customer_segmentation

March 17, 2023

1 Customer Segmentation - Clustering

Nikhil Sharma

1.0.1 Abstract:

In this project, we will be performing different unsupervised clustering of data on the customer's records from a groceries firm's database. Customer segmentation is the practice of separating customers into groups that reflect similarities among customers in each cluster. we will divide customers into segments to optimize the significance of each customer to the business. To modify products according to distinct needs and behaviors of the customers. It also helps the business to cater the concerns of different types of customers.

1.0.2 Keyword:

Customer Segmentation, Business, Data, Cleaning, Analysis, Clustering, Elbow Method, K-Means Algorithm, Visualization.

1.1 Table of Contents

Model Training

Model Evaluation

Visualization

Conclusion

2 Objectives:

After completing this lab you will be able to:

- **Understand** the customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers
- **Apply** customer personality analysis to help business to modify its product based on its target customers from different types of customer segments.
- **Target** Need to perform clustering to summarize customer segments.

3 Introduction:

Nowadays the competition is vast and lot of technologies came into account for effective growth and revenue generation. For every business the most important component is data. With the help of grouped or ungrouped data, we can perform some operations to find customer interests.

Data mining helpful to extract data from the database in a human readable format. But, we may not known the actual beneficiaries in the whole dataset. Customer Segmentation is useful to divide the large data from dataset into several groups based on their age, demographics, spent, income, gender, etc. These groups are also known as clusters. By this, we can get to know that, which product got huge number of sales and which age group are purchasing etc. And, we can supply that product much for better revenue generation.

Initially we are going to take the old data. As we know that old is gold so, by using the old data we are going to apply K-means clustering algorithm and we have to find the number of clusters first. So, at lastly, we have to visualize the data. One can easily find the potential group of data while observing that visualization.

The goal of this notebook is to identify customer segments using the data mining approach, using the partitioning algorithm called as K-means clustering algorithm. The elbow method determines the optimal clusters.

4 Data Understanding:

Context :

Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers.

Customer personality analysis helps a business to modify its product based on its target customers from different types of customer segments. For example, instead of spending money to market a

new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment.

Features :

People * ID: Customer's unique identifier * Year_Birth: Customer's birth year * Education: Customer's education level * Marital_Status: Customer's marital status * Income: Customer's yearly household income * Kidhome: Number of children in customer's household * Teenhome: Number of teenagers in customer's household * Dt_Customer: Date of customer's enrollment with the company * Recency: Number of days since customer's last purchase * Complain: 1 if the customer complained in the last 2 years, 0 otherwise

Products

- MntWines: Amount spent on wine in last 2 years
- MntFruits: Amount spent on fruits in last 2 years
- MntMeatProducts: Amount spent on meat in last 2 years
- MntFishProducts: Amount spent on fish in last 2 years
- MntSweetProducts: Amount spent on sweets in last 2 years
- MntGoldProds: Amount spent on gold in last 2 years

Promotion

- NumDealsPurchases: Number of purchases made with a discount
- AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
- AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

Place

- NumWebPurchases: Number of purchases made through the company's website
- NumCatalogPurchases: Number of purchases made using a catalogue
- NumStorePurchases: Number of purchases made directly in stores
- NumWebVisitsMonth: Number of visits to company's website in the last month

5 Data Preparation:

In this Section: * We will Install and Import the required Libraries. * We will load the data. * We will see few basic information about our data.

5.0.1 Required Libraries:

[1]:
```
#Make Sure to install libraries before importing
#Importing the Libraries

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.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

#Supressing the Warnings
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)
```

5.0.2 Data Loading:

[2]:
```
df = pd.read_csv('data/marketing_campaign.csv',  sep="\t")
df.head(3)
```

[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	\
0	5524	1957	Graduation	Single	58138.0	0	0	
1	2174	1954	Graduation	Single	46344.0	1	1	
2	4141	1965	Graduation	Together	71613.0	0	0	

	Dt_Customer	Recency	MntWines	…	NumWebVisitsMonth	AcceptedCmp3	\
0	04-09-2012	58	635	…	7	0	
1	08-03-2014	38	11	…	5	0	
2	21-08-2013	26	426	…	4	0	

	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	

	Z_CostContact	Z_Revenue	Response
0	3	11	1
1	3	11	0
2	3	11	0

```
[3 rows x 29 columns]
```

5.0.3 Brief Data Information:

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

[4]:
```
print("The total number of data-points after removing the rows without missing␣
↪values are:", len(df))
```

```
The total number of data-points after removing the rows without missing values
```

```
are: 2240
```

[5]:
```
#Missing Values
df.isna().sum()
```

[5]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

From the above output, we can conclude and note that:

- There are missing values in income
- `Dt_Customer` that indicates the date a customer joined the database is not parsed as `DateTime`
- There are some categorical features in our data frame; as there are some features in`dtype: object`). So we will need to encode them into numeric forms later.

6 Data Preprocessing:

In this Section: * We will clean our data. * We will do feature Engineering. * We will do Label encoding fro the categorical features. * We will scale the features using the standard scaler * We Will Create a subset dataframe for dimensionality reduction

6.0.1 Data Cleaning:

First of all, for the missing values, I am simply going to drop the rows that have missing income values.

[6]:
```
#remove or drop the NA values
df = df.dropna()
print("The total number of data-points after removing the rows with missing␣
 ↪values are:", len(df))
```

```
The total number of data-points after removing the rows with missing values are:
2216
```

In the next step, I am going to create a feature out of `Dt_Customer` that indicates the number of days a customer is registered in the firm's database. However, in order to keep it simple, I am taking this value relative to the most recent customer in the record.

Thus to get the values I must check the newest and oldest recorded dates.

[7]:
```
df["Dt_Customer"] = pd.to_datetime(df["Dt_Customer"])
dates = []
for i in df["Dt_Customer"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("The newest customer's enrolment date in therecords:",max(dates))
print("The oldest customer's enrolment date in the records:",min(dates))
```

```
The newest customer's enrolment date in therecords: 2014-12-06
The oldest customer's enrolment date in the records: 2012-01-08
```

Creating a feature (`Customer_For`) of the number of days the customers started to shop in the store relative to the last recorded date

[8]:
```
#Created a feature "Customer_For"
days = []
d1 = max(dates) #taking it to be the newest customer
for i in dates:
    delta = d1 - i
    days.append(delta)
df["Customer_For"] = days
df["Customer_For"] = pd.to_numeric(df["Customer_For"], errors="coerce")
```

Now we will be exploring the unique values in the categorical features to get a clear idea of the data.

[9]:
```
print("Total categories in the feature Marital_Status:\n", df["Marital_Status"].
 ↪value_counts(), "\n")
print("Total categories in the feature Education:\n", df["Education"].
 ↪value_counts())
```

```
Total categories in the feature Marital_Status:
 Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64

Total categories in the feature Education:
 Graduation    1116
PhD             481
Master          365
2n Cycle        200
Basic            54
Name: Education, dtype: int64
```

6.0.2 Feature Engineering:

In this section, we will be performing the following steps to engineer some new features:

- Extract the `Age` of a customer by the `Year_Birth` indicating the birth year of the respective person.
- Create another feature `Spent` indicating the total amount spent by the customer in various categories over the span of two years.
- Create another feature `Living_With` out of `Marital_Status` to extract the living situation of couples.
- Create a feature `Children` to indicate total children in a household that is, kids and teenagers.
- To get further clarity of household, Creating feature indicating `Family_Size`.
- Create a feature `Is_Parent` to indicate parenthood status
- Lastly, we will create three categories in the `Education` by simplifying its value counts and dropping some of the redundant features

[10]:
```
#Feature Engineering
#Age of customer today
df["Age"] = 2021-df["Year_Birth"]

#Total spendings on various items
df["Spent"] = df["MntWines"]+ df["MntFruits"]+ df["MntMeatProducts"]+␣
 ↪df["MntFishProducts"]+ df["MntSweetProducts"]+ df["MntGoldProds"]

#Deriving living situation by marital status"Alone"
df["Living_With"]=df["Marital_Status"].replace({"Married":"Partner", "Together":
 ↪"Partner", "Absurd":"Alone", "Widow":"Alone", "YOLO":"Alone", "Divorced":
 ↪"Alone", "Single":"Alone",})
```

```
#Feature indicating total children living in the household
df["Children"]=df["Kidhome"]+df["Teenhome"]

#Feature for total members in the householde
df["Family_Size"] = df["Living_With"].replace({"Alone": 1, "Partner":2})+␣
 ↪df["Children"]

#Feature pertaining parenthood
df["Is_Parent"] = np.where(df.Children> 0, 1, 0)

#Segmenting education levels in three groups
df["Education"]=df["Education"].replace({"Basic":"Undergraduate","2n Cycle":
 ↪"Undergraduate", "Graduation":"Graduate", "Master":"Postgraduate", "PhD":
 ↪"Postgraduate"})

#For clarity
df=df.rename(columns={"MntWines": "Wines","MntFruits":
 ↪"Fruits","MntMeatProducts":"Meat","MntFishProducts":
 ↪"Fish","MntSweetProducts":"Sweets","MntGoldProds":"Gold"})

#Dropping some of the redundant features
to_drop = ["Marital_Status", "Dt_Customer", "Z_CostContact", "Z_Revenue",␣
 ↪"Year_Birth", "ID"]
df = df.drop(to_drop, axis=1)
```

Now that we have some new features let's have a look at the data's stats.

[11]:
```
df.describe()
```

[11]:

	Income	Kidhome	Teenhome	Recency	Wines	\
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	
mean	52247.251354	0.441787	0.505415	49.012635	305.091606	
std	25173.076661	0.536896	0.544181	28.948352	337.327920	
min	1730.000000	0.000000	0.000000	0.000000	0.000000	
25%	35303.000000	0.000000	0.000000	24.000000	24.000000	
50%	51381.500000	0.000000	0.000000	49.000000	174.500000	
75%	68522.000000	1.000000	1.000000	74.000000	505.000000	
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	

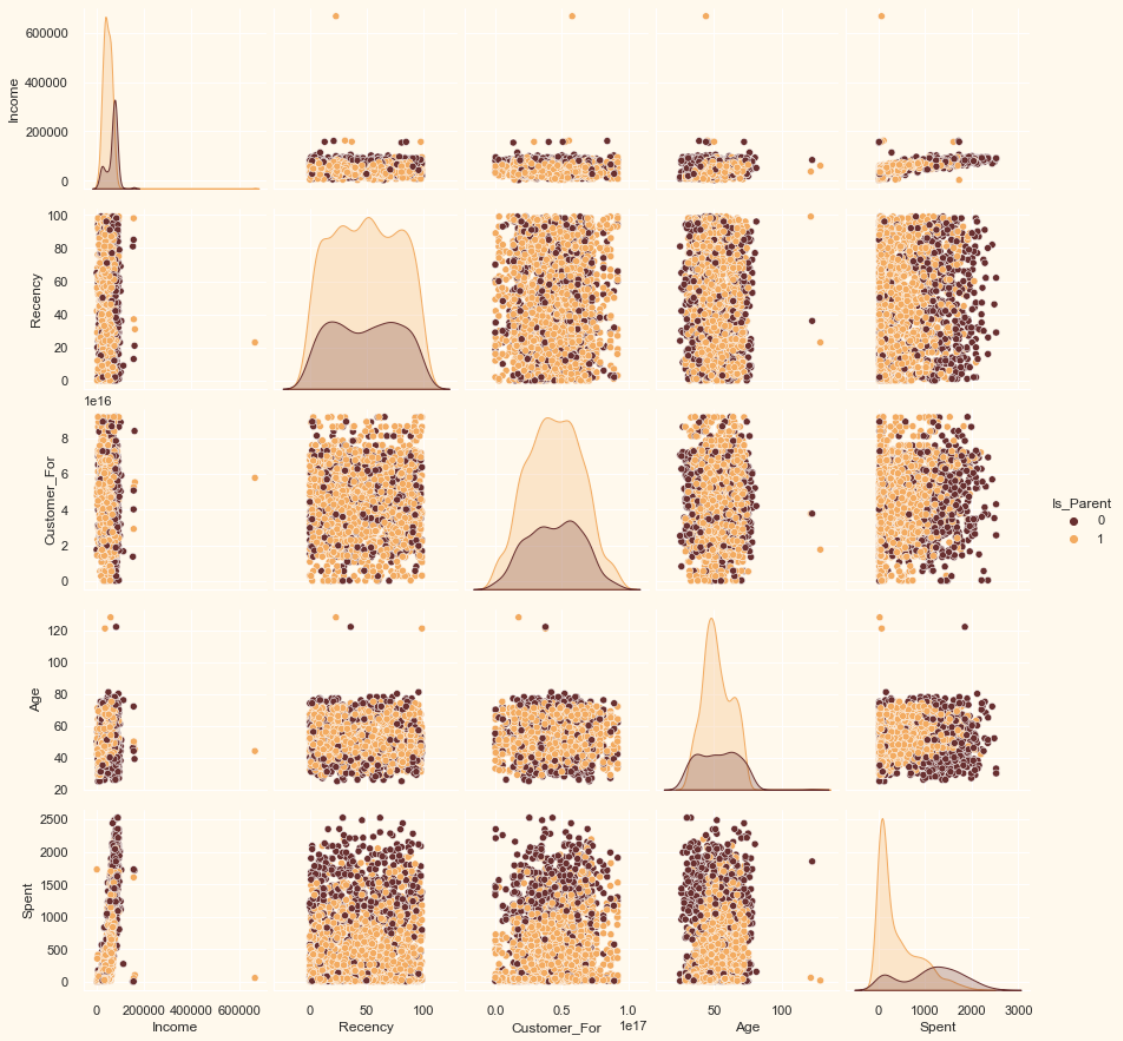
	Fruits	Meat	Fish	Sweets	Gold	…	\
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	…	
mean	26.356047	166.995939	37.637635	27.028881	43.965253	…	
std	39.793917	224.283273	54.752082	41.072046	51.815414	…	
min	0.000000	0.000000	0.000000	0.000000	0.000000	…	
25%	2.000000	16.000000	3.000000	1.000000	9.000000	…	
50%	8.000000	68.000000	12.000000	8.000000	24.500000	…	
75%	33.000000	232.250000	50.000000	33.000000	56.000000	…	

```
max     199.000000  1725.000000   259.000000   262.000000   321.000000  …

       AcceptedCmp1  AcceptedCmp2     Complain     Response  Customer_For  \
count   2216.000000   2216.000000  2216.000000  2216.000000  2.216000e+03
mean       0.064079      0.013538     0.009477     0.150271  4.423735e+16
std        0.244950      0.115588     0.096907     0.357417  2.008532e+16
min        0.000000      0.000000     0.000000     0.000000  0.000000e+00
25%        0.000000      0.000000     0.000000     0.000000  2.937600e+16
50%        0.000000      0.000000     0.000000     0.000000  4.432320e+16
75%        0.000000      0.000000     0.000000     0.000000  5.927040e+16
max        1.000000      1.000000     1.000000     1.000000  9.184320e+16

                Age        Spent     Children  Family_Size    Is_Parent
count  2216.000000  2216.000000  2216.000000  2216.000000  2216.000000
mean     52.179603   607.075361     0.947202     2.592509     0.714350
std      11.985554   602.900476     0.749062     0.905722     0.451825
min      25.000000     5.000000     0.000000     1.000000     0.000000
25%      44.000000    69.000000     0.000000     2.000000     0.000000
50%      51.000000   396.500000     1.000000     3.000000     1.000000
75%      62.000000  1048.000000     1.000000     3.000000     1.000000
max     128.000000  2525.000000     3.000000     5.000000     1.000000

[8 rows x 28 columns]
```

- The above stats show some discrepancies in mean Income and Age and max Income and age.
- Do note that, max-age is 128 years, As we calculated the age that would be today (i.e. 2021) and the data is old.
- we must take a look at the broader view of the data. we will plot some of the selected features.

[12]:
```
#To plot some selected features
#Setting up colors prefrences
sns.set(rc={"axes.facecolor":"#FFF9ED","figure.facecolor":"#FFF9ED"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9",␣
 ↪"#9F8A78", "#F3AB60"])
#Plotting following features
To_Plot = [ "Income", "Recency", "Customer_For", "Age", "Spent", "Is_Parent"]
print("Reletive Plot Of Some Selected Features: A Data Subset")
plt.figure()
sns.pairplot(df[To_Plot], hue= "Is_Parent",palette= (["#682F2F","#F3AB60"]))
#Taking hue
plt.show()
```

```
Reletive Plot Of Some Selected Features: A Data Subset
<Figure size 576x396 with 0 Axes>
```



Clearly, there are a few outliers in the Income and Age features. I will be deleting the outliers in the data.

[13]:
```
#Dropping the outliers by setting a cap on Age and income.
df = df[(df["Age"]<90)]
df = df[(df["Income"]<600000)]
print("The total number of data-points after removing the outliers are:",␣
 ↪len(df))
```

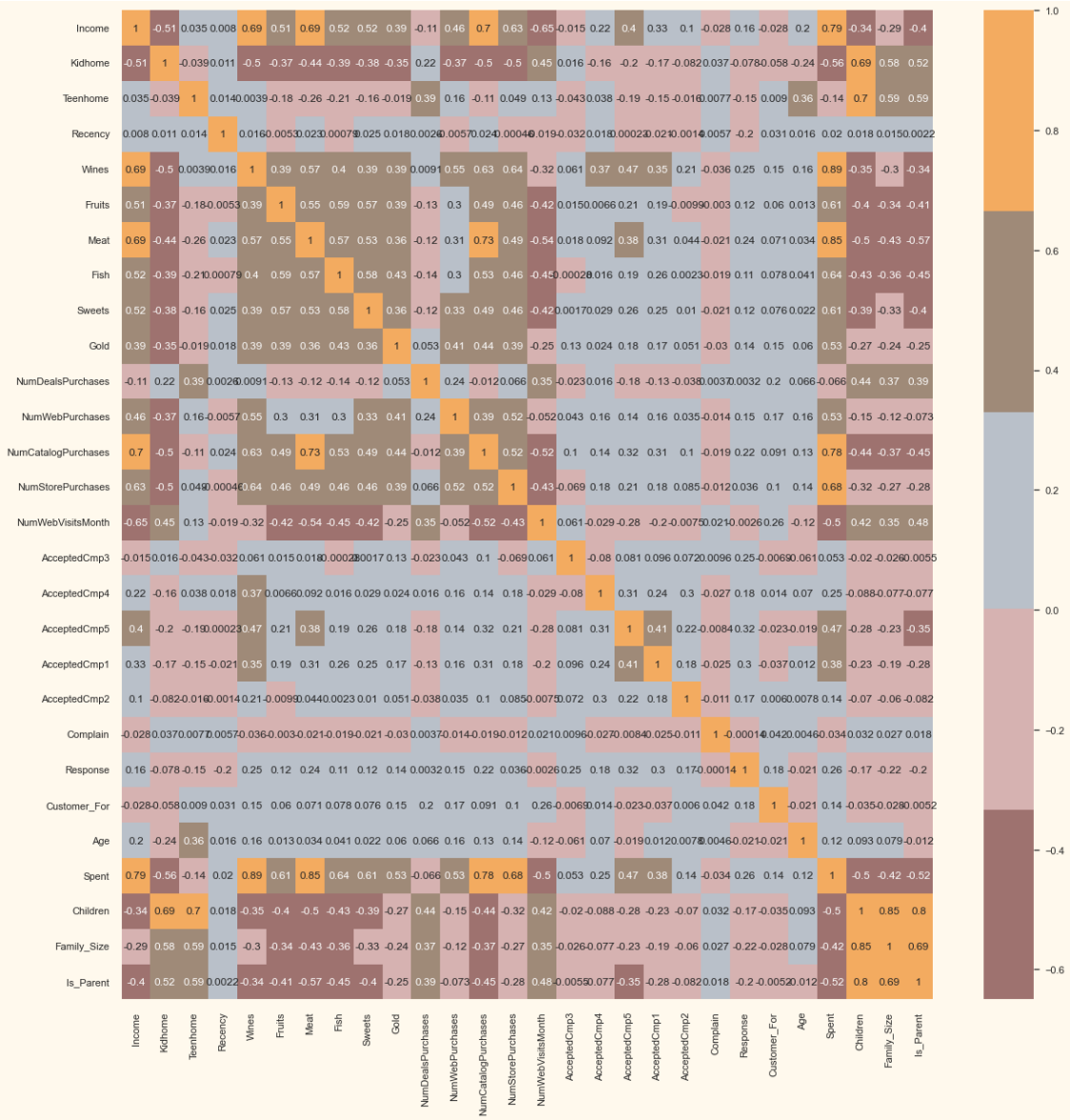
```
The total number of data-points after removing the outliers are: 2212
```

Next, let us look at the correlation amongst the features. (Excluding the categorical attributes at this point.)

[14]:
```
#correlation matrix
corrmat= df.corr()
```

```
plt.figure(figsize=(20,20))
sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```

[14]: <AxesSubplot:>



The data is quite clean and the new features have been included. I will proceed to the next step. That is, preprocessing the data.

6.0.3 Encoding and Scaling

In this section, we will be preprocessing the data to perform clustering operations.

- Label encoding the categorical features

- Scaling the features using the standard scaler
- Creating a subset dataframe for dimensionality reduction

[15]:
```
#Get list of categorical variables
s = (df.dtypes == 'object')
object_cols = list(s[s].index)

print("Categorical variables in the dataset:", object_cols)
```

```
Categorical variables in the dataset: ['Education', 'Living_With']
```

[16]:
```
#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
    df[i]=df[[i]].apply(LE.fit_transform)

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

```
All features are now in numerical.
```

[17]:
```
#Creating a copy of data
df_copy = df.copy()
# creating a subset of dataframe by dropping the features on deals accepted and␣
 ↪promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',␣
 ↪'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
df_copy = df_copy.drop(cols_del, axis=1)
#Scaling
scaler = StandardScaler()
scaler.fit(df_copy)
scaled_df = pd.DataFrame(scaler.transform(df_copy),columns= df_copy.columns )
print("All features are now scaled")
```

```
All features are now scaled
```

[18]:
```
#Scaled data to be used for reducing the dimensionality
print("Dataframe to be used for further modelling process:")
scaled_df.head()
```

```
Dataframe to be used for further modelling process:
```

[18]:

	Education	Income	Kidhome	Teenhome	Recency	Wines	Fruits	\
0	-0.893586	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041	
1	-0.893586	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461	
2	-0.893586	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540	
3	-0.893586	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961	
4	0.571657	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540	

	Meat	Fish	Sweets	…	NumCatalogPurchases	NumStorePurchases	\

```
0  1.690293  2.453472  1.483713  …             2.503607          -0.555814
1 -0.718230 -0.651004 -0.634019  …            -0.571340          -1.171160
2 -0.178542  1.339513 -0.147184  …            -0.229679           1.290224
3 -0.655787 -0.504911 -0.585335  …            -0.913000          -0.555814
4 -0.218684  0.152508 -0.001133  …             0.111982           0.059532

   NumWebVisitsMonth  Customer_For       Age     Spent  Living_With  Children  \
0           0.692181      1.973583  1.018352  1.676245    -1.349603 -1.264598
1          -0.132545     -1.665144  1.274785 -0.963297    -1.349603  1.404572
2          -0.544908     -0.172664  0.334530  0.280110     0.740959 -1.264598
3           0.279818     -1.923210 -1.289547 -0.920135     0.740959  0.069987
4          -0.132545     -0.822130 -1.033114 -0.307562     0.740959  0.069987

   Family_Size  Is_Parent
0    -1.758359  -1.581139
1     0.449070   0.632456
2    -0.654644  -1.581139
3     0.449070   0.632456
4     0.449070   0.632456

[5 rows x 23 columns]
```

7 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 we will be performing dimensionality reduction 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.

Steps to follow in this section:

- Dimensionality reduction with PCA
- Plotting the reduced dataframe
- Dimensionality reduction with PCA

For this project, we will be reducing the dimensions to 3.

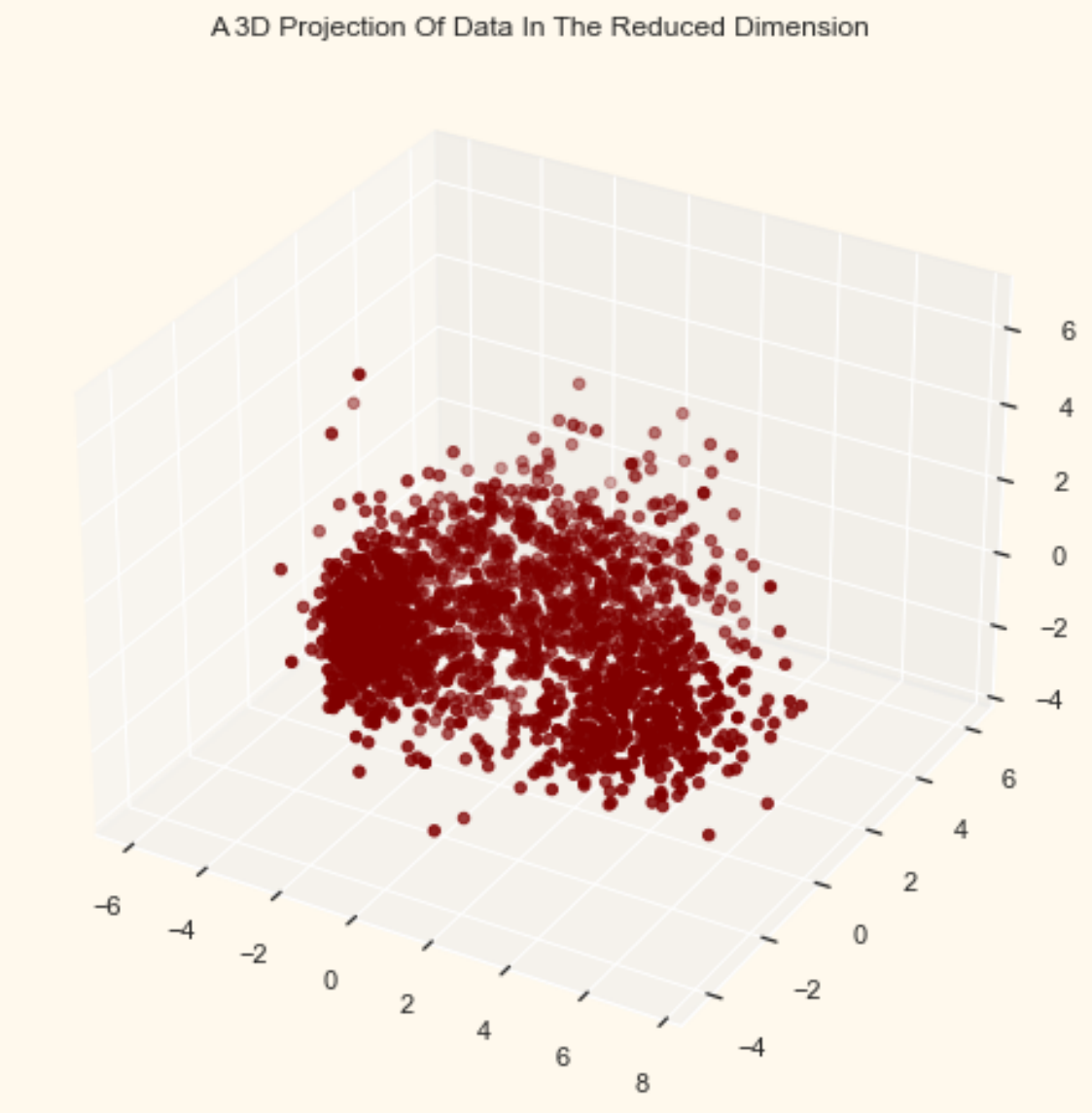
[19]:
```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(scaled_df)
PCA_ds = pd.DataFrame(pca.transform(scaled_df), columns=(["col1","col2",␣
 ↪"col3"]))
PCA_ds.describe().T
```

[19]:

	count	mean	std	min	25%	50%	75%	\
col1	2212.0	-7.970317e-17	2.878377	-5.969394	-2.538494	-0.780421	2.383290	
col2	2212.0	-1.276857e-16	1.706839	-4.312196	-1.328316	-0.158123	1.242289	
col3	2212.0	3.488269e-17	1.221956	-3.530416	-0.829067	-0.022692	0.799895	

	max
col1	7.444305
col2	6.142721
col3	6.611222

[20]:
```
#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()
```



[ ]:

8 Clustering:

Now that we have reduced the attributes to three dimensions, we will perform clustering via Agglomerative clustering. Agglomerative clustering is a hierarchical clustering method. It involves merging examples until the desired number of clusters is achieved.

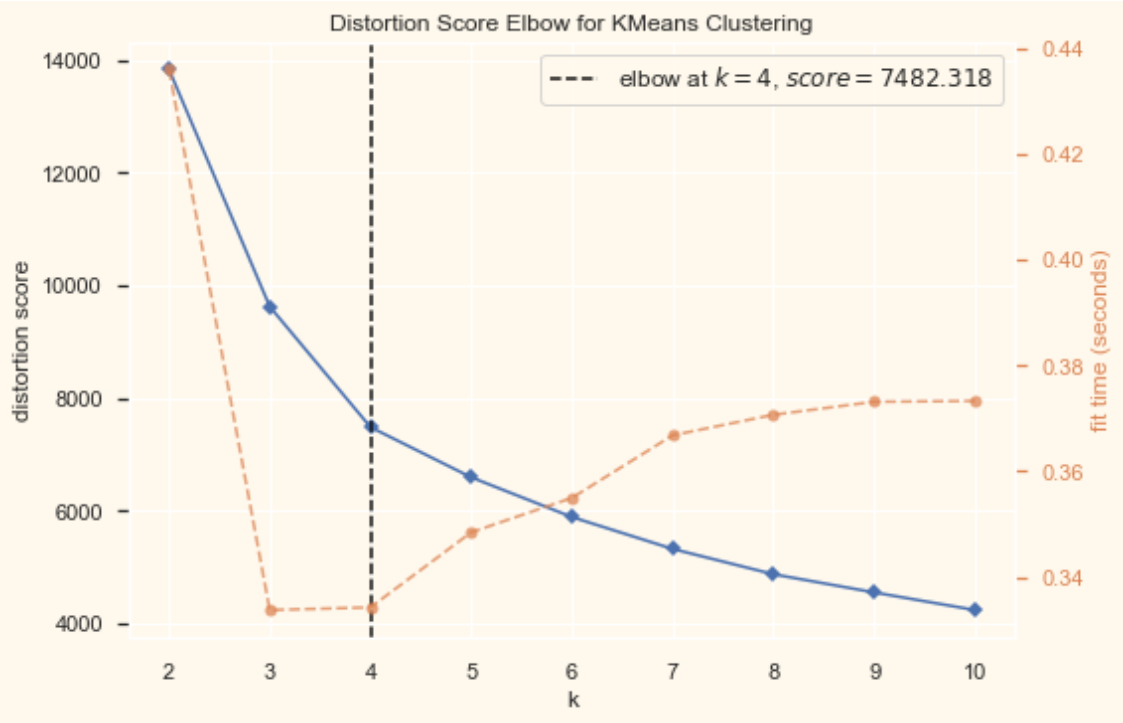
Steps involved in the Clustering

- Elbow Method to determine the number of clusters to be formed
- Clustering via Agglomerative Clustering
- Examining the clusters formed via scatter plot

8.0.1 Model Training:

[21]:
```
# 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:
```



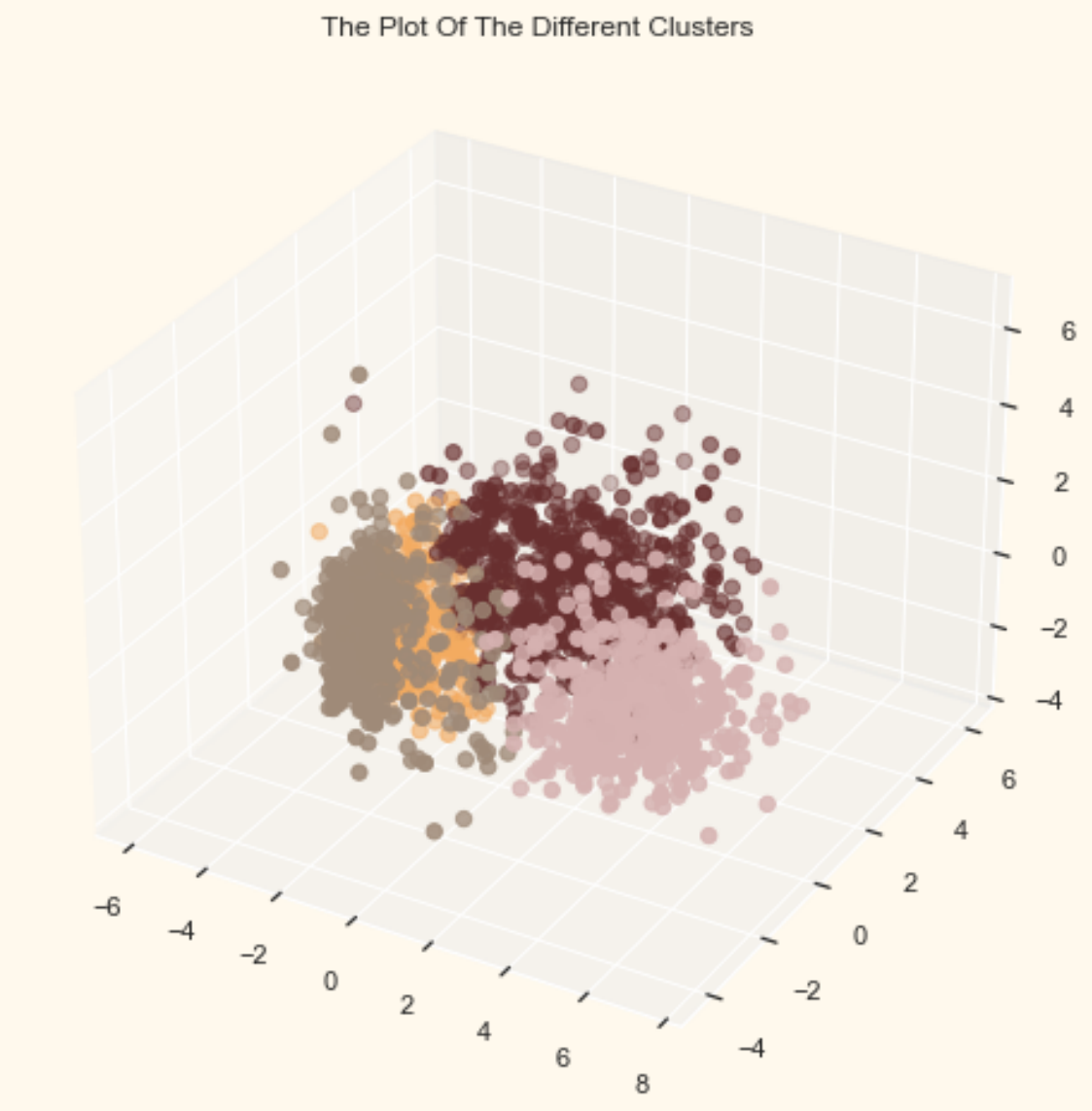
[21]:
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'},
xlabel='k', ylabel='distortion score'>
```

The above cell indicates that four will be an optimal number of clusters for this data. Next, we will be fitting the Agglomerative Clustering Model to get the final clusters.

[22]:
```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
df["Clusters"]= yhat_AC
```

To examine the clusters formed let's have a look at the 3-D distribution of the clusters.

[23]:
```
#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=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Different Clusters")
plt.show()
```



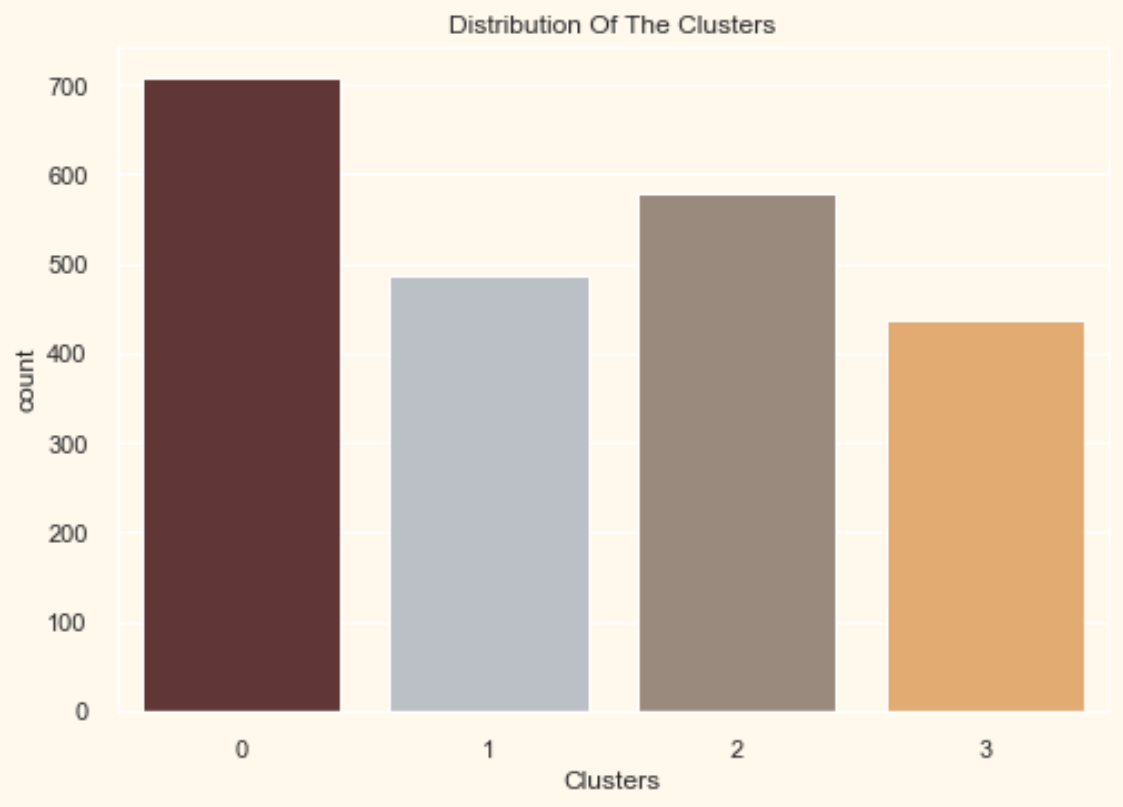
8.0.2 Model Evaluation:

Since this is an unsupervised clustering. We do not have a tagged feature to evaluate or score our model. The purpose of this section is to study the 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.

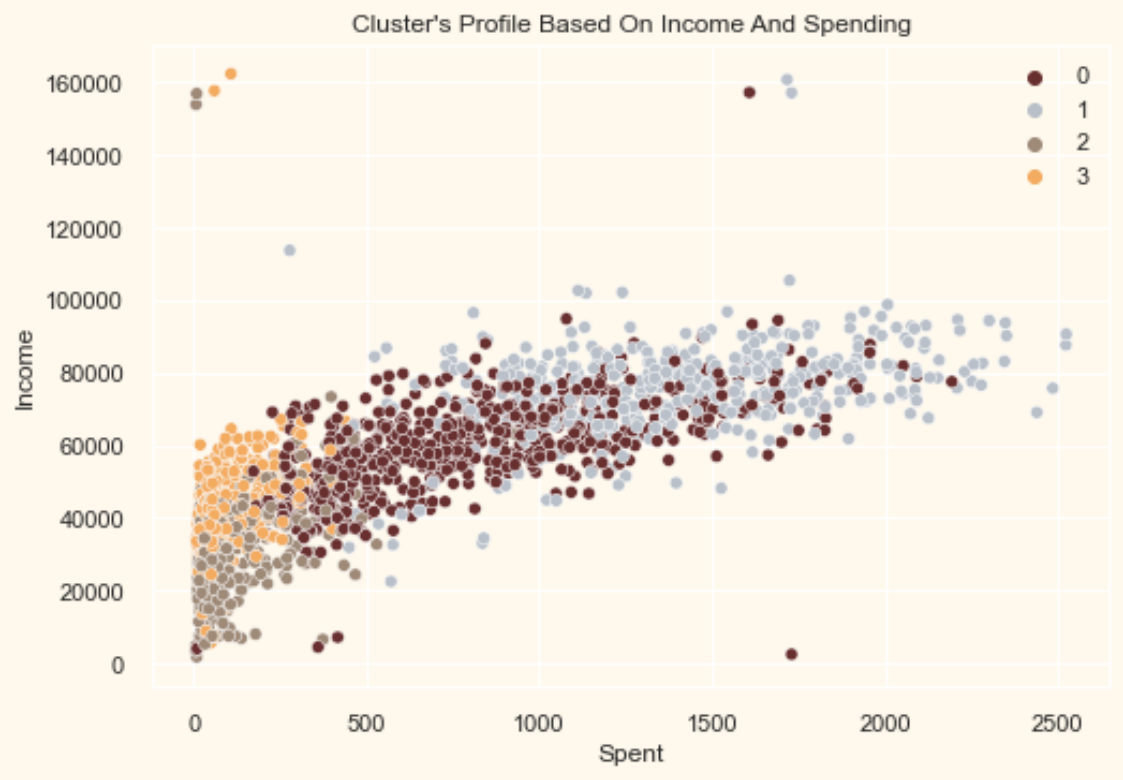
Firstly, let us have a look at the group distribution of clustring

[24]:
```
#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=df["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()
```



The clusters seem to be fairly distributed.

[25]:
```
pl = sns.scatterplot(data = df,x=df["Spent"],␣
↪y=df["Income"],hue=df["Clusters"], palette= pal)
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()
```

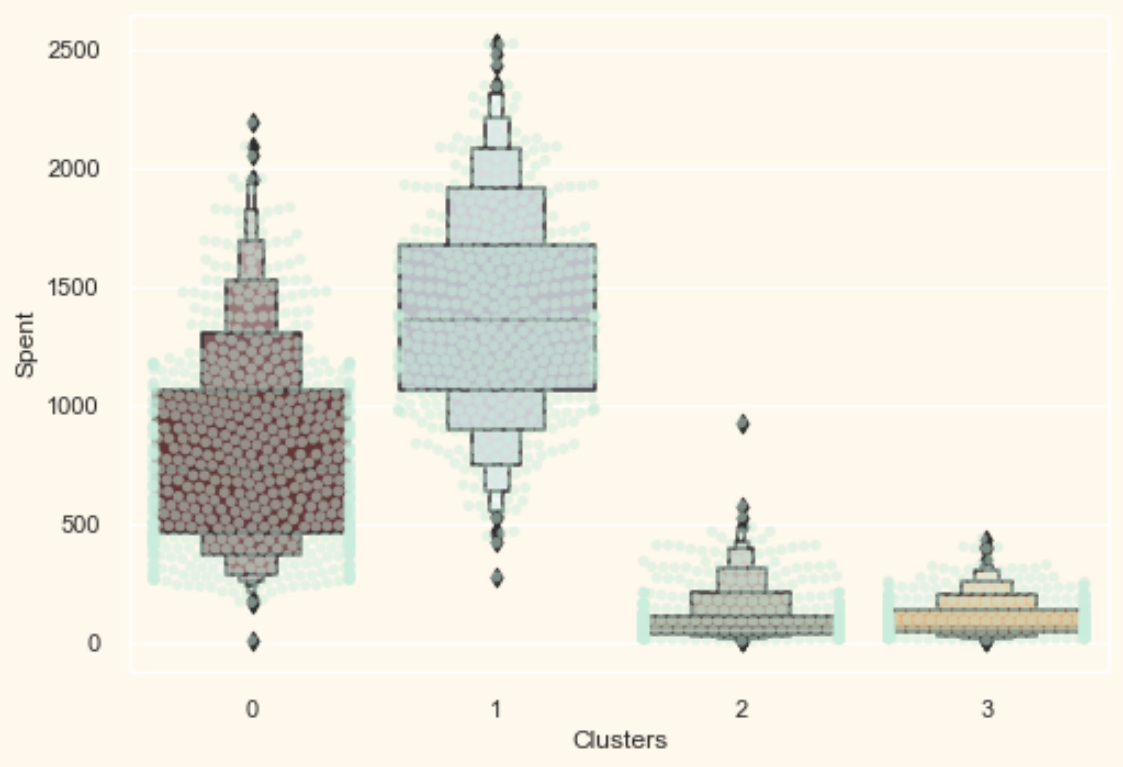


Income vs spending plot shows the clusters pattern

- group 0: high spending & average income
- group 1: high spending & high income
- group 2: low spending & low income
- group 3: high spending & low income.

Next, we will be looking at the detailed distribution of clusters as per the various products in the data. Namely: Wines, Fruits, Meat, Fish, Sweets and Gold

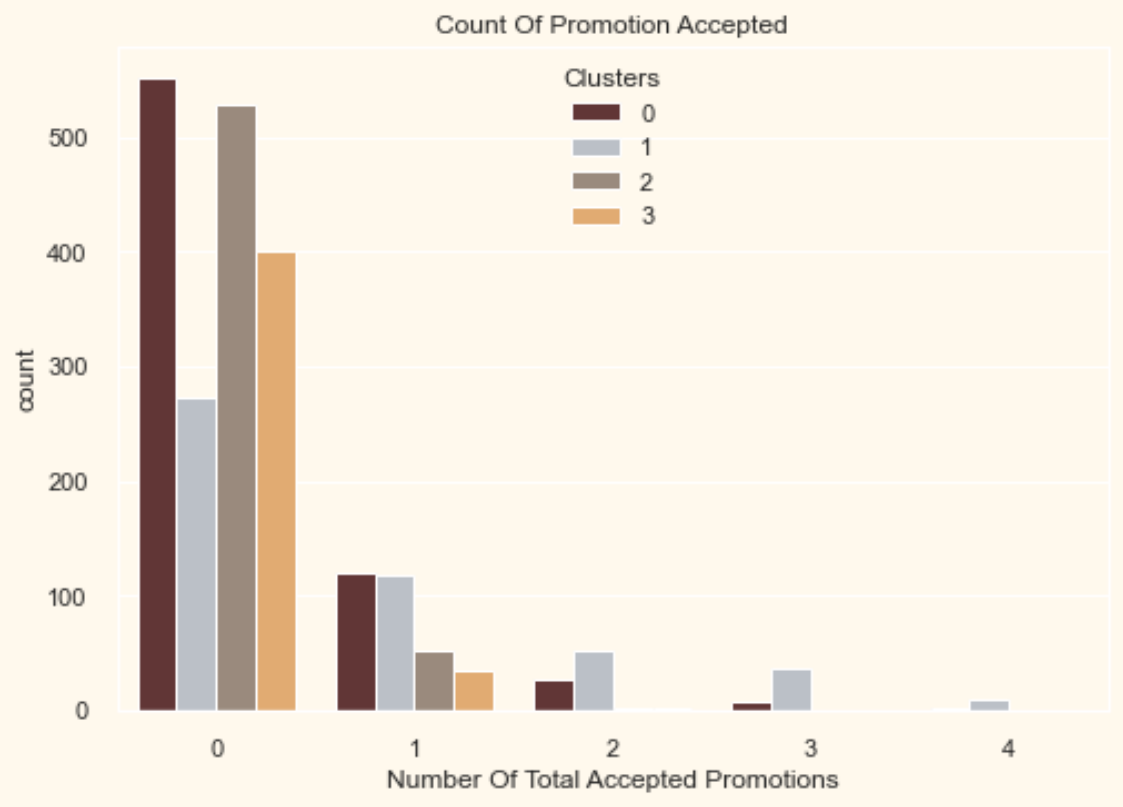
[26]:
```
plt.figure()
pl=sns.swarmplot(x=df["Clusters"], y=df["Spent"], color= "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=df["Clusters"], y=df["Spent"], palette=pal)
plt.show()
```



From the above plot, it can be clearly seen that cluster 1 is our biggest set of customers closely followed by cluster 0. We can explore what each cluster is spending on for the targeted marketing strategies.

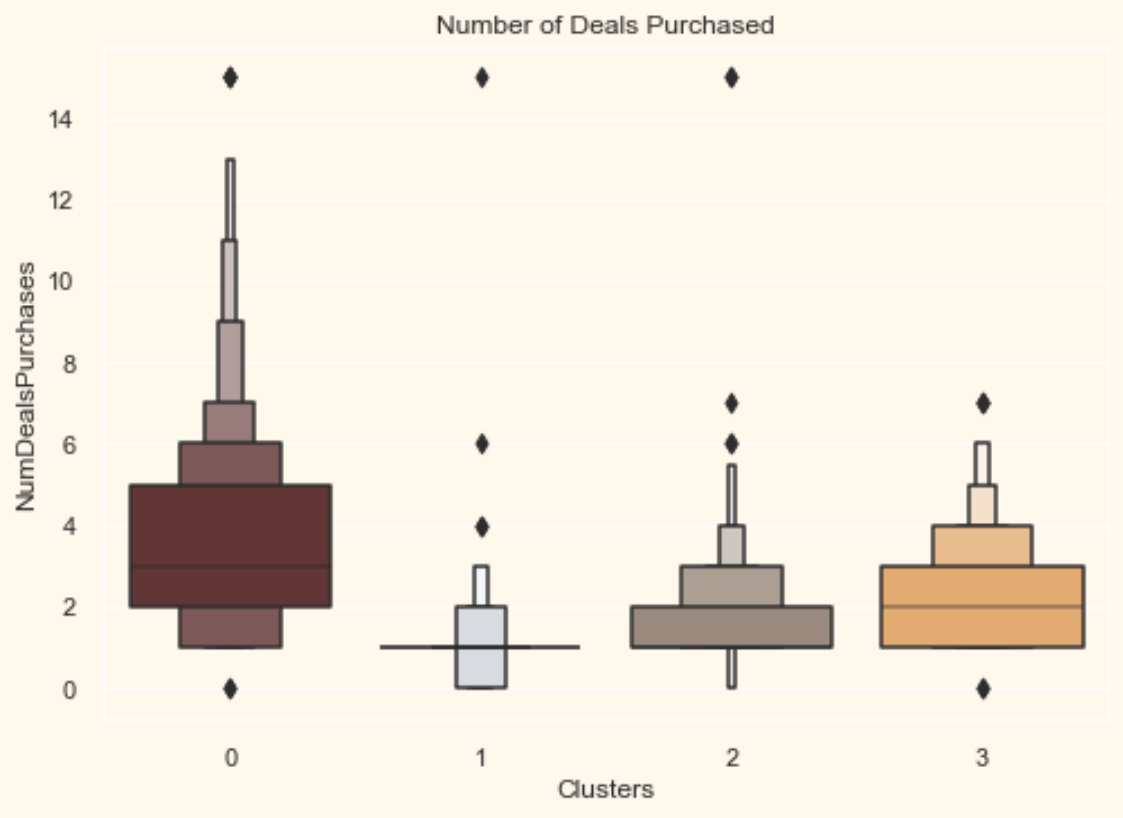
Let us next explore how did our campaigns do in the past.

[27]:
```
#Creating a feature to get a sum of accepted promotions
df["Total_Promos"] = df["AcceptedCmp1"]+ df["AcceptedCmp2"]+␣
 ↪df["AcceptedCmp3"]+ df["AcceptedCmp4"]+ df["AcceptedCmp5"]
#Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=df["Total_Promos"],hue=df["Clusters"], palette= pal)
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.show()
```



There has not been an overwhelming response to the campaigns so far. Very few participants overall. Moreover, no one part take in all 5 of them. Perhaps better-targeted and well-planned campaigns are required to boost sales.

[28]:
```
#Plotting the number of deals purchased
plt.figure()
pl=sns.boxenplot(y=df["NumDealsPurchases"],x=df["Clusters"], palette= pal)
pl.set_title("Number of Deals Purchased")
plt.show()
```



Unlike campaigns, the deals offered did well. It has best outcome with cluster 0 and cluster 3. However, our star customers cluster 1 are not much into the deals. Nothing seems to attract cluster 2 overwhelmingly.

9 Visualization:

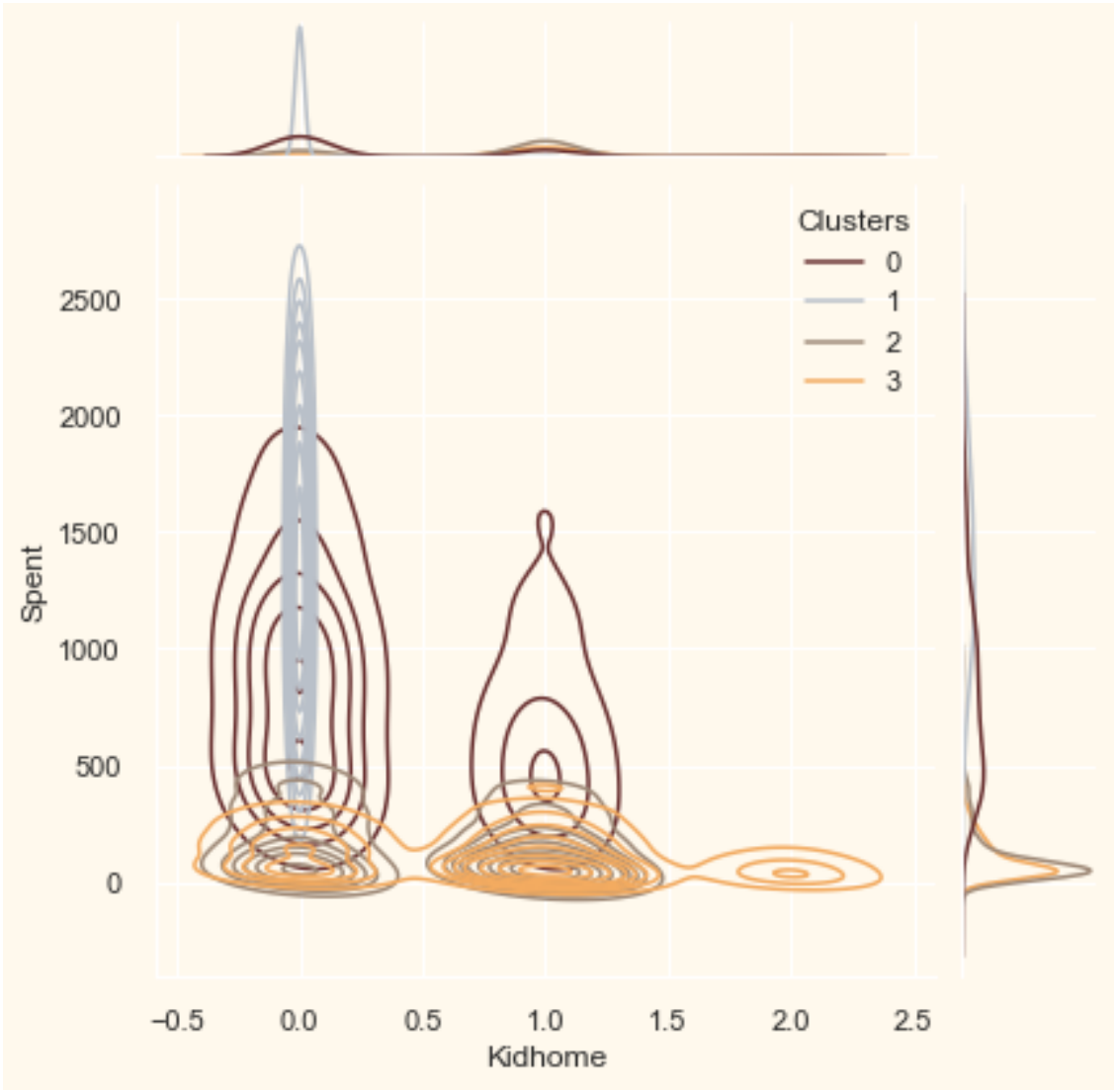
Now that we have formed the clusters and looked at their purchasing habits. Let us see who all are there in these clusters. For that, we will be profiling the clusters formed and come to a conclusion about who is our star customer and who needs more attention from the retail store's marketing team.

To decide that I will be plotting some of the features that are indicative of the customer's personal traits in light of the cluster they are in. On the basis of the outcomes, I will be arriving at the conclusions.

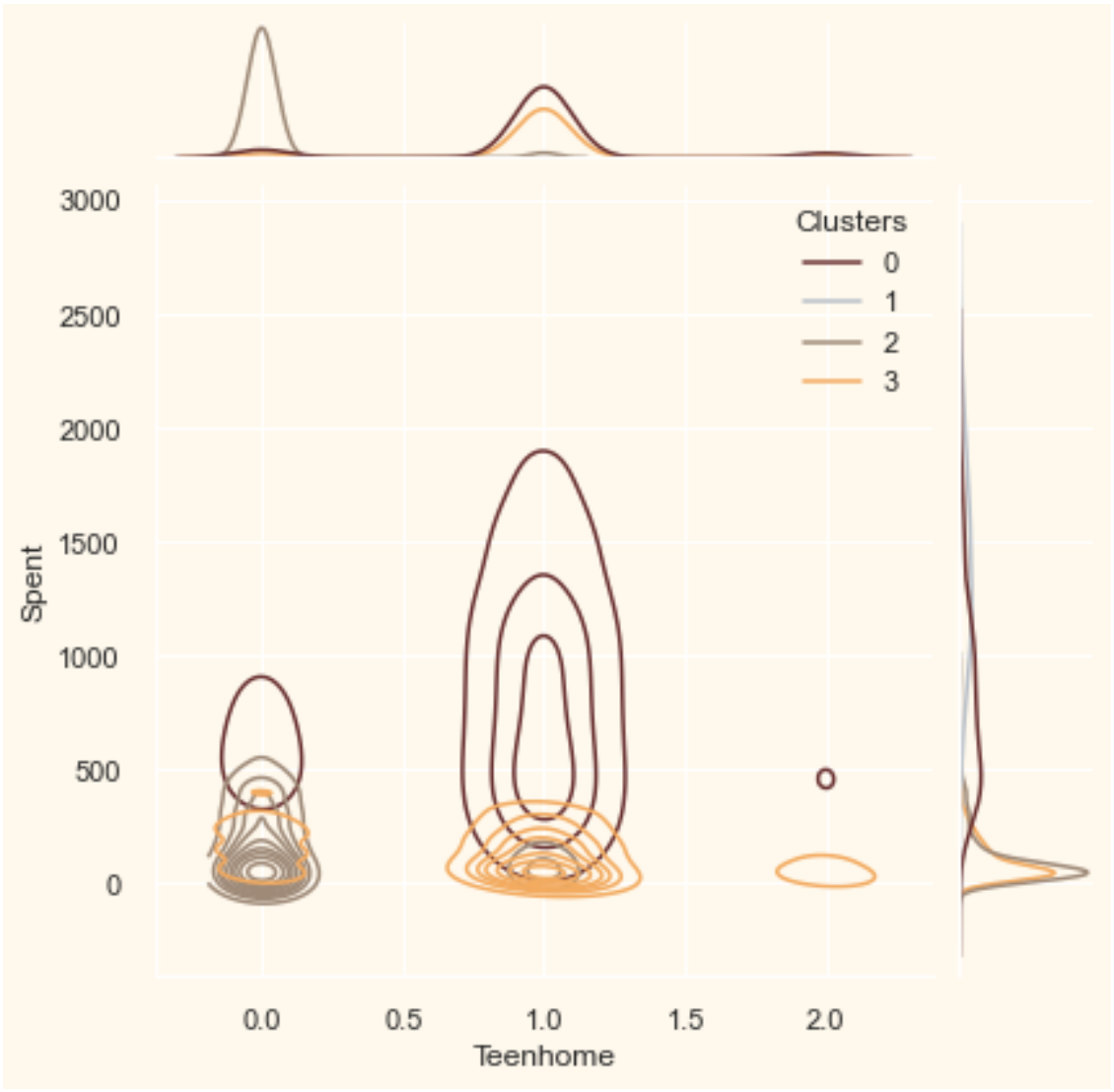
[29]:
```
Personal = [ "Kidhome","Teenhome","Customer_For", "Age", "Children",␣
 ↪"Family_Size", "Is_Parent", "Education","Living_With"]

for i in Personal:
    plt.figure()
    sns.jointplot(x=df[i], y=df["Spent"], hue =df["Clusters"], kind="kde",␣
 ↪palette=pal)
```

```
    plt.show()
```

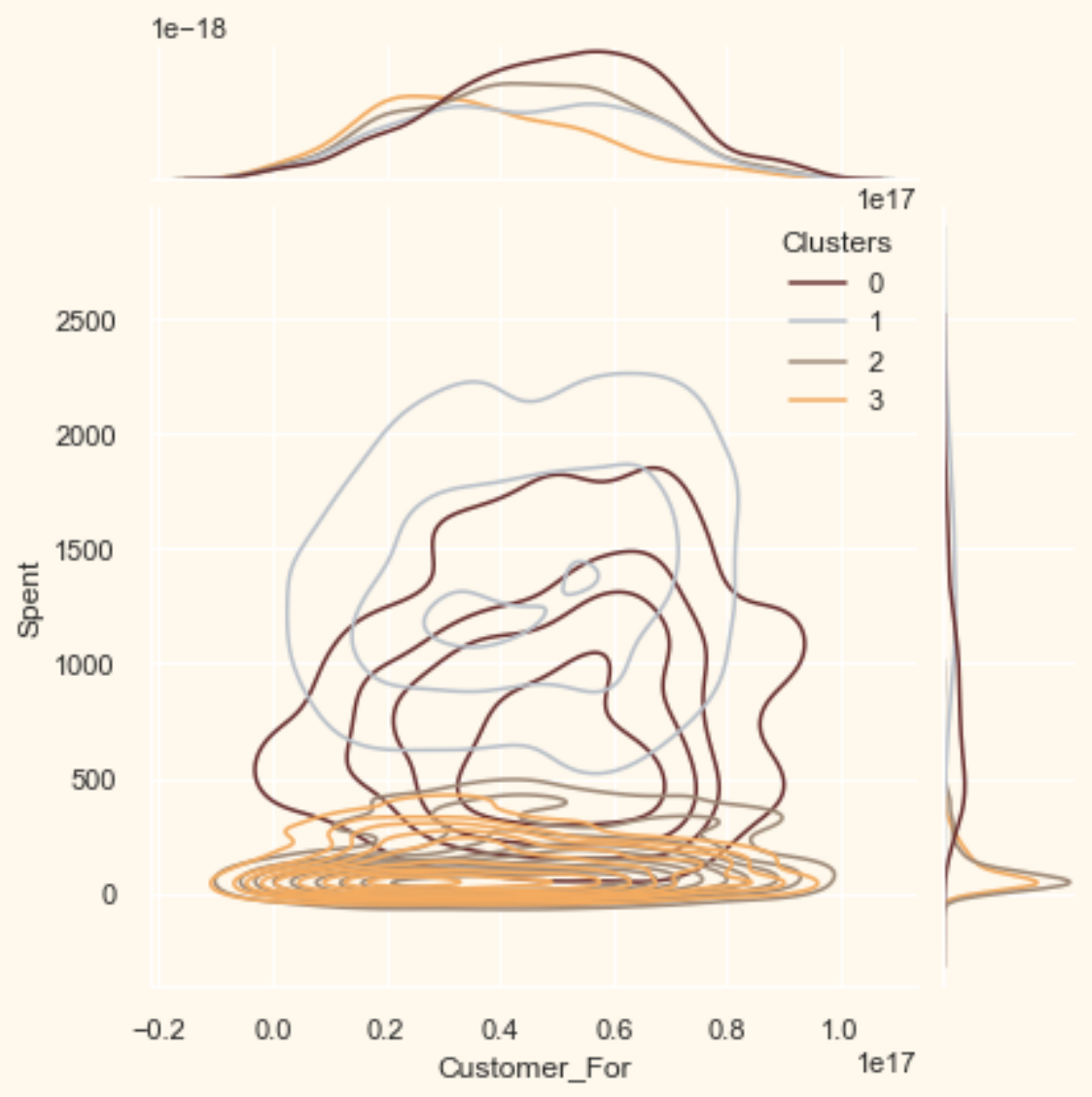
```
<Figure size 576x396 with 0 Axes>
```



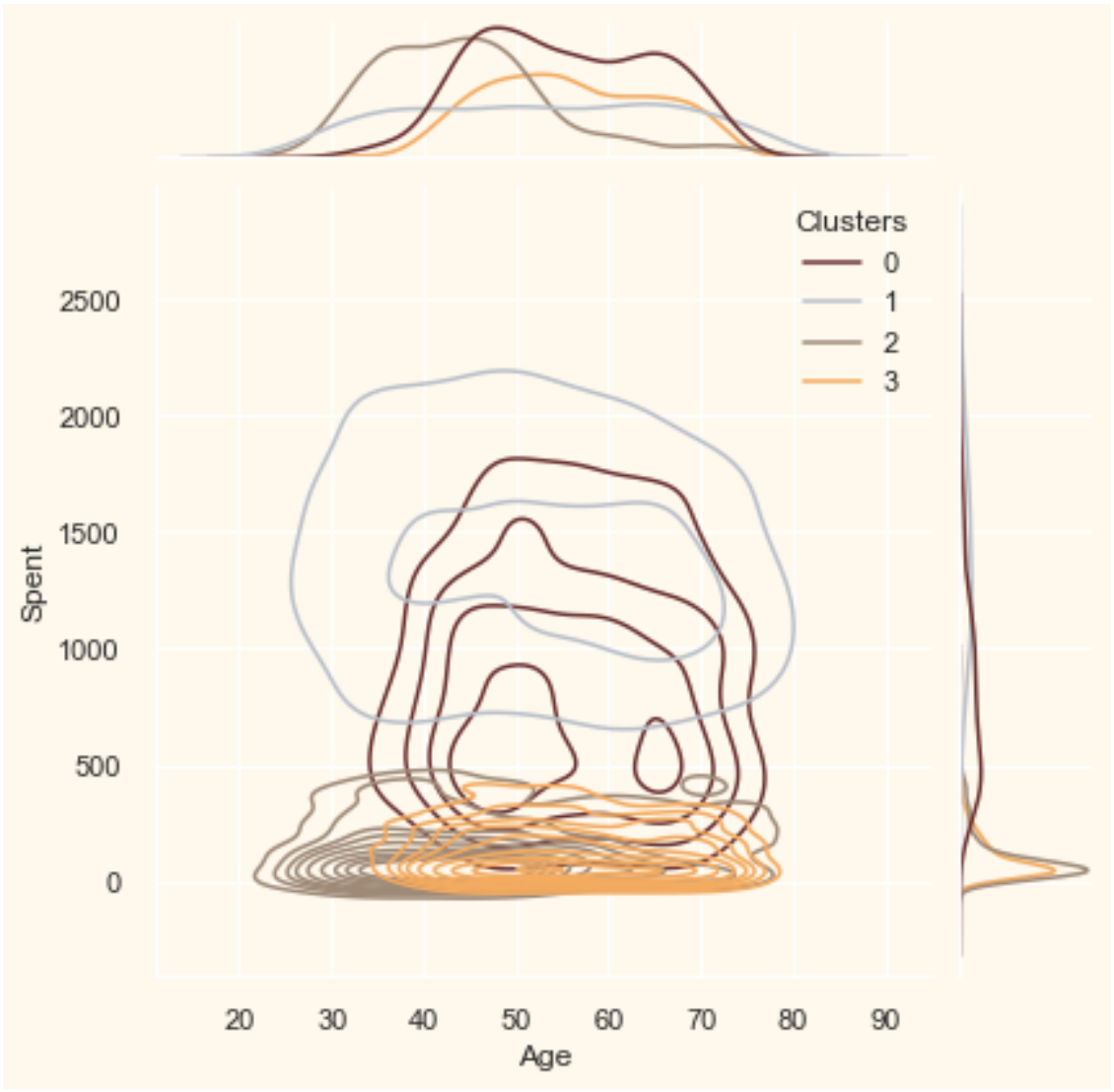

```
<Figure size 576x396 with 0 Axes>
```



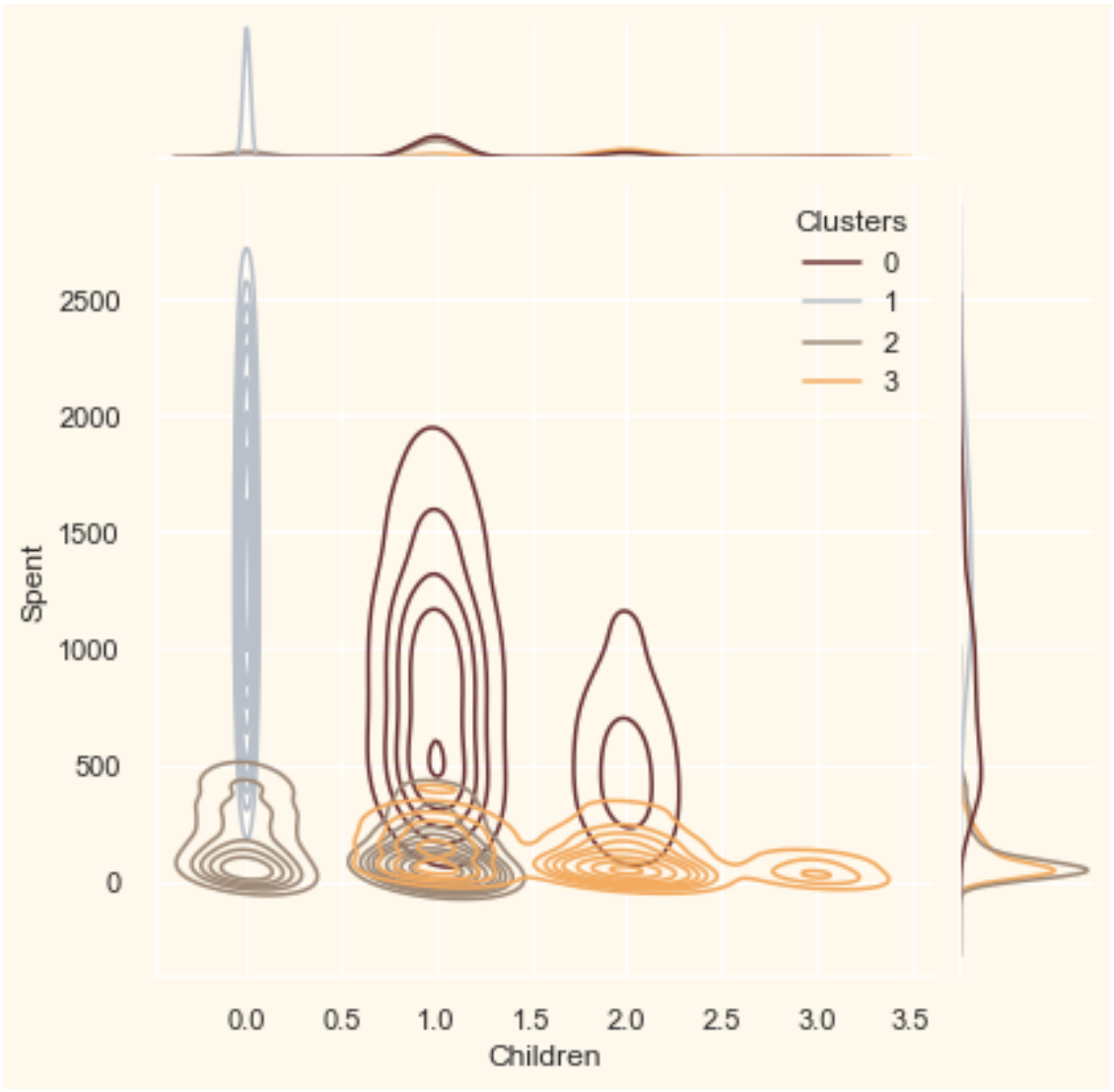
<Figure size 576x396 with 0 Axes>



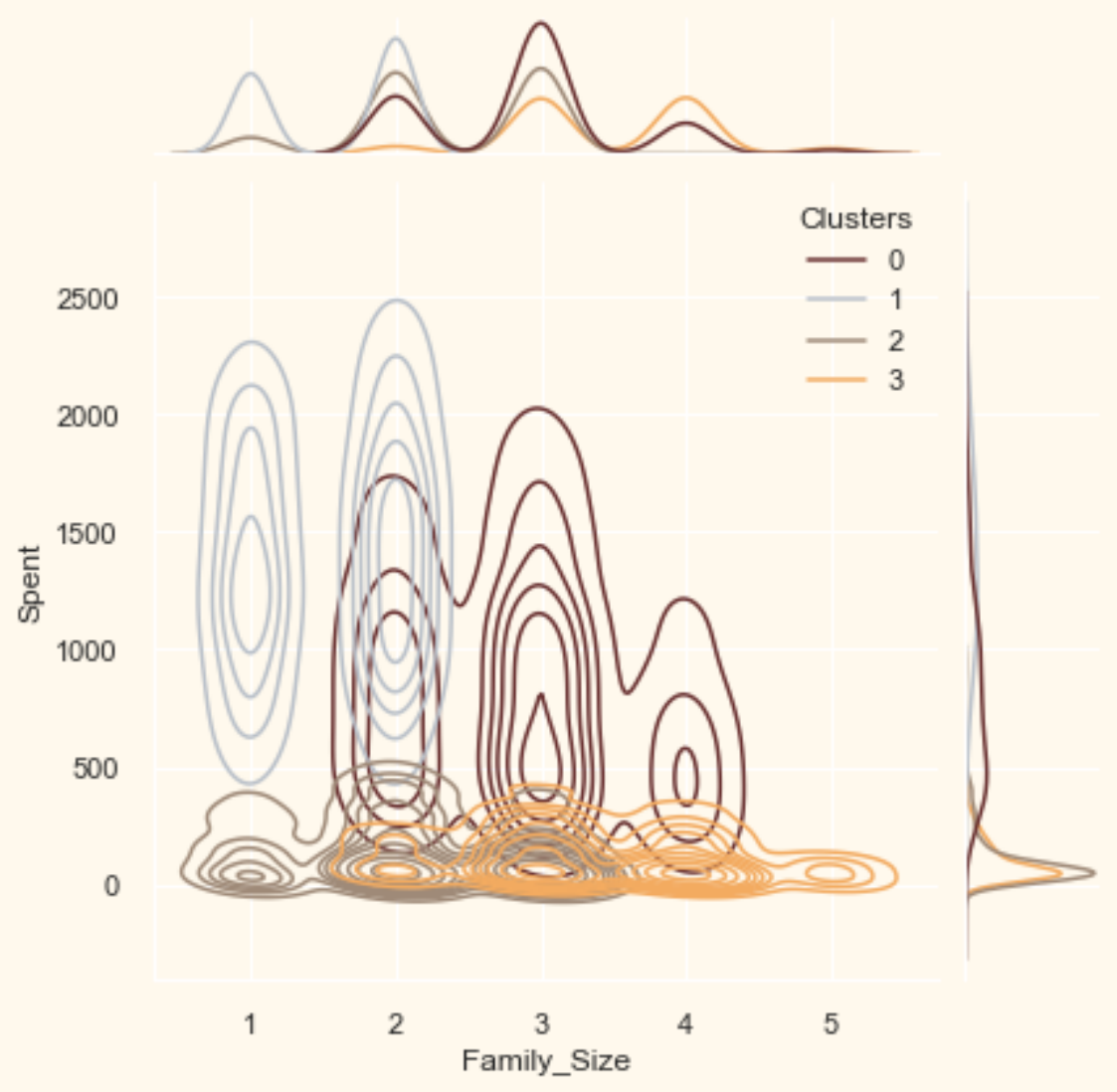
<Figure size 576x396 with 0 Axes>



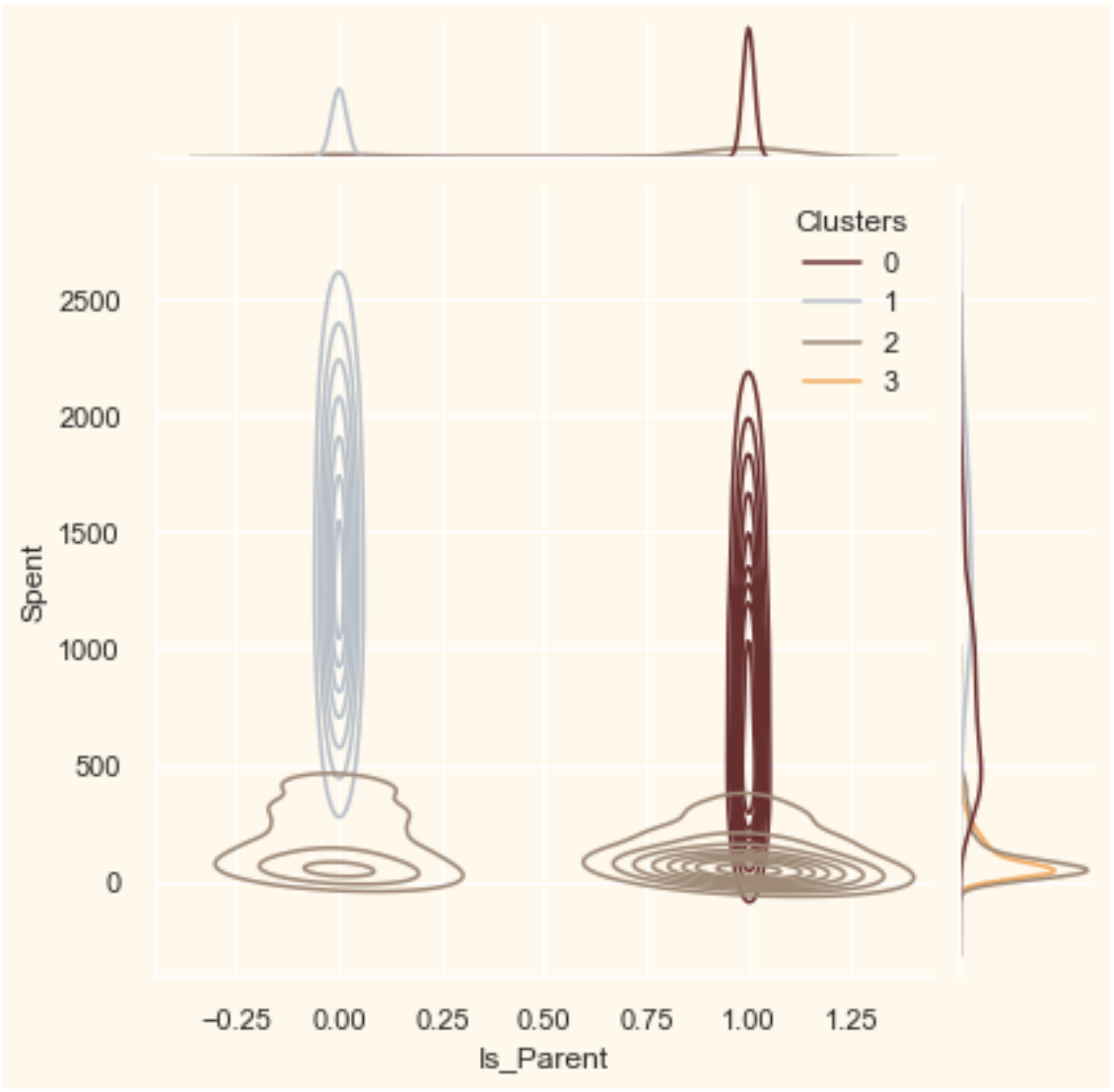

```
<Figure size 576x396 with 0 Axes>
```



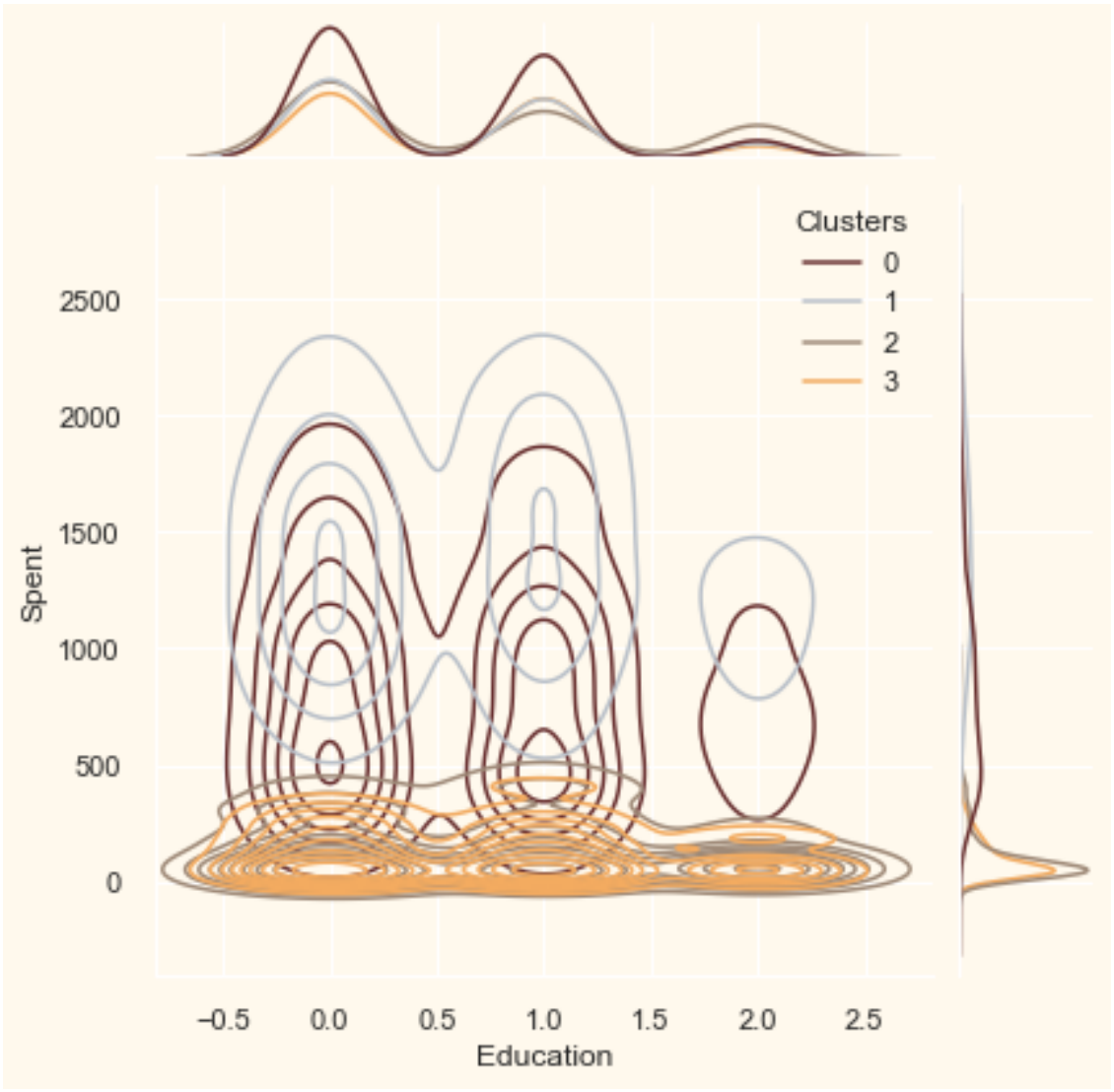
<Figure size 576x396 with 0 Axes>



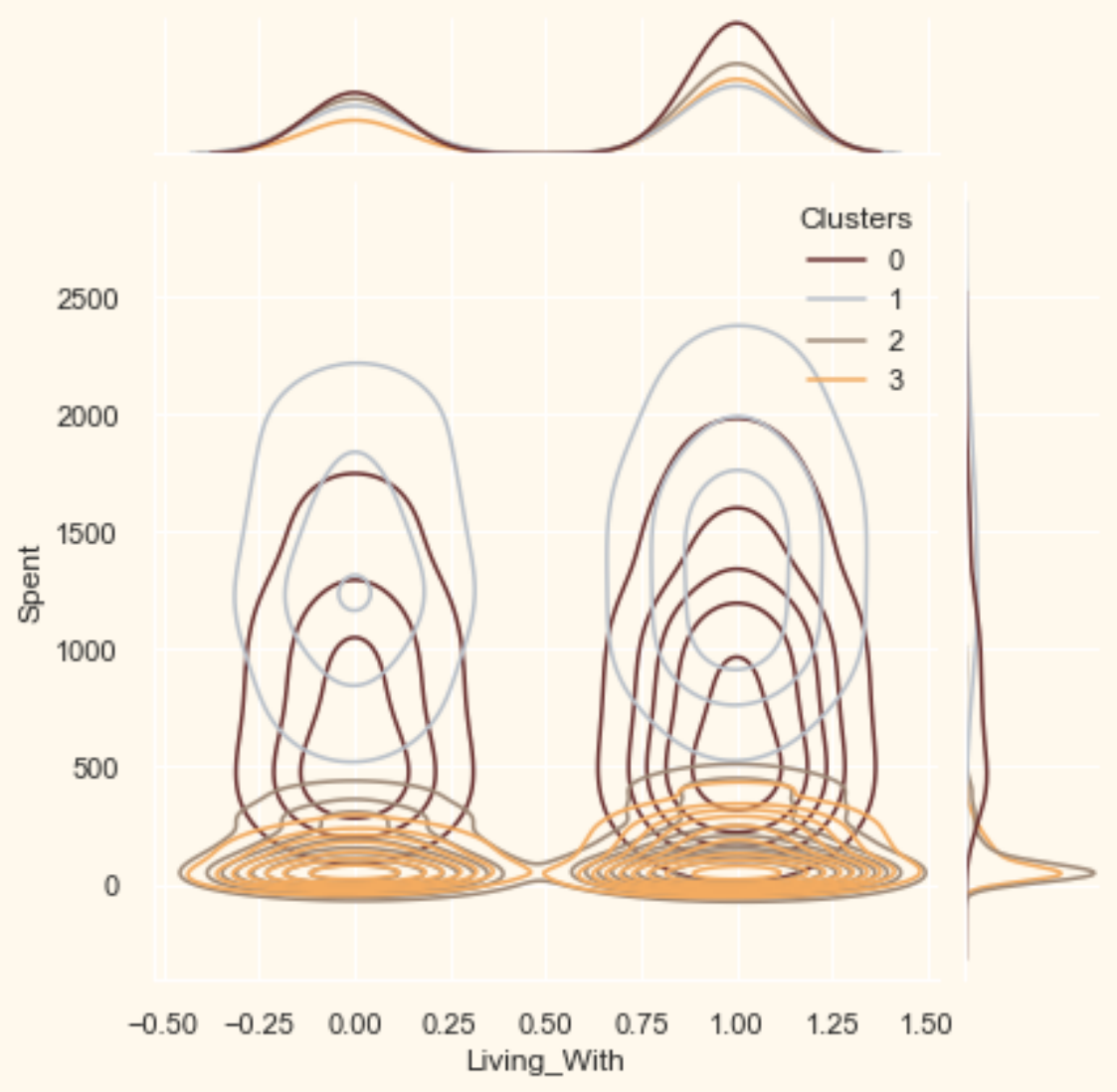
<Figure size 576x396 with 0 Axes>



<Figure size 576x396 with 0 Axes>



<Figure size 576x396 with 0 Axes>



Points to be noted:

The following information can be deduced about the customers in different clusters.

Cluster 0: * They are definitely a parent. * They have max 4 number in their family and minimum 2. * Single parents are a subset of this group * Most of the parents haves teenager in their family. * Relatively younger.

Cluster 1: * They are definitely not a parent. * They have max 2 number in their family. * Slight majority of couples over single people. * span all ages. * A high-income group.

Cluster 2: * The majority of these peoples are parents. * They have max 3 members in their family. * The majority of them have only one kid. * Relatively Younger.

Cluster 3: * They are definitely a parent. * They have max 5 and at least 2 members in their family. * The majority of them have a teenager at home. * Relatively Older. * A low-income

group.

10 Conclusion:

So according to the output and overall analysis conducted on this project on customer personality analysis, we can conclude that the biggest customers of wines are: * Customers with an average income of around `$69,500`. * Customers with an average total spend of approximately `$1,252`. * Customers registered with the company for approximately `21 months`. * Customers with a graduate degree, And * Customers who are also heavy consumers of meat products.

We performed unsupervised clustering. We did use dimensionality reduction followed by agglomerative clustering. I came up with 4 clusters and further used them in profiling customers in clusters according to their family structures and income/spending. This can be used in planning better marketing strategies.