Customer Segmentation Analysis Model of E-Commerce Industry

AI Product Prototype Analysis

Chaganti Venkatarami Reddy, Subesha Sasmal, Parth Sukla

AI Prototype Analysis for Customer Segmentation

To be useful, segments must be measurable, substantial, accessible, differentiable, and accountable.

(Philip Kotler)

Abstract

Artificial intelligence (AI) has the potential to revolutionize pathology. AI refers to the application of modern machine learning techniques to digital tissue images in order to detect, quantify, or characterize specific cell or tissue structures. By automating time-consuming diagnostic tasks, AI can greatly reduce the workload and help to remedy the serious shortage of pathologists. At the same time, AI can make analyses more sensitive and reproducible and it can capture novel biomarkers from tissue morphology for precision medicine. In a survey from the year 2019 of 487 pathologists from 54 countries, a great majority looked forward to using AI as a diagnostic tool.

In this report we are going to analyse and explore a E-Commerce bussiness data. And we will create a model to make customer segments based on new customers and their sales.

KeyWords : *Customers, Segments, E-Commerce, Prototypes, Market segmentation, Stock Code, Basket Price, Cluster analysis, Regression.*

Data Collection

The data has been collected manually, and the sources used for this process are listed below :

- https://www.kaggle.com/datasets
- https://data.gov.in/
- https://www.data.gov/
- https://data.worldbank.org/
- https://datasetsearch.research.google.com/

Business Need Assessment

- Many E-Commerce businessess should have a mandatory analysis of their previous sales and they have to create new segments based on new customers.
- Analysing their datasets and forecatsing their sales will enable them to buy grocery, according, prepare the types of items that their customers like during a particular time of the year.
- There are lot of E-Commerce bussinessess of same type in the city, so this is a big market but we have to **Segment the Market** according to which item is mostly selling.
- Properly done, the model can be extended to whole E-Commerce businessess.

Target Specifications and Characterizations

- The target here is to develop a model that will forecast future sales, cut down costs and new segments(customers) of an E-Commerce Industry.
- The trend recognition has to be done by a data scientist who has some knowledge about the E-Commerce industry.

- Employing someone who has no knowledge about the E-Commerce industry is not a great idea as they will not be able to give the insights of someone who knows about this field.
- The model should be able to handle large volumes of data, as in a E-Commerce industry there will be a lot of features for the model to look at and the size of data depends on the sales of a particular month or week.
- We should also know in advance whether the customers need our model to forecast the sales for a week or for a month.

Benchmarking Alternative Products

Many big companies like **Amazon, Flipkart** have started implementing ML/AI in their outlets and while they are making new products as well. These companies have identified the use of ML and AI and are benefitting from it. A lot of AI companies have also entered the market helping big corporations.

Applicable Patents

Systems and methods for contextual vocabularies and customer segmentation by *SDL Netherlands B.V. , Amsterdam Zuidoost (NL)* was patented on 2011-01-29.

Applicable Regulations:

- Data Protection and Privacy Rules.
- License for the open-source codes that might be used in the model implementation.
- Laws related to AI.

Applicable Constraints

- Data Collection from the customer.
- The customer should know about the time, money and scope of the project before it starts.
- Transperant use of the data obtained from the customer.

Bussiness Opportunities

The target customers here are mainly local E-Commerce Industries or stores who have a good number of customers.

Many local E-Commerce stores might be stuck at the same level for some period of time without knowing what to do, to generate more revenue. The main goal while implementing an ML model for their bussiness will be to reduce their cost by suggesting what sort of materials the customers are more buying at a particular time of the year. They are the primary bussiness targets.

If the customer has a delivery option like **Amazon, Flipkart** etc.,, then we will be able to find the areas that requires more deliveries and weed out areas that are not bringing much profit.

If the model is successfully implemented for the previously mentioned targets, the model can be expanded for E-Commerce Industries that are in multiple.

Concept Development

We must first understand the environment before we start working on a model and the type of items, the people in that region like and what are the traditions there. After gaining sufficient knowledge about the environment we have to start collecting data. After collecting the data, we have to perform EDA which is used to identify patterns in the dataset and it will help us zone in on the areas that are leaking money. Visualization will help a lot here. Once we have found the trend and outliers, the next step is to use the basic regression models and time-series models , in which we will fit our training dataset and see what sort of results we will be getting. After analysing various parameters like squared-error, etc we will know what type of model to use and what type of model should our model be based around. The models will be regression models and time-series models.

Implementation

Packages/Tools Used:

1. **Numpy:** To calculate various calculations related to arrays .
2. **Pandas:** To read or load the datasets.
3. **SKLearn:** This is a Machine Learning library which contains builtin Machine Learning algorithms.

Data Preprocessing

Data Cleaning

The data collected is compact and is partly used for visualization purposes and partly for clustering. Python libraries such as NumPy, Pandas, Scikit-Learn, and SciPy are used for the workflow, and the results obtained are ensured to be reproducible.

```
df_initial = pd.read_csv('data.csv',encoding="ISO-8859-1",
                         dtype={'CustomerID': str,'InvoiceID': str})
print('Dataframe dimensions:', df_initial.shape)
#_____
df_initial['InvoiceDate'] = pd.to_datetime(df_initial['InvoiceDate'])
#_____________________________________________________________
# gives some infos on columns types and numer of null values
tab_info=pd.DataFrame(df_initial.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(index={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shape[0]*100).T.
                         rename(index={0:'null values (%)'}))
display(tab_info)
#________________
# show first lines
display(df_initial[:5])
```

```
Dataframe dimensions: (541909, 8)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
column type	object	object	object	int64	datetime64[ns]	float64	object	object
null values (nb)	0	0	1454	0	0	0	135080	0
null values (%)	0.0	0.0	0.268311	0.0	0.0	0.0	24.926694	0.0

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom

EDA

We start the Exploratory Data Analysis with some data Analysis drawn from the data without Principal Component Analysis and with some Principal Component Analysis in the dataset obtained from the combination of all the data we have. PCA is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. The process helps in reducing

dimensions of the data to make the process of classification/regression or any form of machine learning, cost-effective.

Exploring the content of variables

This dataframe contains 8 variables that correspond to:

- **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.
- **StockCode:** Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
- **Description:** Product (item) name. Nominal.
- **Quantity:** The quantities of each product (item) per transaction. Numeric.
- **InvoiceDate:** Invoice Date and time. Numeric, the day and time when each transaction was generated.
- **UnitPrice:** Unit price. Numeric, Product price per unit in sterling.
- **CustomerID:** Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
- **Country:** Country name. Nominal, the name of the country where each customer resides.

Let's have a look at no.of orders from different countries:

We see that the dataset is largely dominated by orders made from the UK.

Customers and Products

It can be seen that the data concern 4372 users and that they bought 3684 different products. The total number of transactions carried out is of the order of 22'000.

Now we will determine the number of products purchased in every transaction:

```
temp = df_initial.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['InvoiceDate'].count()
nb_products_per_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})
nb_products_per_basket[:10].sort_values('CustomerID')
```

	CustomerID	InvoiceNo	Number of products
0	12346	541431	1
1	12346	C541433	1
2	12347	537626	31
3	12347	542237	29
4	12347	549222	24
5	12347	556201	18
6	12347	562032	22
7	12347	573511	47
8	12347	581180	11
9	12348	539318	17

The first lines of this list shows several things worthy of interest:

- The existence of entries with the prefix C for the InvoiceNo variable: this indicates transactions that have been canceled.
- The existence of users who only came once and only purchased one product (e.g. n°12346).
- The existence of frequent users that buy a large number of items at each order.

Cancelling Orders

First of all, we count the number of transactions corresponding to canceled orders:

```
nb_products_per_basket['order_canceled'] = nb_products_per_basket['InvoiceNo'].apply(lambda x:int('C' in x))
display(nb_products_per_basket[:5])
#_____________________________________________________________________________________
n1 = nb_products_per_basket['order_canceled'].sum()
n2 = nb_products_per_basket.shape[0]
print('Number of orders canceled: {}/{} ({:.2f}%) '.format(n1, n2, n1/n2*100))
```

	CustomerID	InvoiceNo	Number of products	order_canceled
0	12346	541431	1	0
1	12346	C541433	1	1
2	12347	537626	31	0
3	12347	542237	29	0
4	12347	549222	24	0

```
Number of orders canceled: 3654/22190 (16.47%)
```

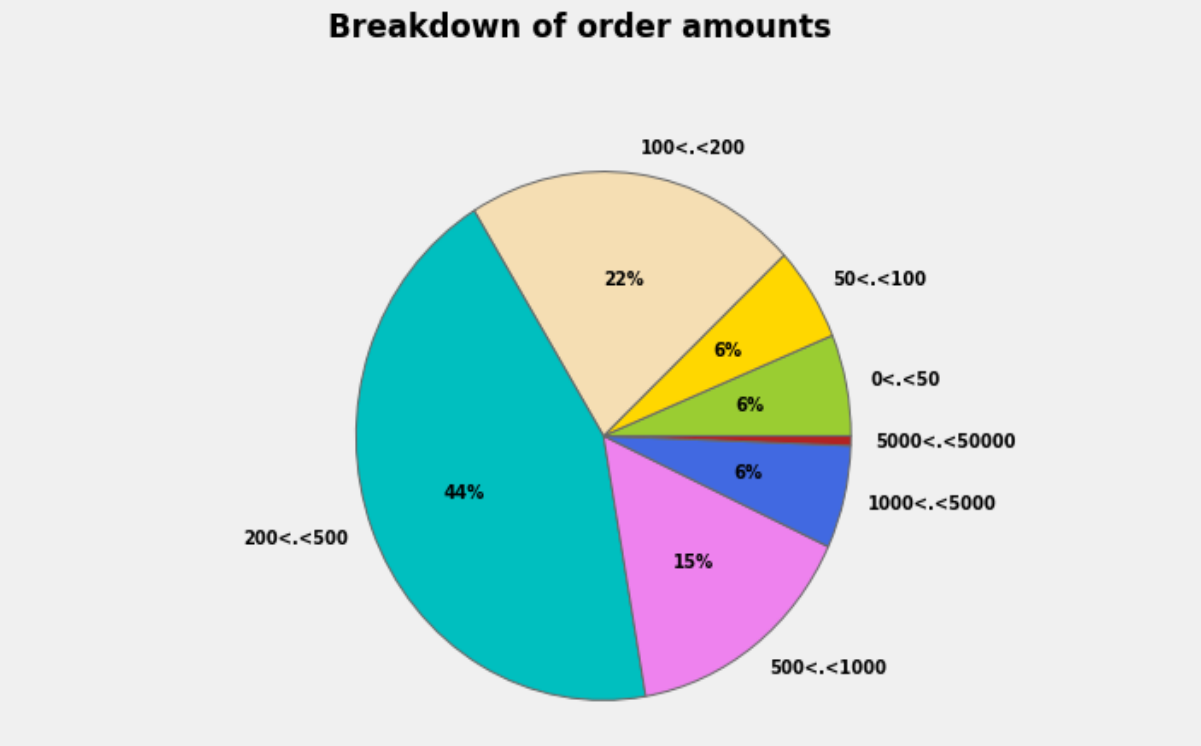
We note that the number of cancellations is quite large (16% of the total number of transactions).

On these few lines, we see that when an order is canceled, we have another transactions in the dataframe, mostly identical except for the **Quantity** and **InvoiceDate** variables. We decide to check if this is true for all the entries. To do this, I decide to locate the entries that indicate a negative quantity and check if there is systematically an order indicating the same quantity (but positive), with the same description (**CustomerID**, **Description** and **UnitPrice**):

```
df_check = df_initial[df_initial['Quantity'] < 0][['CustomerID','Quantity',
                                                   'StockCode','Description','UnitPrice']]
for index, col in  df_check.iterrows():
    if df_initial[(df_initial['CustomerID'] == col[0]) & (df_initial['Quantity'] == -col[1])
                & (df_initial['Description'] == col[2])].shape[0] == 0:
        print(df_check.loc[index])
        print(15*'-'+'>'+' HYPOTHESIS NOT FULFILLED')
        break
```

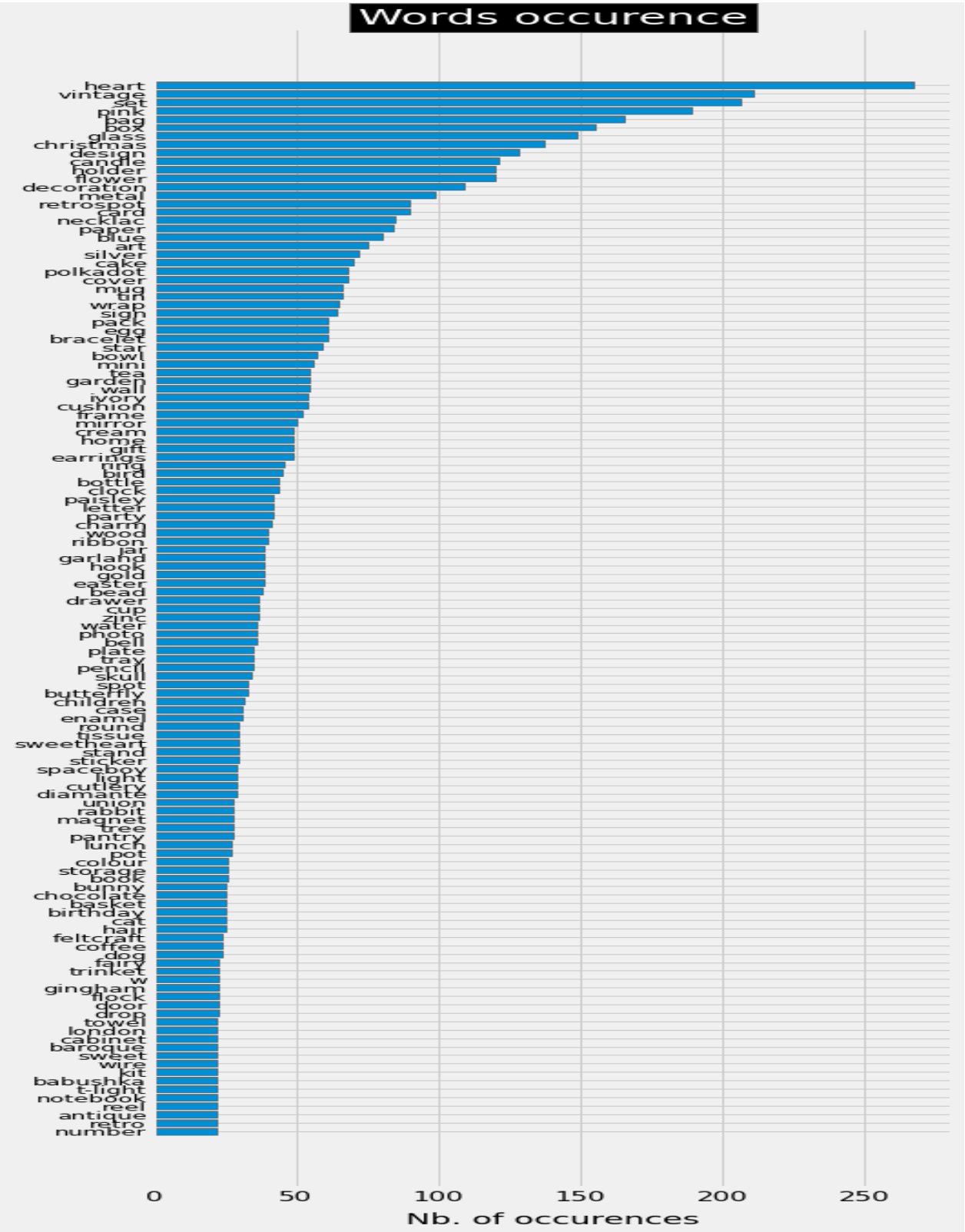
```
CustomerID        14527
Quantity             -1
StockCode             D
Description    Discount
UnitPrice          27.5
Name: 141, dtype: object
---------------> HYPOTHESIS NOT FULFILLED
```

Breakdown of Order Amounts



It can be seen that the vast majority of orders concern relatively large purchases given that 65% of purchases give prizes in excess of £ 200.

Most Common Keywords



Creating Clusters of products

In the case of matrices with binary encoding, the most suitable metric for the calculation of distances is the Hamming's metric. In order to define (approximately) the number of clusters that best represents the data, we use the silhouette score:

```
matrix = X.to_numpy()
for n_clusters in range(3,10):
    kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=30)
    kmeans.fit(matrix)
    clusters = kmeans.predict(matrix)
    silhouette_avg = silhouette_score(matrix, clusters)
    print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)
```

```
For n_clusters = 3 The average silhouette_score is : 0.09687971860732911
For n_clusters = 4 The average silhouette_score is : 0.1268004588393788
For n_clusters = 5 The average silhouette_score is : 0.14562442905455303
For n_clusters = 6 The average silhouette_score is : 0.1450621616291374
For n_clusters = 7 The average silhouette_score is : 0.14308266534595307
For n_clusters = 8 The average silhouette_score is : 0.15342109568144105
For n_clusters = 9 The average silhouette_score is : 0.1154018954074338
```

In practice, the scores obtained above can be considered equivalent since, depending on the run, scores of 0.1 ± 0.05 will be obtained for all clusters with $n_{clusters} > 3$ (we obtain slightly lower scores for the first cluster). On the other hand, we found that beyond 5 clusters, some clusters contained very few elements. we therefore choose to separate the dataset into 5 clusters. In order to ensure a good classification , we iterate until we obtain the best possible silhouette score, which is, in the present case, around 0.15:

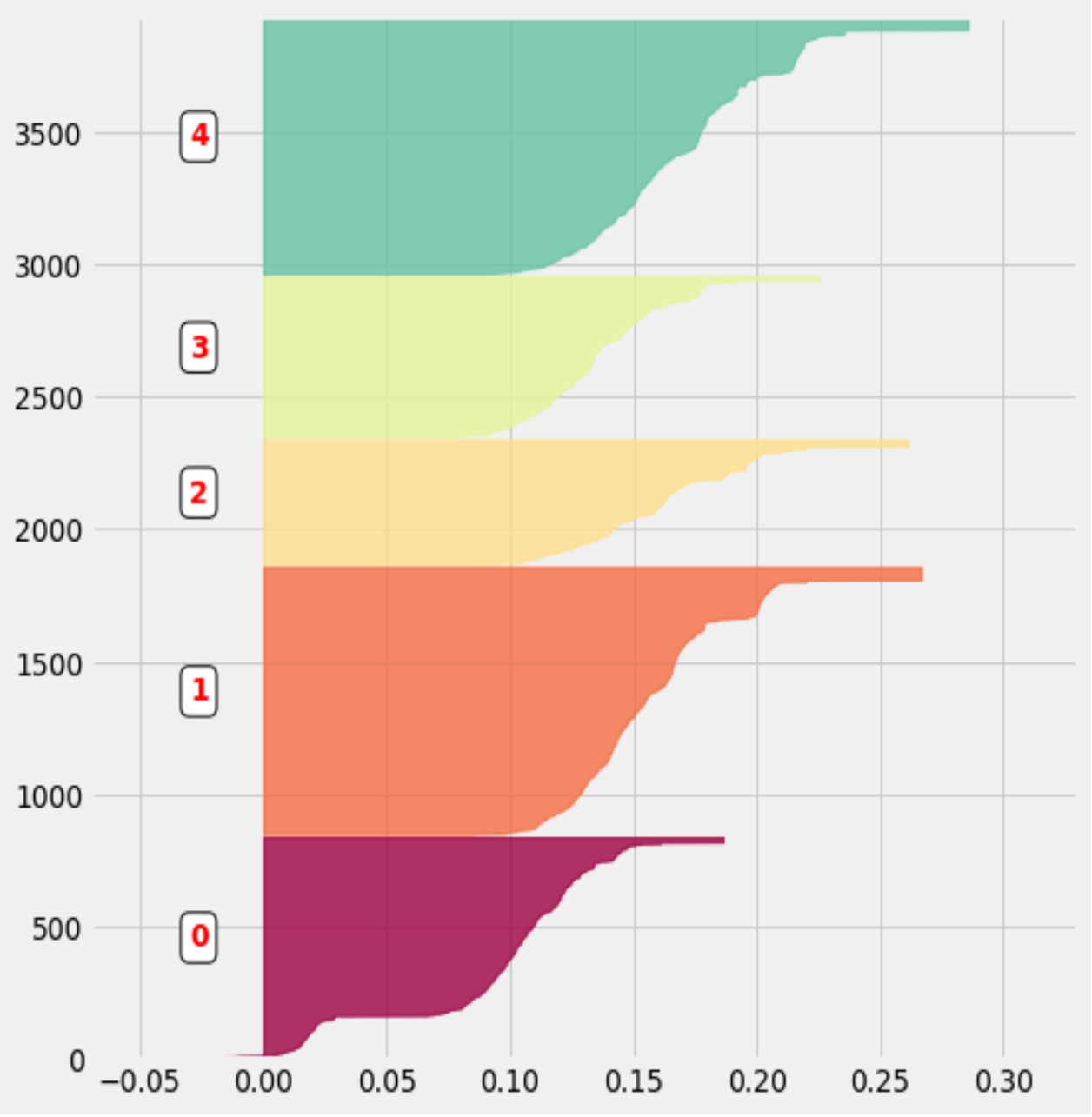
```
n_clusters = 5
silhouette_avg = -1
while silhouette_avg < 0.145:
    kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=30)
    kmeans.fit(matrix)
    clusters = kmeans.predict(matrix)
    silhouette_avg = silhouette_score(matrix, clusters)

    #km = kmodes.KModes(n_clusters = n_clusters, init='Huang', n_init=2, verbose=0)
    #clusters = km.fit_predict(matrix)
    #silhouette_avg = silhouette_score(matrix, clusters)
    print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)
```

```
For n_clusters = 5 The average silhouette_score is : 0.14679607861398997
```

Characterizing the content of clusters

- **Silhouette intra-cluster score**

  In order to have an insight on the quality of the classification, we can represent the silhouette scores of each element of the different clusters.



- **Word Cloud**

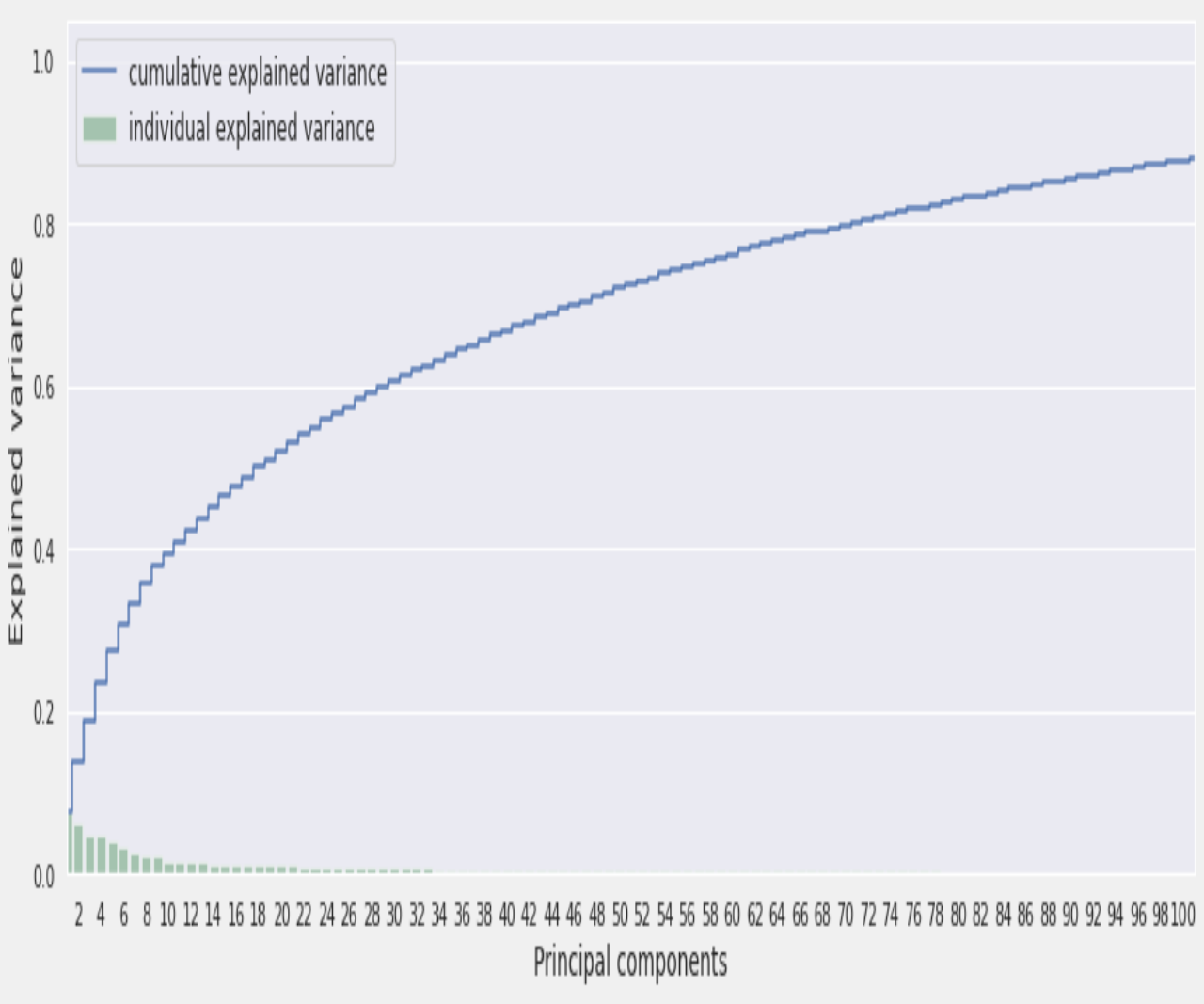
  Now we can have a look at the type of objects that each cluster represents. In order to obtain a global view of their contents, we determine which keywords are the most frequent in each of them and I output the result as wordclouds:

From this representation, we can see that for example, one of the clusters contains objects that could be associated with gifts (keywords: Christmas, packaging, card, ...). Another cluster would rather contain luxury items and jewelry (keywords: necklace,

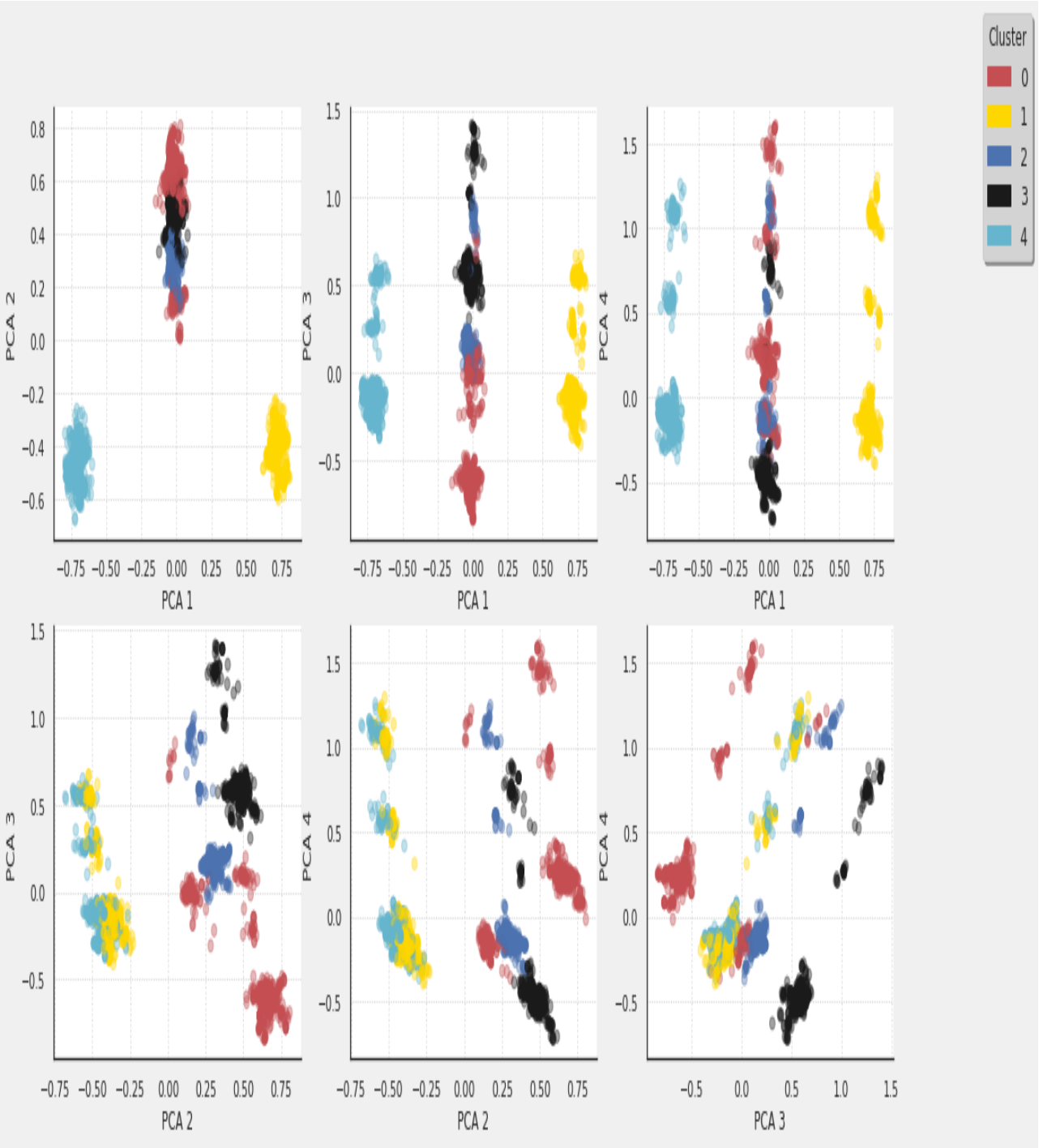
bracelet, lace, silver, ...). Nevertheless, it can also be observed that many words appear in various clusters and it is therefore difficult to clearly distinguish them.

- **Principal Component Analysis**

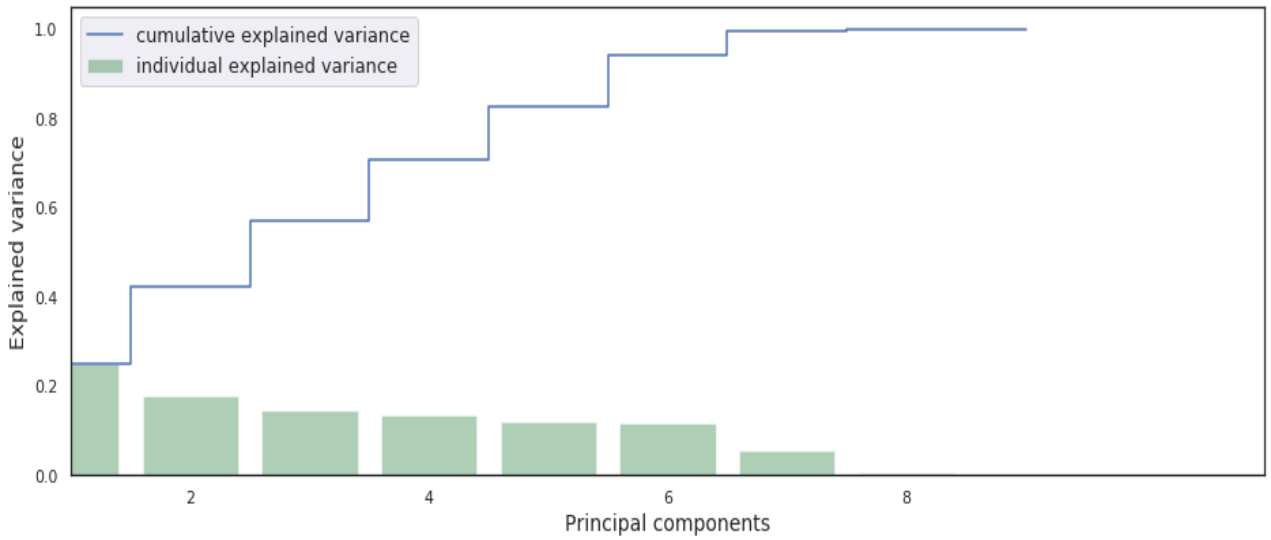
  In order to ensure that these clusters are truly distinct, we look at their composition. Given the large number of variables of the initial matrix, we first perform a PCA, and then check for the amount of variance explained by each component:



Visualizing the Decomposed data

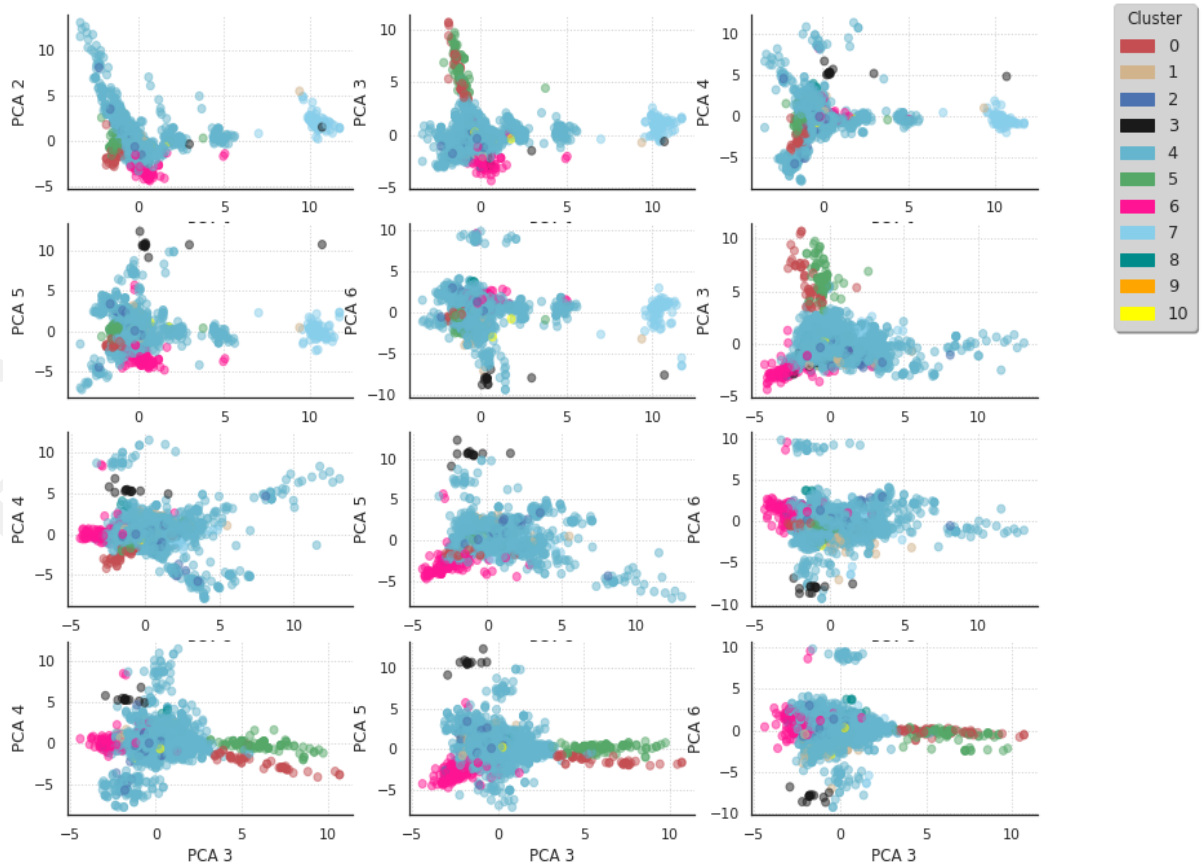


Data Encoding

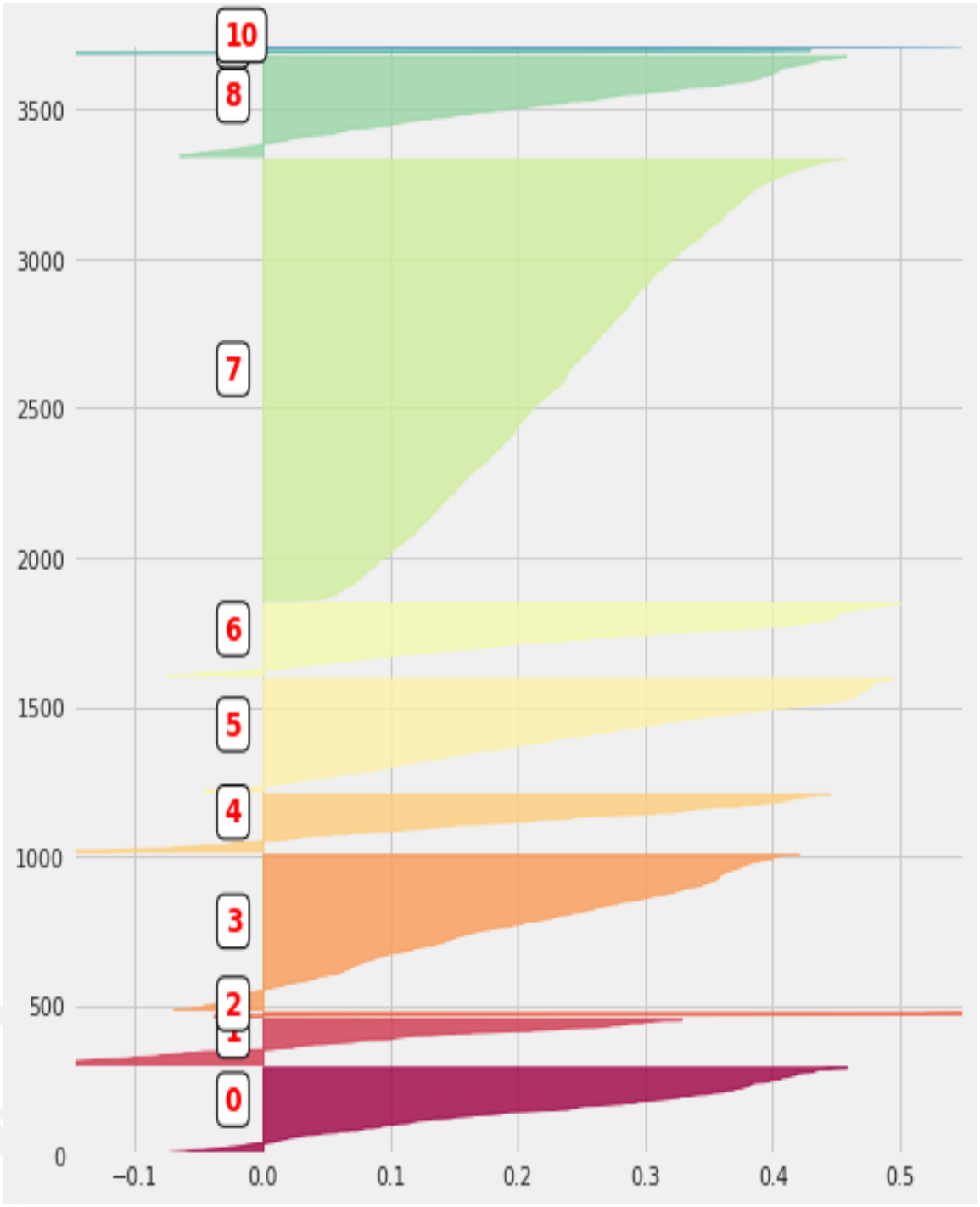


Creation of Customer Categories

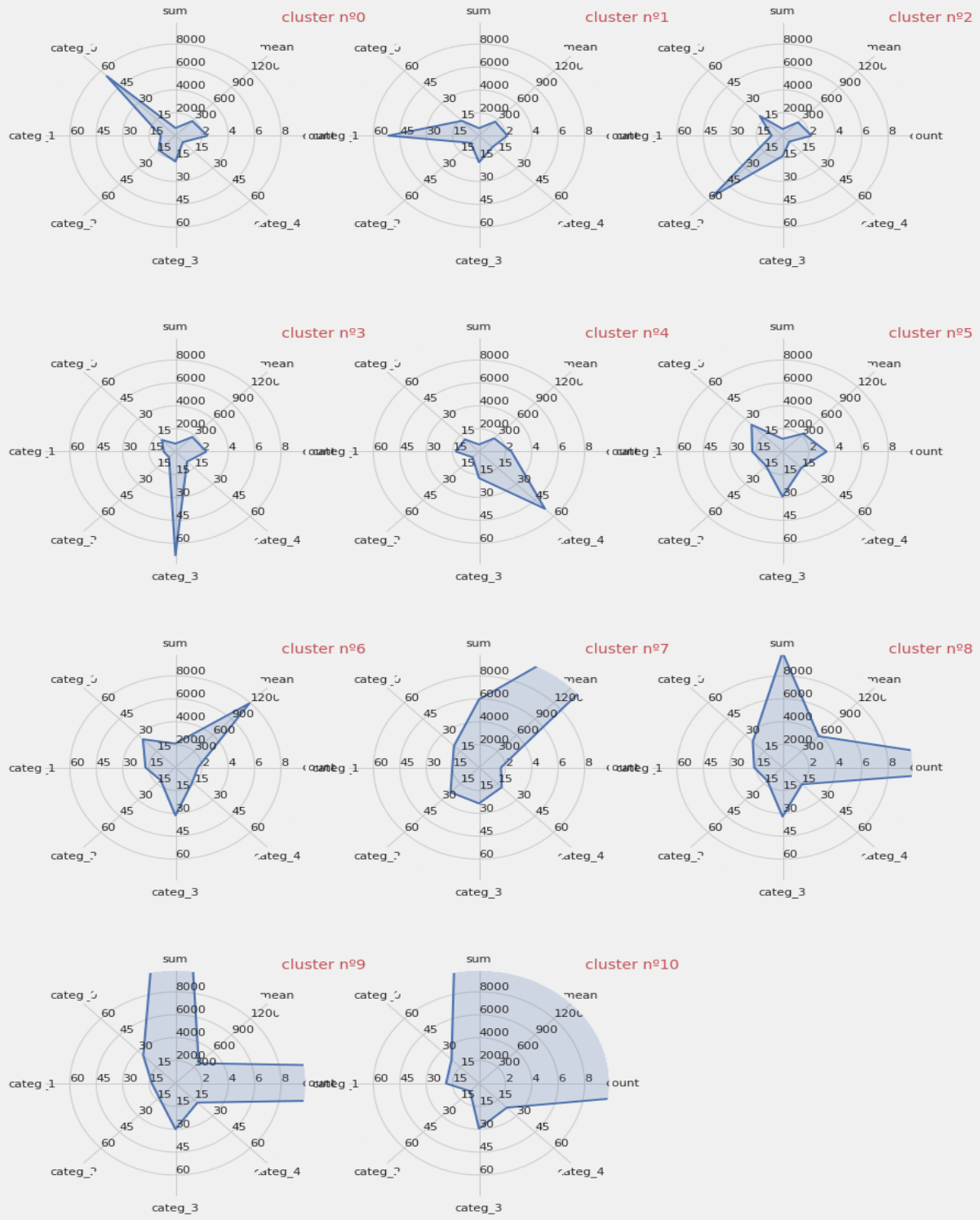
- **Report via PCA**



- **Score of silhouette intra-cluster**



- **Customers morphology**



Classification of Customers

Support Vector Machine Classifier (SVC)

The first classifier we use is the SVC classifier. In order to use it, we create an instance of the $\text{Class}_{\text{Fit}}$ class and then call$\text{grid}_{\text{search}}()$. When calling this method, we provide as parameters:

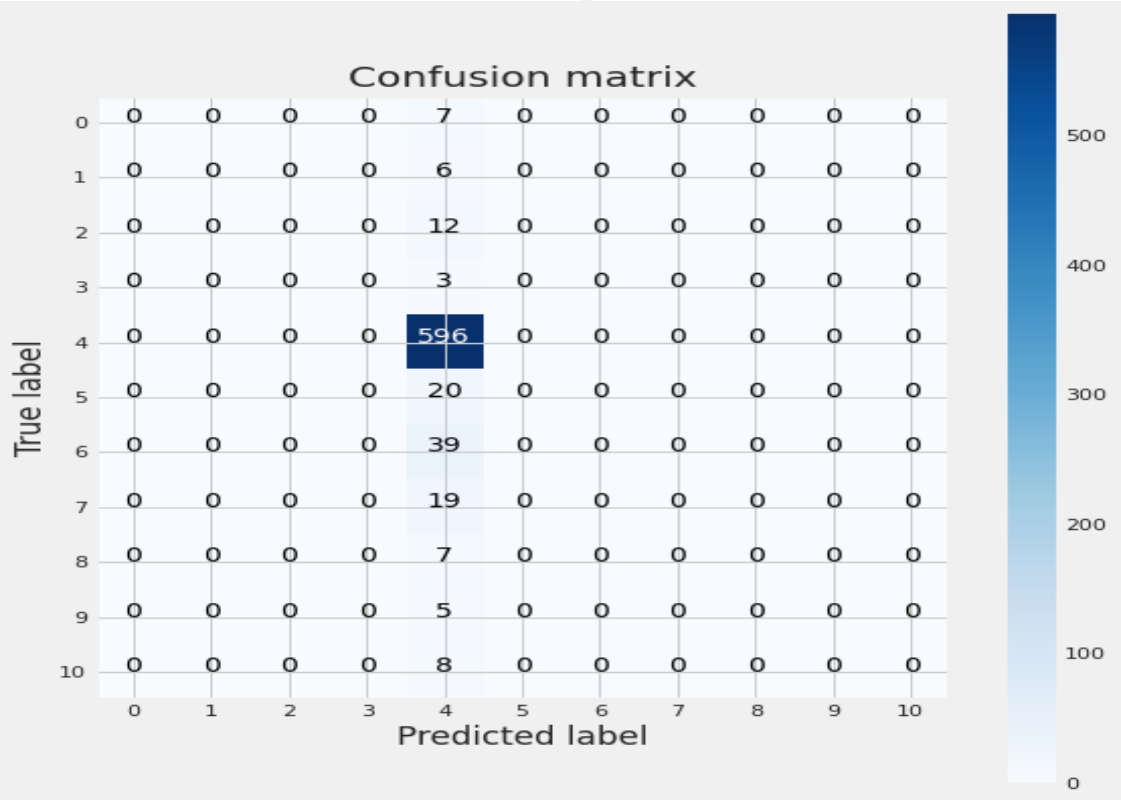
- The hyperparameters for which I will seek an optimal value
- The number of folds to be used for cross-validation

Support Vector Machine Classifier (SVC)

```
svc = Class_Fit(clf = svm.LinearSVC)
svc.grid_search(parameters = [{'C':np.logspace(-2,2,10)}], Kfold = 5)
svc.grid_fit(X = X_train, Y = Y_train)
svc.grid_predict(X_test, Y_test)
```

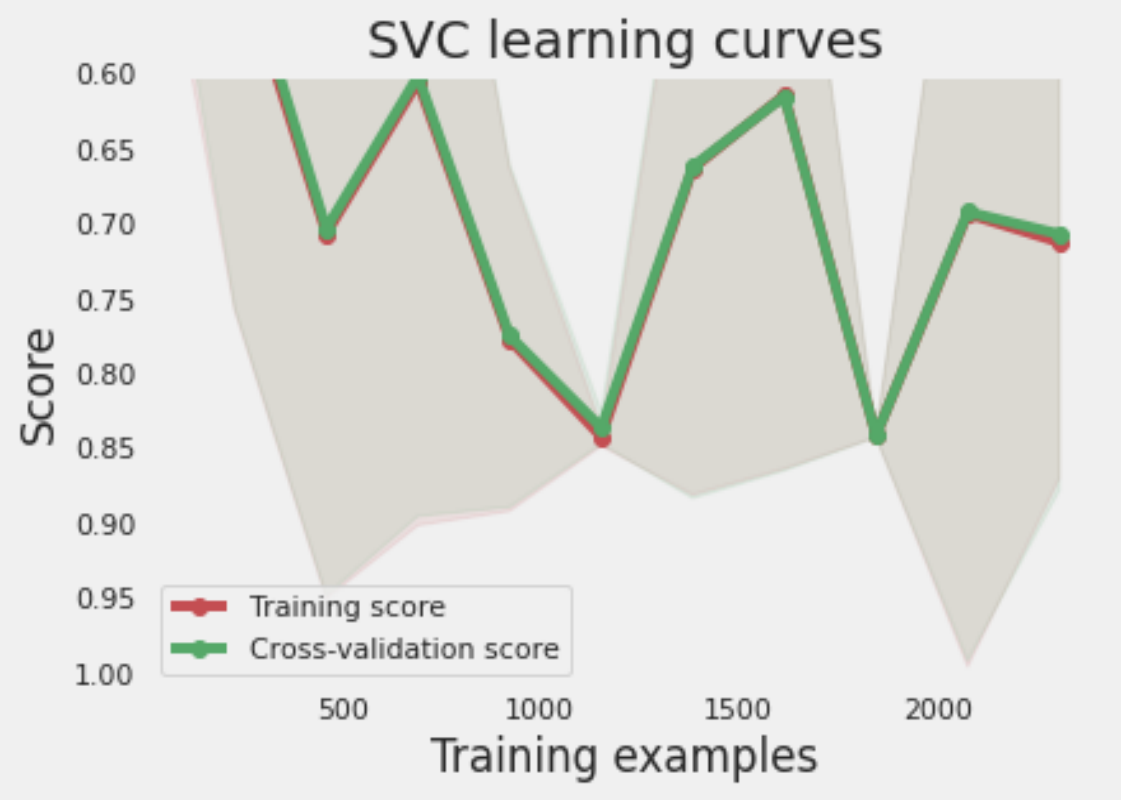
```
Precision: 82.55 %
```

Confusion Matrix



SVC Learning Curve

A typical way to test the quality of a fit is to draw a learning curve. In particular, this type of curves allow to detect possible drawbacks in the model, linked for example to over- or under-fitting. This also shows to which extent the mode could benefit from a larger data sample.



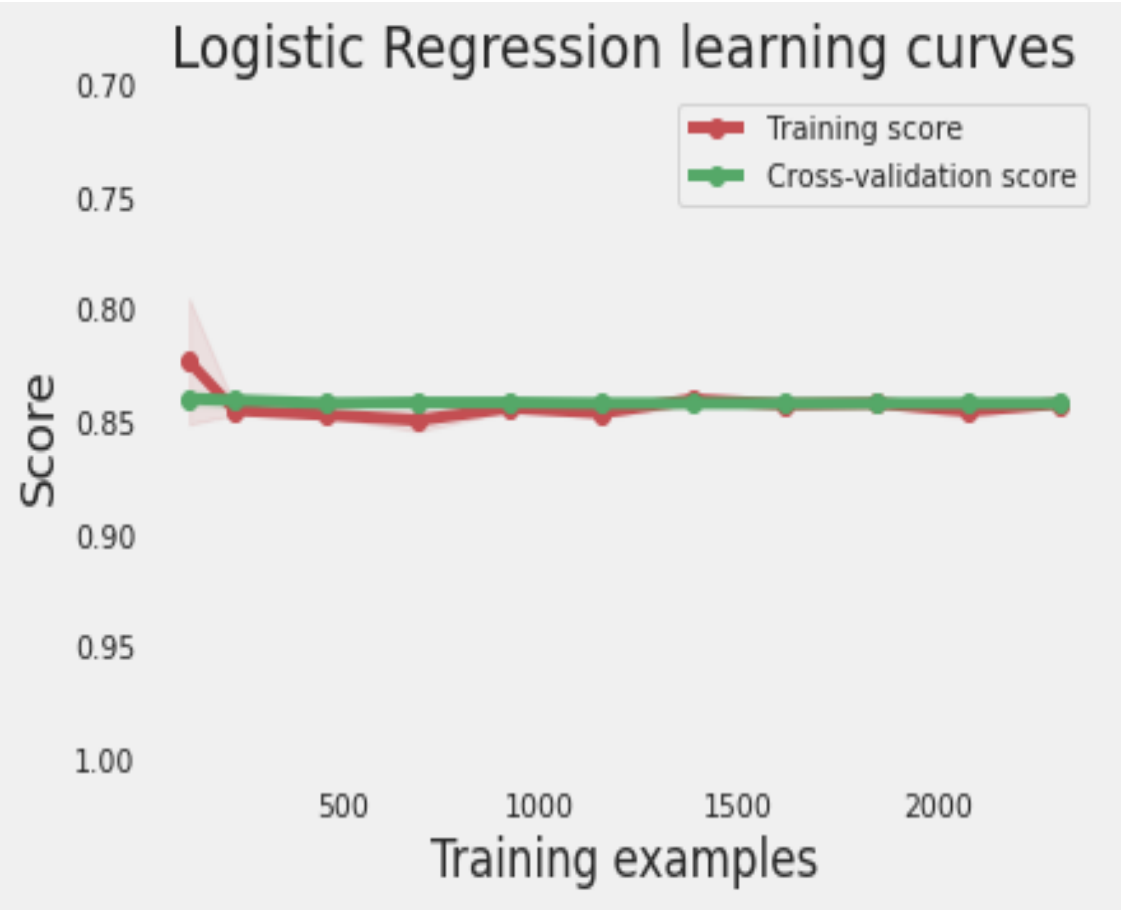
Logistic Regression

Percentage of Accuracy

```
lr = Class_Fit(clf = linear_model.LogisticRegression)
lr.grid_search(parameters = [{'C':np.logspace(-2,2,20)}], Kfold = 5)
lr.grid_fit(X = X_train, Y = Y_train)
lr.grid_predict(X_test, Y_test)
```

```
Precision: 82.55 %
```

Logistic Regression Learning Curve



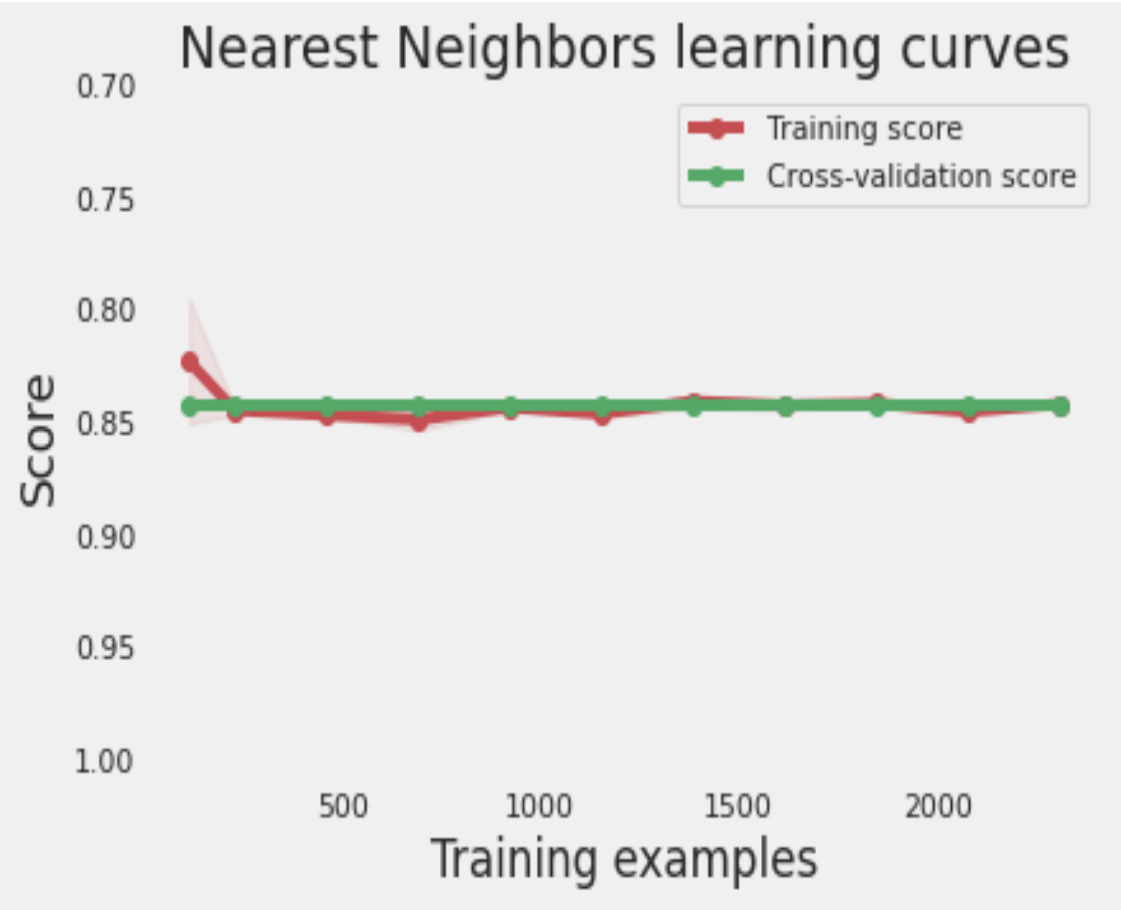
K-Nearest Neighbors

Percentage of Accuracy

```
knn = Class_Fit(clf = neighbors.KNeighborsClassifier)
knn.grid_search(parameters = [{'n_neighbors': np.arange(1,50,1)}], Kfold = 5)
knn.grid_fit(X = X_train, Y = Y_train)
knn.grid_predict(X_test, Y_test)
```

```
Precision: 82.55 %
```

Nearest Neighbors Learning Curve



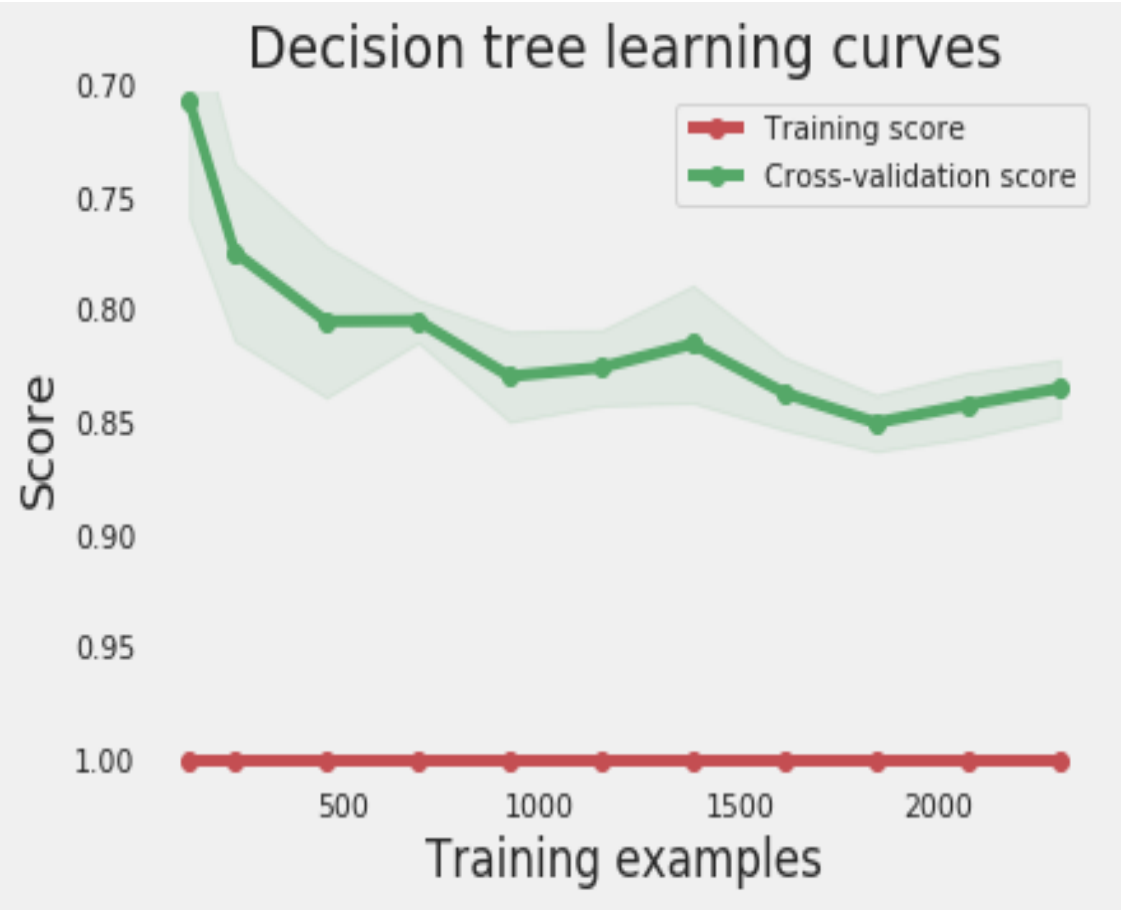
Decision Tree

Percentage of Accuracy

```
tr = Class_Fit(clf = tree.DecisionTreeClassifier)
tr.grid_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max_features' :['sqrt', 'log2']}], Kfold = 5)
tr.grid_fit(X = X_train, Y = Y_train)
tr.grid_predict(X_test, Y_test)
```

```
Precision: 70.50 %
```

Decision Tree Learning Curve



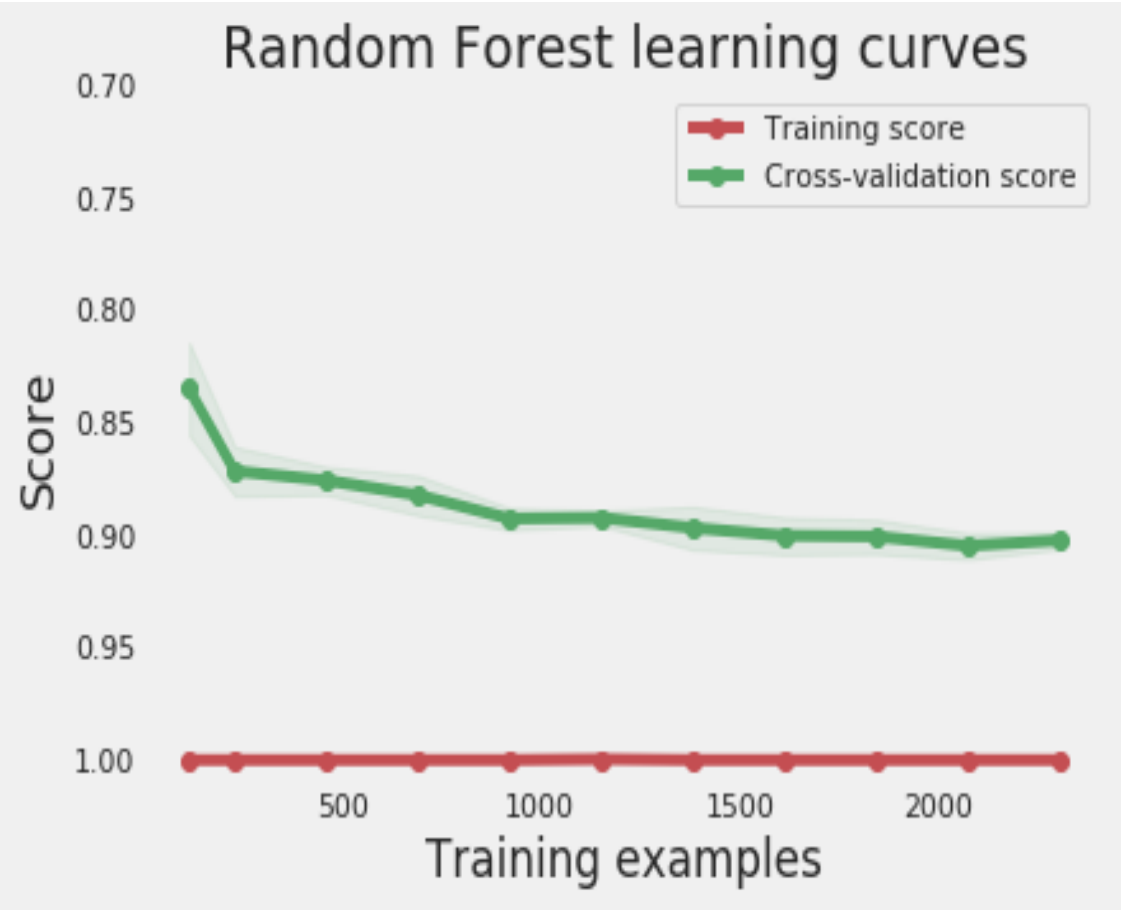
Random Forest

Percentage of Accuracy

```
rf = Class_Fit(clf = ensemble.RandomForestClassifier)
param_grid = {'criterion' : ['entropy', 'gini'], 'n_estimators' : [20, 40, 60, 80, 100],
              'max_features' :['sqrt', 'log2']}
rf.grid_search(parameters = param_grid, Kfold = 5)
rf.grid_fit(X = X_train, Y = Y_train)
rf.grid_predict(X_test, Y_test)
```

```
Precision: 81.99 %
```

Random Forest Learning Curve



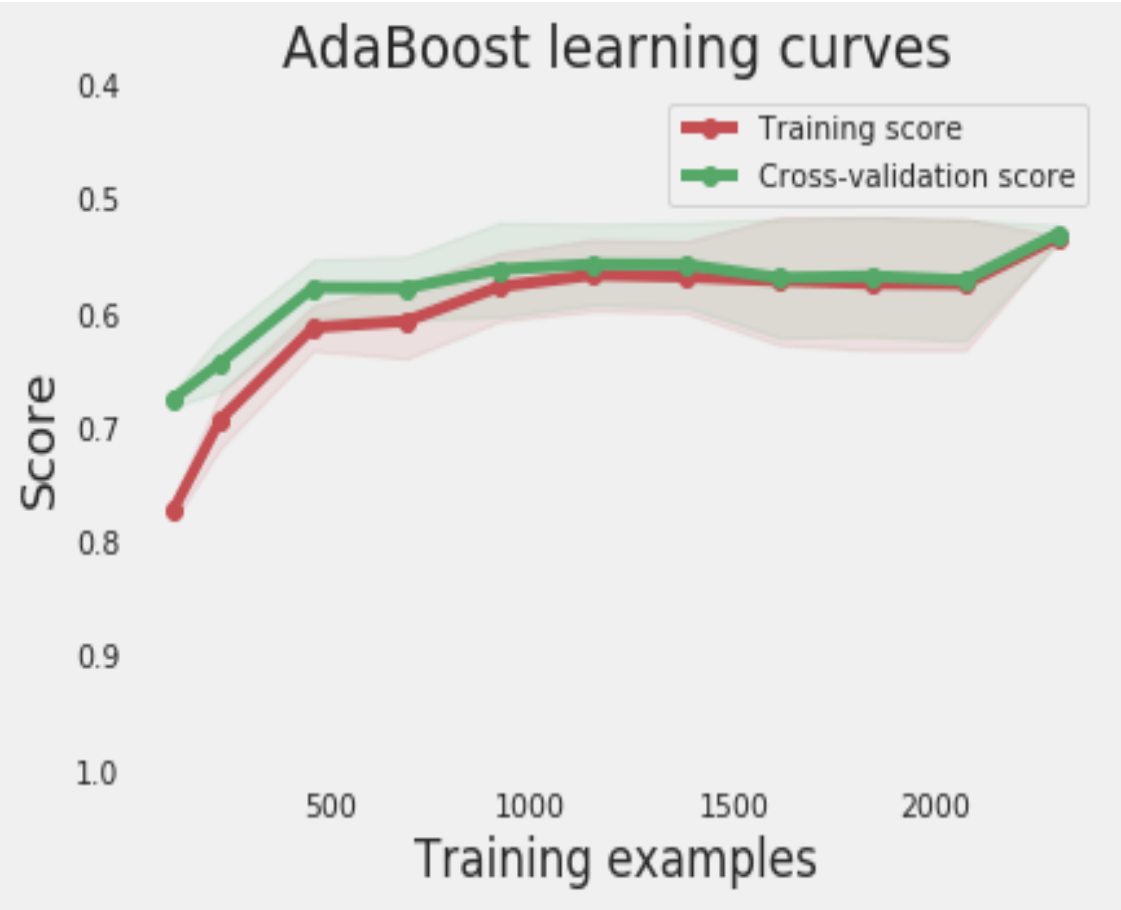
AdaBoost Classifier

Percentage of Accuracy

```
ada = Class_Fit(clf = AdaBoostClassifier)
param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
ada.grid_search(parameters = param_grid, Kfold = 5)
ada.grid_fit(X = X_train, Y = Y_train)
ada.grid_predict(X_test, Y_test)
```

```
Precision: 55.68 %
```

AdaBoost Classifier Learning Curve



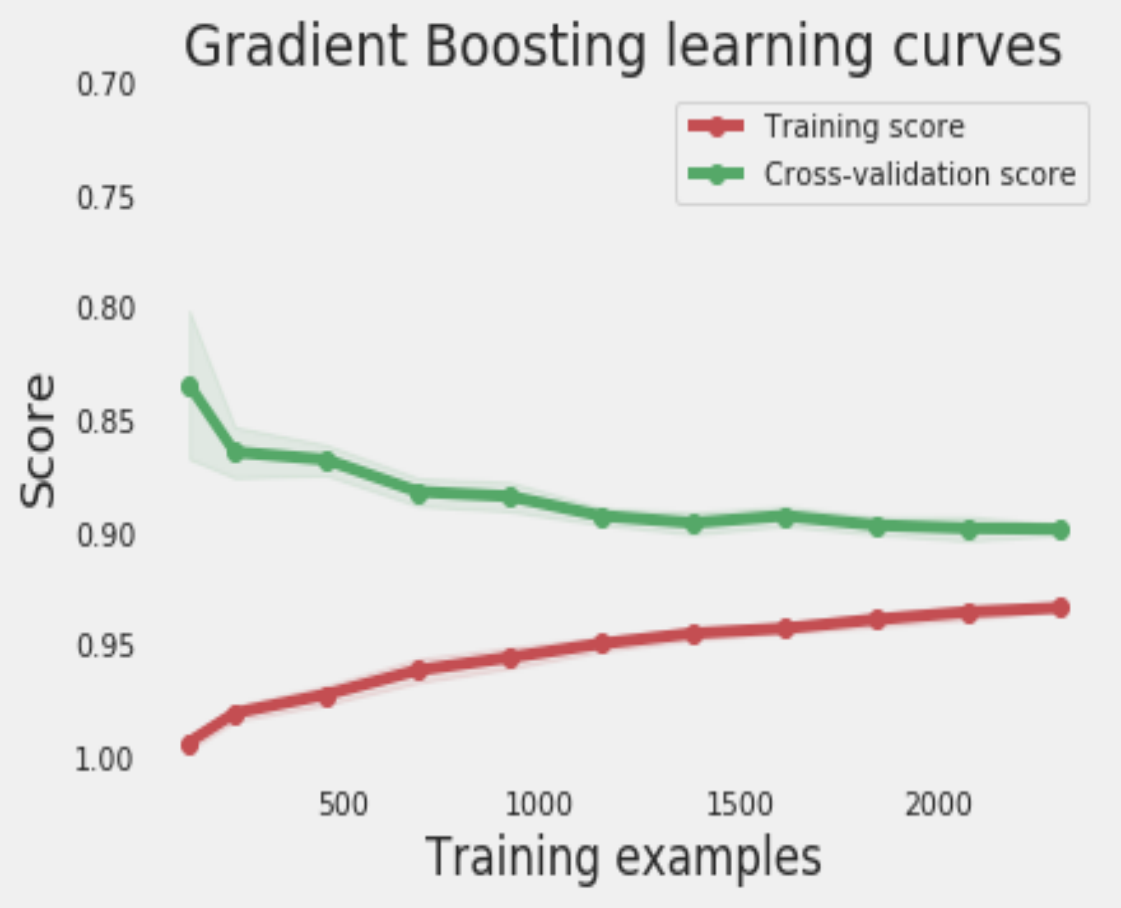
Gradient Boosting Classifier

Percentage of Accuracy

```
gb = Class_Fit(clf = ensemble.GradientBoostingClassifier)
param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
gb.grid_search(parameters = param_grid, Kfold = 5)
gb.grid_fit(X = X_train, Y = Y_train)
gb.grid_predict(X_test, Y_test)
```

```
Precision: 79.64 %
```

Gradient Boosting Classifier Learning Curve



For all these, the classifier is based on 5 variables which are:

- **Mean:** amount of the basket of the current purchase
- **Categ$_N$:** with N[0:4] : percentage spent in product category with index N

Finally, the quality of the predictions of the different classifiers was tested over the last two months of the dataset. The data were then processed in two steps: first, all the data was considered (over the 2 months) to define the category to which each client belongs, and then, the classifier predictions were compared with this category assignment. I then found that 75% of clients are awarded the right classes. The performance of the classifier therefore seems correct given the potential shortcomings of the current model. In particular, a bias that has not been dealt with concerns the seasonality of purchases and the fact that purchasing habits will potentially depend on the time of year (for example, Christmas). In practice, this seasonal effect may cause the categories defined over a 10-month period to be quite different from those extrapolated from the last two months. In order to correct such bias, it would be beneficial to have data that would cover a longer period of time.

Final Product Prototype Details

The final product provides service to operators about the most bought combinations of products for them to analyze customer shopping patterns and helps them manage their inventory and also create new strategies and schemes to increase their sales.The service implements the Customer Segmentation Analysis, i.e Association Rule Mining technique on the dataset of transactions collected from E-Commerce Industries.

Some dynamics of the Apriori Algorithm used in this model and their meaning.

1. **Support:** It tells us about the combination of items bought together frequently. It gives the part of transactions that contain both A and B.

$$Support = \frac{freq(A, B)}{N}$$

2. **Confidence:** It tells us how frequently the items A and B are bought together, for the no. of times A is bought.

$$Confidence = \frac{freq(A, B)}{freq(A)}$$

3. **Lift:** It indicates the strength of a rule over the randomness of A and B being bought together. It basically measures the strength of any association rule.

$$Lift = \frac{Support)}{Supp(A)xSupp(B)}$$

1. Feasibility

This project can be developed and deployed within a few years as SaaS(Software as a Service) for anyone to use.

2. Viability

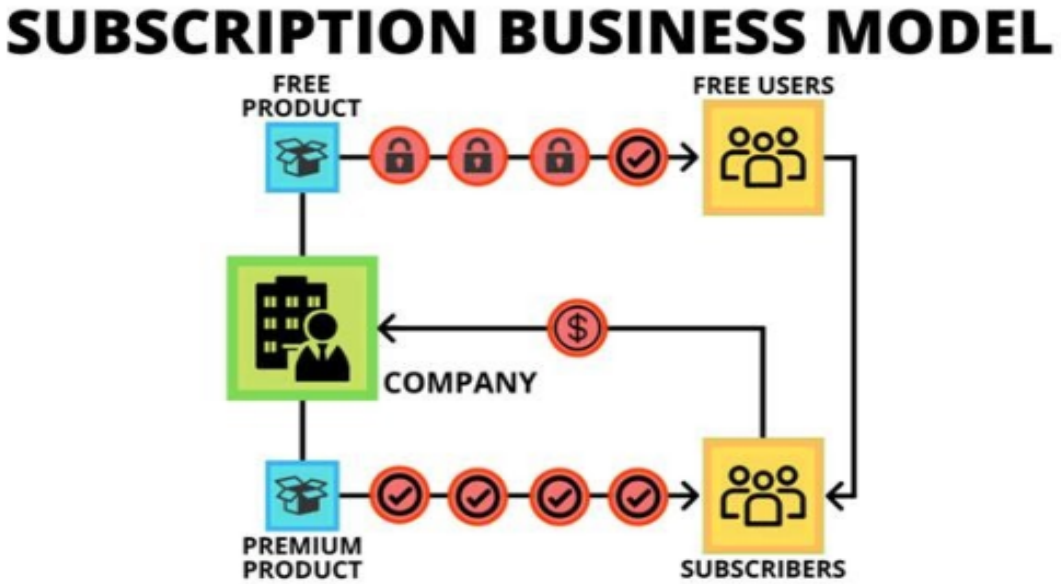
As the retail industry grows in India and the world, there will always be small businesses existing which can use this service to improvise on their sales and data warehousing techniques. So, it is viable to survive in the long-term future as well but improvements are necessary as new technologies emerge.

3. Monetization

This service is directly monetizable as it can be directly released as a service on completion which can be used by businesses.

Business Modeling

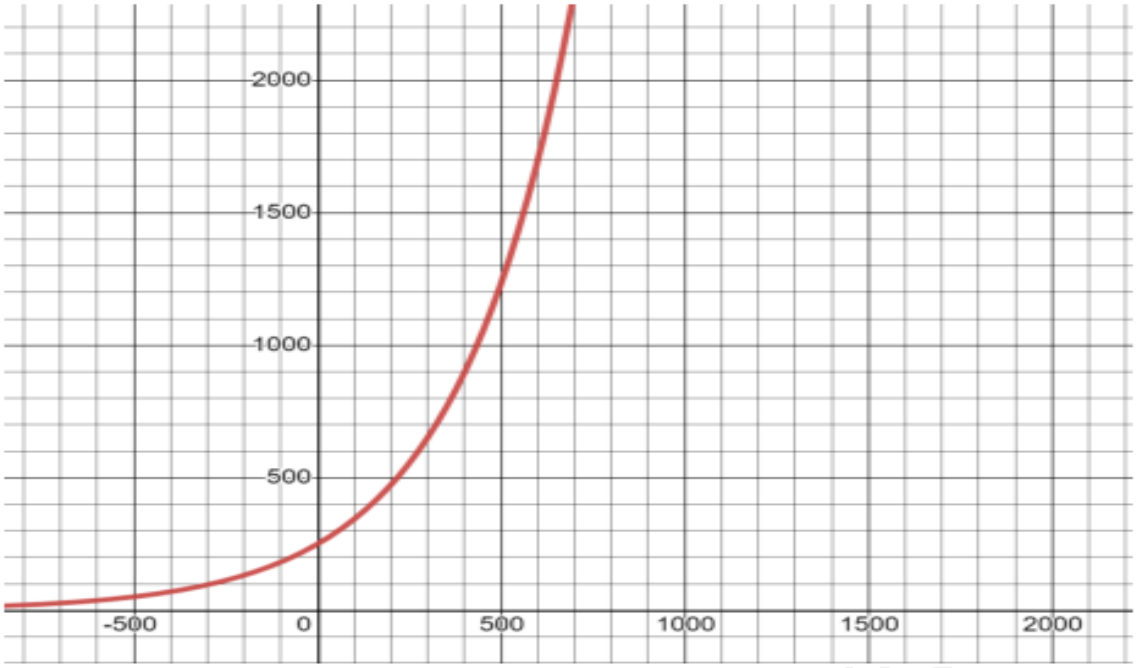
For this service, it is beneficial to use a Subscription Based Model, where initially some features will be provided for free to engage customer retention and increase our customer count. Later it will be charged a subscription fee to use the service further for their business.In the subscription business model, customers pay a fixed amount of money on fixed time intervals to get access to the product or service provided by the company. The major problem is user conversion; how to convert the users into paid users.



Financial Modeling

It can be directly launched into the retail market.

Let's consider our price of product = 250 for getting our graph



Financial Equation

$$Y = X^*(1+r)^t$$

$$Y = (X)^*(3.2)^t$$

Y = Profit over Time

X = Price of our Product

r = Growth Rate

t = Time Interval

1+r = 1 + 3.2% = 1.032

Conclusion

Customer Segmentation analysis is being used by an increasing number of companies to acquire beneficial insights about associations and hidden relationships. However, for small businesses, this extension is a fantastic opportunity to boost sales and help them develop and grow their business.

References

[1] Homeyer A, Lotz J, Schwen LO, Weiss N, Romberg D, Höfener H, et al. *Artificial intelligence in pathology: From prototype to product*, J Pathol Inform 2021;12:13.

[2] Dolnicar, S., Grun Bettina, amp; Leisch, F. (2019). *Market segmentation analysis understanding it, doing it and making it useful*. Springer Nature.

[3] McDonald, M., amp; Dunbar, I. (2003). *Market segmentation*. Butterworth-Heinemann.

[4] Qualtrics AU. 2022. *Customer Segmentation: Definition Methods*. [online] Available at: Customer Segmentation by Qualtrics Experience Management.

Github : Chaganti Reddy/AI-Prototype