customer-segment-analysis

April 26, 2024

1 Customer Segmentation using Unsupervised Machine Learning in Python

2 1. Problem Statement

In today's era, companies work hard to make their customers happy. They launch new technologies and services so that customers can use their products more. They try to be in touch with each of their customers so that they can provide goods accordingly. But practically, it's very difficult and non-realistic to keep in touch with everyone. So, here comes the usage of Customer Segmentation.

Customer Segmentation means the segmentation of customers on the basis of their similar characteristics, behavior, and needs. This will eventually help the company in many ways. Like, they can launch the product or enhance the features accordingly. They can also target a particular sector as per their behaviors. All of these lead to an enhancement in the overall market value of the company.

3 1.2. Dataset Features

3.1 People:

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

3.2 Products:

1. MntWines: Amount spent on wine in last 2 years
2. MntFruits: Amount spent on fruits in last 2 years
3. MntMeatProducts: Amount spent on meat in last 2 years
4. MntFishProducts: Amount spent on fish in last 2 years
5. MntSweetProducts: Amount spent on sweets in last 2 years

6. MntGoldProds: Amount spent on gold in last 2 years

3.3 Promotion:

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

3.4 Place:

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

4 2. 2. Import Libraries and Data

[1]:
```
# handle table-like data and matrices
import pandas as pd
import numpy as np

# visualisation
import seaborn as sns
import matplotlib.pyplot as plt
# import missingno as msno
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff

# preprocessing
from sklearn.preprocessing import StandardScaler

# clustering
# from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering

# evaluations
from sklearn.metrics import confusion_matrix
# ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

```
c:\Users\Shaheen\AppData\Local\anaconda3\Lib\site-
packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

[2]:
```
df = pd.read_csv('marketing_campaign.csv', delimiter='\t')
```

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

[3]:
```
     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
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         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
3  10-02-2014       26        11  …                  6             0
4  19-01-2014       94       173  …                  5             0

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
0             0             0             0             0         0
1             0             0             0             0         0
2             0             0             0             0         0
3             0             0             0             0         0
4             0             0             0             0         0

   Z_CostContact  Z_Revenue  Response
0              3         11         1
1              3         11         0
2              3         11         0
3              3         11         0
4              3         11         0

[5 rows x 29 columns]
```

[4]:
```
df.shape
```

[4]:
```
(2240, 29)
```

[6]:
```
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
```

[7]:
```
df.describe()
```

[7]:
```
                 ID   Year_Birth         Income      Kidhome     Teenhome  \
count   2240.000000  2240.000000    2216.000000  2240.000000  2240.000000
mean    5592.159821  1968.805804   52247.251354     0.444196     0.506250
std     3246.662198    11.984069   25173.076661     0.538398     0.544538
min        0.000000  1893.000000    1730.000000     0.000000     0.000000
25%     2828.250000  1959.000000   35303.000000     0.000000     0.000000
50%     5458.500000  1970.000000   51381.500000     0.000000     0.000000
75%     8427.750000  1977.000000   68522.000000     1.000000     1.000000
max    11191.000000  1996.000000  666666.000000     2.000000     2.000000

           Recency     MntWines    MntFruits  MntMeatProducts  \
```

```
count  2240.000000  2240.000000  2240.000000       2240.000000
mean     49.109375   303.935714    26.302232        166.950000
std      28.962453   336.597393    39.773434        225.715373
min       0.000000     0.000000     0.000000          0.000000
25%      24.000000    23.750000     1.000000         16.000000
50%      49.000000   173.500000     8.000000         67.000000
75%      74.000000   504.250000    33.000000        232.000000
max      99.000000  1493.000000   199.000000       1725.000000

       MntFishProducts  …  NumWebVisitsMonth  AcceptedCmp3  AcceptedCmp4  \
count      2240.000000  …        2240.000000   2240.000000   2240.000000
mean         37.525446  …           5.316518      0.072768      0.074554
std          54.628979  …           2.426645      0.259813      0.262728
min           0.000000  …           0.000000      0.000000      0.000000
25%           3.000000  …           3.000000      0.000000      0.000000
50%          12.000000  …           6.000000      0.000000      0.000000
75%          50.000000  …           7.000000      0.000000      0.000000
max         259.000000  …          20.000000      1.000000      1.000000

       AcceptedCmp5  AcceptedCmp1  AcceptedCmp2     Complain  Z_CostContact  \
count   2240.000000   2240.000000   2240.000000  2240.000000         2240.0
mean       0.072768      0.064286      0.013393     0.009375            3.0
std        0.259813      0.245316      0.114976     0.096391            0.0
min        0.000000      0.000000      0.000000     0.000000            3.0
25%        0.000000      0.000000      0.000000     0.000000            3.0
50%        0.000000      0.000000      0.000000     0.000000            3.0
75%        0.000000      0.000000      0.000000     0.000000            3.0
max        1.000000      1.000000      1.000000     1.000000            3.0

       Z_Revenue     Response
count     2240.0  2240.000000
mean        11.0     0.149107
std          0.0     0.356274
min         11.0     0.000000
25%         11.0     0.000000
50%         11.0     0.000000
75%         11.0     0.000000
max         11.0     1.000000

[8 rows x 26 columns]
```

5 3. Handle missing values

[8]:
```
df.isnull().sum()
```

[8]:
```
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
```

[11]:
```
df['Income'] = df['Income'].fillna(df['Income'].mean())
```

[12]:
```
df.isnull().sum()
```

[12]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
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
```

5.1 let's find if we have duplicate values

[13]:
```
df.duplicated().sum()
```

[13]:
```
0
```

6 4. Feature Engineering:

back to table of content

[14]:
```
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               2240 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
```

6.1 Dt_Customer that indicates the date a customer joined the database is not parsed as DateTime

[19]:
```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')
```

[20]:
```
print("The newest customer's enrolment date in the records:",␣
 ↪max(df['Dt_Customer']))
print("The oldest customer's enrolment date in the records:",␣
 ↪min(df['Dt_Customer']))
```

```
The newest customer's enrolment date in the records: 2014-06-29 00:00:00
The oldest customer's enrolment date in the records: 2012-07-30 00:00:00
```

6.2 Extract the "Age" of a customer by the "Year_Birth" indicating the birth year of the respective person.

[22]:
```
df['Age'] = 2014 - df['Year_Birth']
```

6.3 Create another feature "Spent" indicating the total amount spent by the customer in various categories over the span of two years.

[23]:
```
df['spent'] =␣
 ↪df['MntWines']+df['MntFruits']+df['MntMeatProducts']+df['MntFishProducts']+df['MntSweetProd
```

6.4 Create another feature "Living_With" out of "Marital_Status" to extract the living situation of couples.

[24]:
```
df['Marital_Status'].value_counts()
```

[24]:
```
Marital_Status
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: count, dtype: int64
```

[27]:
```
df['living_with'] = df['Marital_Status'].replace({'Married':'partner',␣
↪'Together':'Partner', 'Absurd':'Alone', 'Widow':'Alone',
                                                'YOLO':'Alone', 'Divorced':
↪'Alone', 'Single':'Alone'})
```

6.5 Create a feature "Children" to indicate total children in a household that is, kids and teenagers.

[26]:
```
df['Children'] = df['Kidhome'] + df['Teenhome']
```

6.6 To get further clarity of household, Creating feature indicating "Family_Size"

[30]:
```
# Define a function to map living arrangements to family sizes
def family_size(living_with):
    if living_with == 'Alone':
        return 1
    elif living_with == 'Partner':
        return 2
    else:
        return 0  # Handle unknown living arrangements gracefully

# Apply the function to create a new column 'Family_Size'
df['Family_Size'] = df['living_with'].apply(family_size) + df['Children']
```

6.7 Create a feature "Is_Parent" to indicate parenthood status

[31]:
```
df['Is_Parent'] = np.where(df.Children > 0, 1, 0)
```

6.8 Segmenting education levels in three groups

[32]:
```
df['Education'].value_counts()
```

[32]:
```
Education
Graduation    1127
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: count, dtype: int64
```

[34]:
```
df['Education'] = df['Education'].replace({'Basic':'Undergraduate', '2n Cycle':
↪'Undergraduate', 'Graduation':'Graduate', 'Master':'Postgraduate', 'PhD':
↪'Postgraduate'})
```

6.9 Dropping some of the redundant features

[35]:
```
to_drop = ['Marital_Status', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue',␣
↪'Year_Birth', 'ID']
df = df.drop(to_drop, axis=1)
```

[36]:
```
df.head()
```

[36]:
```
     Education   Income  Kidhome  Teenhome  Recency  MntWines  MntFruits  \
0     Graduate  58138.0        0         0       58       635         88
1     Graduate  46344.0        1         1       38        11          1
2     Graduate  71613.0        0         0       26       426         49
3     Graduate  26646.0        1         0       26        11          4
4  Postgraduate  58293.0        1         0       94       173         43

   MntMeatProducts  MntFishProducts  MntSweetProducts  …  AcceptedCmp1  \
0              546              172                88  …             0
1                6                2                 1  …             0
2              127              111                21  …             0
3               20               10                 3  …             0
4              118               46                27  …             0

   AcceptedCmp2  Complain  Response  Age  spent  living_with  Children  \
0             0         0         1   57   1617        Alone         0
1             0         0         0   60     27        Alone         2
2             0         0         0   49    776      Partner         0
3             0         0         0   30     53      Partner         1
4             0         0         0   33    422      partner         1

   Family_Size  Is_Parent
0            1          0
```

```
1            3          1
2            2          0
3            3          1
4            1          1

[5 rows x 29 columns]
```

7 data analysis and visualization

[37]:
```
df.shape
```

[37]:
```
(2240, 29)
```

[38]:
```
df.info()
```

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

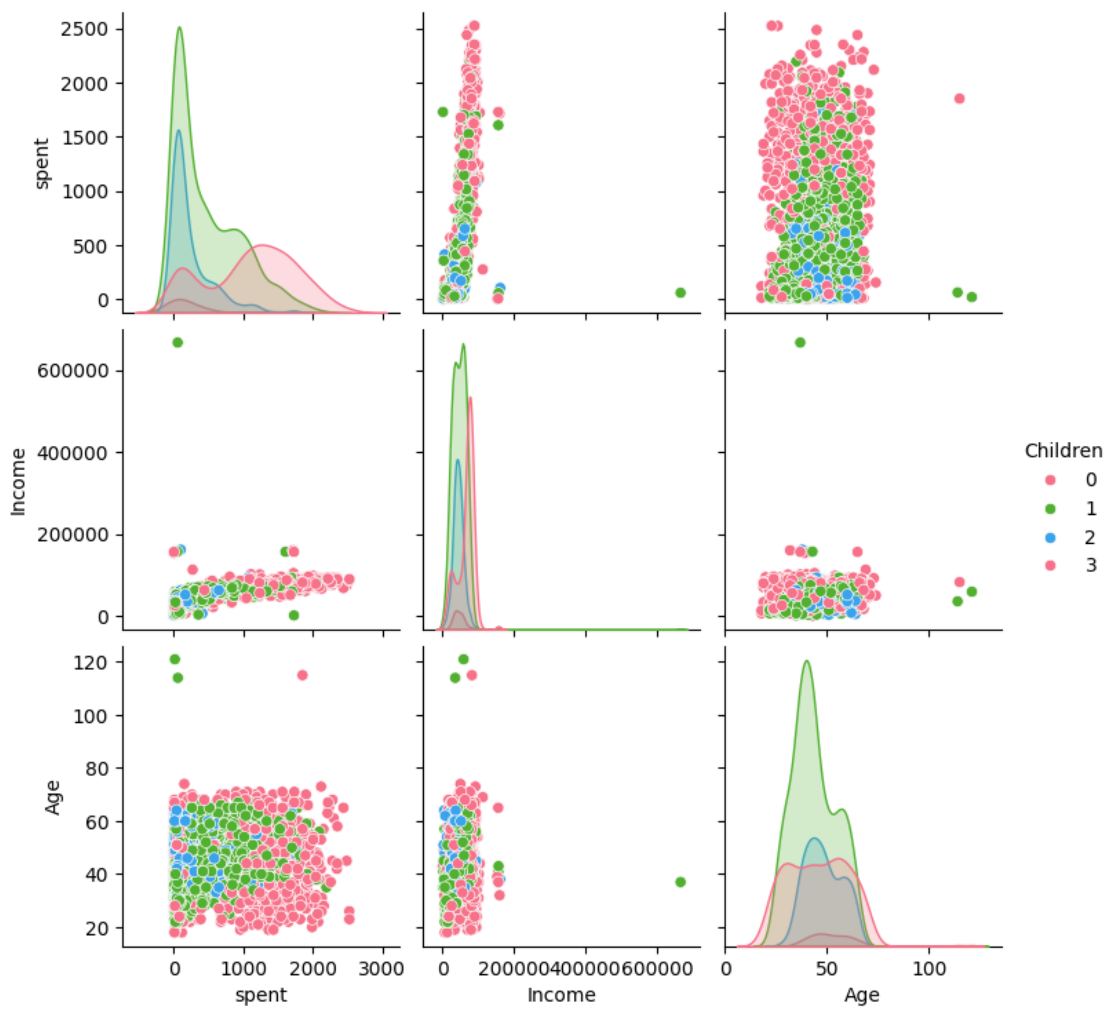
```
 26   Children               2240 non-null   int64
 27   Family_Size            2240 non-null   int64
 28   Is_Parent              2240 non-null   int32
dtypes: float64(1), int32(1), int64(25), object(2)
memory usage: 498.9+ KB
```

[39]:
```python
df.describe(include=object).T
```

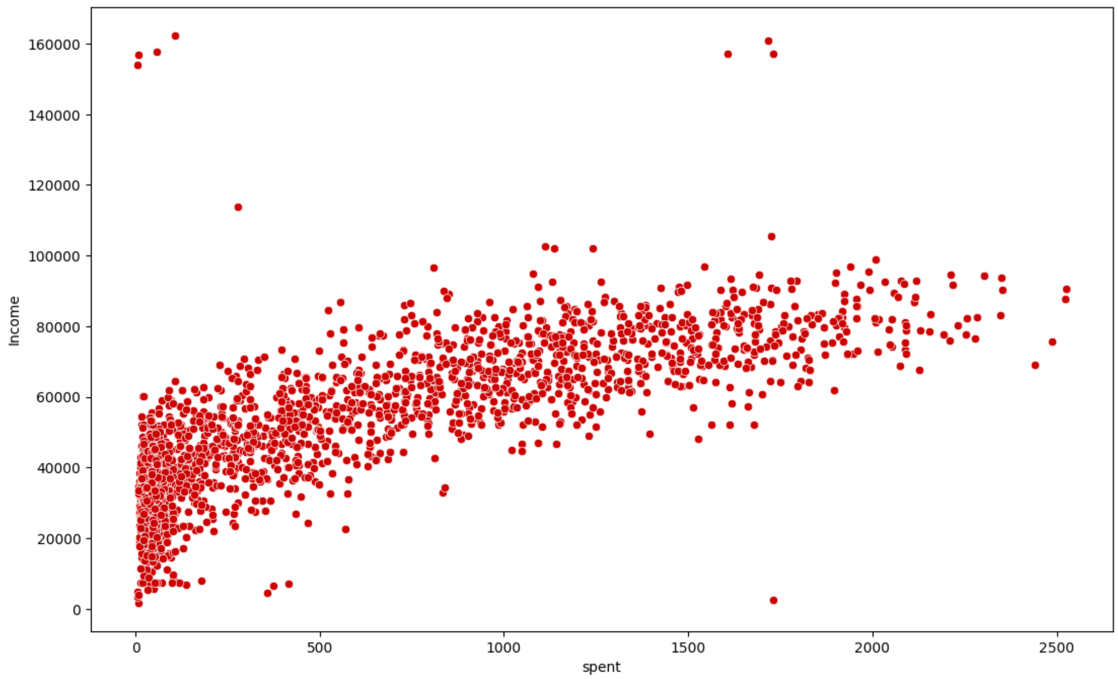
[39]:

	count	unique	top	freq
Education	2240	3	Graduate	1127
living_with	2240	3	partner	864

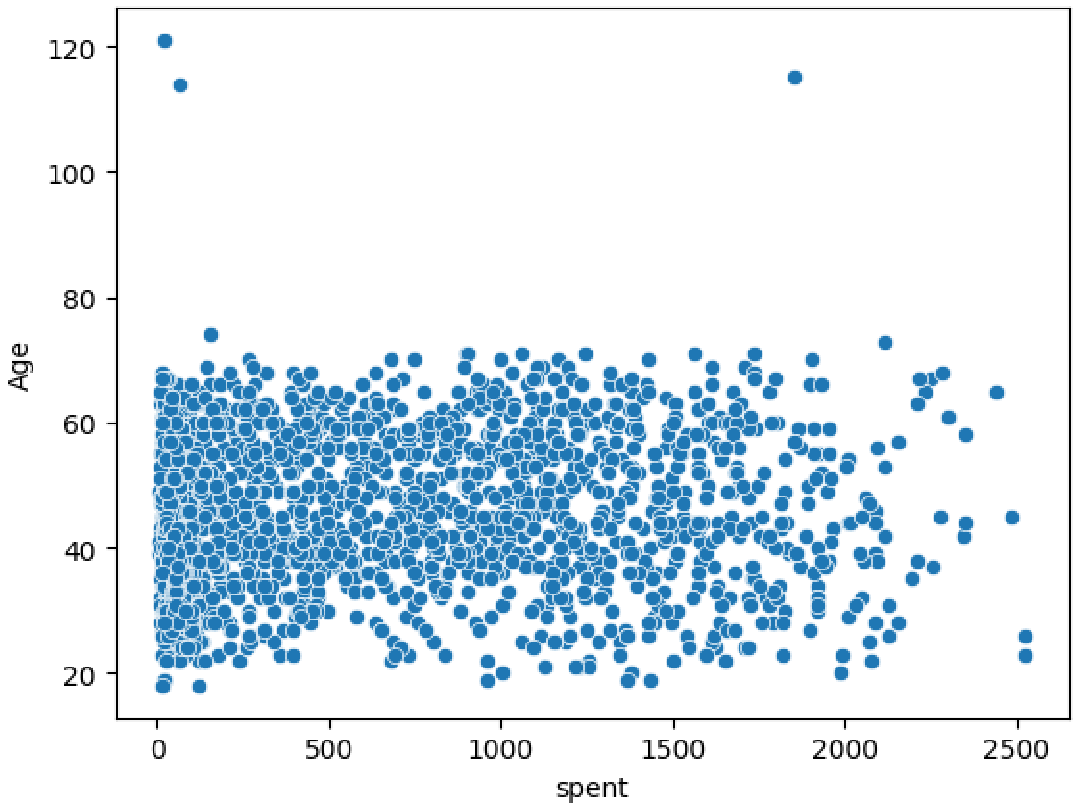
[41]:
```python
sns.pairplot(df , vars=['spent','Income','Age'] , hue='Children',␣
 ↪palette='husl');
```



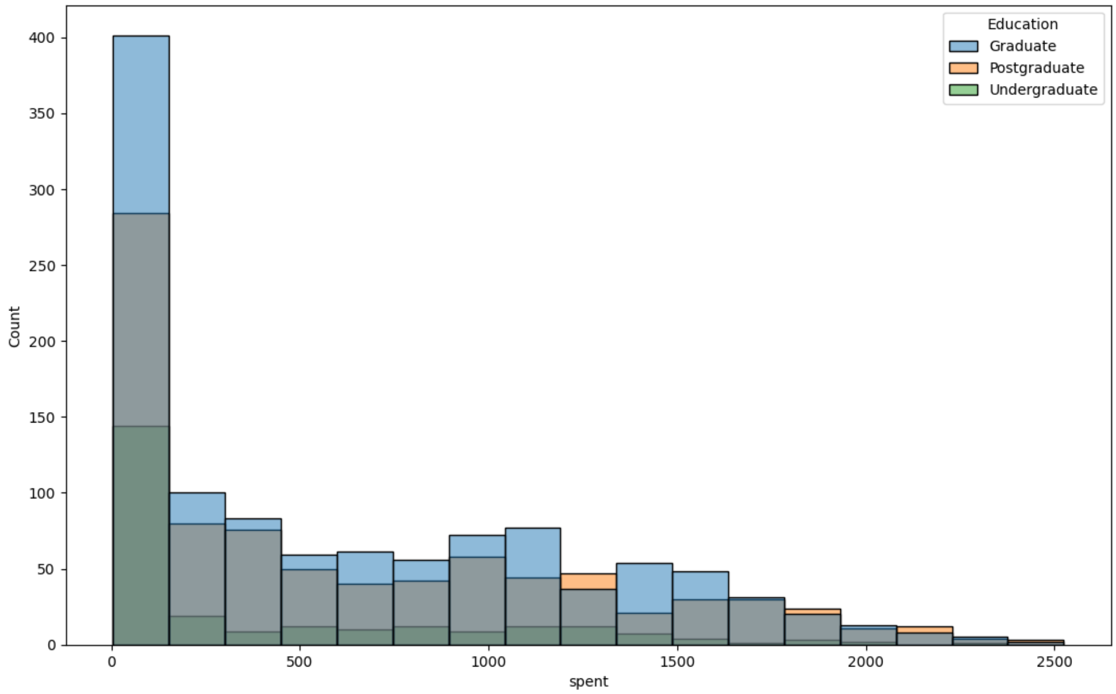
[43]:
```
plt.figure(figsize=(13,8))
sns.scatterplot(x=df[df['Income']<600000]['spent'],
 ↪y=df[df['Income']<600000]['Income'], color='#cc0000');
```



[45]:
```
sns.scatterplot(x=df['spent'], y=df['Age']);
```

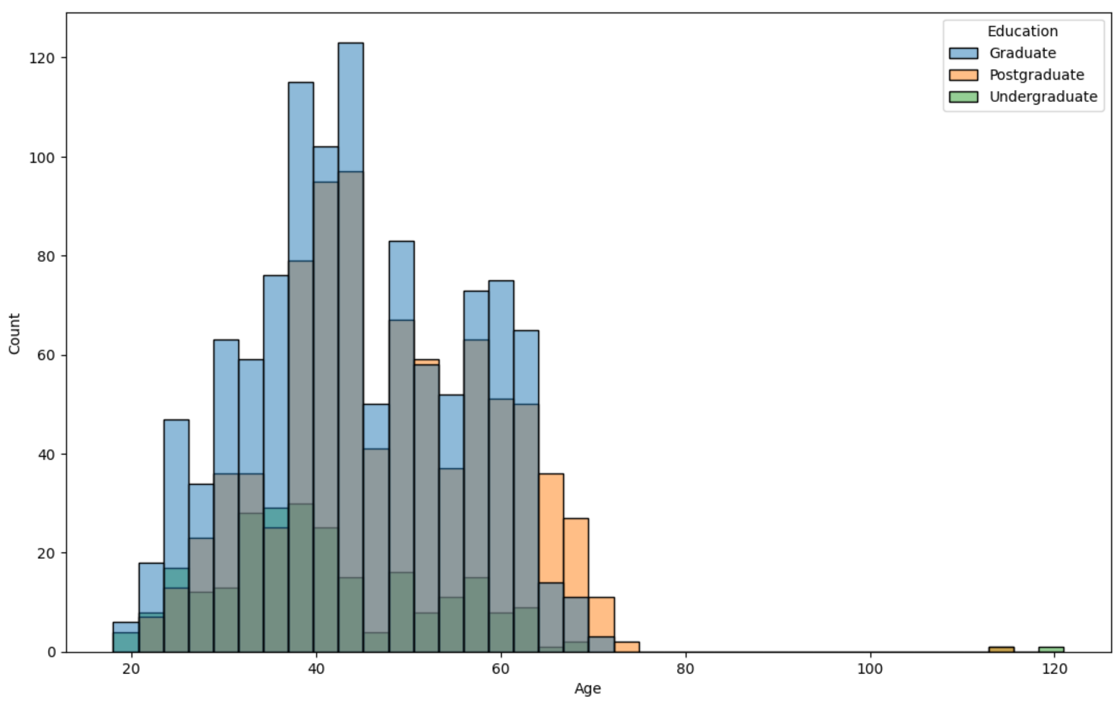


[47]:
```
plt.figure(figsize=(13,8))
sns.histplot(x=df['spent'], hue=df['Education']);
```



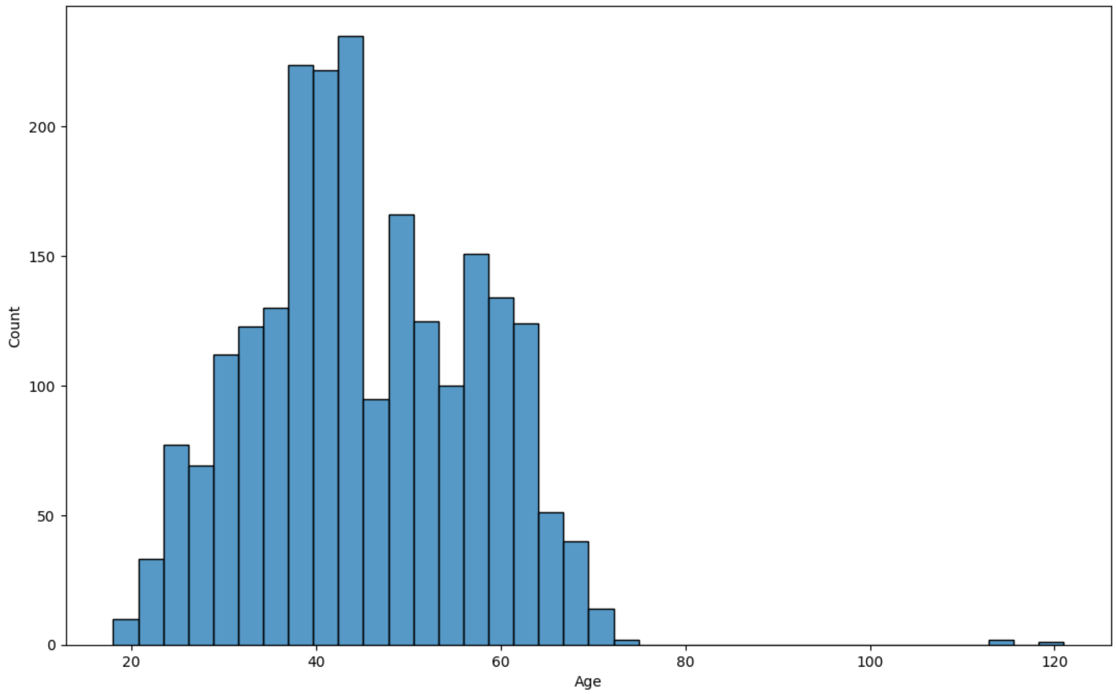
[48]:

```
plt.figure(figsize=(13,8))
sns.histplot(x=df['Age'], hue=df['Education']);
```

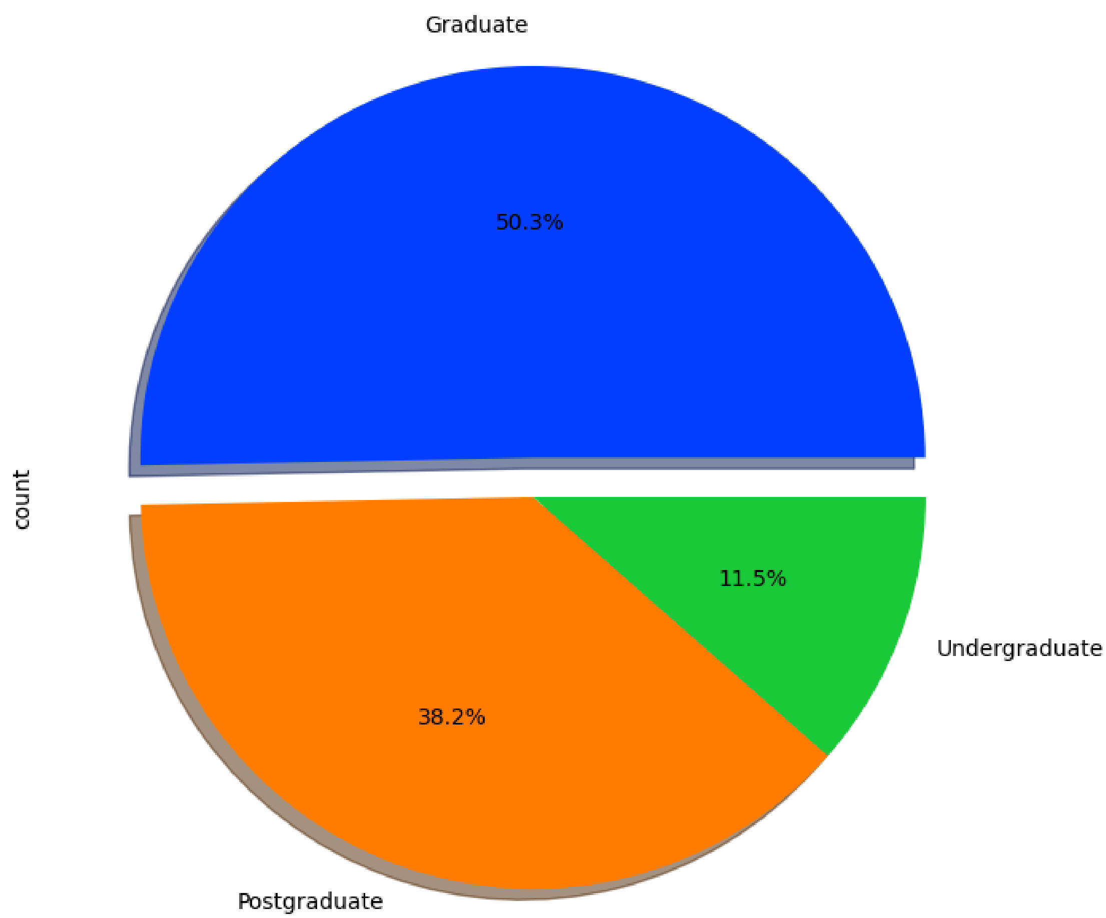


[49]:
```
plt.figure(figsize=(13,8))
sns.histplot(x=df['Age'])
```

[49]: <Axes: xlabel='Age', ylabel='Count'>

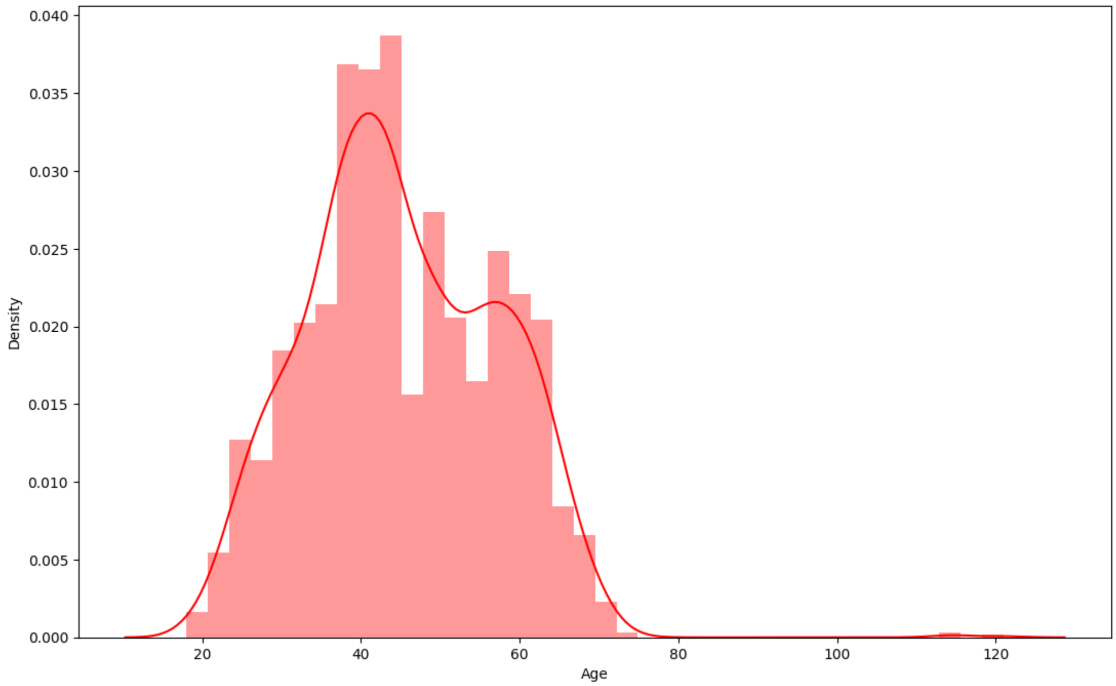


[50]:
```
df['Education'].value_counts().plot.pie(explode=[0.1,0,0], autopct='%1.1f%%',␣
↪shadow=True, figsize=(8,8), colors=sns.color_palette('bright'));
```

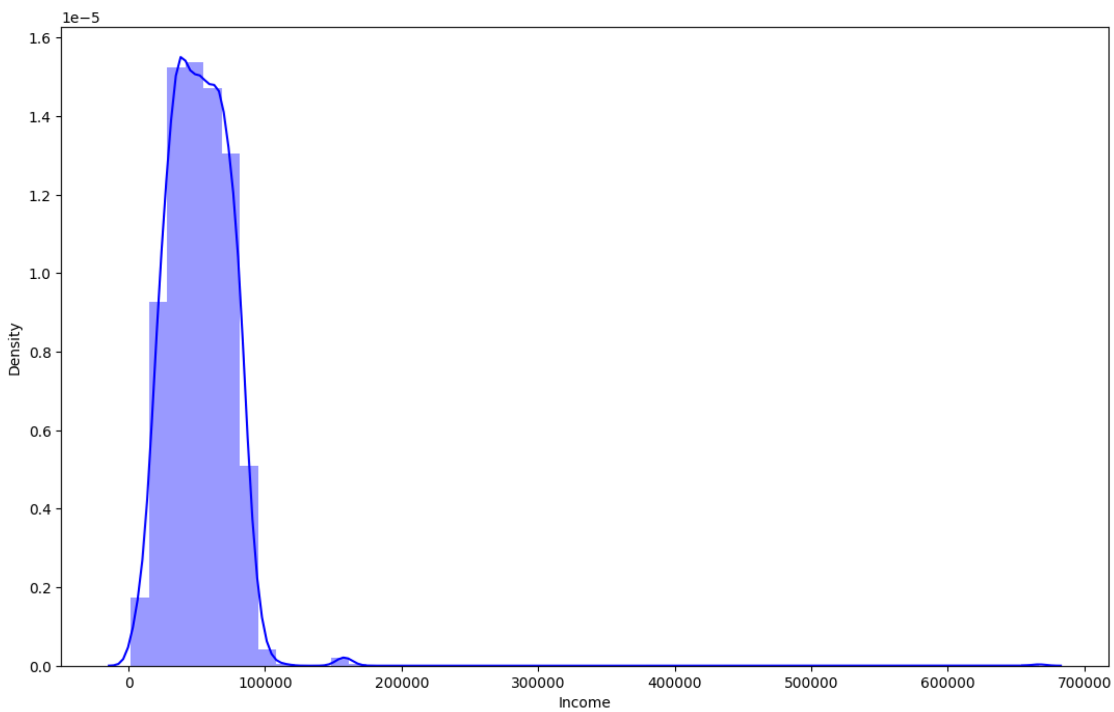


8 6. outlier detection:

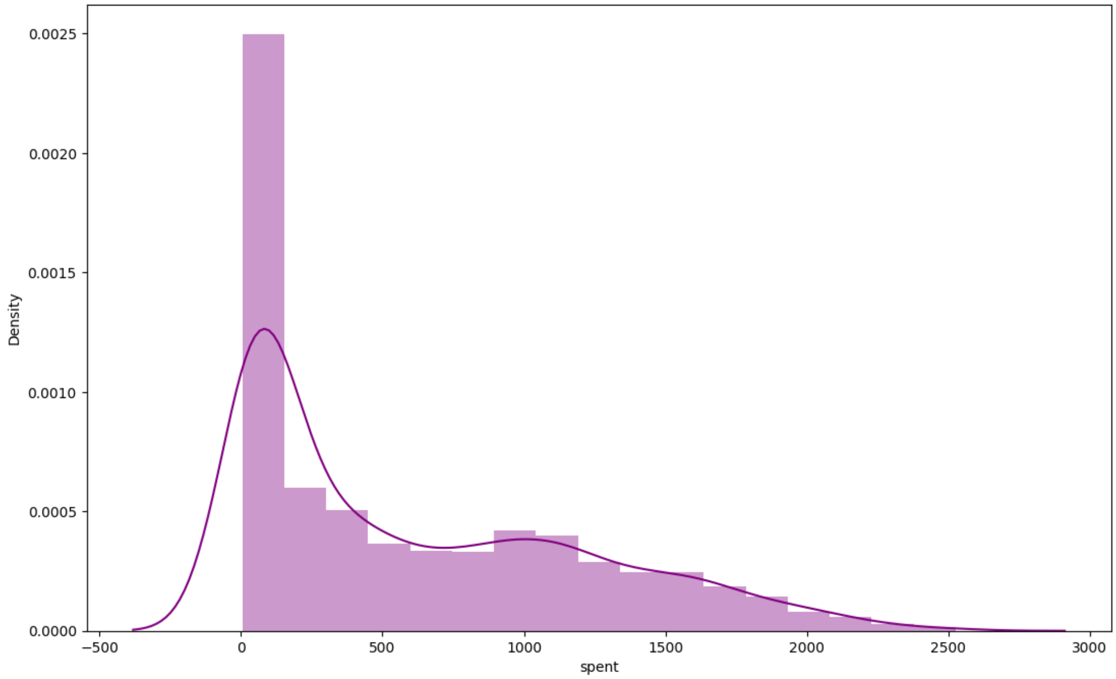
[56]:
```
plt.figure(figsize=(13,8))
sns.distplot(df.Age, color='red');
```



[59]:
```
plt.figure(figsize=(13,8))
sns.distplot(df.Income, color='blue');
```



[60]:
```python
plt.figure(figsize=(13,8))
sns.distplot(df.spent, color='purple');
```



Another way of visualising outliers is using boxplots and whiskers, which provides the quantiles (box) and inter-quantile range (whiskers), with the outliers sitting outside the error bars (whiskers).

All the dots in the plot below are outliers according to the quantiles + 1.5 IQR rule

[63]:
```python
fig = make_subplots(rows=1, cols=3)

fig.add_trace(go.Box(y=df['Age'], notched=True, name='Age', marker_color =␣
 ↪'#6699ff',
                     boxmean=True, boxpoints='suspectedoutliers'), 1, 2)

fig.add_trace(go.Box(y=df['Income'], notched=True, name='Income', marker_color␣
 ↪= '#ff0066',
                     boxmean=True, boxpoints='suspectedoutliers'), 1, 1)

fig.add_trace(go.Box(y=df['spent'], notched=True, name='spent', marker_color =␣
 ↪'lightseagreen',
                     boxmean=True, boxpoints='suspectedoutliers'), 1, 3)

fig.update_layout(title_text='<b>Box Plots for Numerical Variables<b>')

fig.show()
```

[71]:
```
numerical = ['Income', 'Recency', 'Age', 'spent']
```

[72]:
```
def detect_outliers(d):
  for i in d:
    Q3, Q1 = np.percentile(df[i], [75 ,25])
    IQR = Q3 - Q1

    ul = Q3+1.5*IQR
    ll = Q1-1.5*IQR

    outliers = df[i][(df[i] > ul) | (df[i] < ll)]
    print(f'*** {i} outlier points***', '\n', outliers, '\n')
```

[73]:
```
detect_outliers(numerical)
```

```
*** Income outlier points***
 164     157243.0
617     162397.0
655     153924.0
687     160803.0
1300    157733.0
1653    157146.0
2132    156924.0
2233    666666.0
Name: Income, dtype: float64

*** Recency outlier points***
 Series([], Name: Recency, dtype: int64)

*** Age outlier points***
 192    114
239    121
339    115
Name: Age, dtype: int64

*** spent outlier points***
 1179    2525
1492    2524
1572    2525
Name: spent, dtype: int64
```

8.1 We will delete some of the outlier points.

[74]:
```
data = df[(df['Age']<100)]
data = df[(df['Income']<600000)]
```

```
[75]: data.shape
```

```
[75]: (2239, 29)
```

9 7. categorical variable Encoding:

```
[78]: import pandas as pd

def find_categorical_variables(data):
    categorical_vars = []
    for col in data.columns:
        if data[col].dtype == 'object':  # Check if the column has 'object'␣
 ↪datatype
            categorical_vars.append(col)
    return categorical_vars

# Example usage:
# Assuming 'df' is your DataFrame
categorical_variables = find_categorical_variables(df)
print("Categorical Variables:", categorical_variables)
```

```
Categorical Variables: ['Education', 'living_with']
```

```
[79]: df['living_with'].unique()
```

```
[79]: array(['Alone', 'Partner', 'partner'], dtype=object)
```

9.1 Since the education is a ordinal variable, we will encode it with ordinal numbers.

```
[91]: df['Education'] = df['Education'].map({'Undergraduate':0,'Graduate':1,␣
 ↪'Postgraduate':2})
```

```
[92]: df['living_with'] = df['living_with'].map({'Alone':0,'Partner':1})
```

```
[93]: df.dtypes
```

```
[93]: Education                int64
Income                 float64
Kidhome                  int64
Teenhome                 int64
Recency                  int64
MntWines                 int64
MntFruits                int64
MntMeatProducts          int64
MntFishProducts          int64
```

```
MntSweetProducts         int64
MntGoldProds             int64
NumDealsPurchases        int64
NumWebPurchases          int64
NumCatalogPurchases      int64
NumStorePurchases        int64
NumWebVisitsMonth        int64
AcceptedCmp3             int64
AcceptedCmp4             int64
AcceptedCmp5             int64
AcceptedCmp1             int64
AcceptedCmp2             int64
Complain                 int64
Response                 int64
Age                      int64
spent                    int64
living_with            float64
Children                 int64
Family_Size              int64
Is_Parent                int32
dtype: object
```

[94]:
```
df.head()
```

[94]:
```
   Education   Income  Kidhome  Teenhome  Recency  MntWines  MntFruits  \
0          1  58138.0        0         0       58       635         88
1          1  46344.0        1         1       38        11          1
2          1  71613.0        0         0       26       426         49
3          1  26646.0        1         0       26        11          4
4          2  58293.0        1         0       94       173         43

   MntMeatProducts  MntFishProducts  MntSweetProducts  …  AcceptedCmp1  \
0              546              172                88  …             0
1                6                2                 1  …             0
2              127              111                21  …             0
3               20               10                 3  …             0
4              118               46                27  …             0

   AcceptedCmp2  Complain  Response  Age  spent  living_with  Children  \
0             0         0         1   57   1617          0.0         0
1             0         0         0   60     27          0.0         2
2             0         0         0   49    776          1.0         0
3             0         0         0   30     53          1.0         1
4             0         0         0   33    422          NaN         1

   Family_Size  Is_Parent
0            1          0
```

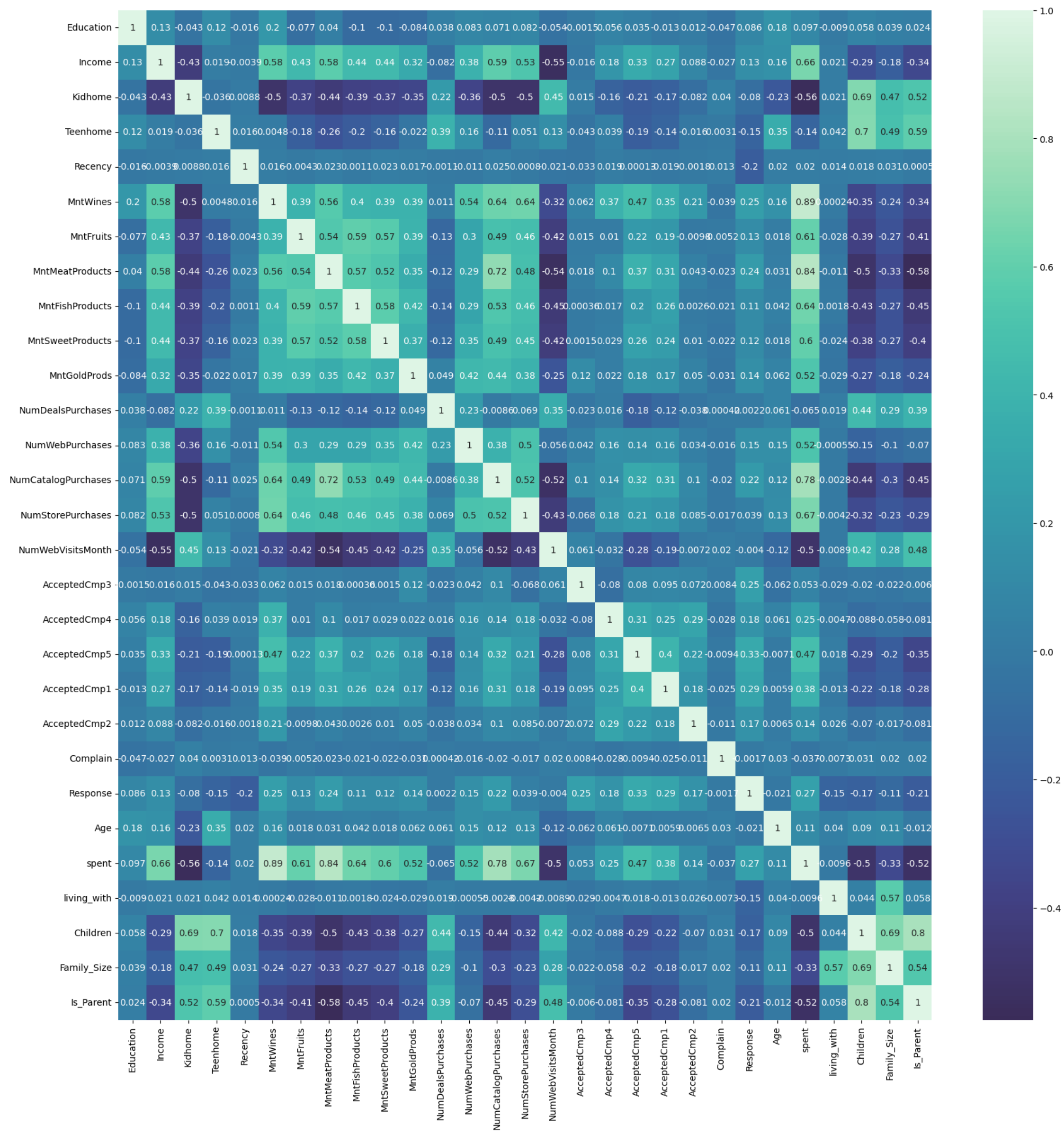
```
1          3          1
2          2          0
3          3          1
4          1          1

[5 rows x 29 columns]
```

[95]:
```python
correlation = df.corr()

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

[95]: <Axes: >



10 8. Feature Scaling:

In this section, numerical features are scaled.

StandardScaler $= \frac{x-\mu}{s}$

[96]:
```python
df_old = df.copy()
```

[97]:
```python
# creating a subset of dataframe by dropping the features on deals accepted and␣
↪promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',␣
↪'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
df = df.drop(cols_del, axis=1)
```

[101]:
```python
scaler = StandardScaler()
df = pd.DataFrame(scaler.fit_transform(df),
           columns = df.columns)

df.head()
```

[101]:
```
   Education    Income   Kidhome  Teenhome   Recency  MntWines  MntFruits  \
0  -0.410013  0.235327 -0.825218 -0.929894  0.307039  0.983781   1.551577
1  -0.410013 -0.235826  1.032559  0.906934 -0.383664 -0.870479  -0.636301
2  -0.410013  0.773633 -0.825218 -0.929894 -0.798086  0.362723   0.570804
3  -0.410013 -1.022732  1.032559 -0.929894 -0.798086 -0.870479  -0.560857
4   1.123256  0.241519  1.032559 -0.929894  1.550305 -0.389085   0.419916

   MntMeatProducts  MntFishProducts  MntSweetProducts  …  NumWebPurchases  \
0         1.679702         2.462147          1.476500  …         1.409304
1        -0.713225        -0.650449         -0.631503  …        -1.110409
2        -0.177032         1.345274         -0.146905  …         1.409304
3        -0.651187        -0.503974         -0.583043  …        -0.750450
4        -0.216914         0.155164         -0.001525  …         0.329427

   NumCatalogPurchases  NumStorePurchases  NumWebVisitsMonth       Age  \
0             2.510890          -0.550785           0.693904  0.985345
1            -0.568720          -1.166125          -0.130463  1.235733
2            -0.226541           1.295237          -0.542647  0.317643
3            -0.910898          -0.550785           0.281720 -1.268149
4             0.115638           0.064556          -0.130463 -1.017761

      spent  living_with  Children  Family_Size  Is_Parent
0  1.679417    -0.853606 -1.264505    -0.753390  -1.584605
1 -0.961275    -0.853606  1.396361     1.075980   0.631072
```

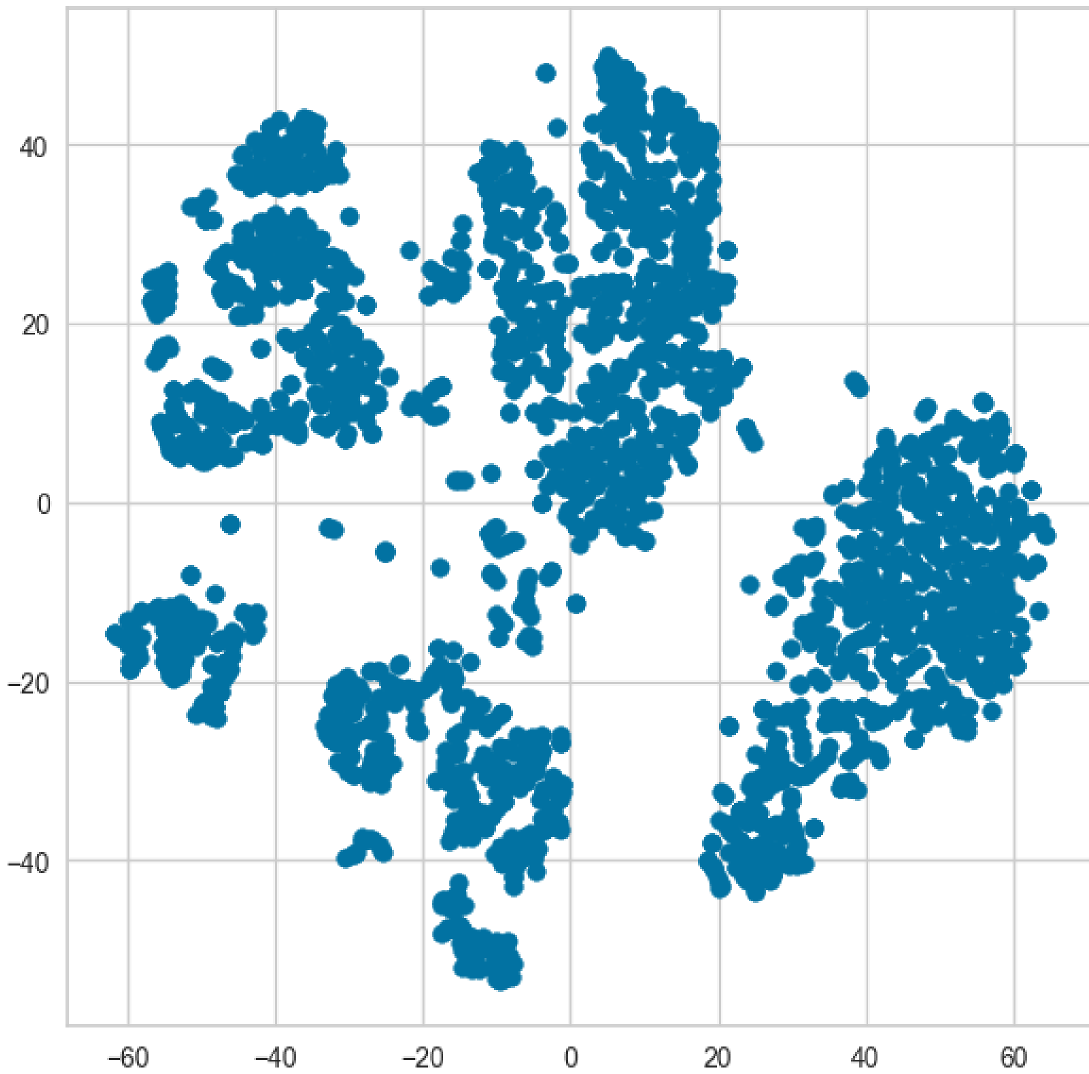
```
2  0.282673     1.171501 -1.264505     0.161295  -1.584605
3 -0.918094     1.171501  0.065928     1.075980   0.631072
4 -0.305254          NaN  0.065928    -0.753390   0.631072

[5 rows x 22 columns]
```

11 10. Clustering analysis:

We will be using T-distributed Stochastic Neighbor Embedding. It helps in visualizing high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the values to low-dimensional embedding.

[115]:
```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0)
tsne_df = model.fit_transform(df)
plt.figure(figsize=(7, 7))
plt.scatter(tsne_df[:, 0], tsne_df[:, 1])
plt.show()
```



[129]:
```
# Create a Plotly figure
fig = px.scatter(x=tsne_df[:, 0], y=tsne_df[:, 1], title='TSNE Scatter Plot')

# Show the figure
fig.show()
```

12 KMeans Clustering:

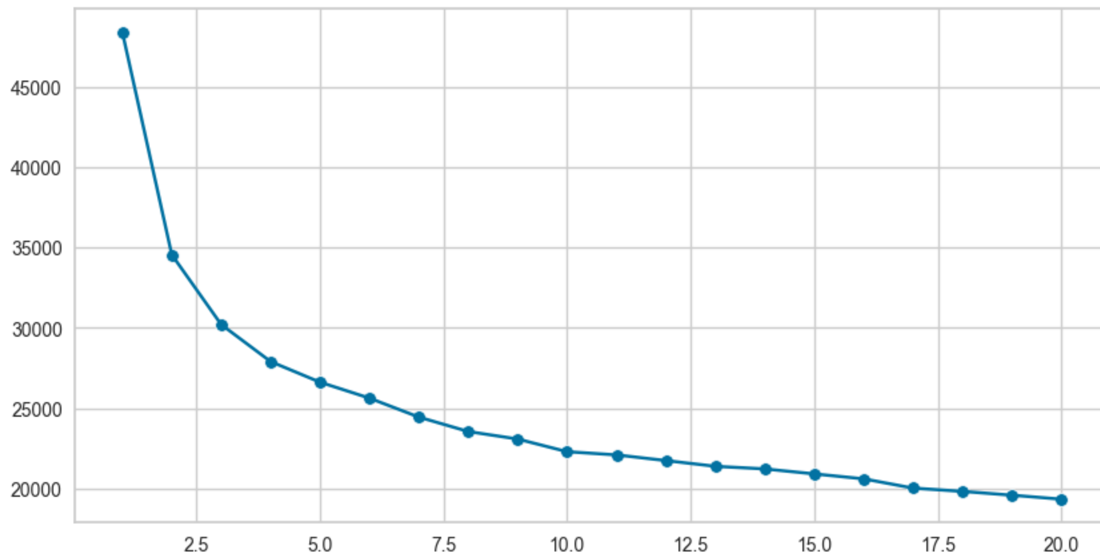
can also be used to cluster the different points in a plane.

[116]:
```
error = []
for n_clusters in range(1, 21):
    model = KMeans(init='k-means++',
```

```
                    n_clusters=n_clusters,
                    max_iter=500,
                    random_state=22)
    model.fit(df)
    error.append(model.inertia_)
```

Here inertia is nothing but the sum of squared distances within the clusters.

[117]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(x=range(1, 21), y=error)
sns.scatterplot(x=range(1, 21), y=error)
plt.show()
```

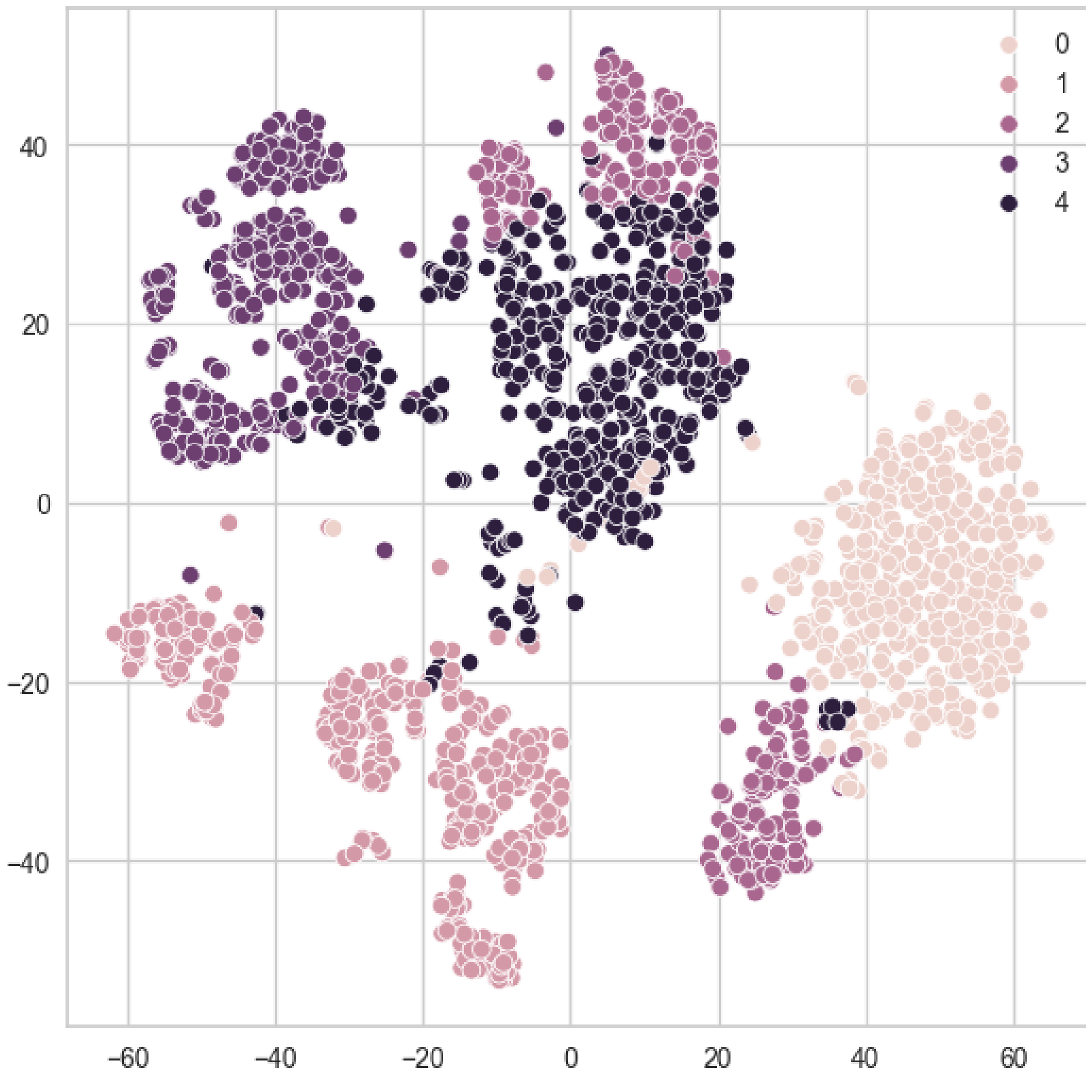


Here by using the elbow method we can say that k = 6 is the optimal number of clusters that should be made as after k = 6 the value of the inertia is not decreasing drastically.

[120]:
```
# create clustering model with optimal k=5
model = KMeans(init='k-means++',
               n_clusters=5,
               max_iter=500,
               random_state=22)
spent = model.fit_predict(df)
```

Scatterplot will be used to see all the 6 clusters formed by KMeans Clustering.

[122]:
```
plt.figure(figsize=(7, 7))
sns.scatterplot(x=tsne_df[:, 0], y=tsne_df[:, 1], hue=spent)
plt.show()
```


0
1
2
3
4
40
20
0
−20
−40
−60
−40
−20
0
20
40
60