**PRESTIGE INSTITUTE OF MANAGEMENT AND RESEARCH, INDORE**

(AnAutonomousInstitutionEstablishedin1994, AccreditedTwiceConsecutivelywith Grade A‟NAAC’’ (UGC) ISO 9001: 2008 Certified Institute, AICTE / UGC Approved Programs affiliated to DAVV, Indore)



**(Session 2022 – 2023)**

**Multivariate Data Analysis Project Report on**

“Customer Personality Analysis”

**Faculty Mentor: Submitted By:**

Mr. Makhan Kumbhkar Manvi Saxena (21110113847)

Tanishq Singh (21110113467)

Tashmeet Kaur Saluja (21110313642)

**Class:**

BBA Business Analytics

V Semester (A)

***Customer Personality Analysis***

**Index**

[Acknowledgement 2](#_Toc121705447)

[Introduction 2](#_Toc121705448)

[What is the study is all about? 2](#_Toc121705449)

[How we are going to do this? 2](#_Toc121705450)

[Machine learning Steps for our Project 2](#_Toc121705451)

[Collecting the data 2](#_Toc121705452)

[Preparing the Data 2](#_Toc121705453)

[Data Preprocessing 2](#_Toc121705454)

[Dimensionality Reduction 2](#_Toc121705455)

[K-Mean Clustering 2](#_Toc121705456)

[Code 2](#_Toc121705457)

[1. Problem Statement 2](#_Toc121705458)

[2. Basics Of Analysis 2](#_Toc121705459)

[3. Data Cleaning 2](#_Toc121705460)

[4. Data Preprocessing 2](#_Toc121705461)

[5. Exploratory Data Analysis 2](#_Toc121705462)

[6. Machine Learning 2](#_Toc121705463)

[7. Cluster Profiling 2](#_Toc121705464)

[Result 2](#_Toc121705465)

[Recommendations 2](#_Toc121705466)

[References 2](#_Toc121705467)

# *Acknowledgement*

We would like to thank **Mr. Makhan Kumbhkar**, our Professor-in-charge and our Director, **Dr. Subramanian Raman Iyer** for their support and guidance in completing our project on the topic, “**Customer Personality Analysis**”.It was a great learning experience.

We would like to take this opportunity to express our gratitude to all of the group members Manvi Saxena, Tanishq Singh and Tashmeet Kaur Saluja. The project would not have been successful without their cooperation and inputs.

# *Introduction*

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. In this project, we are going to introduce you to a data science project on customer personality analysis with Python.

Data Science and Artificial Intelligence are revolutionizing the world through technical transformations. We can observe many machine learning applications in day-to-day lives, but one of the greatest applications of machine learning is to classify individuals based on their personality traits. Each person on this planet is unique and carries a unique personality. The availability of a high-dimensional and large amount of data has paved the way for increasing marketing campaigns' effectiveness by targeting specific people. Such personality-based communications are highly effective in increasing the popularity and attractiveness of products and services. It increased usage, customer satisfaction, and broader acceptance among users. Some common examples are: Personalization of online advertisement campaigns leads to more revenue and click-through rates.

Personality traits are closely associated with an individual’s behavior and preferences. Hence the fusion of a personality-based approach has primarily increased the Recommender System’s attractiveness.

Personality-based adaption’s are also used to provide personalized visualizations and could even suggest better music recommendations.

Data Science and Artificial Intelligence are revolutionizing the world through technical transformations. We can observe many machine learning applications in day-to-day lives, but one of the greatest applications of machine learning is to classify individuals based on their personality traits. Each person on this planet is unique and carries a unique personality. The availability of a high-dimensional and large amount of data has paved the way for increasing marketing campaigns' effectiveness by targeting specific people. Such personality-based communications are highly effective in increasing the popularity and attractiveness of products and services. It increased usage, customer satisfaction, and broader acceptance among users. Some common examples are: Personalization of online advertisement campaigns leads to more revenue and click-through rates.

Personality traits are closely associated with an individual’s behavior and preferences. Hence the fusion of a personality-based approach has primarily increased the Recommender System’s attractiveness.

Personality-based adaption’s are also used to provide personalized visualizations and could even suggest better music recommendations.

Personality traits could also solve the “cold start” problem using scientifically validated and relatively stable latent dimensions of an individual based on personalized systems

With the availability of high-dimensional and fine-grained data about human behavior, it becomes too handy to research and observe human behavior. Using mobile sensing studies, data collected from our day-to-day activities have drastically altered how psychologists perform research and undertake personality assessments. Machine learning models are a boon to researchers and are used to learn highly complex relationships and evaluate their generalizability and robustness using the resampling method. It has the potential to transform research and assessment in personality psychology. Algorithms can handle vast datasets, including thousands of attributes, without succumbing to co-linearity issues. Moreover, ML algorithms are highly efficient in recognizing patterns in datasets that humans cannot even perceive. The use of these ML models can lead to better, more objective, and automated personality assessments.

People interact and express their likes, thoughts, feelings, and opinions on social media, capturing their personality traits. Machine Learning models have been actively using such a wide range of data to predict individuals. Various supervised machine learning algorithms like Naïve Bayes and Support Vector Machines are widely used among industries to predict personality traits. Moreover, recently, researchers have started to apply unsupervised learning methods to identify other psychological constructs in digital data.

As a data scientist we come across datasets where we have the data but we don't know much about what story it contains. In that case we need to figure out something to start with, exploratory data analysis is the best way to start with. EDA gives you a glance at the data story and what it hides. But what if we still need some more from the data or the EDA is not capable of satisfying our desires. Then in that case we use some machine learning algorithm to deep dive into the data and bring out useful insights.

We are doing this case study on a customer dataset to see how a machine learning algorithm can help to find the hidden insights. This case study also helps the reader to understand the way to use the algorithm as we are trying to be as simple as we can. Also this case study will help to understand the flow of code for EDA, Visualization, to algorithm and conclusion. How we can use different libraries to explore and visualize the data set.

## What is the study is all about?

So, here we have a dataset of customers which has some demographic and purchasing details. We are trying to get the costumer's personality and itsbehavior. In this project, unsupervised data clustering is performed on customers' records from a supermarket's database.Customer segmentation is the practice of dividing customers into groups that reflect similarities among each other. Customers will be divided into segments to support customer targeting campaigns.

## How we are going to do this?

We don't have any target variable. We will divide the customers into different clusters based on the similarities and dissimilarities with the help of KMeans clustering algorithm.

# *Machine learning Steps for our Project*

3.1 Collecting Data:

3.2 Preparing the Data:

3.3 Choosing a Model:

3.4 Training the Model:

3.5 Evaluating the Model:

## Collecting the data

The dataset consists of 2240 datapoints and 29 attributes. This can be re-organized into the following subsets:

**Customer's Information**

* ID
* Year\_Birth
* Education
* Marital\_Status
* Income
* Kidhome
* Teenhome
* Dt\_Customer
* Recency
* Complain

**Products**

Amount spent on different products in the last 2 years

* MntWines
* MntFruits
* MntMeatProducts
* MntFishProducts
* MntSweetProducts
* MntGoldProds

**Place**

* NumWebPurchases
* NumCatalogPurchases
* NumStorePurchases
* NumWebVisitsMonth

**Promotion**

* NumDealsPurchases
* AcceptedCmp1
* AcceptedCmp2
* AcceptedCmp3
* AcceptedCmp4
* AcceptedCmp5
* Response

## Preparing the Data

An overview of the work carried out in this section encompasses the following:

1. Removal of missing values
2. Creation of a new feature ("Customer\_For") out of "Dt\_Customer"
3. Engineering further new features
   * "Age" from "Year\_Birth"
   * "Spent" indicating the total amount spent on products over the past two years
   * "Living\_With" from "Marital\_Status"
   * "Children" indicating the total amount of children in a household, including kids and teenagers
   * "Family\_Size" indicating the total amount of people in a household
   * "Is\_Parent" to indicate parenthood status
4. Devise three new categories in "Education" by simplifying existent value counts
5. Dropping redundant features

## Data Preprocessing

The following steps are taken to preprocess the data:

1. Label encoding categorical features
2. Scaling features using a standard scalar
3. Generating a subset data frame for reduction of dimensionality

## Dimensionality Reduction

Steps taken in this subsection:

* Dimensionality reduction with Principal Component Analysis.
* Plotting of the reduced dataframe.

## K-Mean Clustering

To apply clustering, the following steps have been taken:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third step, which means reassign each data point to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

# *Code*

## 1. Problem Statement

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.

## 2. Basics Of Analysis

**2.1 Basic Libraries**

**# data analysis and wrangling**

**import** pandas as pd

import numpy as np

import random as rnd

**# visualization**

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from matplotlib import colors

from matplotlib.colors import ListedColormap

**# Clustering**

from yellowbrick.cluster import KElbowVisualizer

from sklearn.cluster import KMeans

from sklearn.preprocessing import LabelEncoder

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib.colors import ListedColormap

from sklearn import metrics

**2.2 Import Dataset**

from google.colab import files

uploaded = files.upload()

import io

data=pd.read\_csv("marketing\_campaign.csv",sep='\t')

df=data

**2.3 Understanding the Dataset**

df.head()

df.info()

We have total 29 columns(variables) in this dataset most of them are int64 datatype, we have 1 float64 and 3 (object type) categorical variables.

Let's check for the null values in the dataset.

## 3. Data Cleaning

**3.1 Treat Null Values**

df.isnull().sum()

df[(df.Income).isnull()==True]

We have 24 null values in the Income column. We can see the NaN in the income column of table above. This Income column is crucial for our analysis so we need to find a way to treat these null values.

Here I am using a way to replace null with some value which I think is more relevant than dropping the records. Calculate the average income based on the education level and replacing it with the null values.

**3.2 Check for Duplicate Data**

df.duplicated().sum()

**3.3 Remove Irrelevent Data**

data['Marital\_Status']=data['Marital\_Status'].replace(['Absurd','Alone','YOLO'],['Single','Single','Single'])

data['age']=2022-data['Year\_Birth']

data['children']=data['Kidhome']+data['Teenhome']

data.drop(['Year\_Birth','Kidhome','Teenhome','avg\_income'],axis=1,inplace=True)

**3.4 Feature Engineering**

data["Dt\_Customer"] = pd.to\_datetime(data["Dt\_Customer"])

dates = []

for value in data["Dt\_Customer"]:

    value = value.date()

    dates.append(value)

print("Oldest customer join date: ", min(dates))

print("Newest customer join date:", max(dates))

**# Get newest customer date**

number\_of\_days = []

ref\_date = max(dates)

for d in dates:

    delta = ref\_date - d

    number\_of\_days.append(delta)

**# Create 'Customer\_For' feature**

data["Customer\_For"] = number\_of\_days

data["Customer\_For"] = pd.to\_numeric(data["Customer\_For"], errors="raise")

Explore unique values in categorical features to get a clearer picture of data.

print("Marital\_Status:\n", data["Marital\_Status"].value\_counts(), "\n")

print("Education:\n", data["Education"].value\_counts(), "\n")

**3.5 Further feature engineering**

**# Age of customers as of 01/11/2021**

data["Age"] = 2021 - data["Year\_Birth"]

**# Total amount spend**

data["Spent"] = data["MntWines"] + data["MntFruits"] + data["MntMeatProducts"] + \

    data["MntFishProducts"] + data["MntSweetProducts"] + data["MntGoldProds"]

**# Derive household living situation by marital status**

data["Living\_With"] = data["Marital\_Status"].replace(

    {"Married": "Partner",

     "Together": "Partner",

     "Absurd": "Alone",

     "Widow": "Alone",

     "YOLO:": "Alone",

     "Divorced": "Alone",

     "Single": "Alone",

     })

**# Total children living at home**

data["Children"] = data["Kidhome"] + data["Teenhome"]

**# Total household members**

data["Family\_Size"] = data["Living\_With"].map(

    {"Alone":1,

     "Partner":2}

) + data["Children"]

**# Drop NAs acquired due to failed parsings**

data["Family\_Size"].dropna().astype(int)

**# Parenthood feature**

data["Is\_Parent"] = np.where(data.Children > 0, 1, 0)

**# Divide education levels in three categories**

data["Education"] = data["Education"].replace(

    {"Basic": "Undergraduate",

     "2n Cycle": "Undergraduate",

     "Graduation": "Graduate",

     "Master": "Postgraduate",

     "PhD": "Postgraduate"}

)

# Improve clarity

data = data.rename(columns={"MntWines": "Wines",

                            "MntFruits": "Fruits",

                            "MntFishProducts": "Fish",

                            "MntSweetProducts": "Sweets",

                            "MntGoldProducts": "Gold"}

                   )

**# Drop redundant features**

to\_drop = ["Marital\_Status", "Dt\_Customer", "Z\_CostContact", "Z\_Revenue",

           "Year\_Birth", "ID"]

data = data.drop(to\_drop, axis=1)

data.describe()

**Plot select features for visual analysis**

# Set up colours

sns.set(rc={

    "axes.facecolor":"#FFF9EA",

    "figure.facecolor":"#FFF9EA"

})

pallet = ["#682F33", "#9E713A", "#D3B3B1", "#B9C0C9", "#9F8A78", "#F3DD60"]

cmap = colors.ListedColormap(pallet)

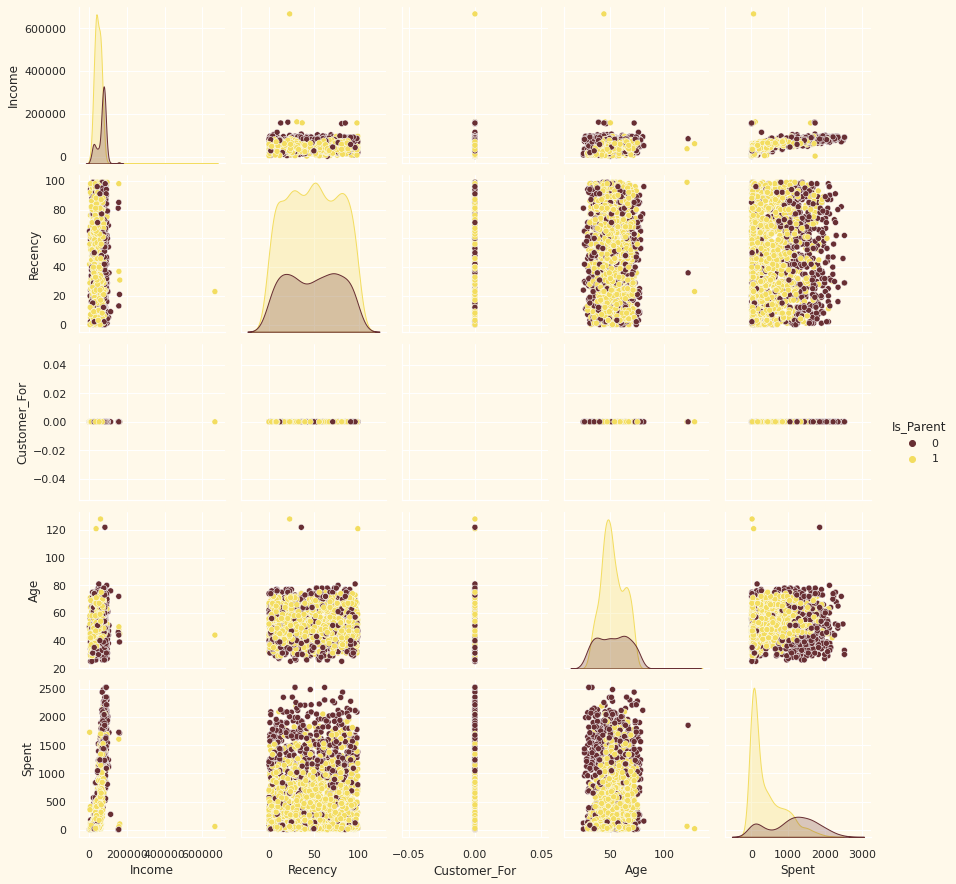
**# Plot features**

feats = ["Income", "Recency", "Customer\_For", "Age", "Spent", "Is\_Parent"]

plt.figure()

sns.pairplot(data[feats], hue = "Is\_Parent", palette = (["#682F33", "#F3DD60"]))

plt.show()

****

**Drop outliers by setting limits on Income and Age**

data = data[(data["Age"] < 90)]

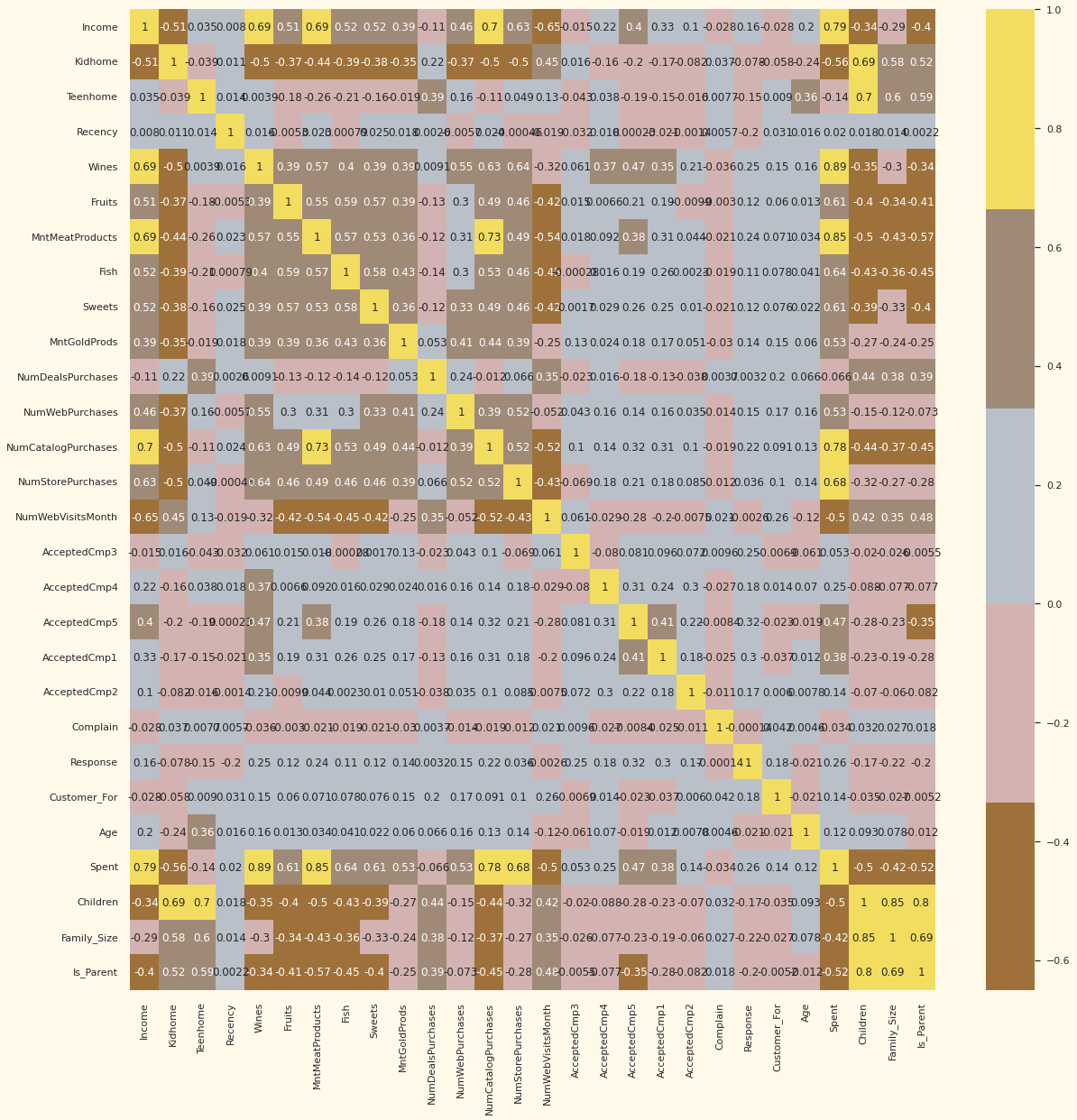
data = data[(data["Income"] < 600000)]

Analysis of correlation amongst features (quantitative attributes only)

cm\_plot = data.corr()

plt.figure(figsize=(20,20))

sns.heatmap(cm\_plot, annot=True, cmap=cmap, center=0)

****

## 4. Data Preprocessing

**# Build a list of categorical variables**

objs = (data.dtypes == 'object')

object\_cols = list(objs[objs].index)

print("Categorical variables: ", object\_cols)

**# Label encode object dtypes**

LabEnc = LabelEncoder()

for obj in object\_cols:

    data[obj] = data[[obj]].apply(LabEnc.fit\_transform)

**All features have been encoded to numerical values.**

**# Subset dataframe**

df\_subset = data.copy()

cols\_delete = ["AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "AcceptedCmp1",

                "AcceptedCmp2", "Complain", "Response"]

df\_subset = df\_subset.drop(cols\_delete, axis=1)

**# Scale data**

scaler = StandardScaler()

scaler.fit(df\_subset)

scaled\_df = pd.DataFrame(scaler.transform(df\_subset), columns=df\_subset.columns)

**All features have now been scaled.**

**Dimensionality Reduction**

The final classification to be applied is based on several factors - basically, features. A higher number of features increase the difficulty of implementation. Since many such features are related, many of them are therefore redundant. Therefore, a suitable solution to this is dimensionality reduction on selected features before classification.

Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables.

A common technique for reducing dimensionality of data sets is Principal Component Analysis (PCA). This technique improves readability with minimal loss of data.

For this data set, the dimensionality will be reduced to 3.

pca = PCA(n\_components=3)

scaled\_df.dropna(inplace=True)

pca.fit(scaled\_df)

PCA\_df = pd.DataFrame(pca.transform(scaled\_df), columns=(["col1", "col2", "col3"]))

PCA\_df.describe().T

**# 3D data visualisation in reduced dimension**

x = PCA\_df["col1"]

y = PCA\_df["col2"]

z = PCA\_df["col3"]

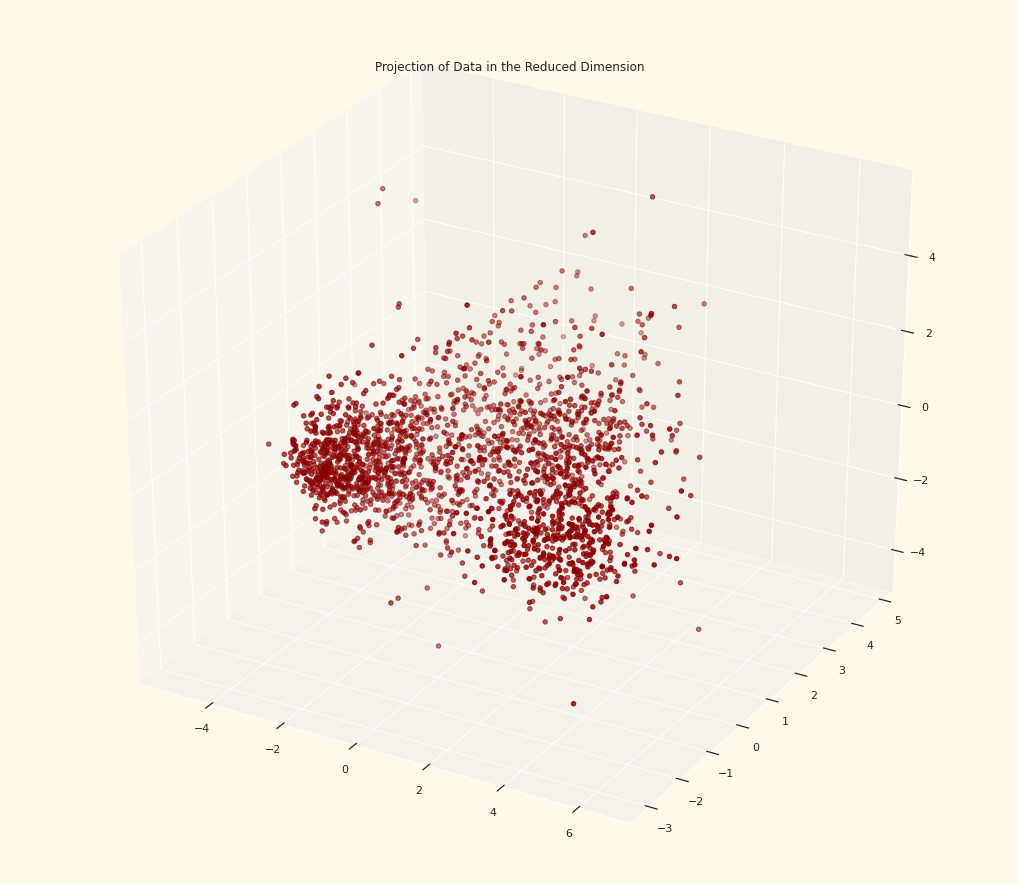
fig = plt.figure(figsize=(18, 16))

ax = fig.add\_subplot(111, projection = "3d")

ax.scatter(x, y, z, c = "darkred", marker = "o")

ax.set\_title("Projection of Data in the Reduced Dimension")

plt.show()

****

## 5. Exploratory Data Analysis

Now, we have treated the null values in the dataset. We don't have any missing values in the dataset.

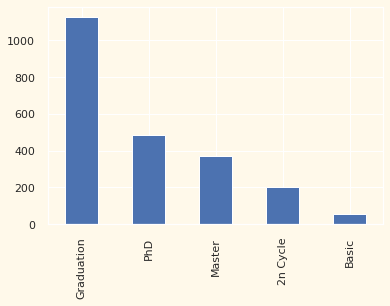
Let's see what we have in the categorical columns.

**5.1 Categorical Variables**

df.describe(include='object')

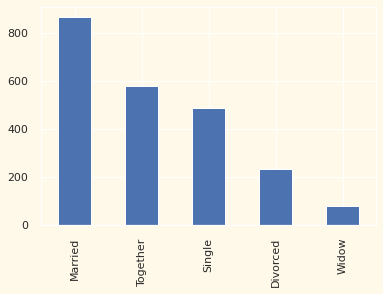
print(df.Education.value\_counts())

df.Education.value\_counts().plot(kind="bar")



print(df['Marital\_Status'].value\_counts())

df['Marital\_Status'].value\_counts().plot(kind="bar")

****

**5.2 Statistical description of continuous variables**

data[['Income','age','children','MntWines','MntFruits','MntMeatProducts','MntFishProducts','MntSweetProducts','MntGoldProds']].describe()

plt.figure(figsize=(8,4))

plt.subplot(1,2,1)

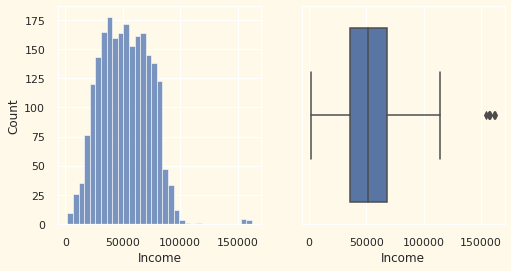
plt.xlabel='Income'

sns.histplot(data=data,x='Income')

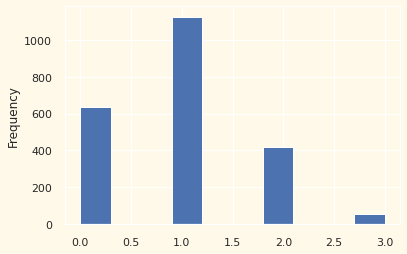
plt.subplot(1,2,2)

plt.xlabel='Income'

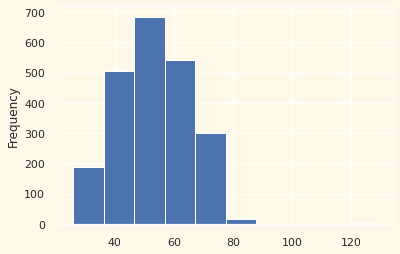
sns.boxplot(data=data,x='Income')



data.children.plot(kind="hist")



data.age.plot(kind="hist")

******

**5.3 Visualization**

**5.3.1 Continuous Variable**

Let's visualise all variables. This trick will help us to understand how the variables are skewed or distributed and gives an overall picture of the dataset. With the help of for loop we can plot all variables in one go rather than plotting each variable at a time.

cont\_col=['Income','children','age','MntWines','MntFruits','MntMeatProducts','MntFishProducts','MntSweetProducts','MntGoldProds']

for i in cont\_col:

    plt.figure(figsize=(18,4))

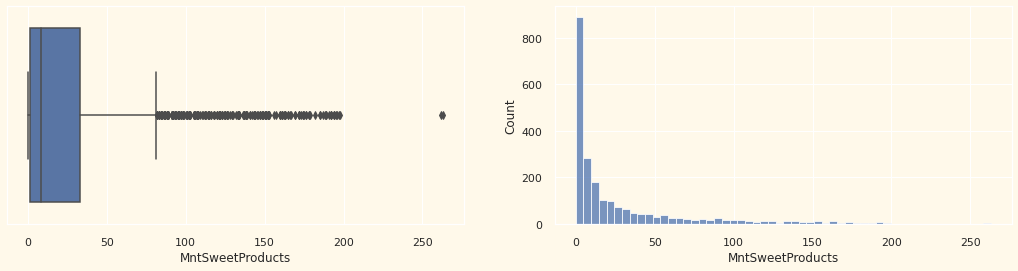
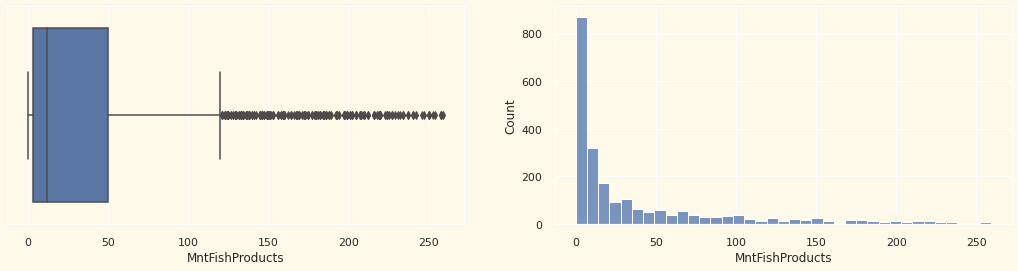
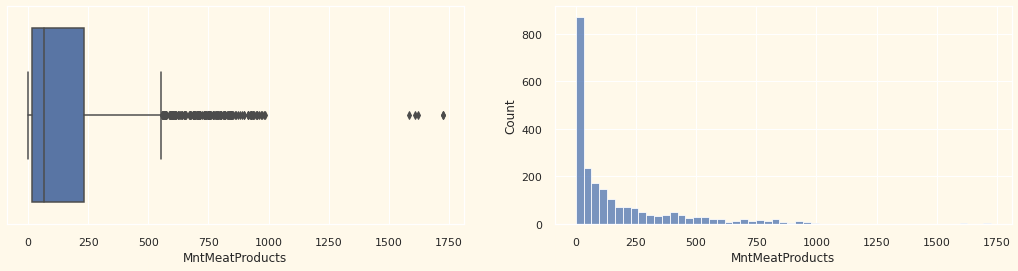
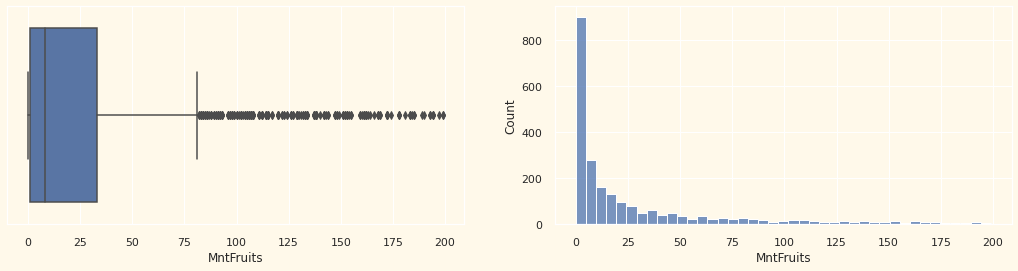
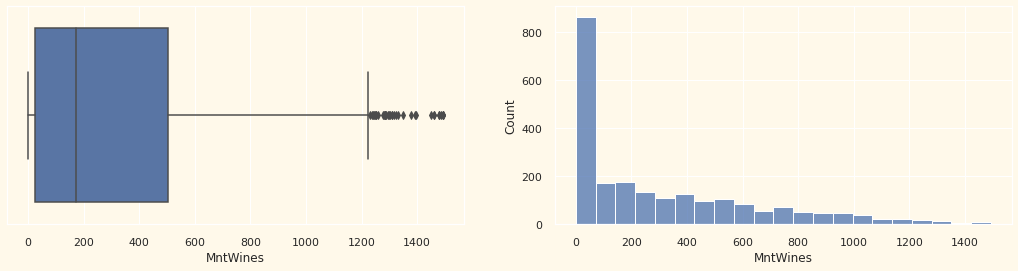
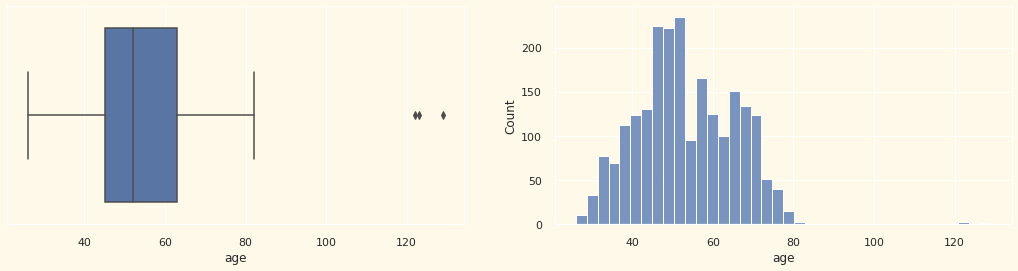
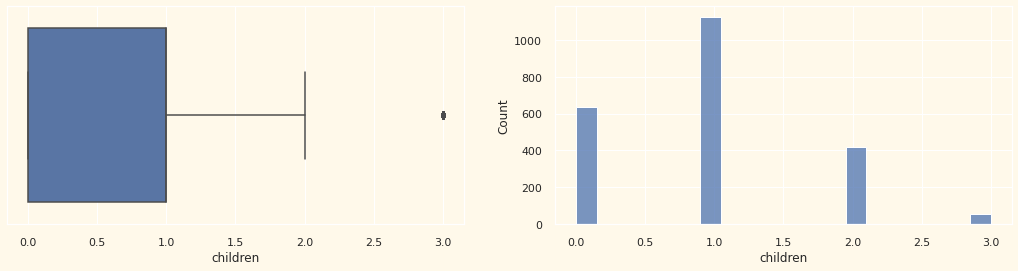
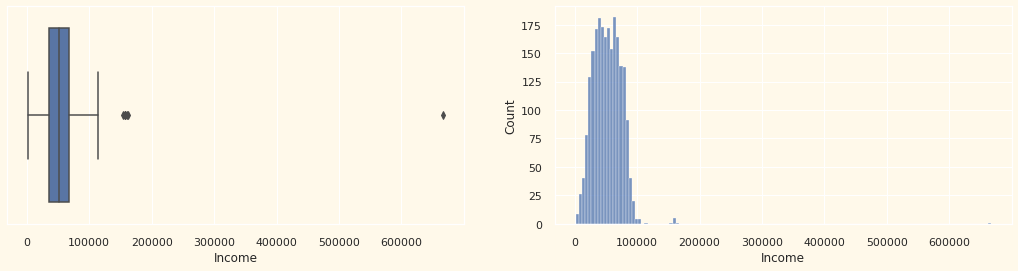
    plt.subplot(1,2,1)

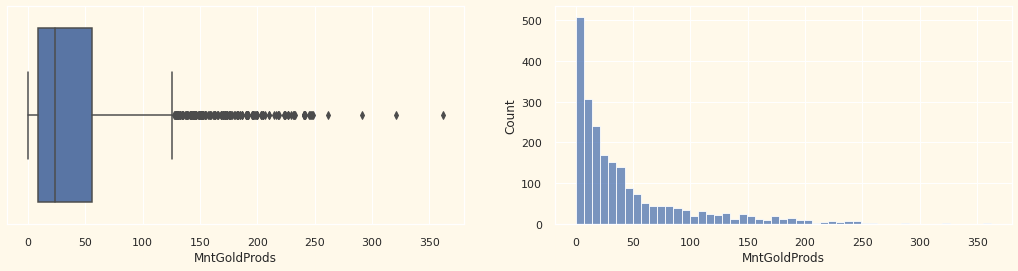
    sns.boxplot(x=data[i],data=data)

    plt.subplot(1,2,2)

    sns.histplot(x=data[i],data=data)

    plt.show()

****

****

**5.3.2 Categorical Variable**

cat\_col=['Education','Marital\_Status']

for i in cat\_col:

    plt.figure(figsize=(12,4))

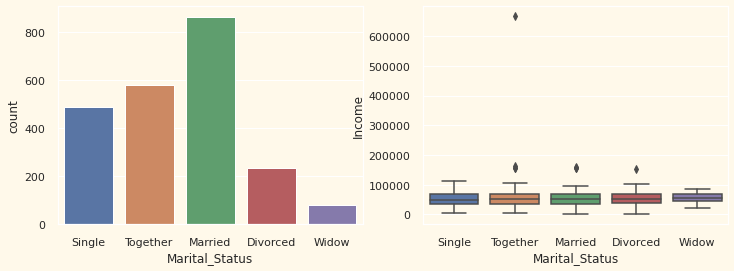
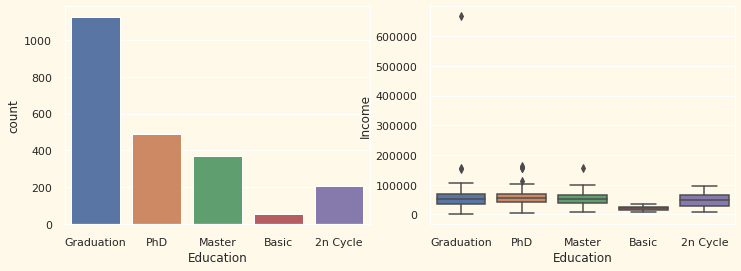
    plt.subplot(1,2,1)

    sns.countplot(x=data[i],data=data)

    plt.subplot(1,2,2)

    sns.boxplot(x =i, y ='Income', data =data)

    plt.show()



So, this is all about the EDA. There is lot more to do as far as EDA is concerned.

After this is exploration of the dataset, time to proceed for the machine learning algorithm. As our intention is to divide dataset into groups as per the similarity of the observations. These groups are called clusters. To make clusters we are using KMeans clustering algorithm.

## 6. Machine Learning

To proceed with the clustering we need to make another copy of the dataset. This will make the dataset immune to any changes.

cluster=data.copy()

We don't need variables 'ID' and 'Dt\_Customer' for clustering so we drop them.

cluster.drop(['ID','Dt\_Customer'],axis=1,inplace=True)

Now, we have two categorical variables and they are very important for clustering. We need to convert the variables into numerical codes to convert them into continuous variable. This needs to be done because machine learning doesn't accept categorical variables.

for i in cat\_col:

    cluster[i] = cluster[i].astype('category')

    cluster[i] = cluster[i].cat.codes

cluster.head()

cluster.info()

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

Standard Scaler is a common technique used for standardization of the dataset. There are some manual and libraries available for this in python.

scaler=StandardScaler()

data\_scaled=pd.DataFrame(scaler.fit\_transform(cluster),columns=cluster.columns)

**K-Mean Algorithm**

Here we are using KMeans algorithm. For this algorithm to work we need to define number of clusters for it. We need to decide the value of k for KMeans. For deciding the value of K we can use any of two techniques:

~Sum of Squared Errors

~Silhouette Score

**Sum Of Squared Errors**

Theoretically, for Kmeans to perform well, we recommend SSE to be minimum. But we generally look out for a elbow shape in the plot of "number of clusters Vs value of SSE".

sse={}

for k in range(1,11):

    kmean=KMeans(n\_clusters=k,random\_state=1).fit(data\_scaled)

    lables=kmean.predict(data\_scaled)

    sse[k]=kmean.inertia\_

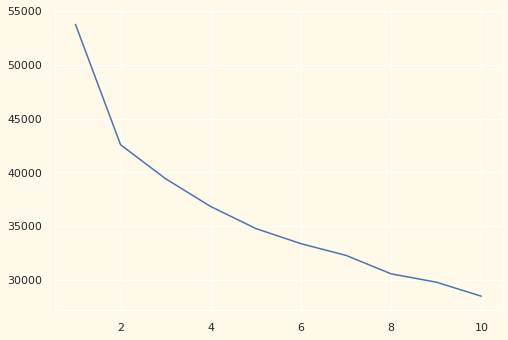
plt.figure()

plt.plot(list(sse.keys()), list(sse.values()), 'bx-')

plt.xlable='no. of clusters'

plt.ylable= 'SSE'

plt.show()



As per the plot we select value of 2(number of clusters)

**Silhouette Score**

Sometimes we don't get clear indications for the value of K for the algorithm through SSE. In that case we use Silhouette score for the decision of value of K.

Choose value of K at which silhouette score is highest and close to 1.

sc={}

for k in range(2,10):

    km=KMeans(n\_clusters=k,random\_state=1).fit(data\_scaled)

    l=km.predict(data\_scaled)

    sc[k]=silhouette\_score(data\_scaled,l)

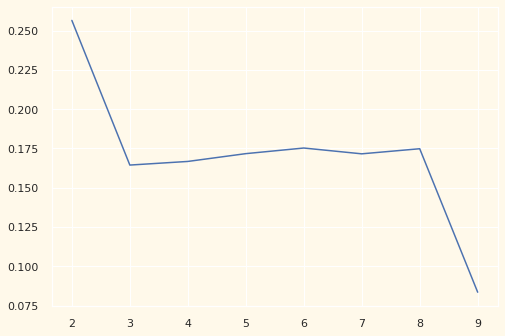
plt.figure()

plt.plot(list(sc.keys()), list(sc.values()), 'bx-')

plt.xlable='no. of clusters'

plt.ylable= 'SSE'

plt.show()



With both the methods, we got 2(number of clusters) as the best value of K for KMeans.

kmeans=KMeans(n\_clusters=2, random\_state=1)

kmeans.fit(data\_scaled)

# to predict cluster for each entity of cluster dataset

cluster['cluster\_no']=kmeans.predict(data\_scaled)

# to predict cluster for each entity of original "data" dataset

data['cluster']=kmeans.predict(data\_scaled)

An extra column is added to both datasets copied as well as original.

data.head()

data.cluster.value\_counts()

## 7. Cluster Profiling

means=data.groupby('cluster').mean()

df\_mean=pd.DataFrame(means)

df\_mean.index=['cluster1','cluster2']

df\_mean.T

data

cluster1=data.loc[data.cluster==0]

cluster2=data.loc[data.cluster==1]

cluster3=data.loc[data.cluster==2]

cluster1

for i in cat\_col:

    print(cluster1[i].value\_counts())

    print(cluster2[i].value\_counts())

    plt.figure(figsize=(18,4))

    plt.subplot(1,4,1)

    plt.title("data")

    sns.countplot(x=data[i],data=data)

    plt.subplot(1,4,2)

    plt.title("cluster1")

    sns.countplot(x=cluster1[i],data=cluster1)

    plt.subplot(1,4,3)

    plt.title("cluster2")

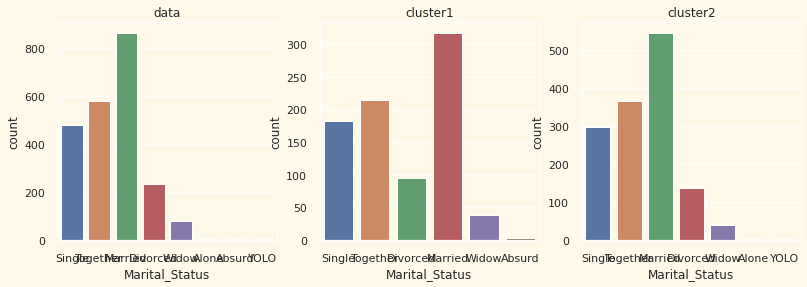
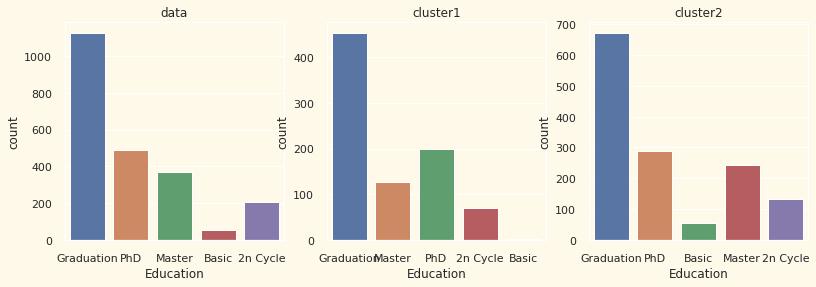
    sns.countplot(x=cluster2[i],data=cluster2)

    #plt.subplot(1,4,4)

    #plt.title("cluster3")

    #sns.countplot(x=cluster3[i],data=cluster3)

    plt.show()



for i in cont\_col:

    plt.figure(figsize=(20,4))

    plt.subplot(1,4,1)

    sns.histplot(x=data[i],data=data,hue='cluster')

    plt.subplot(1,4,2)

    plt.title("cluster0")

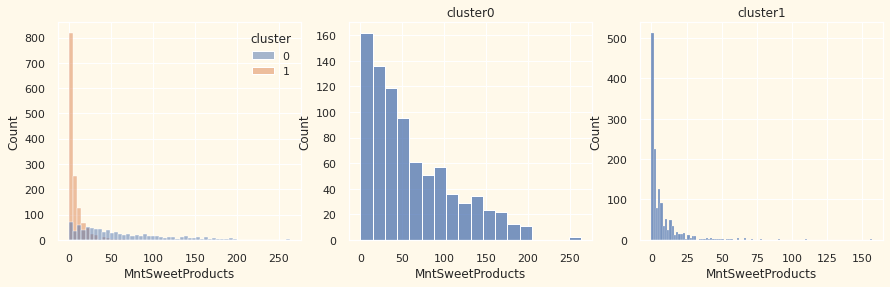
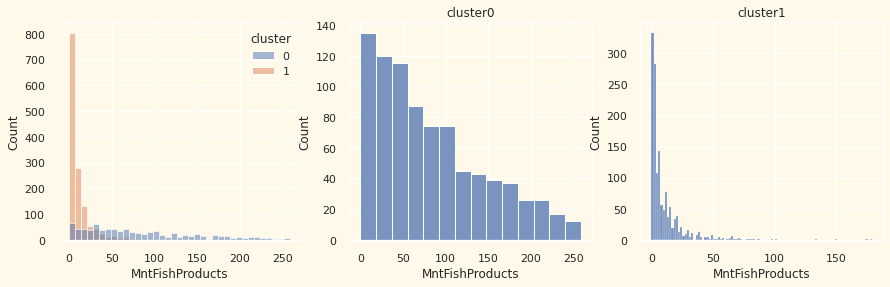
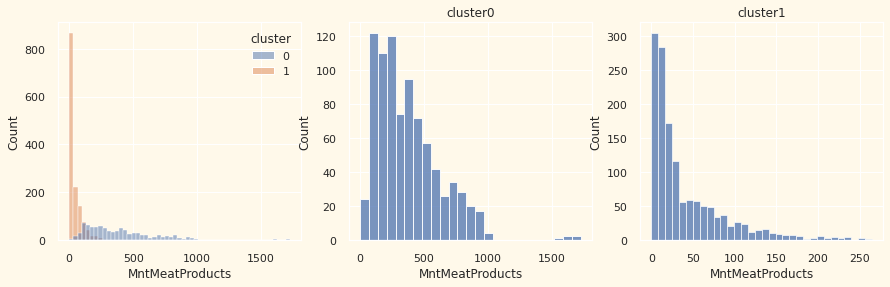
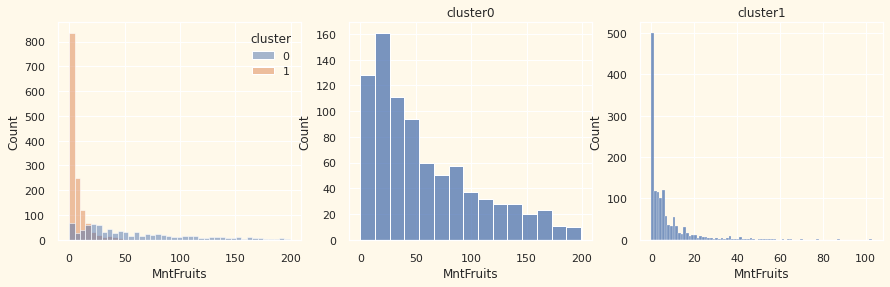
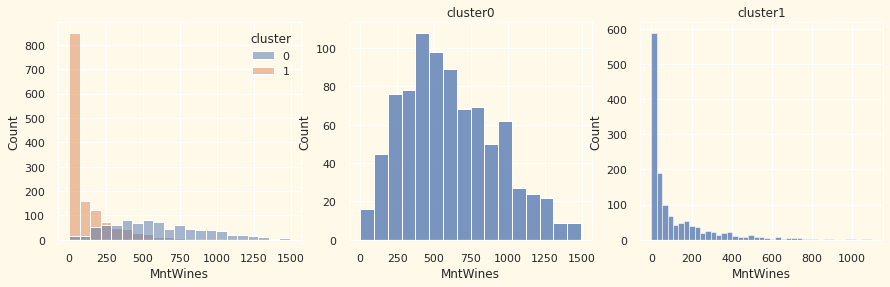
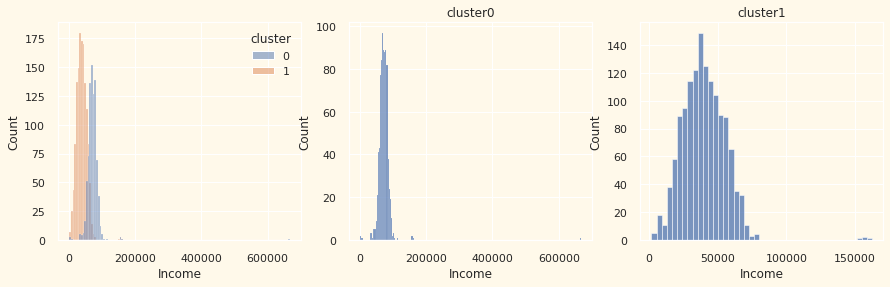
    sns.histplot(x=cluster1[i],data=cluster1)

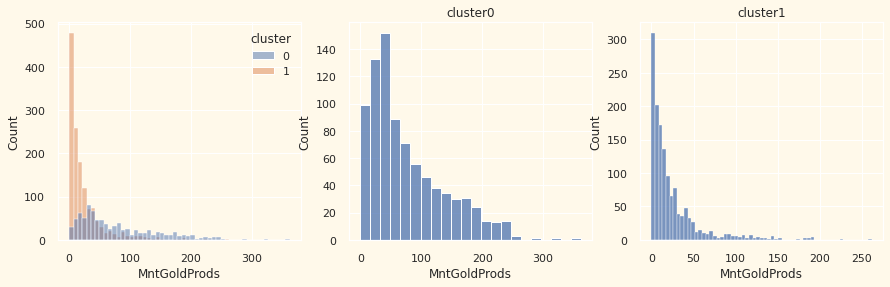
    plt.subplot(1,4,3)

    plt.title("cluster1")

    sns.histplot(x=cluster2[i],data=cluster2)

    plt.show()

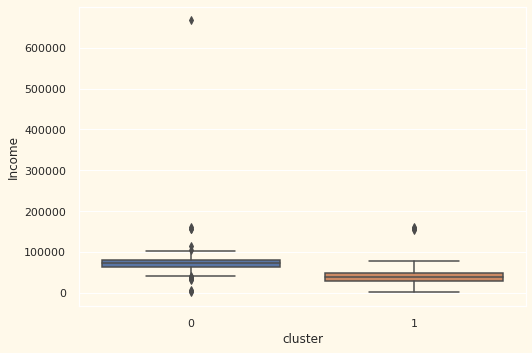


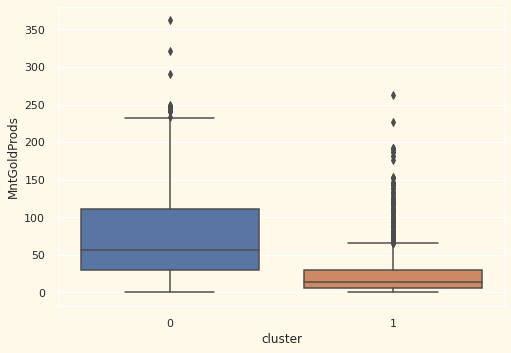
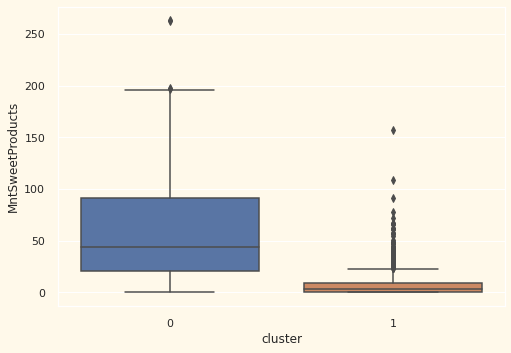
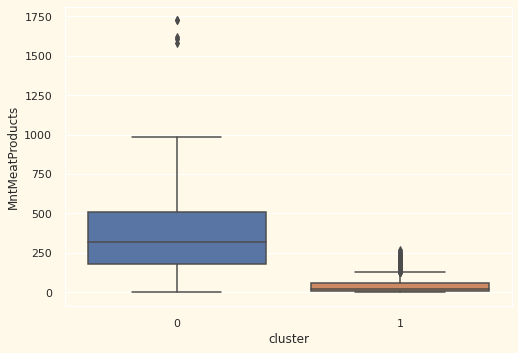
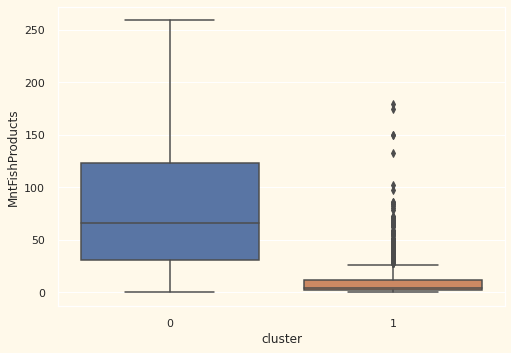
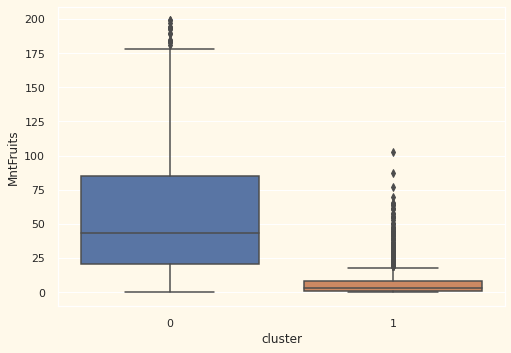
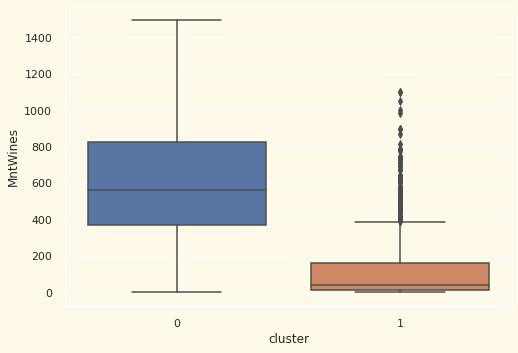


for i in cont\_col:

    sns.boxplot(x='cluster', y=i, data=data)

    plt.show()





#created a column showing the total amount a customer spent

df1['TotalMnt'] = df1.MntMeatProducts + df1. MntWines + df1.MntFruits + df1.MntFishProducts + df1.MntSweetProducts + df1.MntGoldProds

#checks how much of a customer's income is spent purchasing these products

df1['Income\_to\_spend'] = round(df1['TotalMnt']/df1['Income'],3)

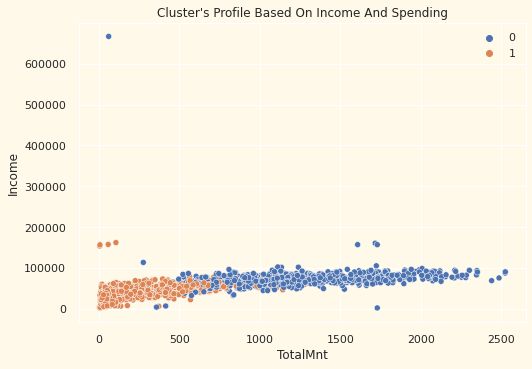
#check if there is a clear distinction between clusters with reference to income and total amount spent

pl = sns.scatterplot(data = df1,x=df1["TotalMnt"], y=df1["Income"],hue=data["cluster"])

pl.set\_title("Cluster's Profile Based On Income And Spending")

plt.legend()

plt.show()



# *Result*

Throughout this project, unsupervised clustering has been performed. Dimensionality reduction was employed, followed by agglomerative clustering. Four clusters have been produced and used for customer profiling per cluster according to their household structures, income and spending. The information extracted can be used for planning future marketing strategies.

Before concluding the analysis, first understand the motive behind this analysis. The intention is to understand the process of clustering the dataset to discover unseen personality of users. How the whole process from raw dataset to conclusions is carried out. What methodology we can use for feeding a clean dataset to KMeans algorithm. And how KMeans algorithm works.

By looking on the charts and tables, we can make different conclusions based on our own ability to understand the subjected matter.

# *Recommendations*

The amount of spending exhibited by Groups 1 and 2 appears to be proportional to their income, namely average income/spending, low income/spending and high income/spending, respectively.

It must be noted that the nature of the promotional campaigns conducted in the past is not defined by this data set. For example, the data provided in this data set could be describing engagement towards campaigns for products with typically low purchase frequency, such as turkeys around Thanksgiving - whilst a promotional campaign for this product is likely to attract one large wave of purchases; the customers are very unlikely to re-purchase a turkey after Thanksgiving, even when discounted.

Further analysis must be carried out to extract information to support the design of future promotional campaigns, such as:

* Average income of customers
* Average total spend of customers
* Average amount of time customers have been registered with the retailer for
* The most frequent education level amongst customers

# *References*

1. [**https://github.com/iulianghg/py-customer-personality-analysis**](https://github.com/iulianghg/py-customer-personality-analysis)
2. [**https://thecleverprogrammer.com/2021/02/08/customer-personality-analysis-with-python/**](https://thecleverprogrammer.com/2021/02/08/customer-personality-analysis-with-python/)
3. [**https://www.kaggle.com/code/govindfordata/personality-analysis-with-kmeans-clustering#5.-Machine-Learning**](https://www.kaggle.com/code/govindfordata/personality-analysis-with-kmeans-clustering#5.-Machine-Learning)
4. [**https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning**](https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning)
5. [**https://medium.com/analytics-vidhya/clustering-and-profiling-customers-using-k-means-9afa4277427**](https://medium.com/analytics-vidhya/clustering-and-profiling-customers-using-k-means-9afa4277427)
6. [**https://www.turing.com/kb/guide-to-principal-component-analysis**](https://www.turing.com/kb/guide-to-principal-component-analysis)
7. [**https://www.geeksforgeeks.org/dimensionality-reduction/**](https://www.geeksforgeeks.org/dimensionality-reduction/)
8. [**https://www.enjoyalgorithms.com/blog/personality-prediction-using-ml**](https://www.enjoyalgorithms.com/blog/personality-prediction-using-ml)
9. [**https://www.logility.com/es/blog/improve-forecast-quality-and-reliability-with-value-add-forecasting-part-1/**](https://www.logility.com/es/blog/improve-forecast-quality-and-reliability-with-value-add-forecasting-part-1/)
10. [**https://www.kaggle.com/piercedee/customer-analysis/**](https://www.kaggle.com/piercedee/customer-analysis/)
11. [**https://www.cobuildlab.com/blog/ai**](https://www.cobuildlab.com/blog/ai)