Customer Segmentation Project

Group Information

- Group Name: M.A.S
- Specialization: Data Science
- Submitted to: Data Glacier canvas platform
- Internship Batch: LISUM10: 30

Group members

Moath Mohammed Bin Musallam | [moathmusallam@gmail.com (mailto:moathmusallam@gmail.com)](mailto:moathmusallam@gmail.com) | Saudi Arabia | University of East Anglia

Andrew Kojo Mensah-Onumah | [kojoakmo@gmail.com (mailto:kojoakmo@gmail.com)](mailto:kojoakmo@gmail.com) |Ghana | Data Glacier

Shaimaa Saleh Obad Al-khawlani | [s.khawlany@gmail.com (mailto:s.khawlany@gmail.com)](mailto:s.khawlany@gmail.com) | Yemen | Data Glacie

In [ ]:

```
1
```

In this project, the KMeans algorithm will be applied to develop insights about different clusters that can be formed from analysing the data.

In [35]:

```
#Importing the Libraries
!pip install yellowbrick
import numpy as np
import pandas as pd
import datetime
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import AgglomerativeClustering
from matplotlib.colors import ListedColormap
from sklearn import metrics
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)
```

```
Requirement already satisfied: yellowbrick in c:\users\kojoa\anaconda3\lib\s
ite-packages (1.3.post1)
Requirement already satisfied: cycler>=0.10.0 in c:\users\kojoa\anaconda3\li
b\site-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: scikit-learn>=0.20 in c:\users\kojoa\anaconda
3\lib\site-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\kojoa\anaconda3\lib
\site-packages (from yellowbrick) (1.8.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\kojoa\a
naconda3\lib\site-packages (from yellowbrick) (3.5.1)
Requirement already satisfied: numpy<1.20,>=1.16.0 in c:\users\kojoa\anacond
a3\lib\site-packages (from yellowbrick) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kojoa\anaconda3
\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\kojoa\anaconda3\l
ib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kojoa\anacon
da3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kojoa\anaconda3
\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\kojoa\anaconda3
\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kojoa\anaconda3\lib
\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\kojoa\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbric
k) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kojoa\anacon
da3\lib\site-packages (from scikit-learn>=0.20->yellowbrick) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\kojoa\anaconda3\lib
\site-packages (from scikit-learn>=0.20->yellowbrick) (1.1.0)
```

In [36]:

```python
# Set view option to view all the columns
pd.set_option('display.max_columns', None)
data = pd.read_csv('finaldata')
```

In [37]:

```python
# Replace column with value '999999' with mode
data['Customer seniority (in months)'].mode()
data.replace([-999999.0, 21], inplace = True)
```

This should have appeared in the data cleaning and transformation section but was missed.

Feature Engineering

Inspecting the dataframe, columns starting from Saving Account down to Direct debit can be termed as products or offers offered by the bank to its clients or customers.

In [38]:

```python
# Create column that totals the total number of bank products used by a customer
data['Total products'] = data.iloc[:, 20:45].sum(axis= 1)
data
```

Out[38]:

	Unnamed: 0	Customer code	Employee index	Customer's Country residence	Customer's sex	Age	Date of first contract(account was created)	cust I
0	0	1375586	N	ES	M	35.0	2015-01-12	
1	1	1050611	N	ES	F	23.0	2012-08-10	
2	2	1050612	N	ES	F	23.0	2012-08-10	
3	3	1050613	N	ES	M	22.0	2012-08-10	
4	4	1050614	N	ES	F	23.0	2012-08-10	
...	...	...	...	...	...	...	...	
999995	999995	1183296	N	ES	M	27.0	2013-09-25	
999996	999996	1183295	N	ES	M	56.0	2013-09-25	
999997	999997	1183294	N	ES	F	39.0	2013-09-25	
999998	999998	1183293	N	ES	F	36.0	2013-09-25	
999999	999999	1183289	N	ES	M	38.0	2013-09-25	

1000000 rows × 45 columns

In [39]:

```
# Print columns in dataframe
data.columns
```

Out[39]:

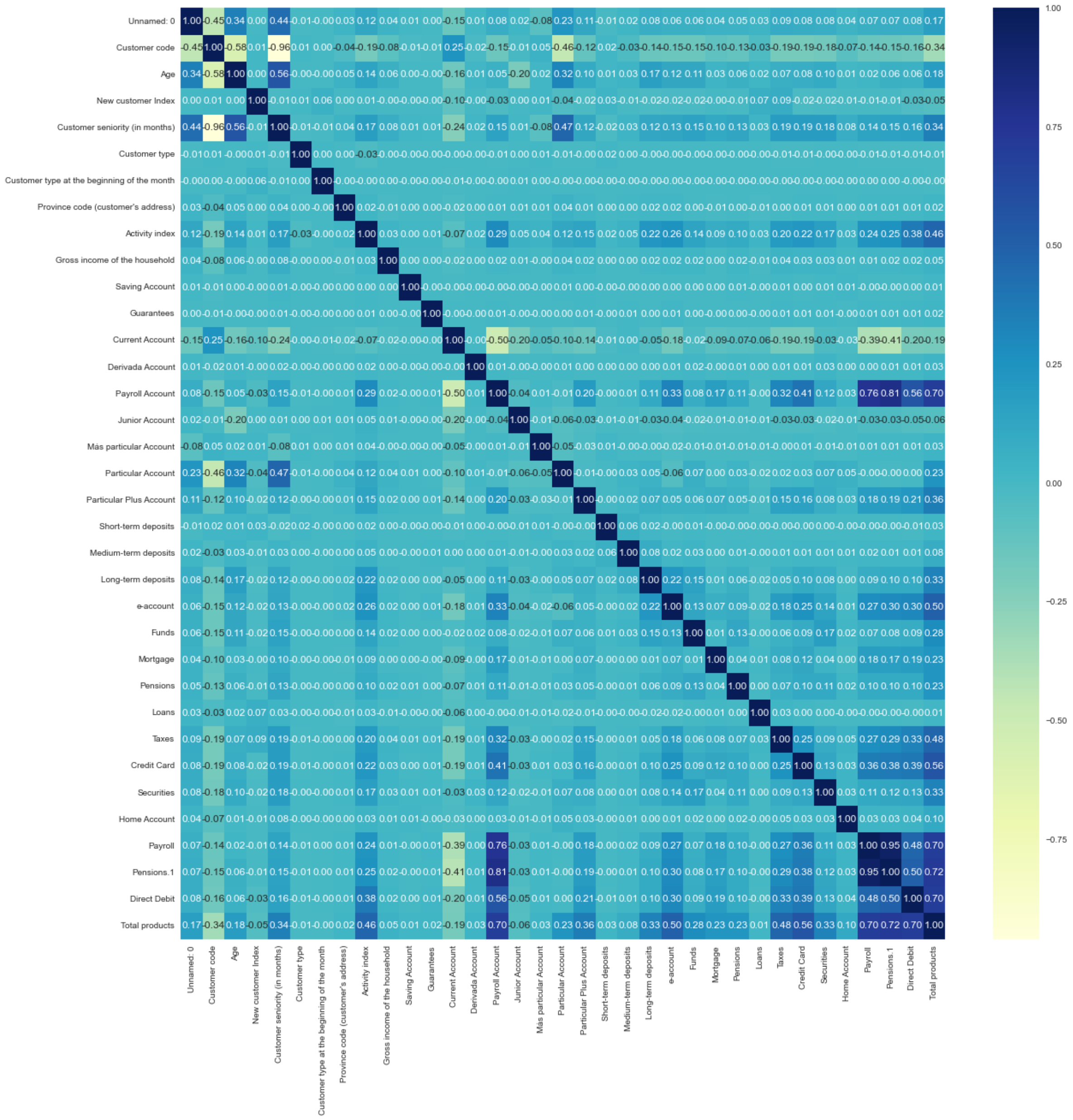
```
Index(['Unnamed: 0', 'Customer code', 'Employee index',
       'Customer's Country residence', 'Customer's sex', 'Age',
       'Date of first contract(account was created)', 'New customer Index',
       'Customer seniority (in months)', 'Customer type',
       'Customer type at the beginning of the month',
       'Customer relation type at the beginning of the month',
       'Residence index', 'Foreigner index',
       'Channel used by the customer to join', 'Deceased index',
       'Province code (customer's address)', 'Province name', 'Activity inde
x',
       'Gross income of the household', 'Saving Account', 'Guarantees',
       'Current Account', 'Derivada Account', 'Payroll Account',
       'Junior Account', 'Más particular Account', 'Particular Account',
       'Particular Plus Account', 'Short-term deposits',
       'Medium-term deposits', 'Long-term deposits', 'e-account', 'Funds',
       'Mortgage', 'Pensions', 'Loans', 'Taxes', 'Credit Card', 'Securitie
s',
       'Home Account', 'Payroll', 'Pensions.1', 'Direct Debit',
       'Total products'],
      dtype='object')
```

In [40]:

```
# Checking correlation of columns
matrix = data.corr()
plt.figure(figsize = (20,20))
sns.heatmap(matrix, cmap = 'YlGnBu', annot = True, fmt = '.2f')
```

Out[40]:

<AxesSubplot:>



PREPROCESSING

In [41]:

```
# Selecting dominant column values
ds= data[data['Customer\'s Country residence']=='ES']
ds= data[data['Customer type']==1]
ds= data[data['Customer relation type at the beginning of the month']!='P']
ds= data[data['Customer type at the beginning of the month']==1]
ds= data[data['New customer Index']==0]
ds= data[data['Residence index']=='Y']
```

Data exploration indicated that some features have dominant occurrence across respective columns. Analysis should be geared towards features with marginal imapact, hence, selection of these dominant categories for analysis.

In [42]:

```
# Let's subset the data to keep only the records from six major channels
subset = [ "KAT", "KFC", "KHE", "KFA", "KAS", "KAG"]
ds = data.loc[data['Channel used by the customer to join'].isin(subset)]
```

Label Encoding

In [43]:

```
# Get list of categorical variables
s = (ds.dtypes == 'object')
object_cols = list(s[s].index)
```

In [44]:

```
# Label Encoding the object data types
LE=LabelEncoder()
for i in object_cols:
    ds[i]=ds[[i]].apply(LE.fit_transform)

print("All features are now numerical")
```

```
All features are now numerical
```

Feature Scaling

In [45]:

```
# Creating a copy of data
ds = ds.copy()

# Dropping unwanted features
cols_del = ['Date of first contract(account was created)', 'Customer code', 'Unnamed: 0
ds = ds.drop(cols_del, axis=1)

# Scaling
scaler = preprocessing.MinMaxScaler()
scaled_frame= scaler.fit_transform(ds)
scaled_df = pd.DataFrame(scaled_frame, columns=ds.columns)
print("All features are now scaled")
scaled_df
```

```
All features are now scaled
```

Out[45]:

	Employee index	Customer's Country residence	Customer's sex	Age	New customer Index	Customer seniority (in months)	Customer type	Custom type t beginni of t mon
0	0.75	0.33	0.0	0.184211	0.0	0.142276	0.0	0
1	0.75	0.33	0.0	0.184211	0.0	0.142276	0.0	0
2	0.75	0.33	0.0	0.184211	0.0	0.142276	0.0	0
3	0.75	0.33	1.0	0.184211	0.0	0.142276	0.0	0
4	0.75	0.33	1.0	0.184211	0.0	0.142276	0.0	0
...	...	...	...	...	...	...	...	
884973	0.75	0.33	1.0	0.219298	0.0	0.089431	0.0	0
884974	0.75	0.33	1.0	0.473684	0.0	0.089431	0.0	0
884975	0.75	0.33	0.0	0.324561	0.0	0.089431	0.0	0
884976	0.75	0.33	0.0	0.298246	0.0	0.089431	0.0	0
884977	0.75	0.33	1.0	0.315789	0.0	0.089431	0.0	0

884978 rows × 41 columns

The columns below helped guide decisions that were made.

In [75]:

```
data['Guarantees'].value_counts()
```

Out[75]:

```
0    999961
1        39
Name: Guarantees, dtype: int64
```

In [76]:

```
scaled_df['Guarantees'].value_counts()
```

Out[76]:

```
0.0    884945
1.0        33
Name: Guarantees, dtype: int64
```

Dimensionality reduction

In this problem, there are many factors on the basis of which the final classification will be done. These factors are basically attributes or features. The higher the number of features, the harder it is to work with it. Many of these features are correlated, and hence redundant. This is why performing dimensionality reduction will be performed on the selected features before putting them through a classifier.
Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables.

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss.

In [48]:

```
# Applying PCA
pca = PCA()
df_pca = pd.DataFrame(pca.fit_transform(scaled_df))
df_pca
```

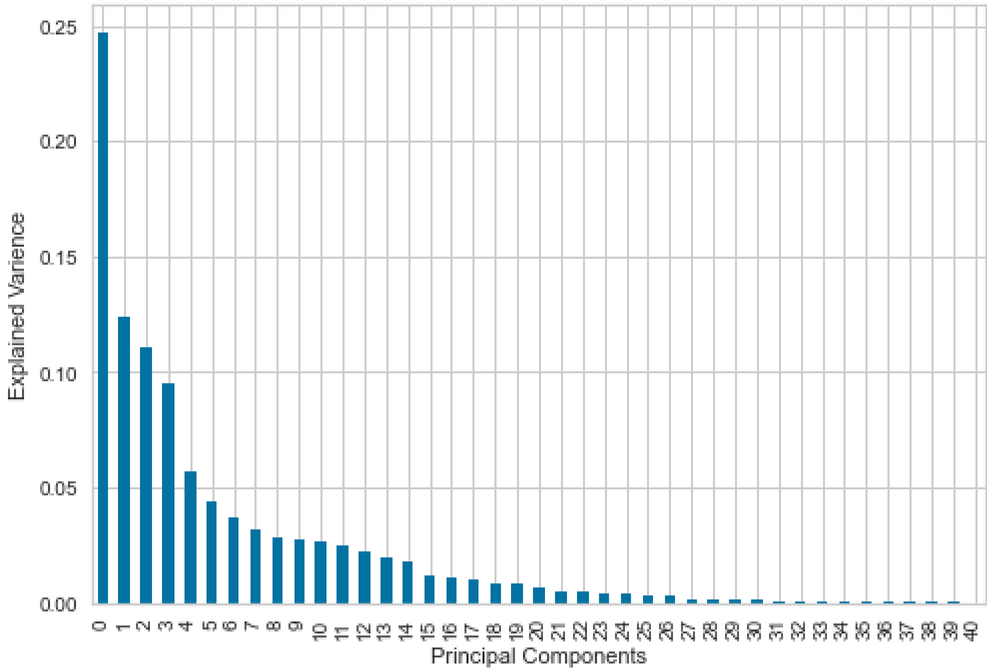
Out[48]:

	0	1	2	3	4	5	6	
0	-0.712855	-0.257173	0.410866	-0.416723	0.136229	-0.121720	0.075445	-0.03877
1	-0.706083	-0.267758	0.405402	-0.408695	0.146397	-0.111334	0.135252	-0.02231
2	-0.080420	-0.379649	-0.380180	-0.538693	-0.262172	-0.295965	0.154506	-0.01225
3	-0.815085	0.593963	0.089616	-0.029966	0.171408	-0.026739	0.084467	-0.02369
4	-0.820375	0.595844	0.093138	-0.038711	0.172298	-0.037885	0.086291	-0.01597
...	...	...	...	...	...	...	...	.
884973	0.502435	0.720096	-0.156811	-0.052363	-0.447939	-0.294095	-0.349484	-0.76884
884974	-0.165388	0.452036	-0.689411	-0.122724	-0.239226	-0.150697	0.031233	0.06629
884975	0.955279	0.022548	0.387933	-0.483552	-0.376240	-0.504188	-0.152122	-0.61135
884976	0.313500	-0.269193	-0.350001	-0.695217	0.173520	-0.313969	-0.281548	-0.54520
884977	0.199984	0.584977	-0.663145	-0.326545	0.211066	-0.242308	-0.269788	-0.51047

884978 rows × 41 columns

In [49]:

```
# Viewing variance of principal components
pd.DataFrame(pca.explained_variance_ratio_).plot.bar()
plt.legend('')
plt.xlabel('Principal Components')
plt.ylabel('Explained Varience');
```



Majority of information is captured by components 3 (first 3 or 4, clarify)

In [50]:

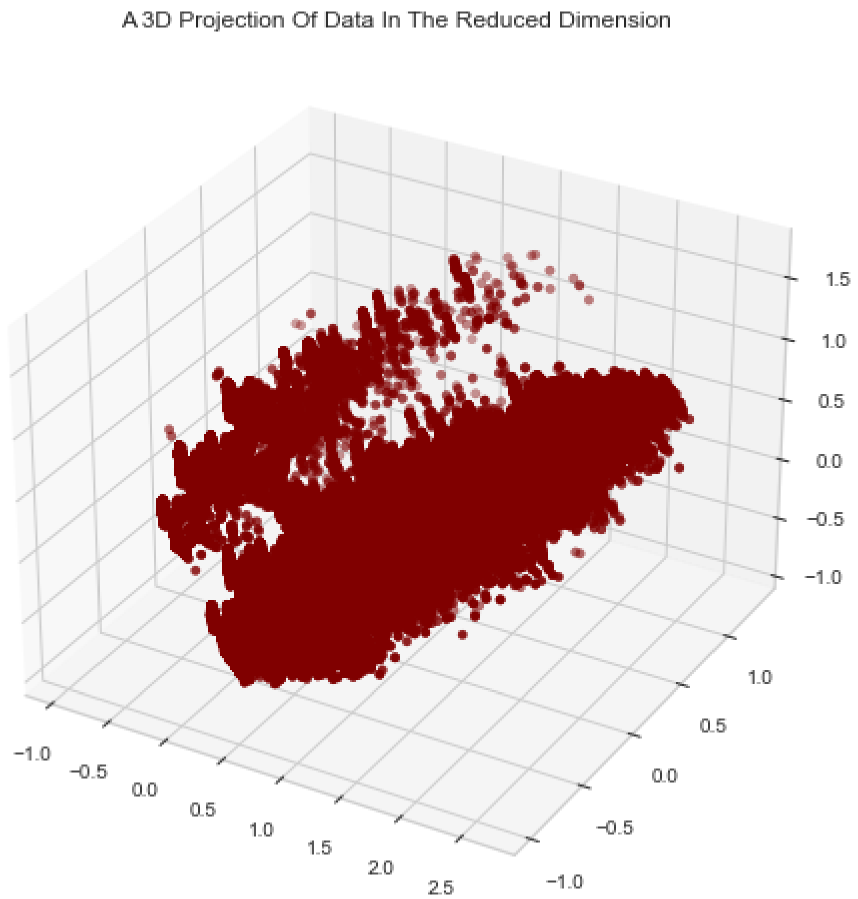
```
#Initiating PCA to reduce dimensions/ features to 3
pca = PCA(n_components=3)
pca.fit(scaled_df)
PCA_ds = pd.DataFrame(pca.transform(scaled_df), columns=(["col1","col2", "col3"]))
PCA_ds.describe().T
```

Out[50]:

	count	mean	std	min	25%	50%	75%	max
col1	884978.0	-5.774881e-15	0.712837	-0.966343	-0.607943	-0.091553	0.311279	2.676103
col2	884978.0	1.467206e-15	0.503943	-0.979636	-0.396459	-0.191932	0.495904	1.347070
col3	884978.0	4.017727e-17	0.477371	-0.914709	-0.381797	0.090090	0.402379	1.699702

In [51]:

```
# A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]
#To plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("A 3D Projection Of Data In The Reduced Dimension")
plt.show()
```

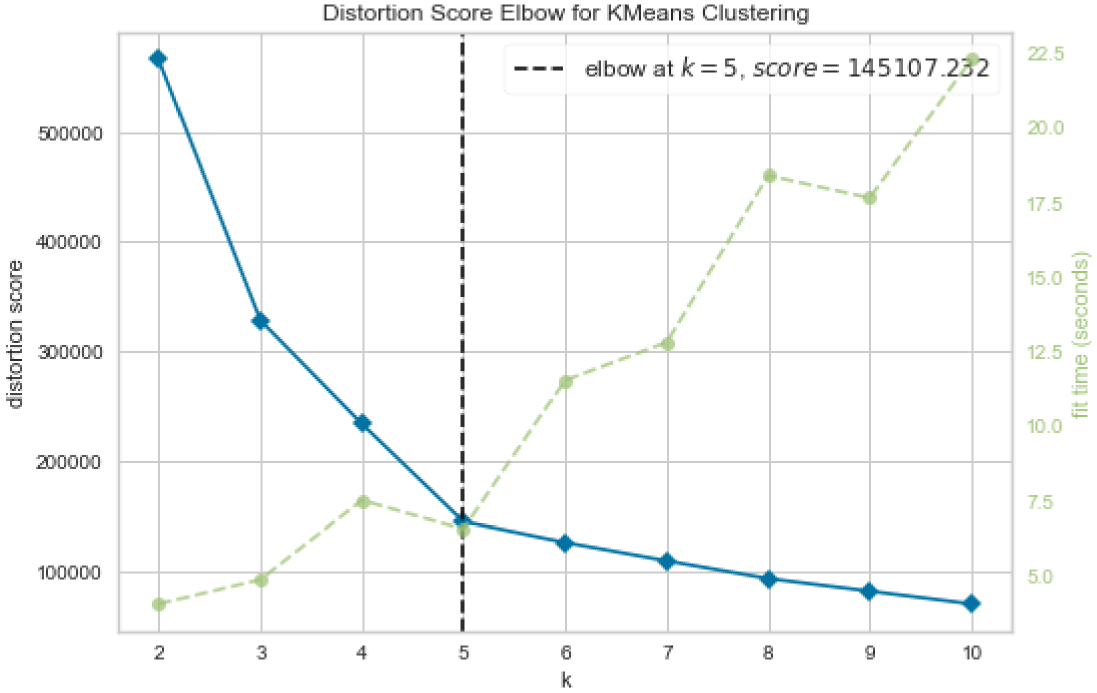


KMeans Clustering

In [52]:

```
# Quick examination of elbow method to find numbers of clusters to make.
print('Elbow Method to determine the number of clusters to be formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()
```

```
Elbow Method to determine the number of clusters to be formed:
```



Out[52]:

```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clusterin
g'}, xlabel='k', ylabel='distortion score'>
```

In [53]:

```
# Setting up color map
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#
```

In [54]:

```
# Initiating the K-Means Clustering model
kmeans5 = KMeans(n_clusters=5)
# Fit model and predict clusters
y_kmeans5 = kmeans5.fit_predict(PCA_ds)
print(y_kmeans5)
# Adding the Clusters feature to dataframe 'ds'.
ds["Clusters"]= y_kmeans5

```

```
[4 4 1 ... 2 1 3]
```

In [55]:

```
# Print the centers of 5 clusters
kmeans5.cluster_centers_
```

Out[55]:

```
array([[-0.68857289,  0.53487507,  0.15038306],
       [ 0.26116138, -0.51029659, -0.30844875],
       [ 1.6203424 ,  0.42620128,  0.51653659],
       [ 0.09953252,  0.38492716, -0.60257166],
       [-0.4865194 , -0.36257366,  0.4975291 ]])
```

In [56]:

```
# Printing the interia value

# Inertia actually calculates the sum of distances of all the points within a cluster f
# tells us how far the points within a cluster are. The distance between them should be


kmeans5.inertia_
```

Out[56]:

```
145107.2968211346
```

In [57]:

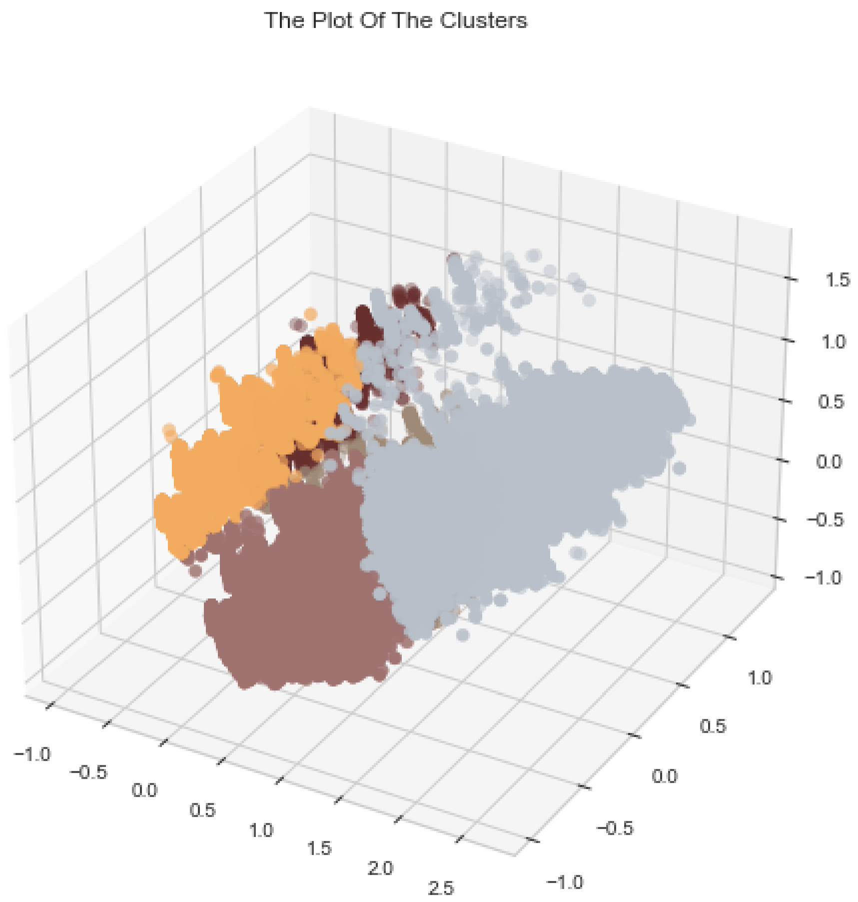
```
# Viewing data points in the 5 clusters

frame = pd.DataFrame(PCA_ds)
frame['cluster'] = y_kmeans5
frame['cluster'].value_counts()
```

Out[57]:

```
1    240380
4    208493
0    177593
3    169598
2     88914
Name: cluster, dtype: int64
```

In [58]:

```
# Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=y_kmeans5, marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
```

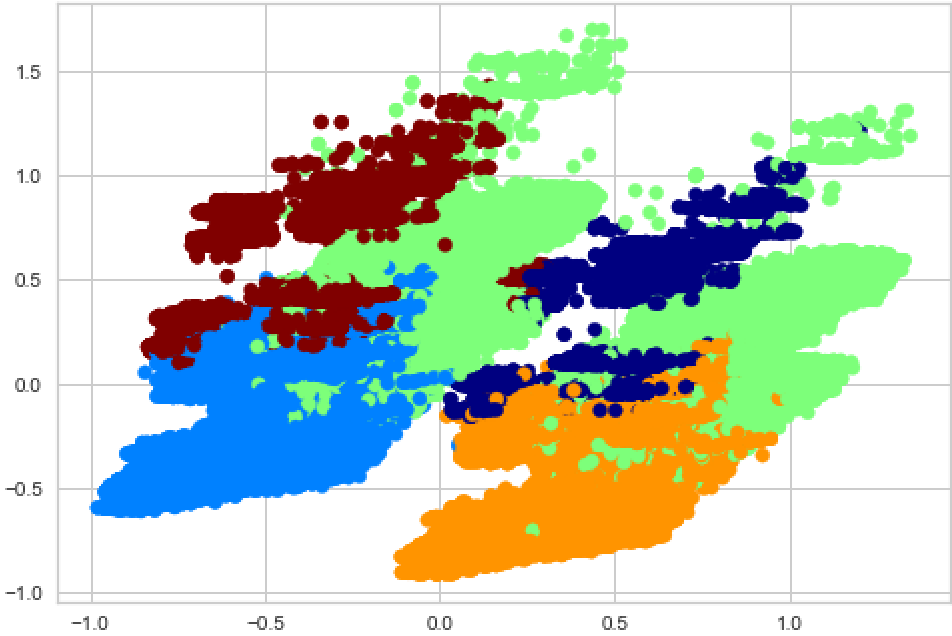


In [59]:

```
# Visualising in 2D
plt.scatter(PCA_ds.iloc[:,1],PCA_ds.iloc[:,2], c = y_kmeans5, cmap='jet')
```

Out[59]:

```
<matplotlib.collections.PathCollection at 0x1a785454d90>
```



Model Evaluation

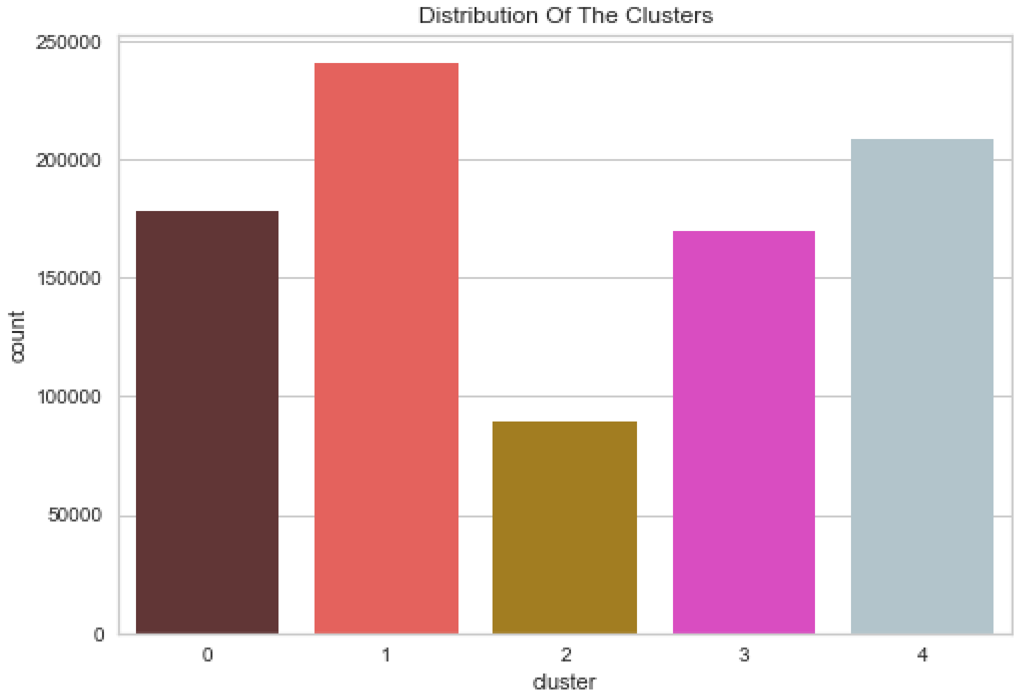
Since this is an unsupervised clustering, there is no tagged feature to evaluate or score the model. The purpose of this section is to study patterns in the clusters formed and determine the nature of the clusters' patterns.

For that, we will be having a look at the data in light of clusters via exploratory data analysis and drawing conclusions. Visualisations are grouped into the following three categories:

1. Distribution of data points in clusters
2. Scatter plots of features against age
3. Scatter plots of features against clusters

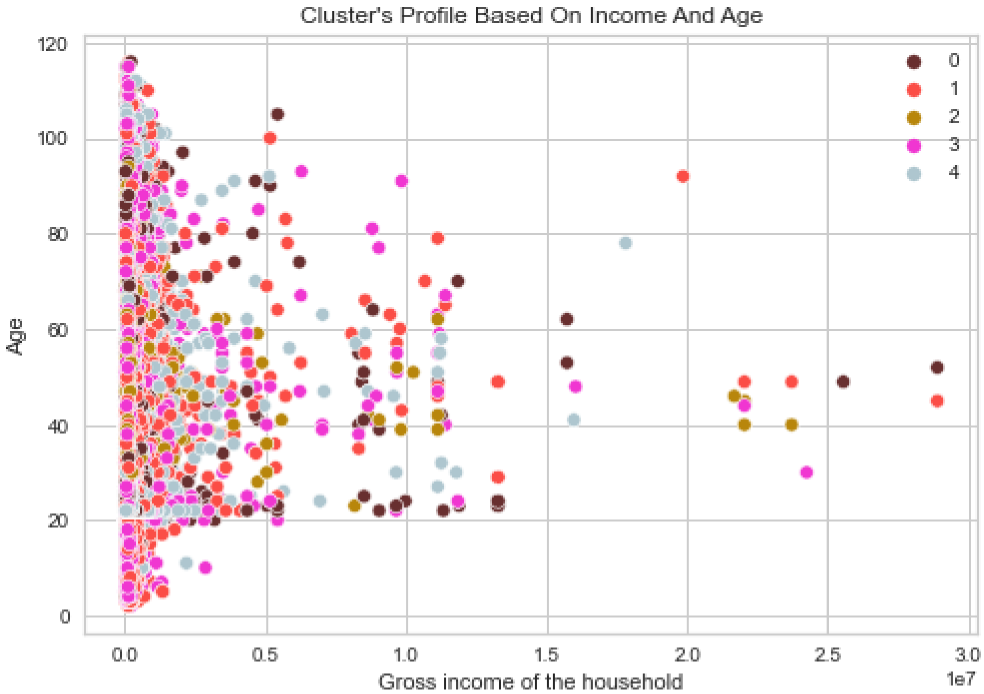
In [60]:

```
# Plotting countplot of clusters
pal = ["#682F2F","#FB4D46", "#B8860B","#F036D1", "#AEC6CF"]
pl = sns.countplot(x=PCA_ds["cluster"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()
```



In [61]:

```
# Income feature against age
pl = sns.scatterplot(data = ds,x=ds["Gross income of the household"], y=ds["Age"],hue=c
pl.set_title("Cluster's Profile Based On Income And Age")
plt.legend()
plt.show()
```



In [62]:

```
# Channel feature against age
pl = sns.scatterplot(data = ds,y=ds["Age"], x=ds["Channel used by the customer to join'
pl.set_title("Cluster's Profile Based On 'Channel used by the customer to join' And Age
plt.legend()
plt.show()
```

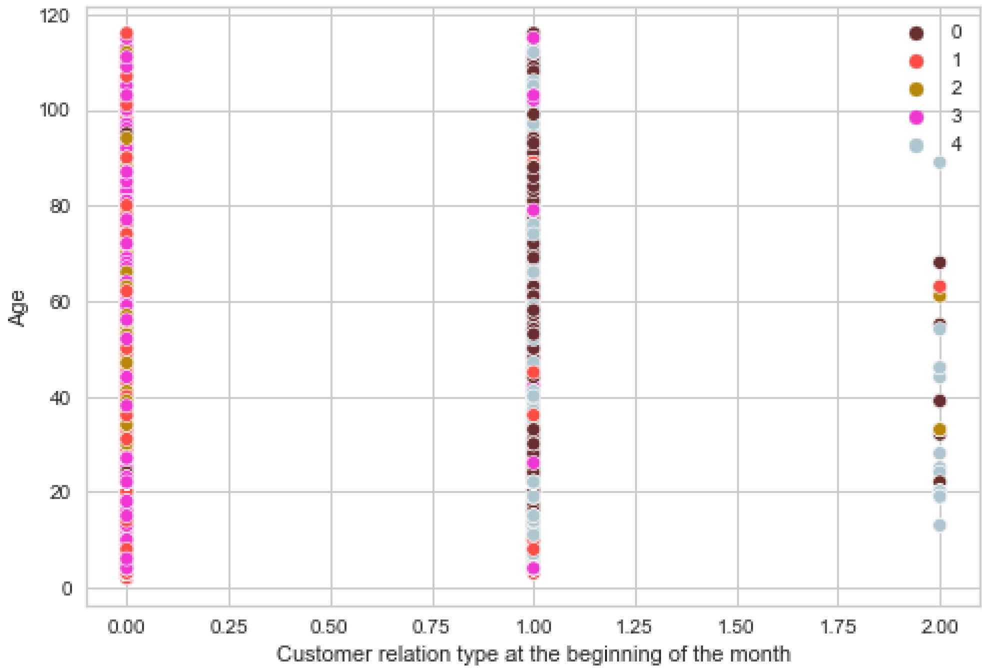
In [63]:

```
# Province feature against age
pl = sns.scatterplot(data = ds,x=ds["Province code (customer's address)"], y=ds["Age"],
pl.set_title("Cluster's Profile Based On Province code And Age")
plt.legend()
plt.show()
```

In [64]:

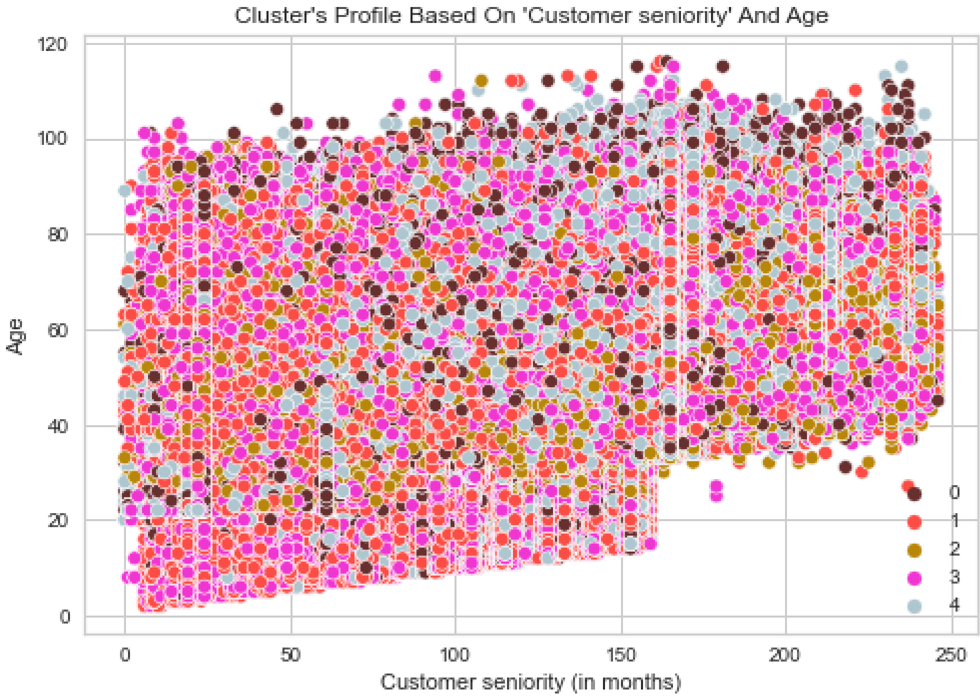
```
# Customer relation type at the beginning of the month against age
pl = sns.scatterplot(data = ds,x=ds["Customer relation type at the beginning of the mor
pl.set_title("Cluster's Profile Based On 'Customer relation type at the beginning of th
plt.legend()
plt.show()

```

Cluster's Profile Based On 'Customer relation type at the beginning of the month' And Age



In [65]:

```
# Customer seniority feature against age
pl = sns.scatterplot(data = ds,x=ds["Customer seniority (in months)"], y=ds["Age"],hue=
pl.set_title("Cluster's Profile Based On 'Customer seniority' And Age")
plt.legend()
plt.show()

```



Insights

- Majority of customers are between 40 and 80 as that portion has dense concentration.
- Majority of bank holder's income range falls below 5,000,000.
- Clusters 0, 3 and 4 dominate type A at the beginning of the month with 2 dominating type P.
- KFC's age bracket starts at 40
- Clusters 1 and 3 boast of more data points with least customer seniority
- Cluster 0 has a bit more older customers

In [66]:

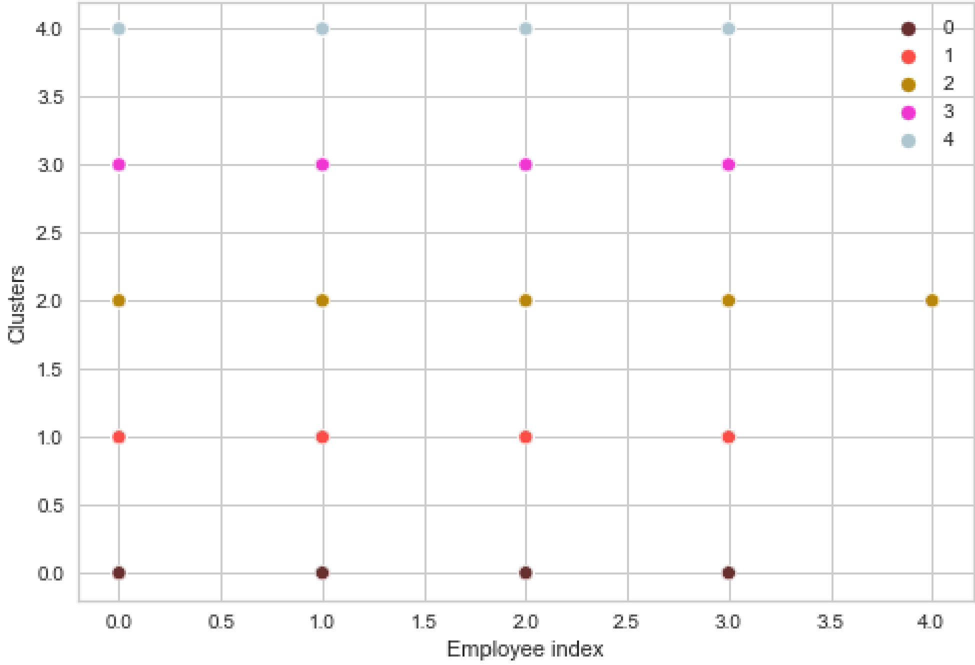
```
len(ds.columns)
```

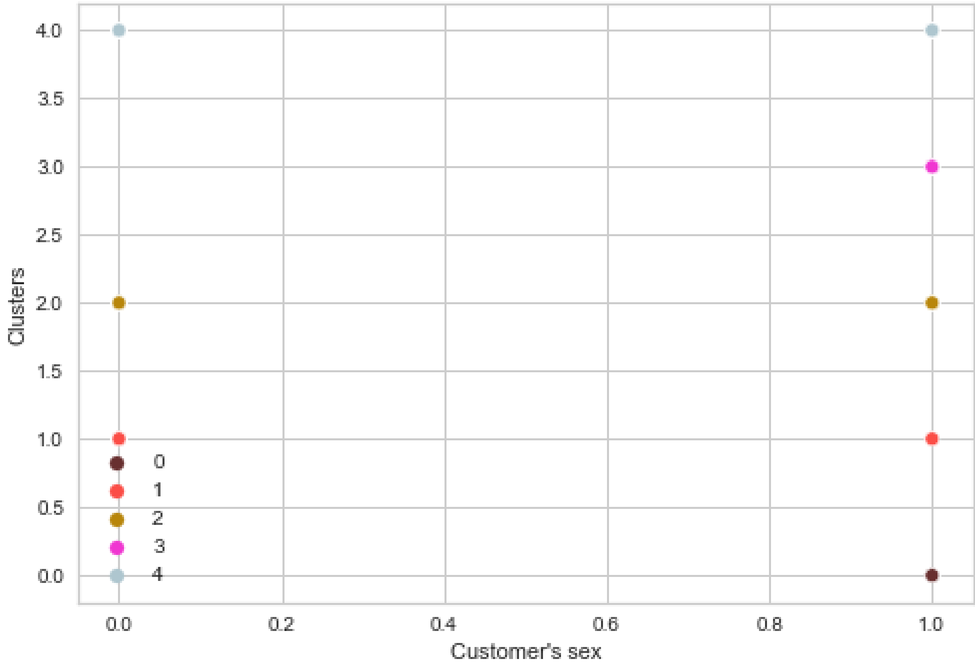
Out[66]:

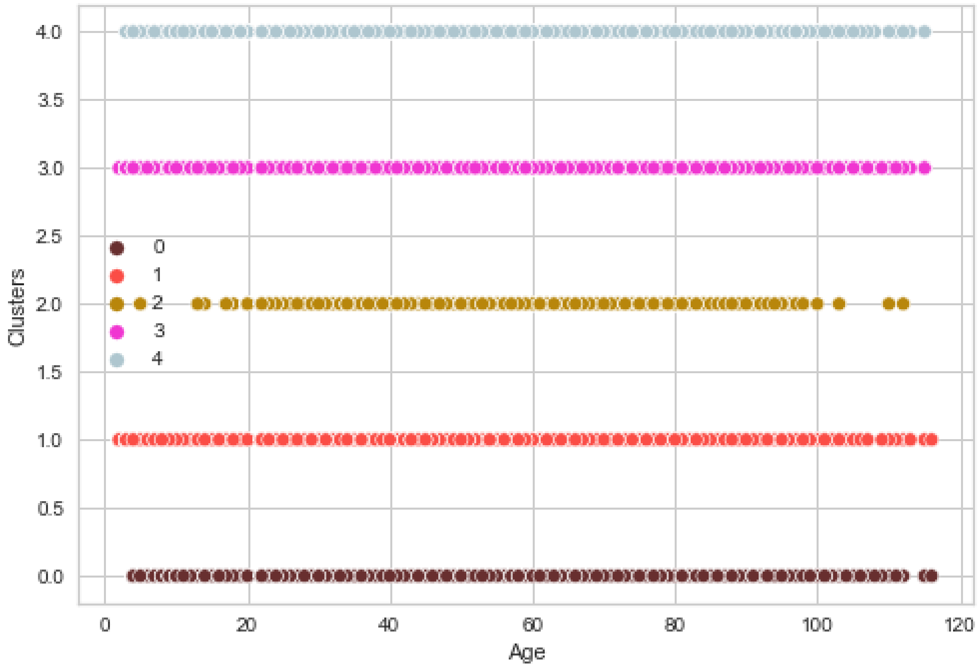
42

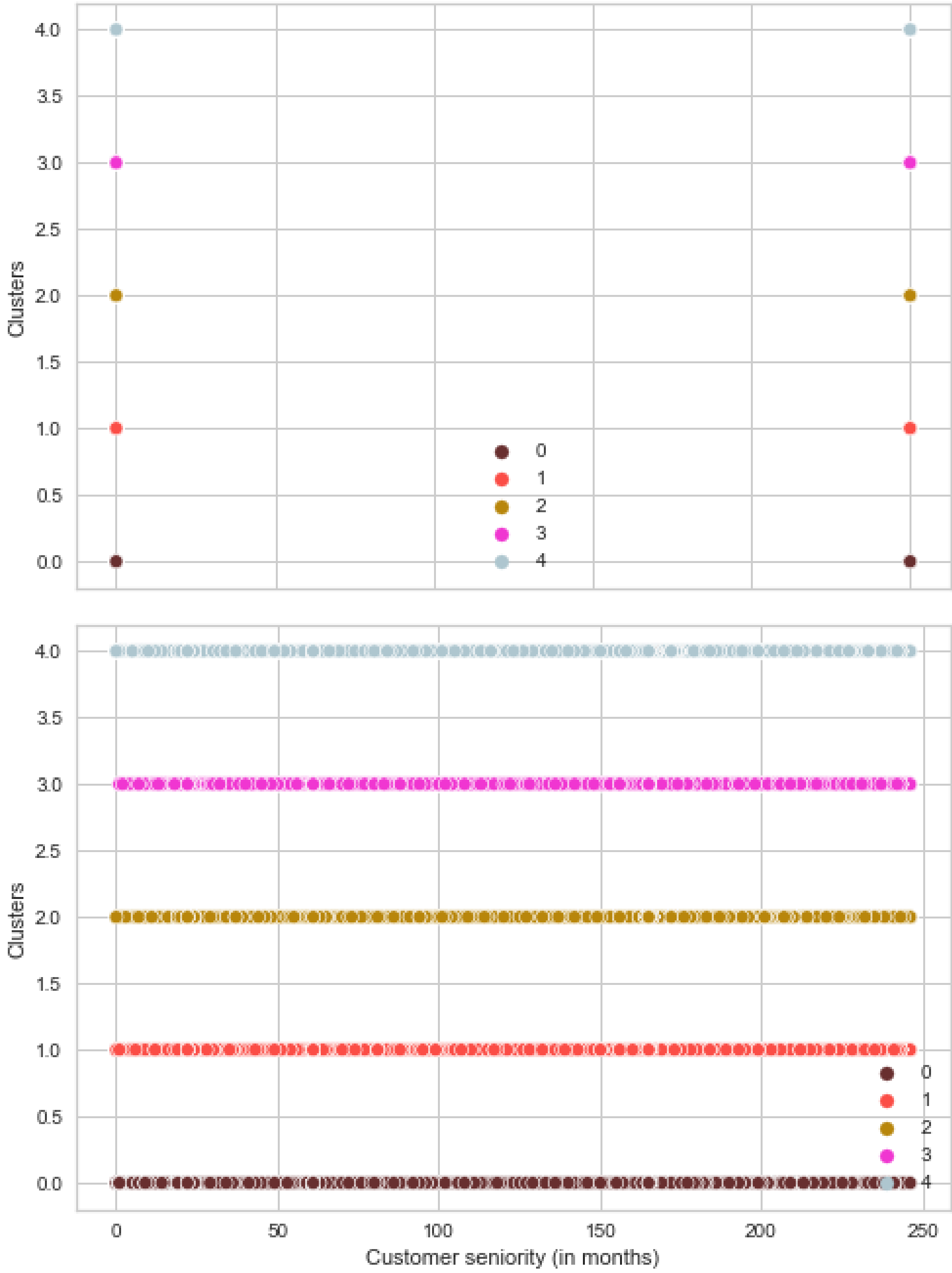
In [67]:

```
# Plotting individual features against cluster groups
col = list(ds.columns)
for i in range(42):
    pl = sns.scatterplot(data = ds, x= col[i], y= ds["Clusters"], hue= ds["Clusters"],
    plt.legend()
    plt.show()
```

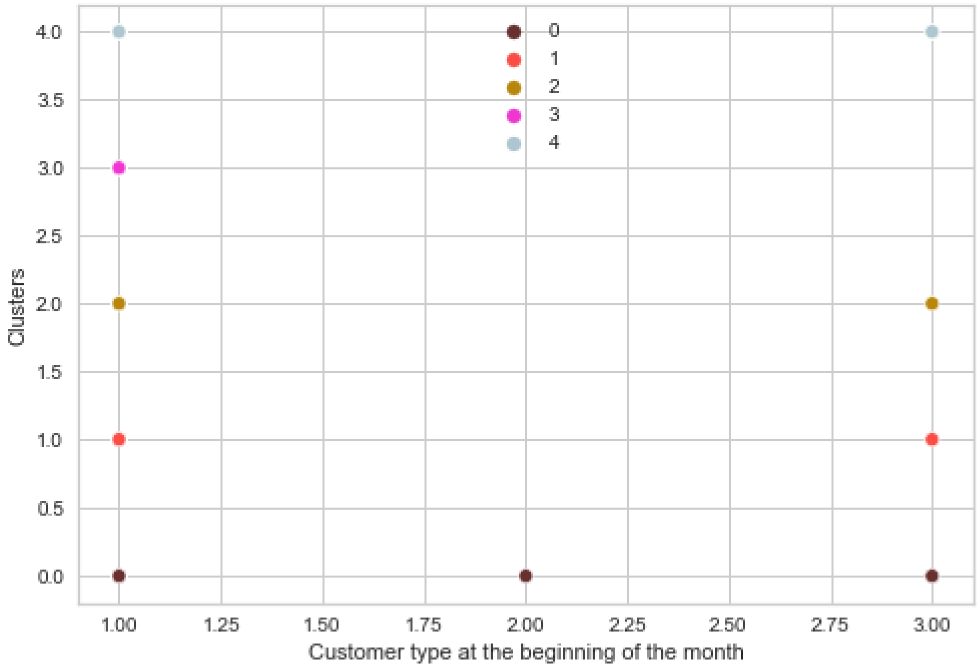



4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Customer's sex

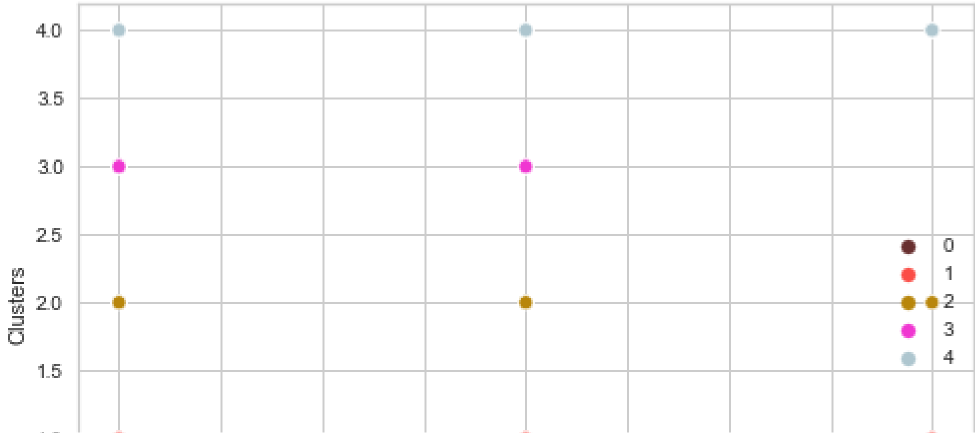

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0
20
40
60
80
100
120
Age

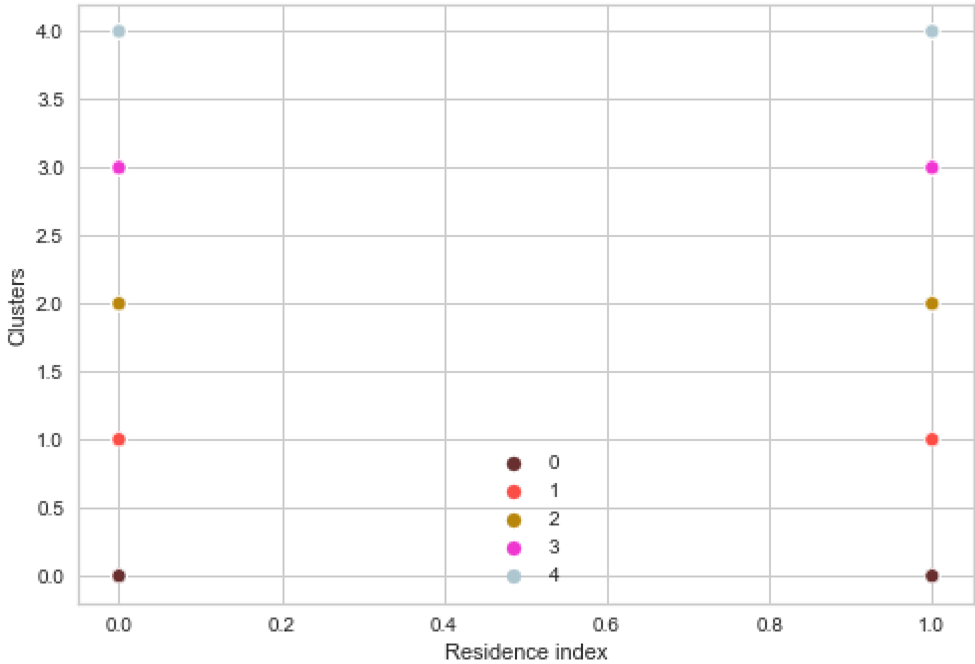

Clusters
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
1
2
3
4
0
50
100
150
200
250
Customer seniority (in months)

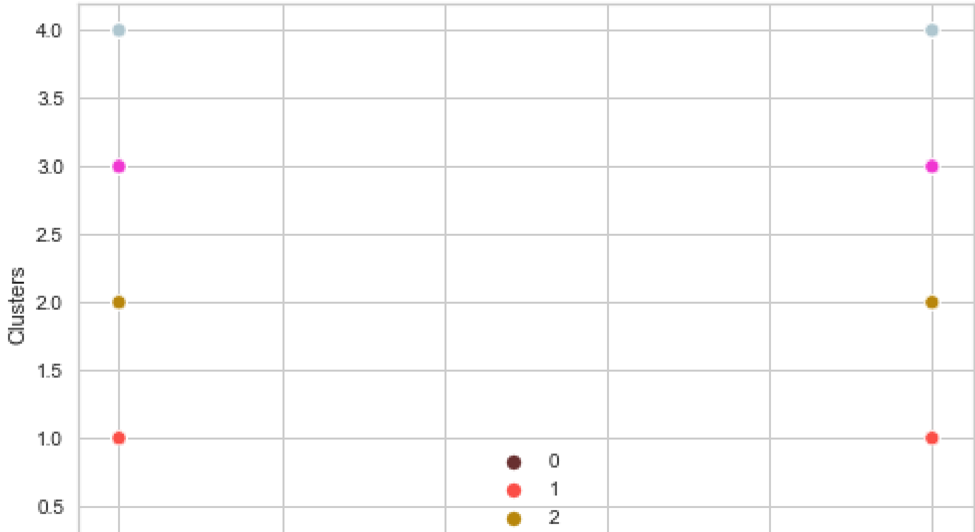
Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0

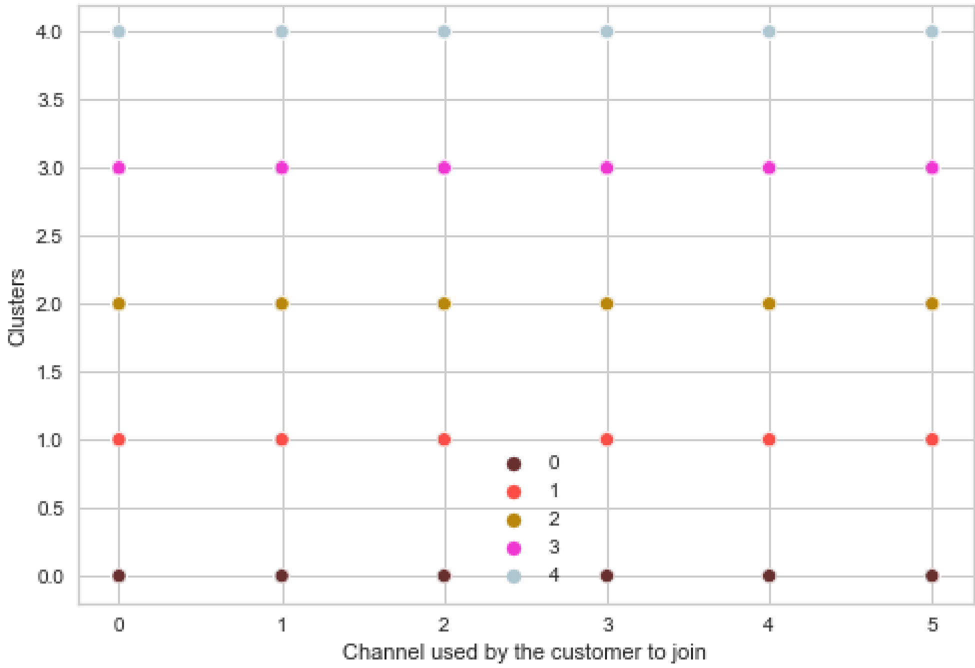


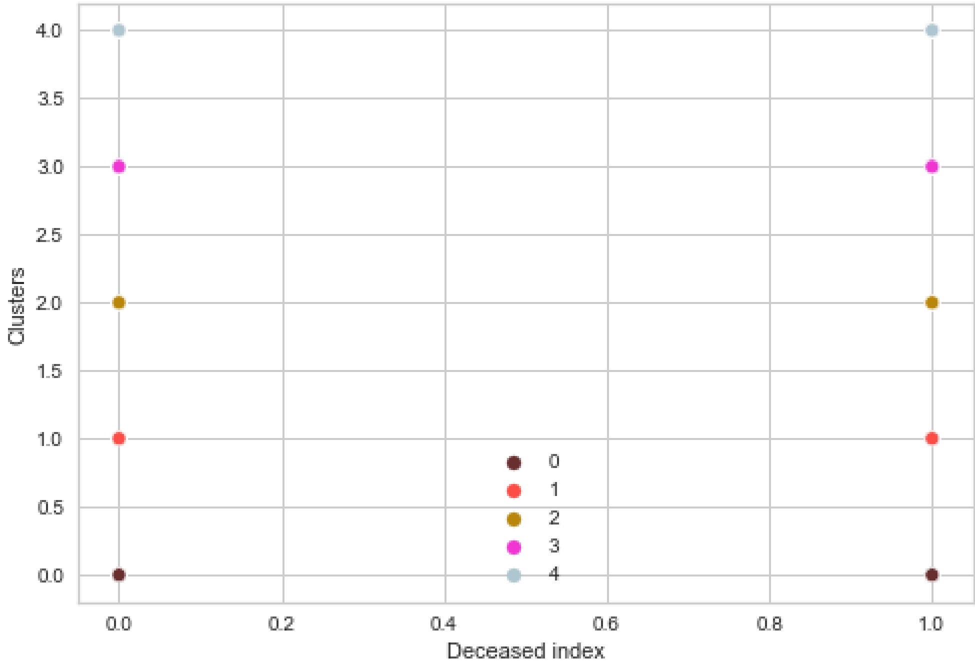
Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Customer type at the beginning of the month

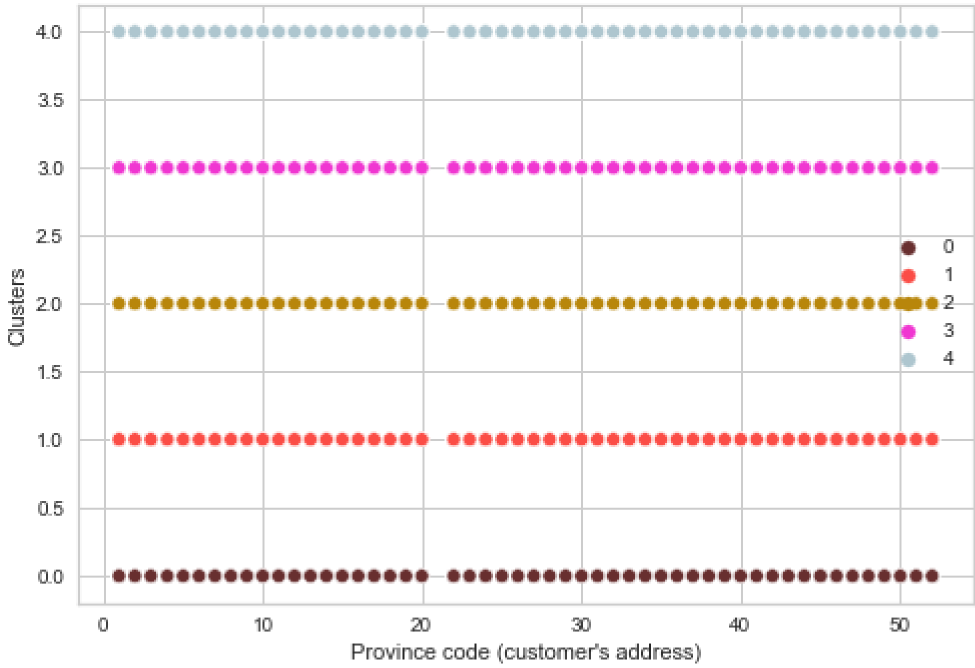

4.0
3.5
3.0
2.5
2.0
1.5
Clusters
0
1
2
3
4

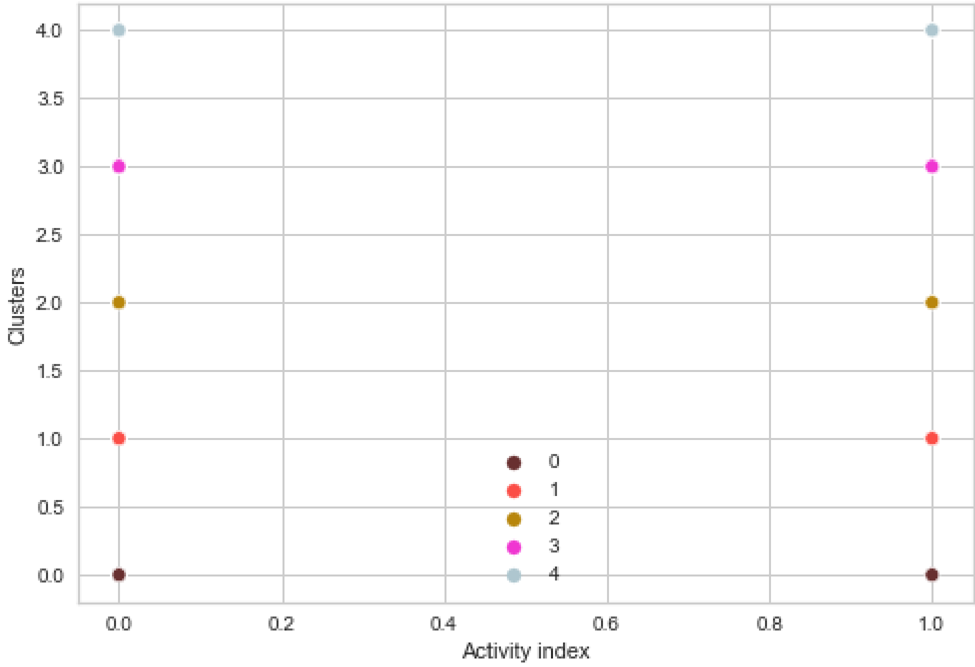

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Residence index

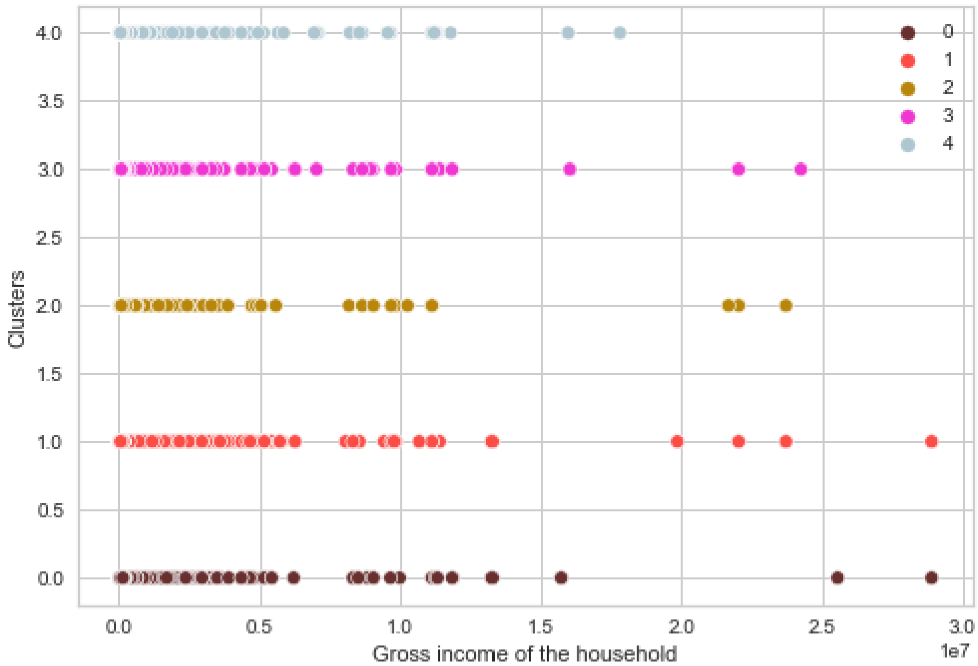

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
1
2

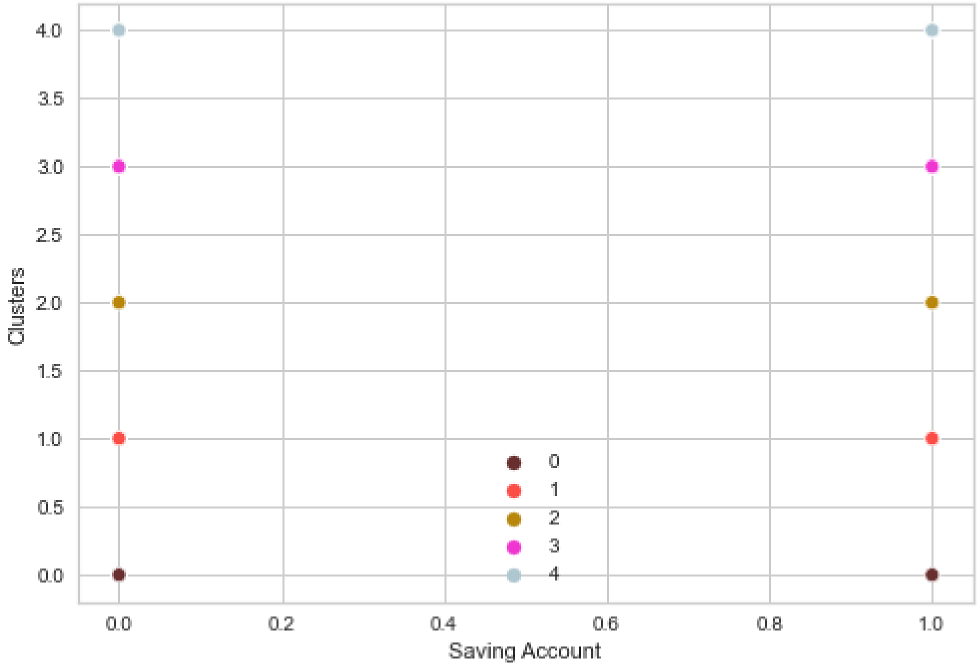

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
0
1
2
3
4
5
Channel used by the customer to join

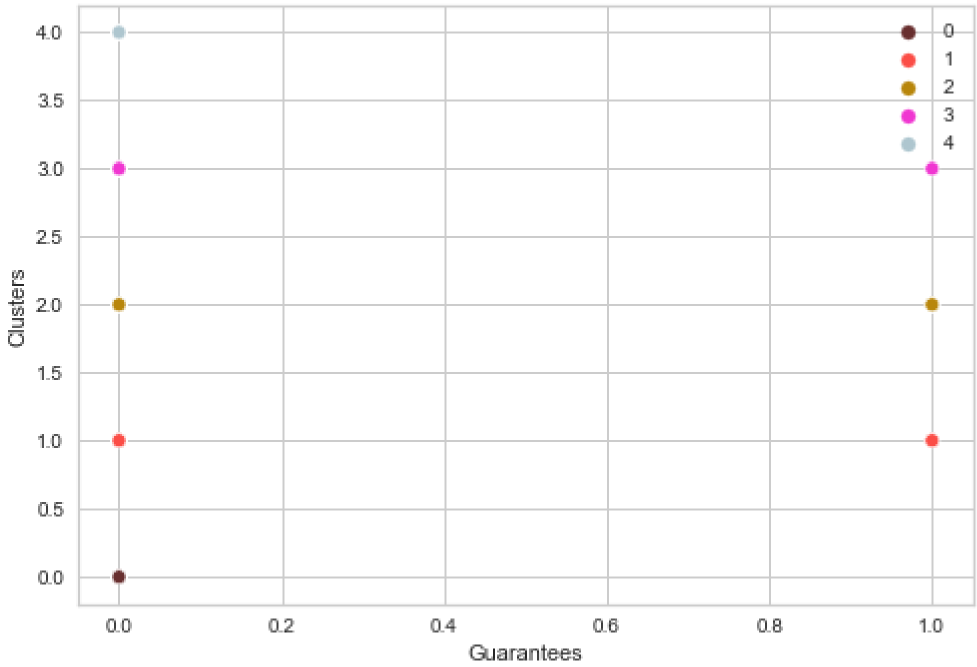

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Deceased index

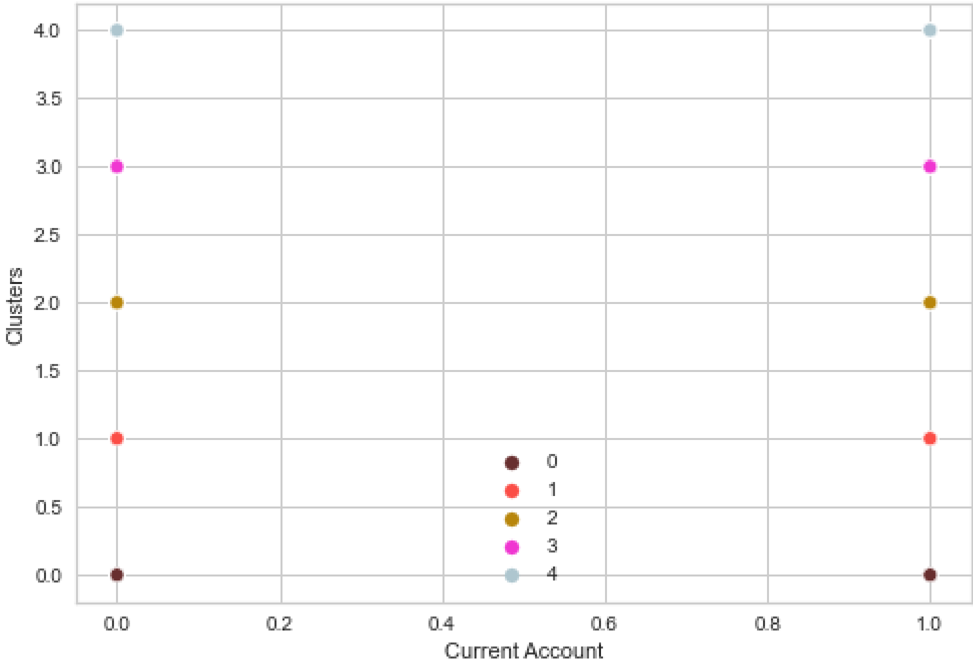

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0
10
20
30
40
50
Province code (customer's address)

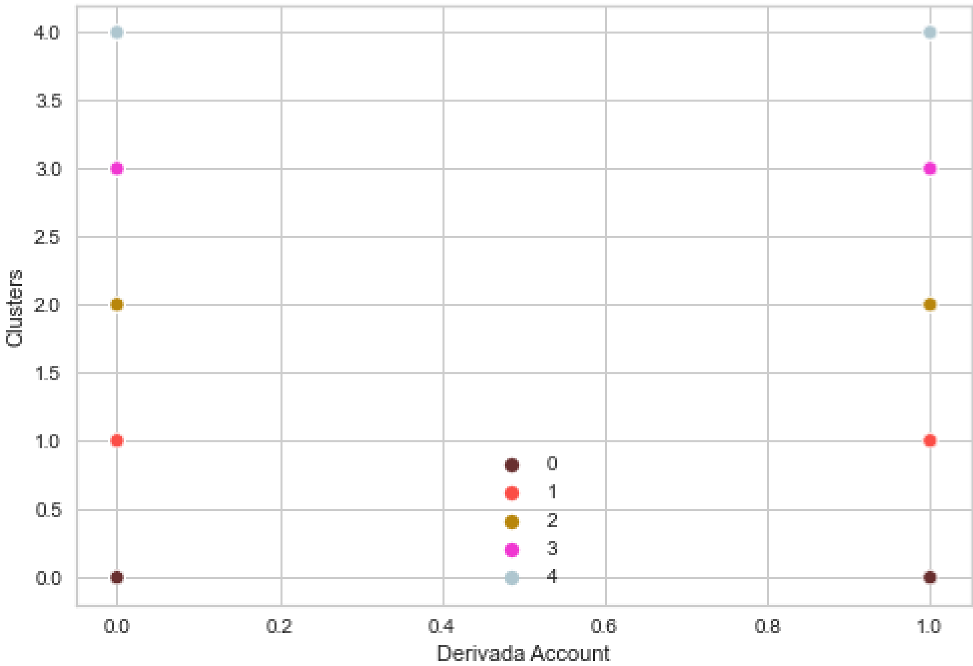

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Activity index

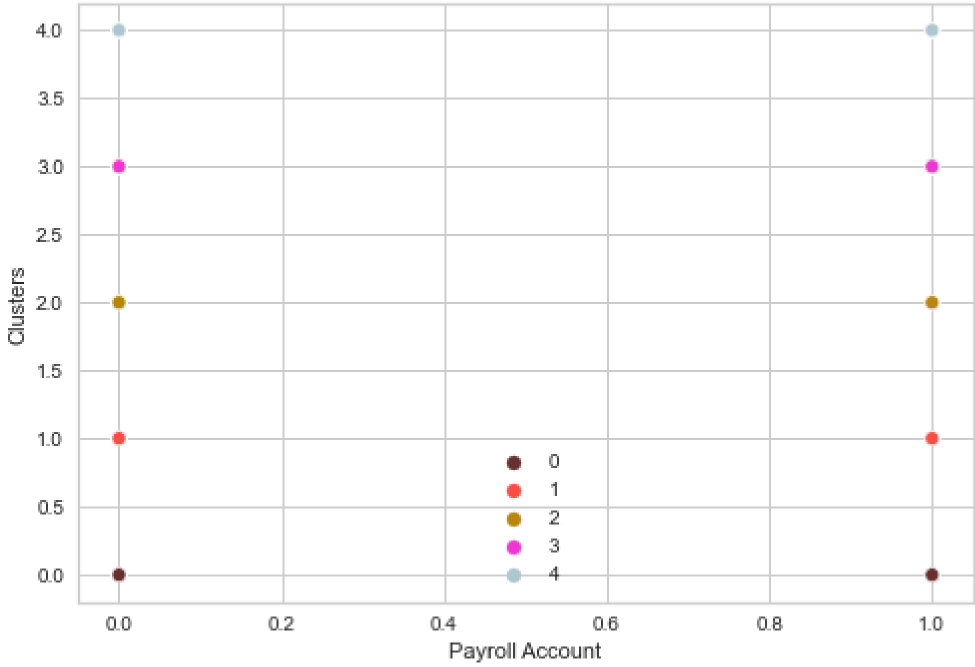

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e7
Gross income of the household


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Saving Account

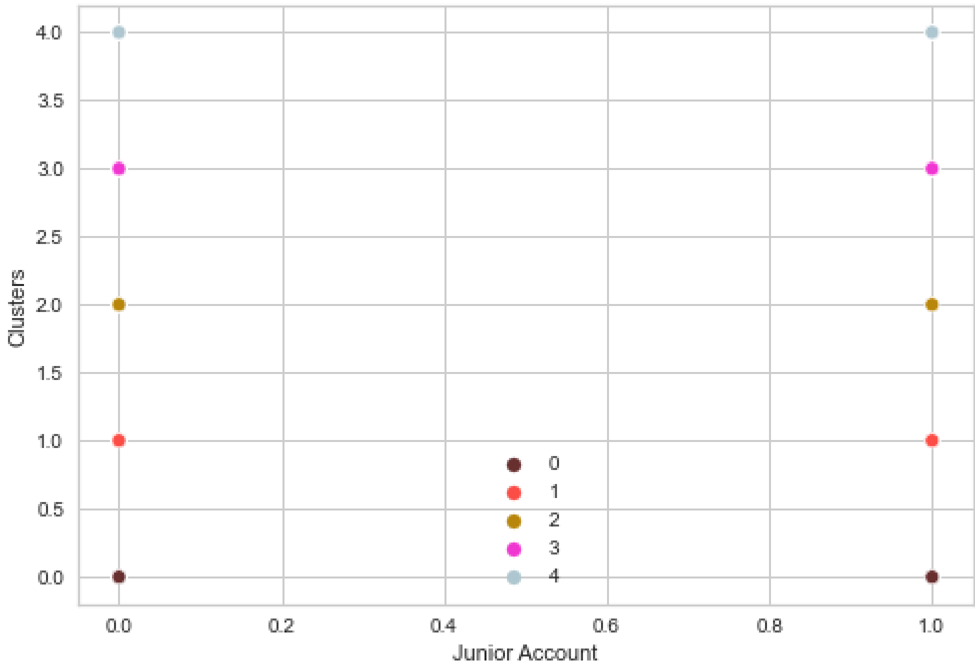

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Guarantees


Clusters
Current Account
0
1
2
3
4

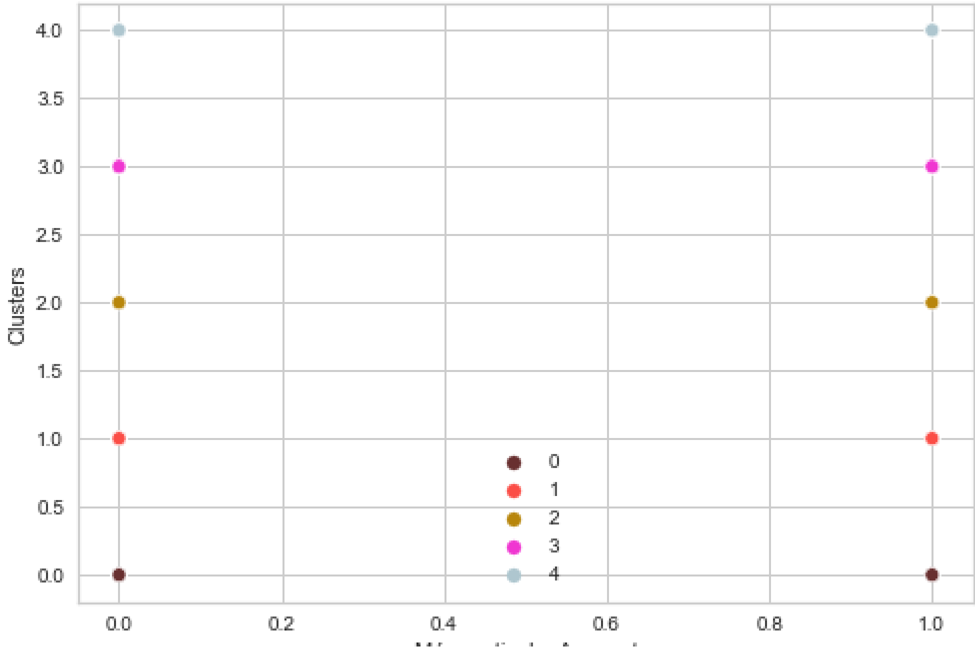

Clusters
Derivada Account
0
1
2
3
4

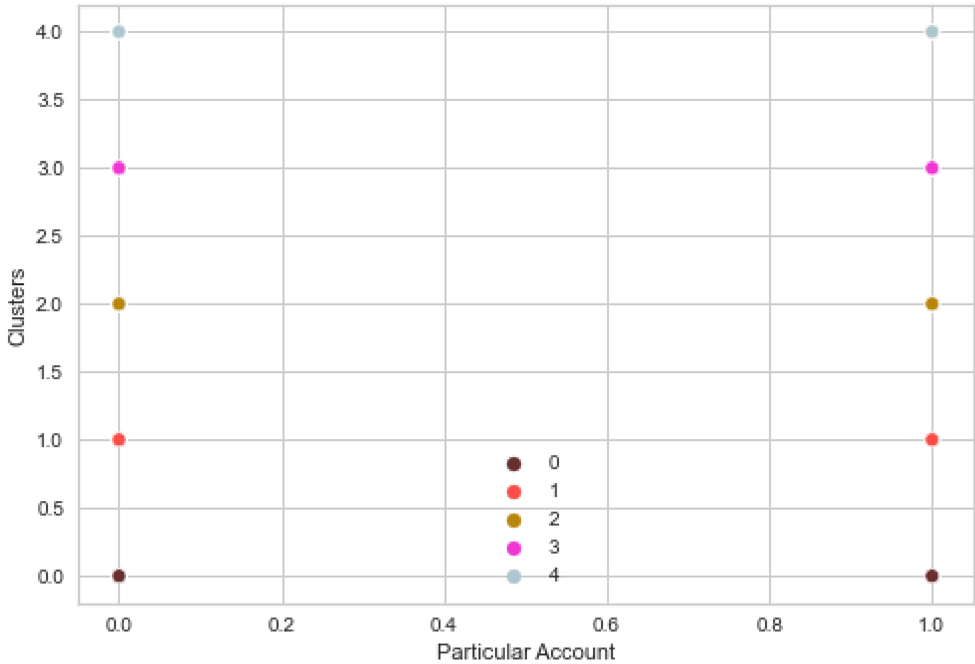


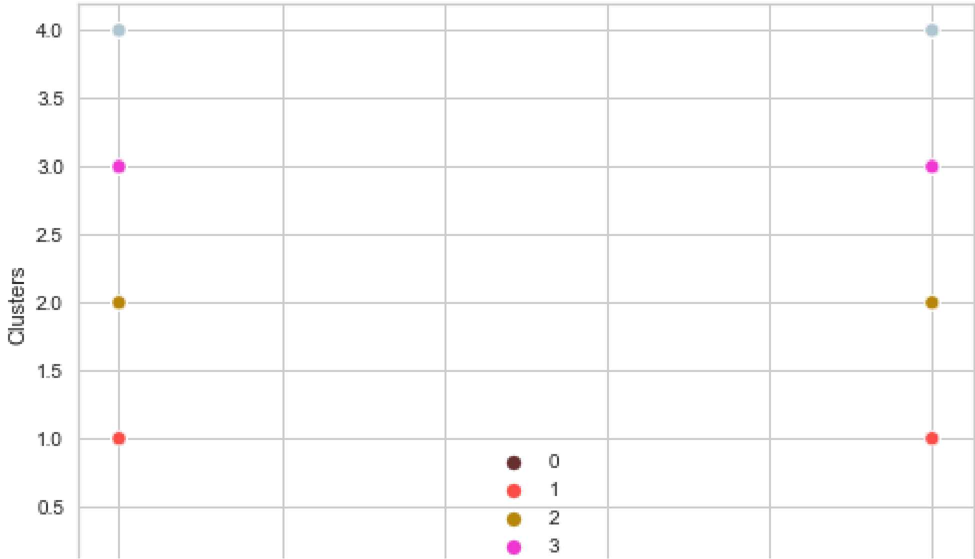
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Payroll Account

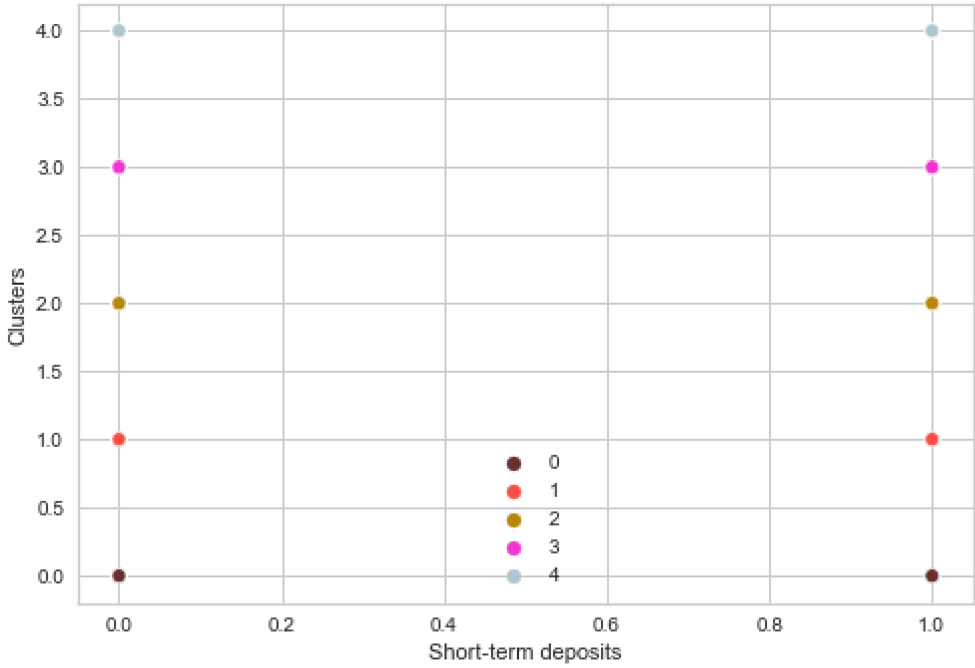


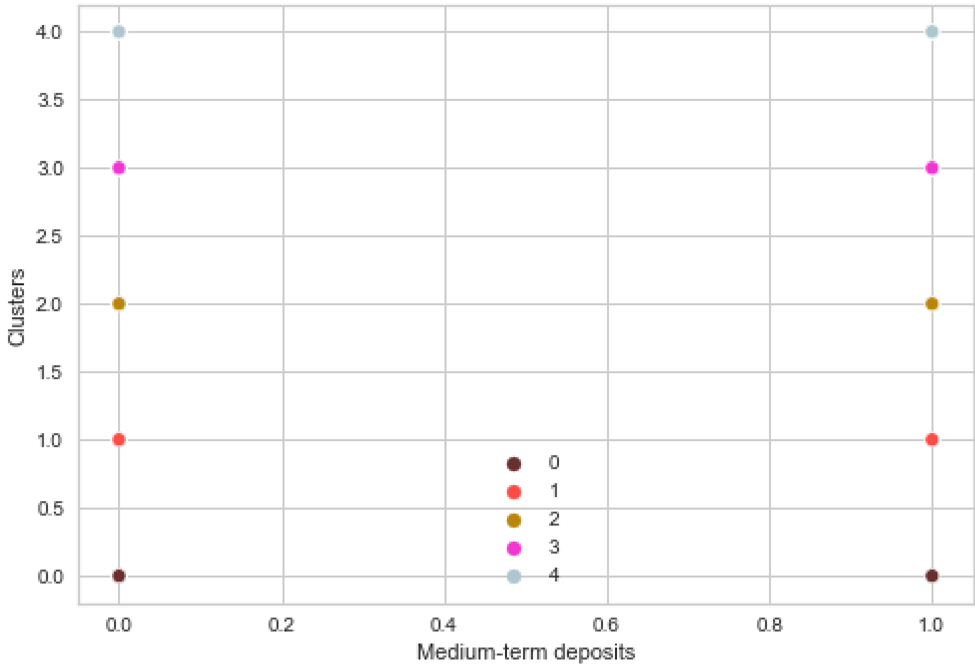
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Junior Account

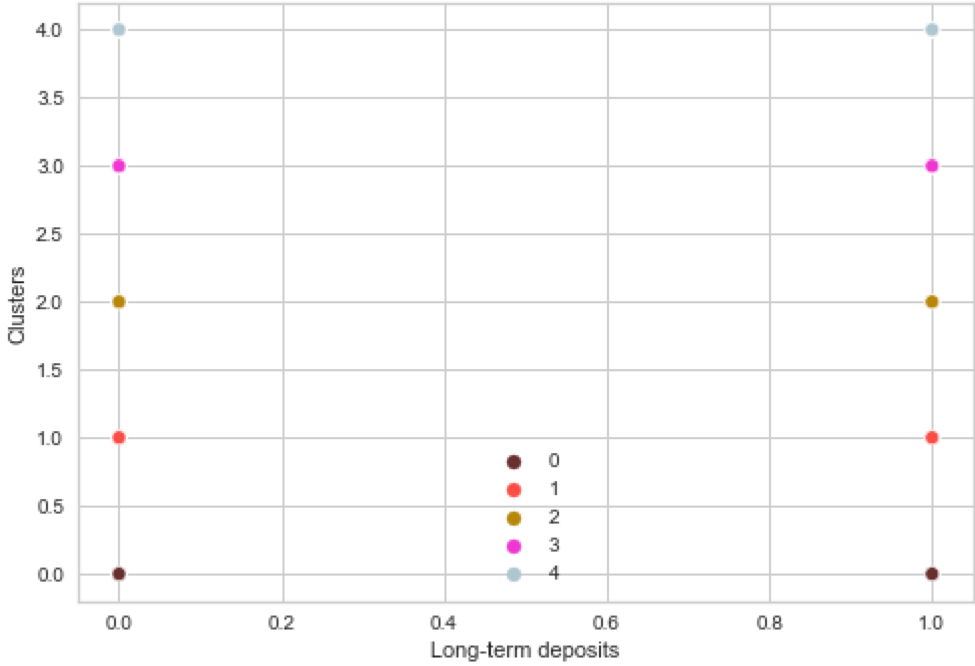

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4

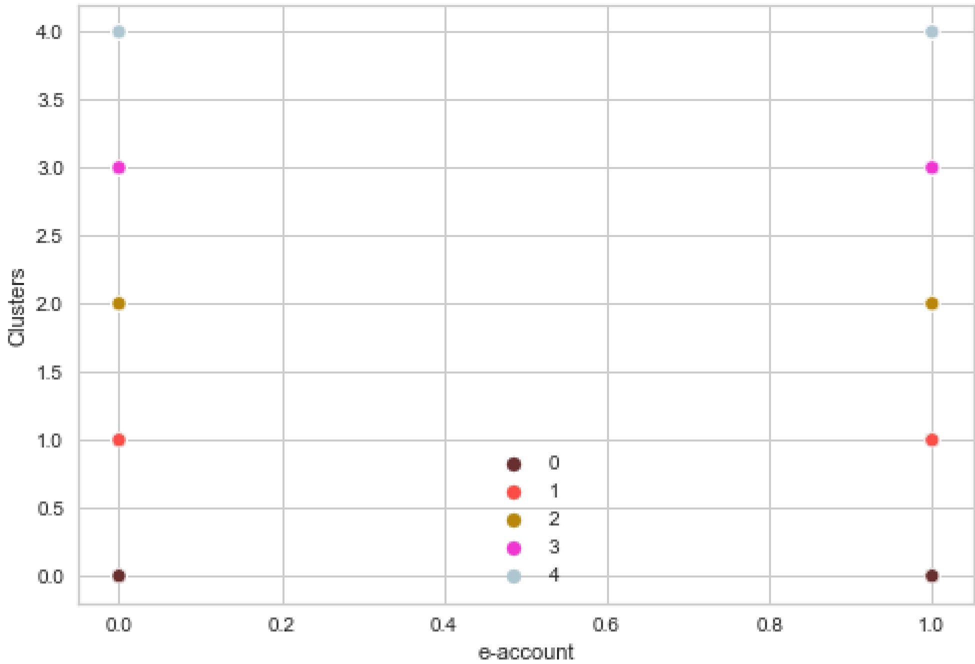

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Particular Account
0
1
2
3
4

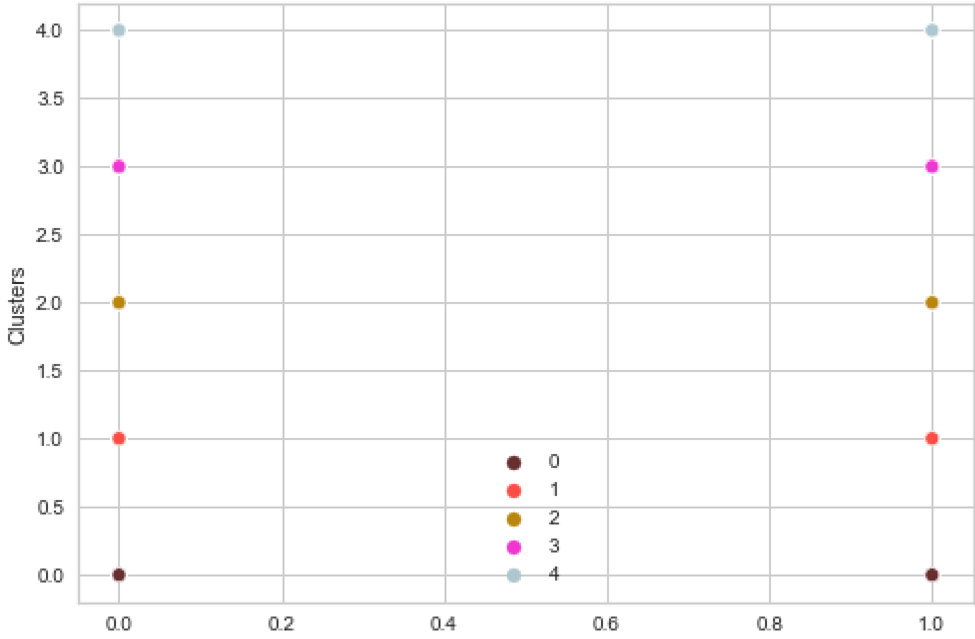

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
1
2
3

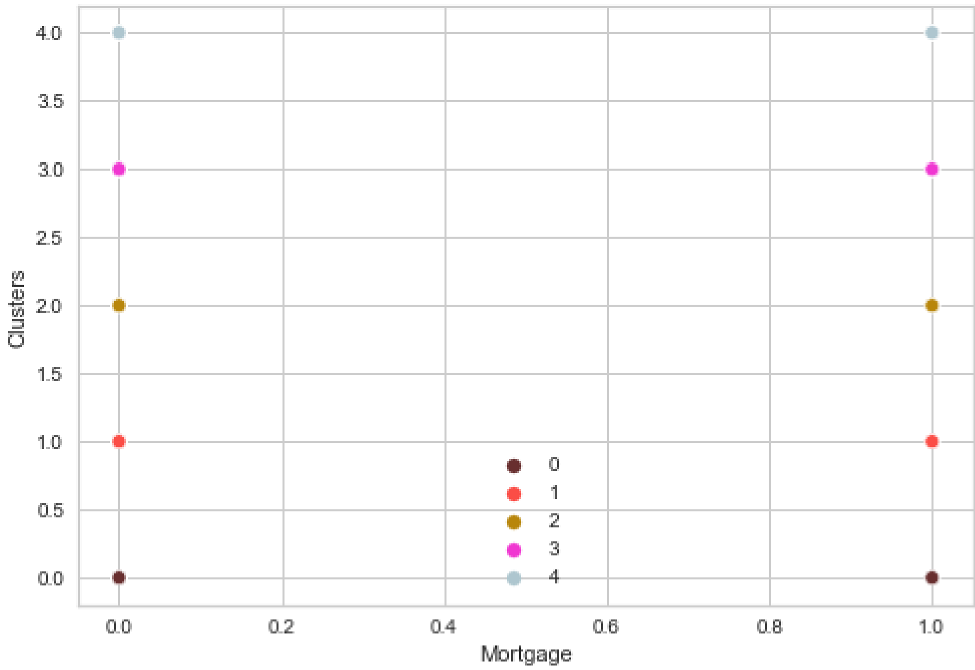

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Short-term deposits

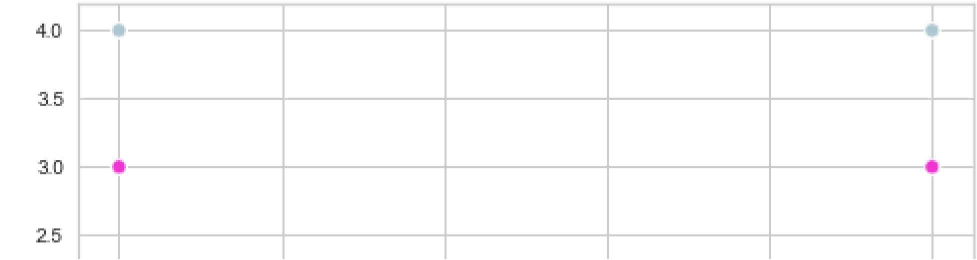

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Medium-term deposits

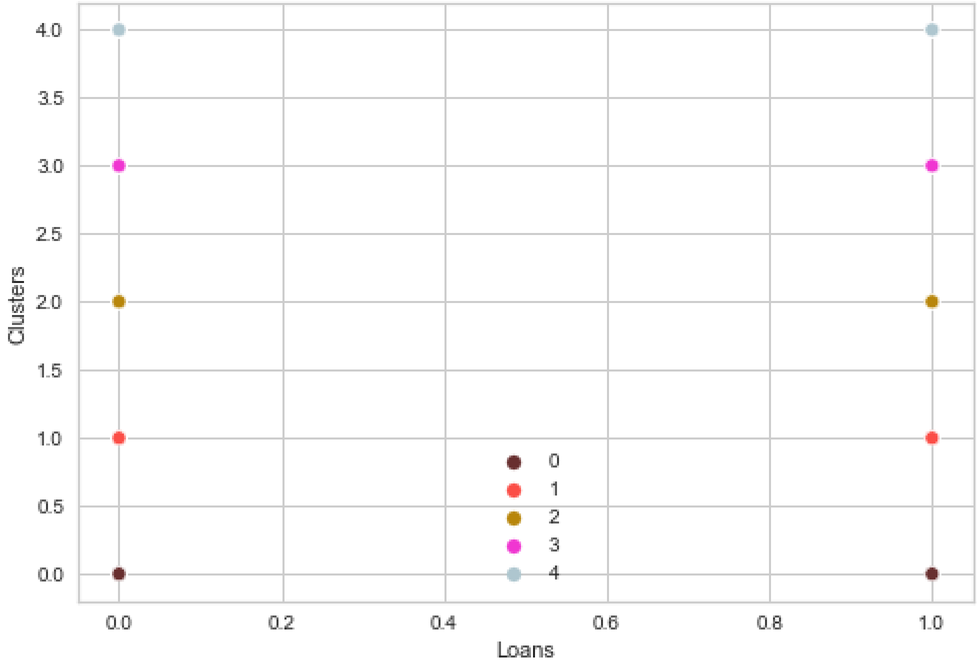

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Long-term deposits

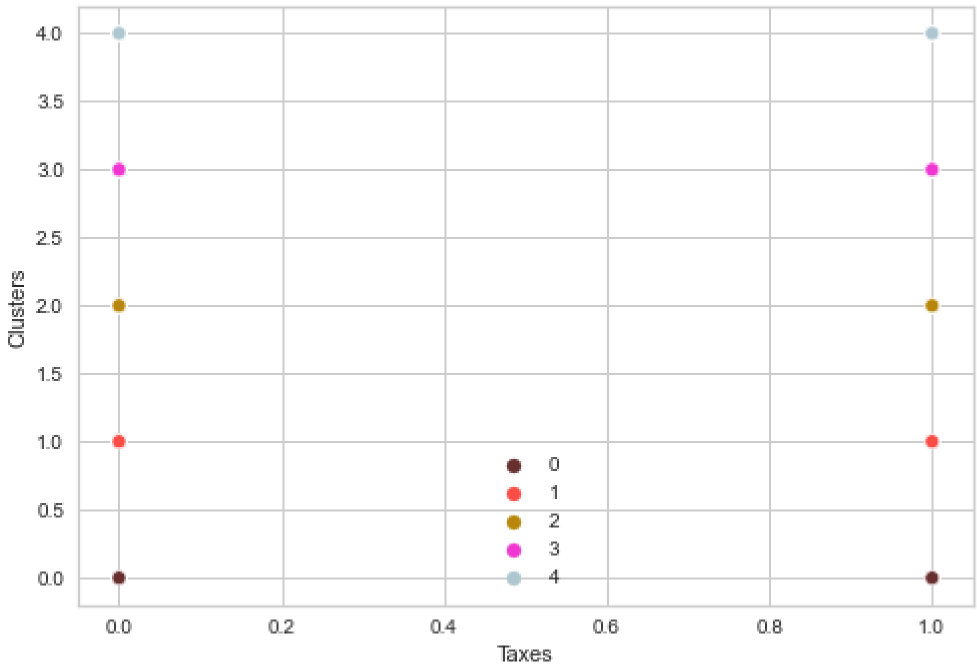

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
e-account

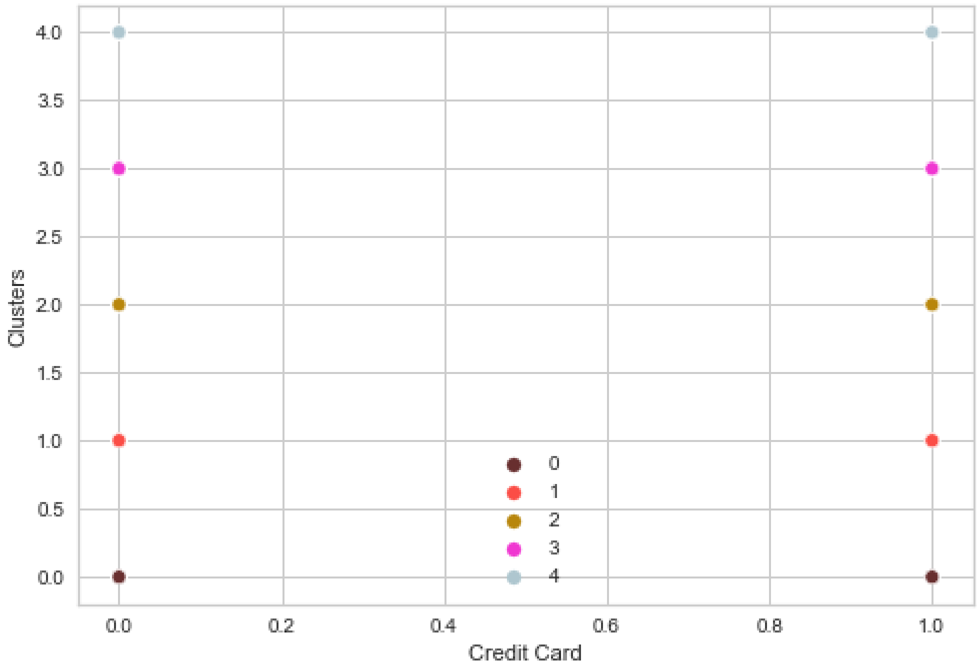

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4

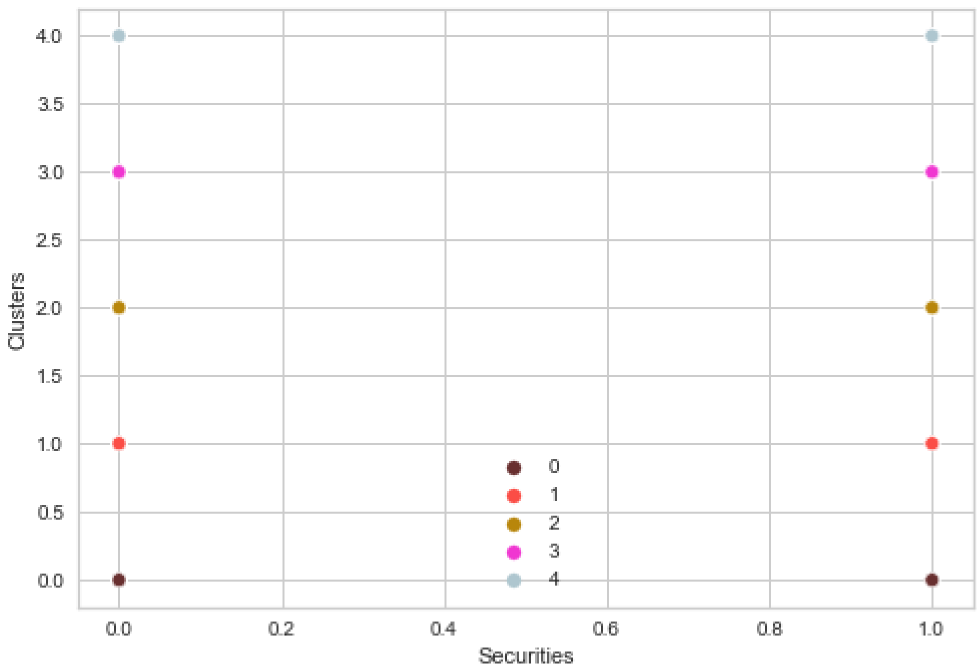

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Mortgage
0
1
2
3
4

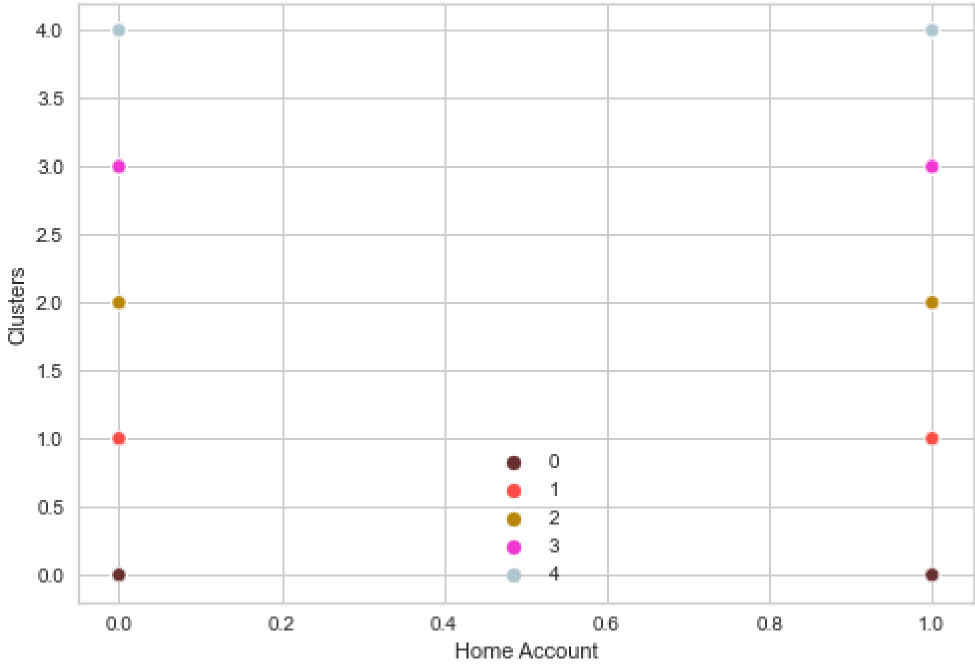

4.0
3.5
3.0
2.5

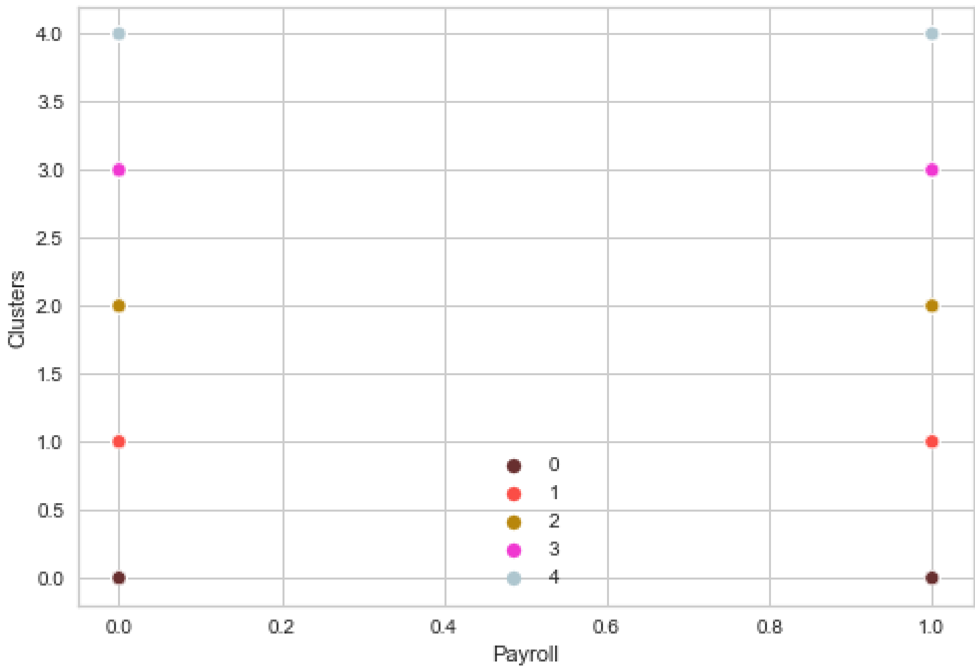

Clusters
Loans
0
1
2
3
4

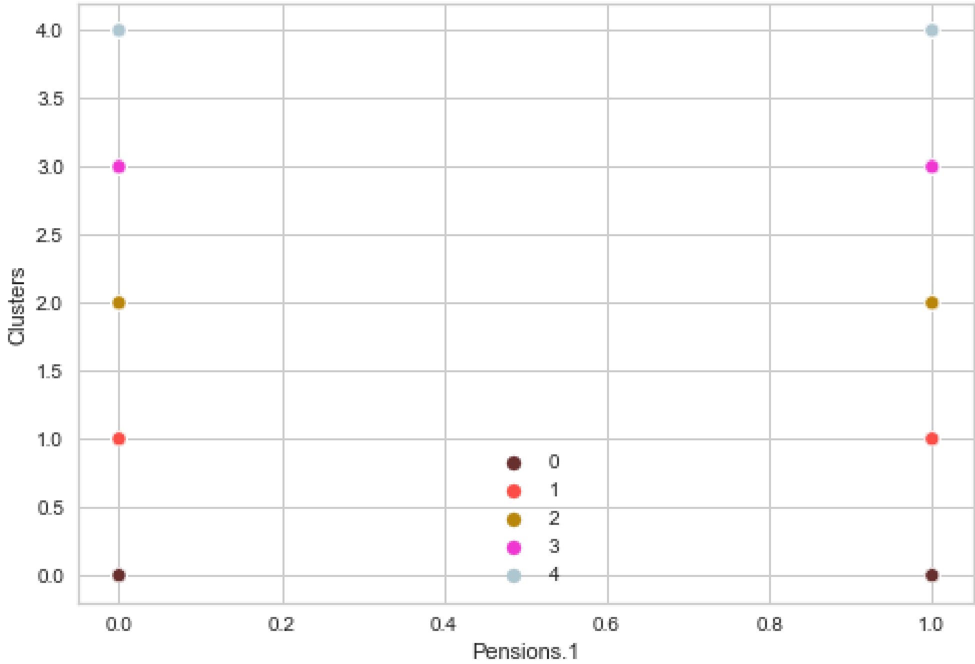

Clusters
Taxes
0
1
2
3
4

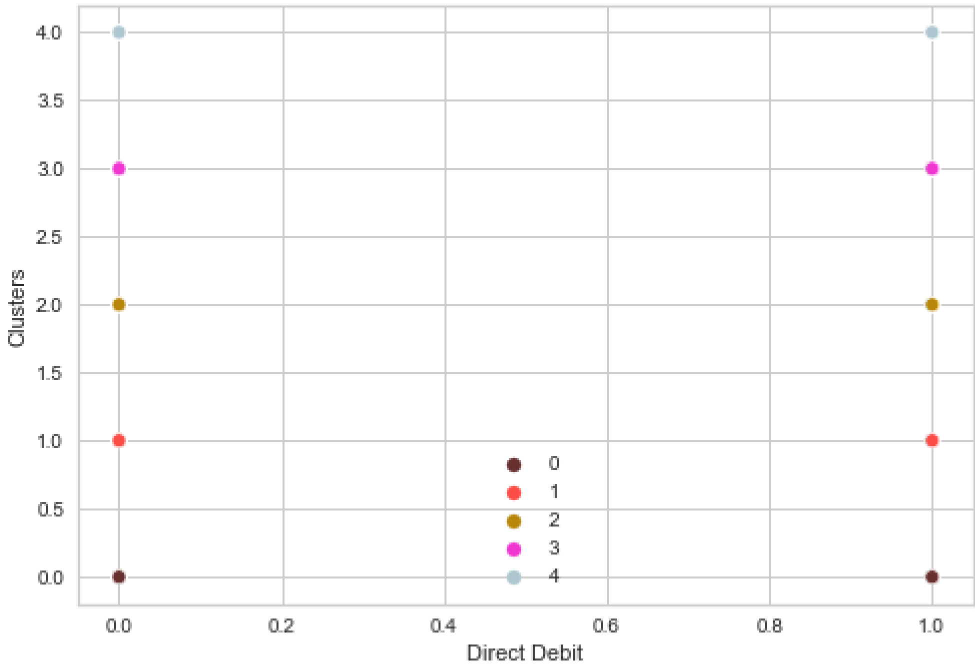

Clusters
Credit Card
0
1
2
3
4

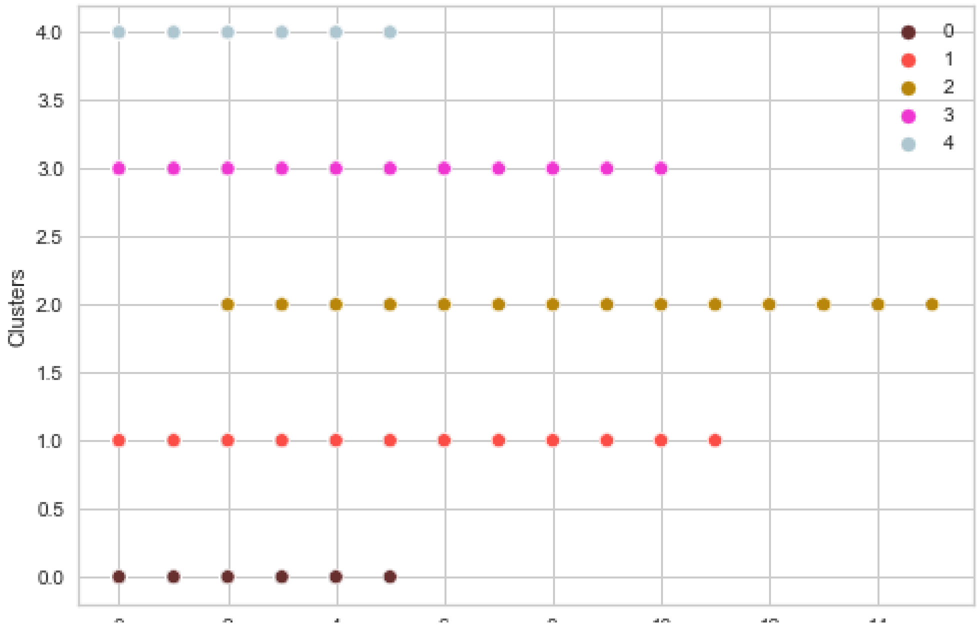

Clusters
Securities
0
1
2
3
4

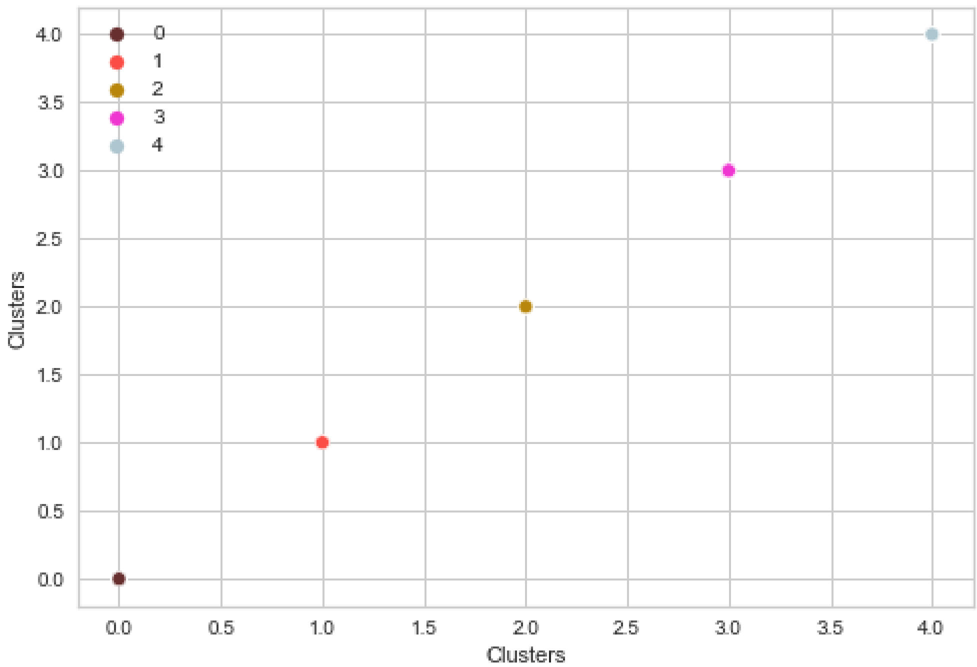

Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Home Account


Clusters
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Payroll


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Pensions.1


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Clusters
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Direct Debit





Based on the Exploratory analysis done, the following assumptions are made about the clusters:

1. Cluster 0

- Male only
- Spans all customer types at the beginning of the month

- No guarantees
- Uses maximum 5 products
- 3rd largest representation

2. Cluster 1

- Wide income range, no limit
- Uses maximum 11 products
- Largest representation

3. Cluster 2

- Few teenagers and few older ones(above 100)
- Income limit 25,000,000
- Minimum of 2 and maximum of 15 products.
- Least representation

4. Cluster 3

- Male only
- Customer relation type A and I only at the beginning of the month
- Customer type 1 only at the beginning of the month
- Income limit 25,000,000
- Uses maximum 10 products
- Fairly large share in all 6 major channels used to join

5. Cluster 4

- Income limit 20,000,000
- No guarantees
- Uses maximum 5 products
- 2nd largest representation

Aside these variances, all other products have a fair representation among the clusters

Miscellaneous

In [68]:

```
# Special customer using 15 products
data[data['Total products']>14]
```

Out[68]:

	Unnamed: 0	Customer code	Employee index	Customer's Country residence	Customer's sex	Age	Date of first contract(account was created)	cust[illegible] l
797816	797816	121159	N	ES	F	47.0	1999-02-11	