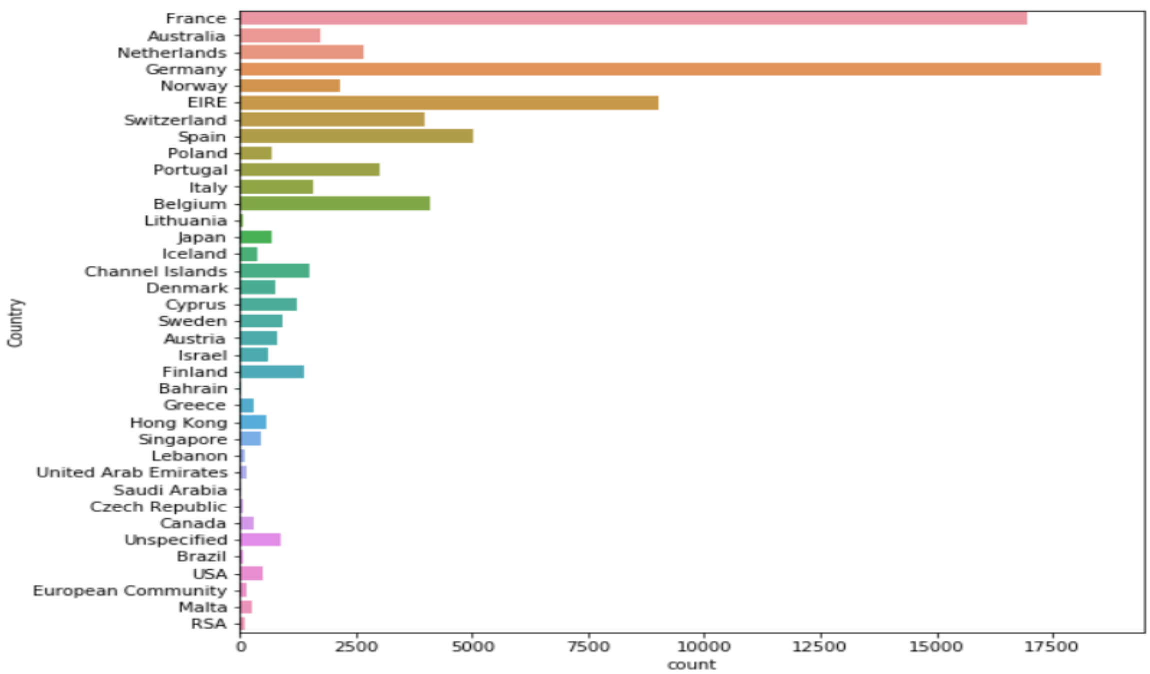


Data Wrangling and Basic EDA

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

```python
# Make figsize 9x9
plt.figure(figsize=(9,9))

# Bar plot by country
sns.countplot(y = 'Country', data=df)
plt.show()
```



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

Display the number of missing observations for each feature.

```
In [73]:  df.isnull().sum()
Out[73]:  InvoiceNo         0
          StockCode         0
          Description       0
          Quantity          0
          InvoiceDate       0
          UnitPrice         0
          CustomerID     2898
          Country           0
          dtype: int64
```

Drop observations with missing customer ID's.

```
In [74]:  df.dropna(inplace=True)

In [75]:  df.isnull().sum()
Out[75]:  InvoiceNo      0
          StockCode      0
          Description    0
          Quantity       0
          InvoiceDate    0
          UnitPrice      0
          CustomerID     0
          Country        0
          dtype: int64
```

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'***

In [90]:
```
customer_df = invoice_data.join([product_data, sales_data, agg_cart_data])
customer_df.head()
```

Out[90]:

CustomerID	total_transactions	total_products	total_unique_products	total_sales	avg_product_value	avg_cart_value	min_cart_value	max_cart_value
12347	7	364	103	8620.00	23.681319	1231.428571	449.64	2588.64
12348	4	62	22	3594.48	57.975484	898.620000	454.88	1785.60
12349	1	146	73	3515.10	24.076027	3515.100000	3515.10	3515.10
12350	1	34	17	668.80	19.670588	668.800000	668.80	668.80
12352	11	180	59	4051.45	22.508056	368.313636	-463.80	1680.60

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.

Create a toy datframe that only contains transactions for 2 customers.(#12817, #12755)

[3]:
```
toy_df = df[df.CustomerID.isin([12734,12755])]
toy_df.head()
```

t[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
949	537899	22328	ROUND SNACK BOXES SET OF 4 FRUITS	1488	12/9/2010 10:44	2.55	12755	Japan	3794.4
2181	539829	82613D	METAL SIGN CUPCAKE SINGLE HOOK	20	12/22/2010 12:47	0.42	12734	France	8.4
2182	539829	82613B	METAL SIGN,CUPCAKE SINGLE HOOK	20	12/22/2010 12:47	0.42	12734	France	8.4
2183	539829	22223	CAKE PLATE LOVEBIRD PINK	24	12/22/2010 12:47	1.95	12734	France	46.8
2184	539829	22222	CAKE PLATE LOVEBIRD WHITE	24	12/22/2010 12:47	1.95	12734	France	46.8

Create a dataframe of dummy variables for 'StockCode'.

In [4]:
```
toy_item_dummies  = pd.get_dummies(toy_df.StockCode)
```

In [5]:
```
toy_item_dummies['CustomerID'] = toy_df['CustomerID']
```

In [6]:
```
toy_item_dummies
```

Out[6]:

	22222	22223	22328	22652	22654	22968	82613B	82613D	M	CustomerID
949	0	0	1	0	0	0	0	0	0	12755
2181	0	0	0	0	0	0	0	1	0	12734
2182	0	0	0	0	0	0	1	0	0	12734
2183	0	1	0	0	0	0	0	0	0	12734
2184	1	0	0	0	0	0	0	0	0	12734

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'**

In [9]:

```
# Create item_data by aggregating at customer level
item_data = item_dummies.groupby('CustomerID').sum()

# Display first 5 rows of item_data
item_data.head(5)
```

Out[9]:

CustomerID	10002	10120	10125	10133	10135	11001	15034	15036	15039	15044A	...	90204	90205A	90205C	90208	902
12347	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
12348	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
12349	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
12350	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
12352	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 2796 columns

The total number times each item was purchased.

In [10]:

```
# Total times each item was purchased
item_data.sum()
```

Out[10]:

```
10002        25
10120         2
10125        26
```

<u>Created a Threshold to reduce dimensionality of entire dataframe</u>

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`**

Here, take a look:

In [15]:

```
# Display first 5 rows of top_20_item_data
top_20_item_data.head(5)
```

Out[15]:

CustomerID	23245	22961	21080	22630	20726	20719	20750	85099B	23084	20725	21212	22551	22629	21731	22328	22556	22554	22423	22326	POST
12347	0	0	0	0	0	8	0	0	6	0	0	0	0	10	0	0	0	8	0	0
12348	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
12349	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
12350	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	2
12352	4	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	4	0	10

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'

First, scale `item_data`, imported just before.

```
]:
# Initialize instance of StandardScaler
sc = StandardScaler()

# Fit and transform item_data
item_data_scaled = sc.fit_transform(item_data)

# Display first 5 rows of item_data_scaled
item_data_scaled[:5]
```

```
]: array([[-0.13647354, -0.04873702, -0.11083616, ..., -0.08461622,
          -0.17314913, -0.67616652],
         [-0.13647354, -0.04873702, -0.11083616, ..., -0.08461622,
          -0.17314913,  0.37821601],
         [-0.13647354, -0.04873702, -0.11083616, ..., -0.08461622,
          -0.17314913, -0.41257089],
         [-0.13647354, -0.04873702, -0.11083616, ..., -0.08461622,
          -0.17314913, -0.41257089],
         [-0.13647354, -0.04873702, -0.11083616, ..., -0.08461622,
           5.5954509 ,  0.64181164]])
```

Next, initialize and fit an instance of the PCA transformation and generate new "principal component features" from **`item_data_scaled`**.

Initialize and fit an instance of the PCA transformation.

- Keeping all of the components for now, not passing any argument

In [37]:
```
# Initialize and fit a PCA transformation
pca = PCA()
```

Generating new "principal component features" from `item_data_scaled`.

In [38]:
```
# Generate new features
PC = pca.fit_transform(item_data_scaled)

# Display first 5 rows
PC[:5]
```

Out[38]:
```
array([[  5.18063695e+00,   7.18817541e-01,   1.54869085e+00, ...,
         -2.14468752e-05,   9.72318157e-15,   2.89688263e-17],
       [ -3.36799957e+00,  -3.44660478e+00,   7.61253706e-01, ...,
         -1.62949609e-04,   9.72318157e-15,   2.89688263e-17],
```

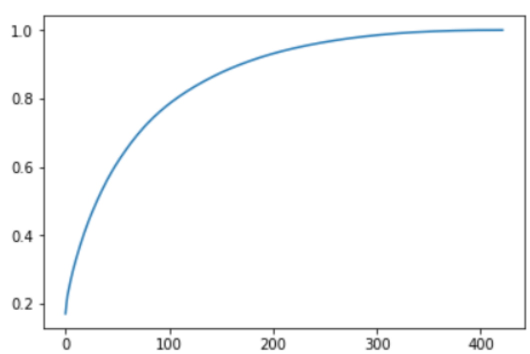
Explained Variance

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.

In [39]:
```
# Cumulative explained variance
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)

# Plot cumulative explained variance
plt.plot(range(len(cumulative_explained_variance)), cumulative_explained_variance)
plt.show()
```



In [40]:
```
# How much variance we'd capture with the first 150 components
cumulative_explained_variance[150]
```

Out[40]: 0.87588010485156753

Initialize and fit another PCA transformation with 150 components and save it as '**pca_item_data.csv**'

```
items_pca.head(5)
```

Out[42]:

CustomerID	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC141	PC142	PC143	PC14
12347	5.180637	0.718818	1.548691	-0.007762	-2.314253	-2.536359	0.967663	-0.735692	0.997183	6.819645	...	-0.042104	0.093376	0.351750	-0.21
12348	-3.368000	-3.446605	0.761256	0.806926	-1.031881	-1.086750	-0.431494	0.157940	-0.439192	-0.363441	...	-0.358197	1.824972	-2.023484	4.60
12349	-0.569100	-1.702421	0.993412	0.784203	0.098077	-1.821577	-1.916209	-0.307812	-0.832751	1.634552	...	-0.976433	-0.931319	-4.301407	-1.62
12350	-3.942596	-3.817902	1.186881	-0.839754	-1.408507	0.811212	-0.541209	0.313549	0.416172	0.503025	...	-1.509877	0.895428	0.005363	0.12

Cluster Analysis

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

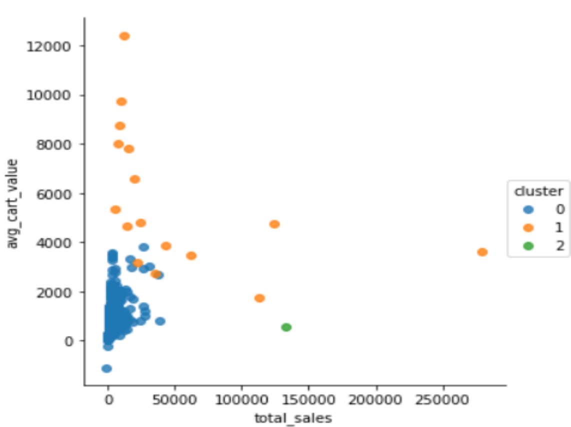
- Here, we'll apply the K-Means algorithm to 3 different feature sets for comparison

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`.

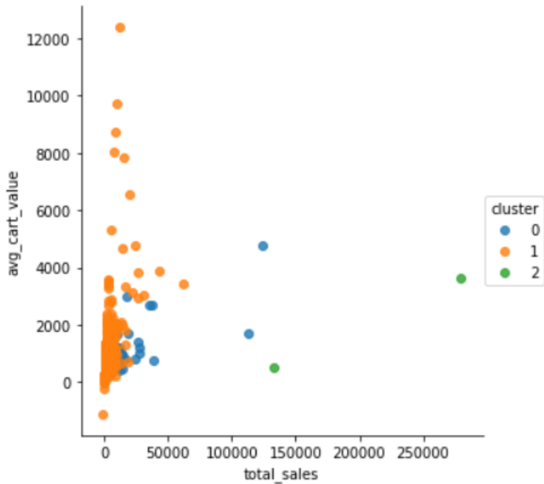
Feature1- Result after applying K-Means algorithm to base_df with random_seed=126 and three cluster

```
# Scatterplot, colored by cluster
sns.lmplot(x='total_sales',y = 'avg_cart_value', data=base_df, hue='cluster', fit_reg=False)
plt.show()
```



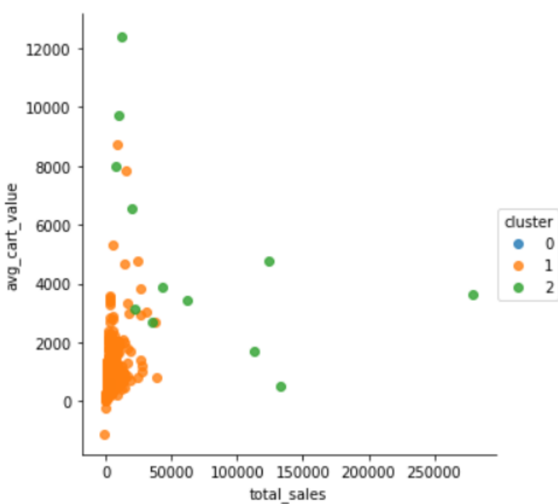
Feature2- Result after applying K-Means algorithm to threshold_item_data with random_seed=126 and three cluster

```
# Scatterplot, colored by cluster
sns.lmplot(x='total_sales',y = 'avg_cart_value', data=threshold_df, hue='cluster', fit_reg=False)
plt.show()
```



Feature3- Result after applying K-Means algorithm to pca_item_data with random_seed=126 and three cluster

```
# Scatterplot, colored by cluster
sns.lmplot(x='total_sales',y = 'avg_cart_value', data=pca_df, hue='cluster', fit_reg=False)
plt.show()
```



Model comparison-

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

1.Compare base_df.cluster and threshold_df.cluster

We can see the adjusted Rand index between `base_df.cluster` and `threshold_df.cluster` like so:

In [153]:
```
# Similary between base_df.cluster and threshold_df.cluster
adjusted_rand_score(base_df.cluster, threshold_df.cluster)
```

Out[153]: 0.11352673471033736

2. Compare base_df.cluster and pca_df.cluster

Finally, display the adjusted Rand index between `base_df.cluster` and `pca_df.cluster`.

In [154]:
```
# Similary between base_df.cluster and pca_df.cluster
adjusted_rand_score(base_df.cluster, pca_df.cluster)
```

Out[154]: 0.78161752101988669

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