Enhancing Marketing Strategies with Customer Personality Analysis

Problem Statement

The problem we propose to solve is to enhance marketing strategies by leveraging customer personality analysis. Traditional marketing approaches often rely on generic segmentation based on demographics or historical purchase behavior. However, this approach fails to capture the nuances of individual customer preferences and behaviors, limiting the effectiveness of marketing campaigns. By leveraging customer personality analysis, we aim to develop a machine learning algorithm model that will target specific customers and personalize the marketing approach.

Dataset Description

The dataset Customer Personality Analysis from Kaggle provides valuable insights into customer behavior and preferences. It contains information about customers, including their purchasing history. The dataset is a comprehensive collection of customer attributes, providing a rich source of information for understanding and analyzing customer behavior.

Data preparation and preprocessing

```
# Import libraries
import sys
import warnings

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import colors
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from yellowbrick.cluster import KElbowVisualizer

if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)
```

```
# Load dataset into DataFrame data
dataset = pd.read_csv('marketing_campaign.csv', sep="\t")
```

```
# Check number of columns and rows
dataset.shape
```

```
(2240, 29)
```

This dataset contains 29 variables and 2240 observations about different customers.

```
# Check columns name
dataset.columns
```

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
```

```
           'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
           'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
           'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
           'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
          dtype='object')
```

```
# Look for description of number of values in each column
dataset.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
```

The Income variable has 2016 where other has 2040 values, which means in Income variable there is missing values, then we have to fill it with NA

```
# Fill NA where there is missing values in Income variable
dataset['Income'] = dataset['Income'].fillna(dataset['Income'].mean())
```

```
# Let's check whether there are no any other missing values
dataset.isnull().sum()
```

```
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
```

```
# Check for duplicate
dataset.duplicated().sum()
```

```
0
```

```
dataset.head(2)
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebV
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	

2 rows × 29 columns

From the above output we can see that there are some variable that need some data cleaning, we are going to modify some features as well as create new ones for further analysis and modeling

Age: The age of the customers are the age in 2014 as it's the last record we have (6th Dec 2014) or we can round that up to 2023

Marital_Status: Narrowing down to 2 categories

Total_Children: Merging Kidhome and Teenhome columns into 1 column which describes the number of children living in the household

Dt_Customer: Extracting new features out of dates to make Day, Dayofweek, Month, and Year features

Education: Narrowing down to 3 categories

Is_Parent: Referring to the parenthood status

Total_Spent: Customer's total spent on products

Feature engineering

```
# Round age up to 2023
dataset['Age'] = 2023 - dataset["Year_Birth"]
```

```
dataset['Age'].max()
```

```
130
```

possible outliers in Age variable

```
# Let's classify values in Marital status variable into two categories: Relationship and Single
dataset['Marital_Status'] = dataset['Marital_Status'].replace(['Married', 'Together'],'Relationship')
dataset['Marital_Status'] = dataset['Marital_Status'].replace(['Divorced', 'Widow', 'Alone', 'YOLO', 'Absurd'],'Single')
```

```
dataset['Marital_Status'].value_counts()
```

```
Relationship    1444
Single           796
Name: Marital_Status, dtype: int64
```

```
from pandas.core.internals.construction import dataclasses_to_dicts
# Let's merge some features which are in same category into one single column
# All Kidhome and Teenhome will be into one single feature ChildrenHome
dataset['ChildrenHome'] = dataset['Kidhome'] + dataset['Teenhome']
# All amount spent in MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts and MntGoldProds into one sin
dataset['AmountSpent'] = dataset['MntWines'] + dataset['MntFruits'] + dataset['MntMeatProducts'] + dataset['MntFishProducts
# All promotion made and accepted in AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5 and Response into
dataset['TotalAcceptedCmp'] = dataset['AcceptedCmp1'] + dataset['AcceptedCmp2'] + dataset['AcceptedCmp3'] + dataset['Accept
# All purchases in NumWebPurchases, NumCatalogPurchases, NumStorePurchases and NumDealsPurchases into one single feature Nu
dataset['NumTotalPurchases'] = dataset['NumWebPurchases'] + dataset['NumCatalogPurchases'] + dataset['NumStorePurchases'] +
```

```
dataset['ChildrenHome'].value_counts()
```

```
1    1128
0     638
2     421
3      53
Name: ChildrenHome, dtype: int64
```

```
dataset['AmountSpent'].max()
```

```
2525
```

```
dataset['TotalAcceptedCmp'].max()
```

```
5
```

```
dataset['NumTotalPurchases'].max()
```

```
44
```

```
# parse Dt_Customer values into DateTime format
dataset['Dt_Customer'] = pd.to_datetime(dataset.Dt_Customer)
# Initialize the first day to be able to count days of customers engagement
dataset['first_day'] = '01-01-2023'
# Convert First_day value into DateTime format
dataset['first_day'] = pd.to_datetime(dataset.first_day)
# Count days of customer engagement
dataset['DaysEngaged'] = (dataset['first_day'] - dataset['Dt_Customer']).dt.days
```

```
<ipython-input-123-f171a89967b2>:2: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) w
  dataset['Dt_Customer'] = pd.to_datetime(dataset.Dt_Customer)
```

```
dataset['DaysEngaged'].min()
```

```
2948
```

```
# Let's classify values in Education variable into two categories: Post Graduate and Under Graduate
dataset['Education'] = dataset['Education'].replace(['PhD','2n Cycle','Graduation', 'Master'],'Post Graduate')
dataset['Education'] = dataset['Education'].replace(['Basic'], 'Under Graduate')
```

```
dataset['Education'].value_counts()
```

```
Post Graduate     2186
Under Graduate      54
Name: Education, dtype: int64
```

```
# For clarity, renaming some columns
dataset = dataset.rename(columns={"MntWines": "Wines", "MntFruits": "Fruits", "MntMeatProducts": "Meat", "MntSweetProducts"
```

```
# Feature pertaining to parenthood
dataset["Is_Parent"] = np.where(dataset["ChildrenHome"] > 0, 1, 0)
```

```
# Dropping some redundant features
to_drop = ["Dt_Customer", "Z_CostContact", "Z_Revenue"]
dataset = dataset.drop(to_drop, axis=1)
```

```
dataset.describe()
```

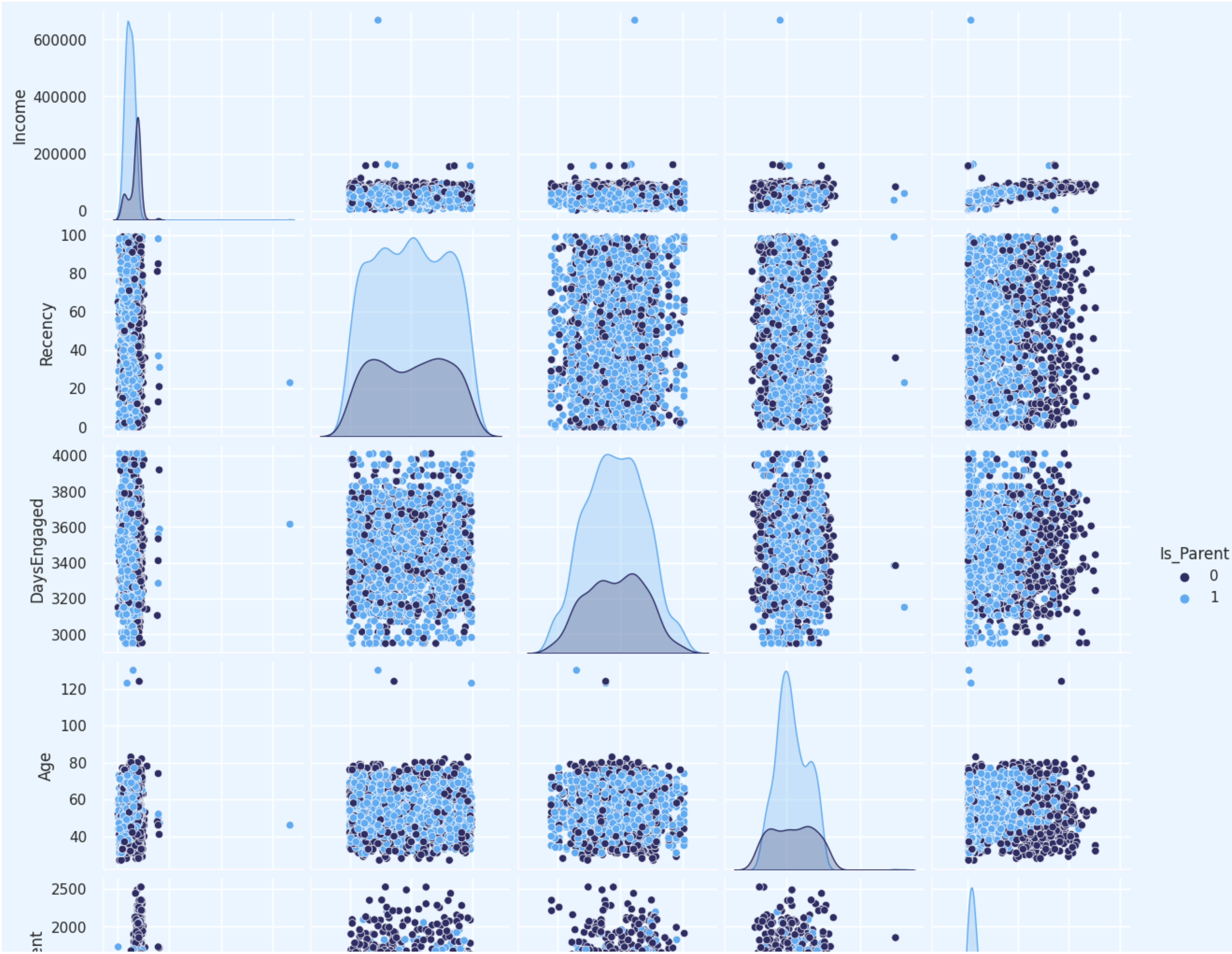
	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Me
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.0000
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.9500
std	3246.662198	11.984069	25037.797168	0.538398	0.544538	28.962453	336.597393	39.773434	225.7153
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	2828.250000	1959.000000	35538.750000	0.000000	0.000000	24.000000	23.750000	1.000000	16.0000
50%	5458.500000	1970.000000	51741.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.0000
75%	8427.750000	1977.000000	68289.750000	1.000000	1.000000	74.000000	504.250000	33.000000	232.0000
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.0000

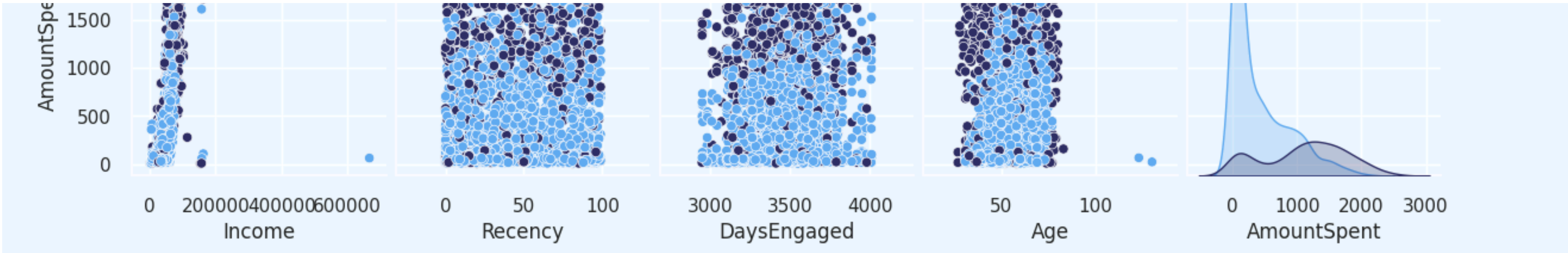
8 rows × 31 columns

```
# Initializing Color pallets
sns.set(rc={"axes.facecolor": "#edf9ff", "figure.facecolor": "#edf9ff"})
pallet = ["#2f2f68", "#6f729e", "#b1b2d6", "#c9c0b9", "#788a9f", "#60abf3"]
cmap = colors.ListedColormap(["#2f2f68", "#6f729e", "#D6B2B1", "#b1b2d6"])
pal = ["#2f2f68", "#c9c0b9", "#788a9f", "#60acf3"]
```

```
# Assuming you have a DataFrame called 'data' with the required columns

To_Plot = ["Income", "Recency", "DaysEngaged", "Age", "AmountSpent", "Is_Parent"]

print("Relative Plot Of Some Selected Features: A Data Subset")
plt.figure()
sns.pairplot(dataset[To_Plot], hue="Is_Parent", palette=["#2f2f68", "#60abf3"])
plt.show()
```

Relative Plot Of Some Selected Features: A Data Subset
<Figure size 800x550 with 0 Axes>





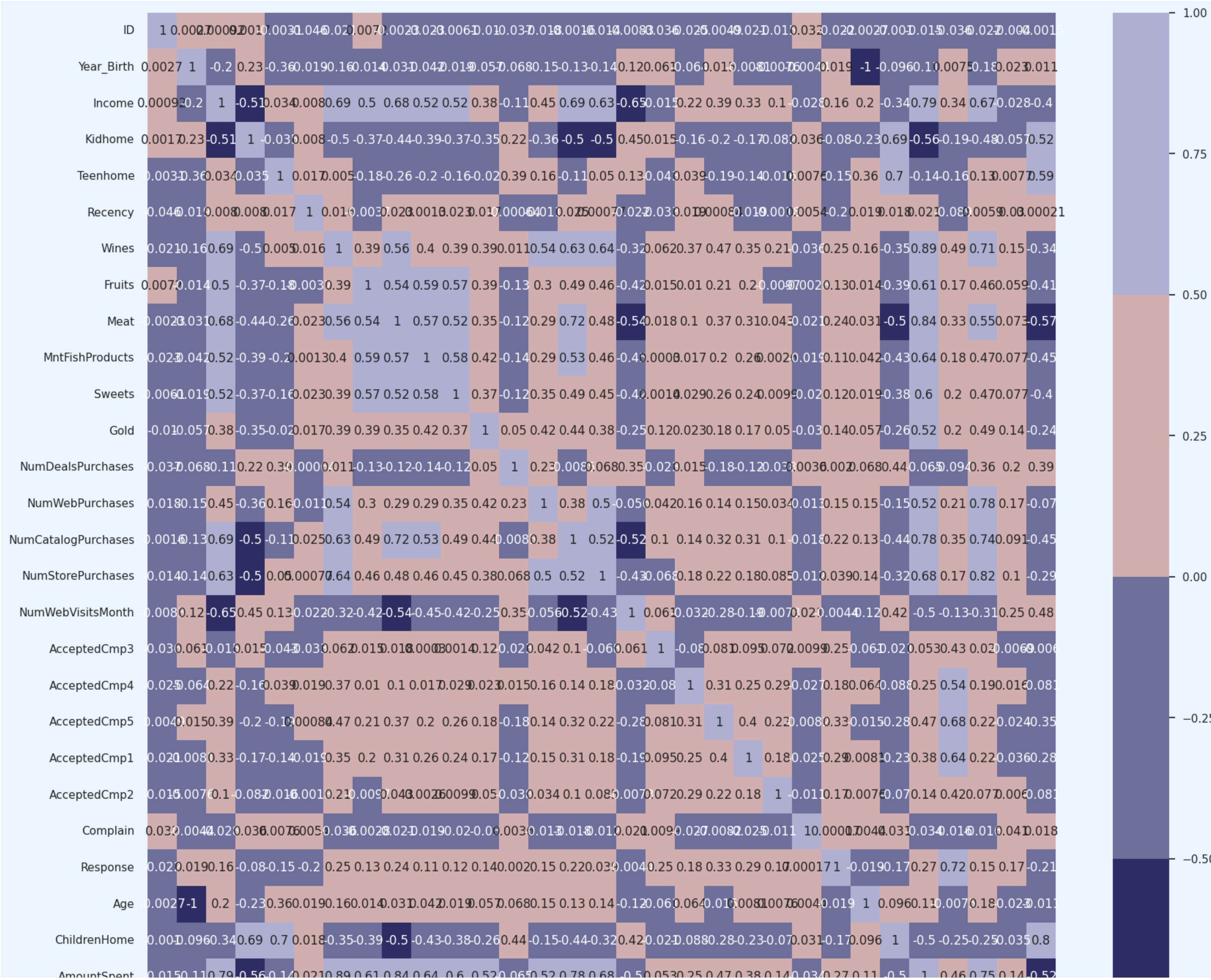

```
# Assuming you have a DataFrame called 'data' with the required columns

# Dropping the outliers by setting a cap on Age and Income
dataset = dataset[(dataset["Age"] < 90)]
dataset = dataset[(dataset["Income"] < 600000)]

print(f"The total data points after removing the outliers are: {len(dataset)}")
```

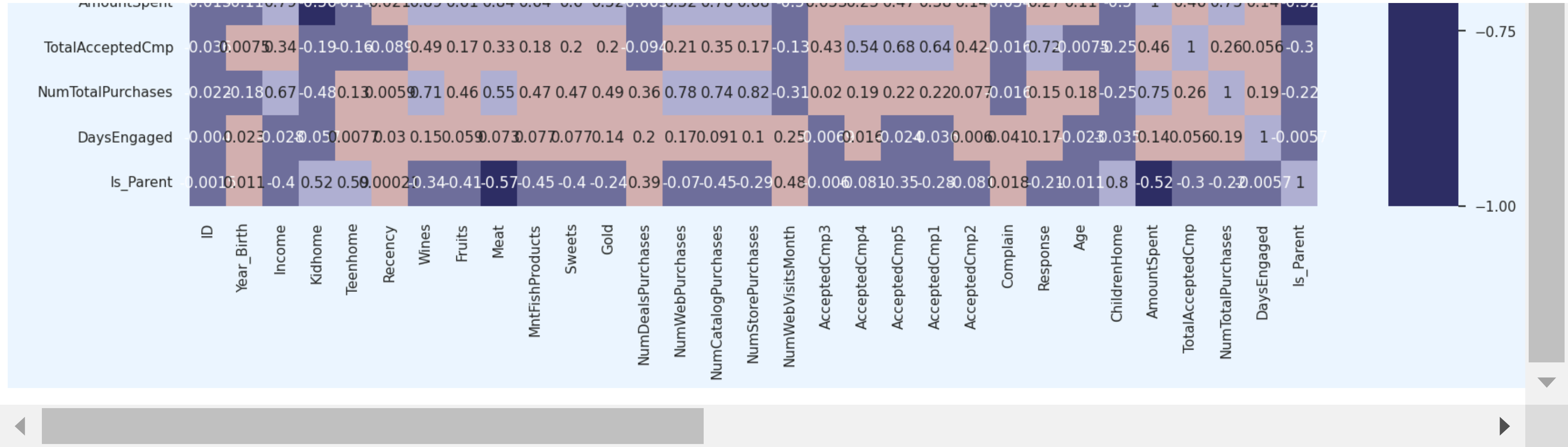
```
The total data points after removing the outliers are: 2236
```

```
#correlation matrix
cor_mat = dataset.corr()
plt.figure(figsize=(20, 20))
sns.heatmap(cor_mat, annot=True, cmap=cmap, center=0)
```

```
cor_mat = dataset.corr()
```

```
<Axes: >
```




TotalAcceptedCmp
NumTotalPurchases
DaysEngaged
Is_Parent
ID
Year_Birth
Income
Kidhome
Teenhome
Recency
Wines
Fruits
Meat
MntFishProducts
Sweets
Gold
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases
NumWebVisitsMonth
AcceptedCmp3
AcceptedCmp4
AcceptedCmp5
AcceptedCmp1
AcceptedCmp2
Complain
Response
Age
ChildrenHome
AmountSpent
TotalAcceptedCmp
NumTotalPurchases
DaysEngaged
Is_Parent
−0.75
−1.00

```python
# Get list of categorical variables
s = (dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables in the dataset:", object_cols)
```

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

```python
from sklearn.preprocessing import LabelEncoder

# Assuming you have a DataFrame called 'data' with categorical columns

LE = LabelEncoder()
object_cols = dataset.select_dtypes(include=['object']).columns

for col in object_cols:
    dataset[col] = dataset[col].astype(str)  # Convert column to string
    dataset[col] = LE.fit_transform(dataset[col])

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

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

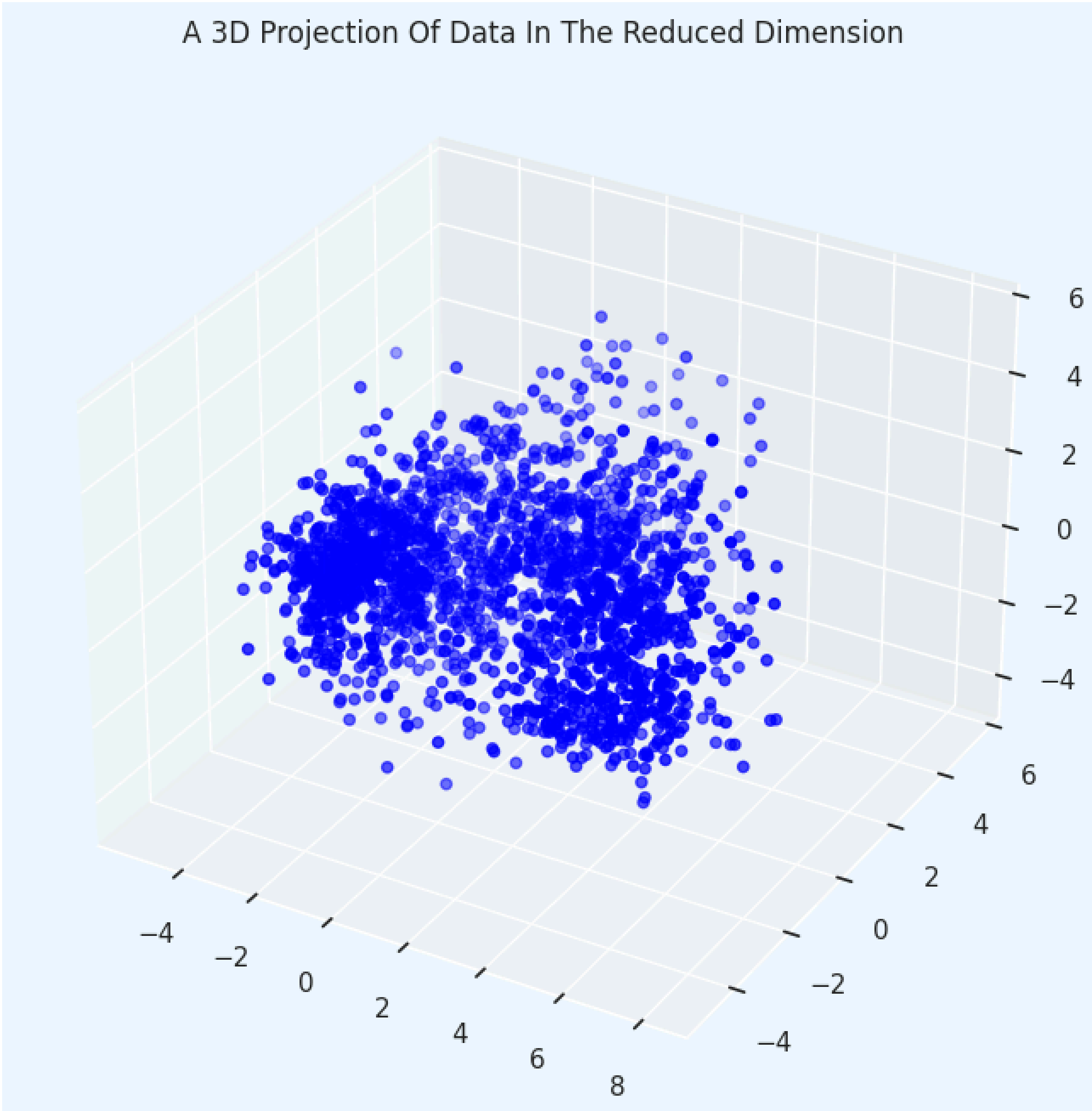
```
from sklearn.preprocessing import StandardScaler

# Assuming you have a DataFrame called 'data_copy' with the required columns

# Creating a copy of the data
data_copy = dataset.copy()

# Creating a subset of the DataFrame by dropping the features on deals accepted and promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','first_day']
data_copy = data_copy.drop(cols_del, axis=1)

# Scaling the data
scaler = StandardScaler()
scaler.fit(data_copy)
scaled_data = pd.DataFrame(scaler.transform(data_copy), columns=data_copy.columns)

print("All features are now scaled using StandardScaler.")
```

```
All features are now scaled using StandardScaler.
```

```
from sklearn.decomposition import PCA

# Assuming you have a DataFrame called 'scaled_ds' with the required columns

# Initiating PCA to reduce dimensions (features) to 3
pca = PCA(n_components=3)
pca.fit(scaled_data)
pca_ds = pd.DataFrame(pca.transform(scaled_data), columns=["col1", "col2", "col3"])

# Describing the PCA transformed dataset
pca_ds.describe().T
```

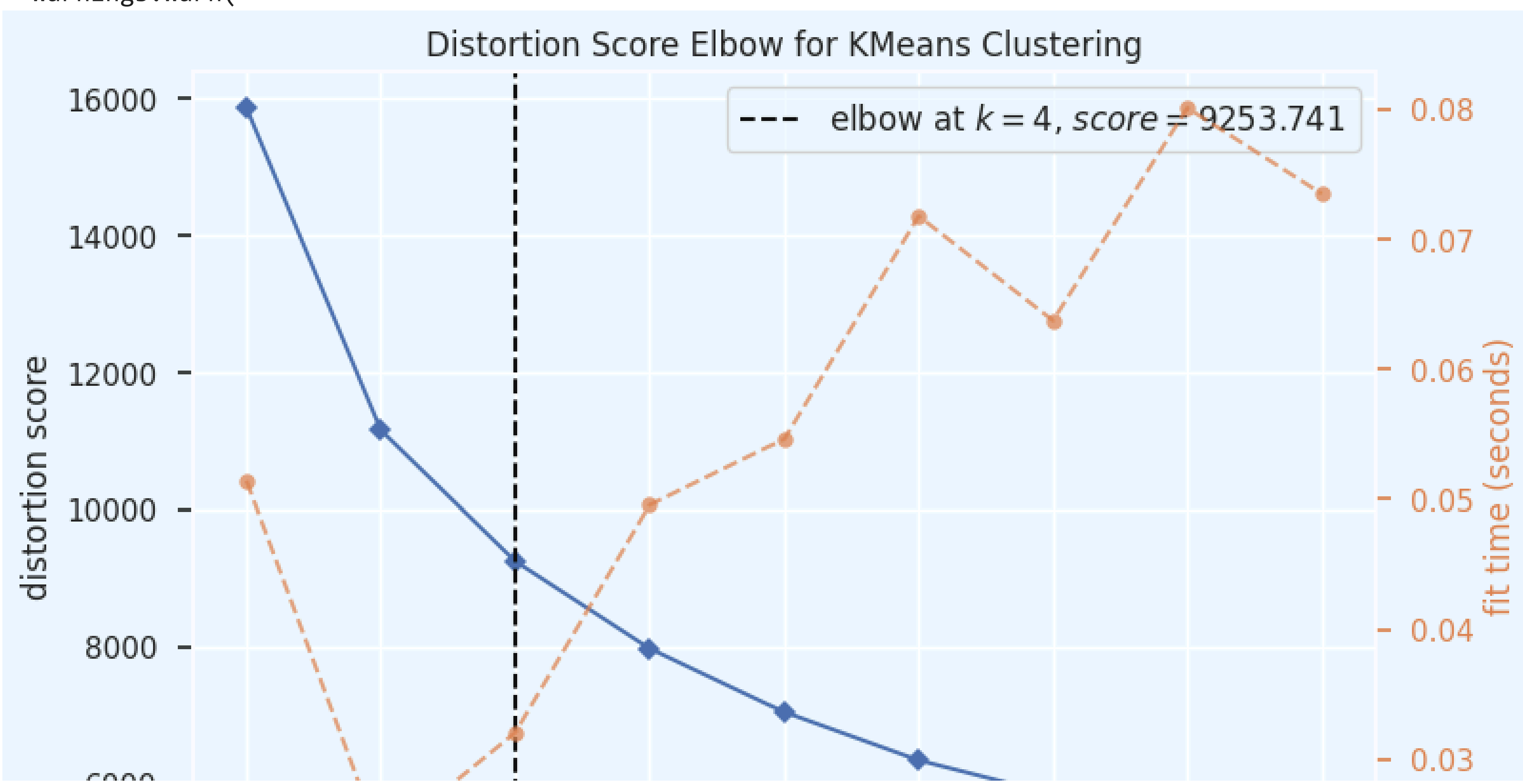
	count	mean	std	min	25%	50%	75%	max
col1	2236.0	4.766610e-17	2.960707	-5.247826	-2.735873	-0.682513	2.525520	8.187958
col2	2236.0	1.143987e-16	1.765851	-4.938153	-1.434204	-0.078224	1.345422	5.551127
col3	2236.0	9.215447e-17	1.443823	-4.407257	-1.000811	-0.080779	0.855019	5.538405

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


A 3D Projection Of Data In The Reduced Dimension
6
4
2
0
−2
−4
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8

```
# 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()
```

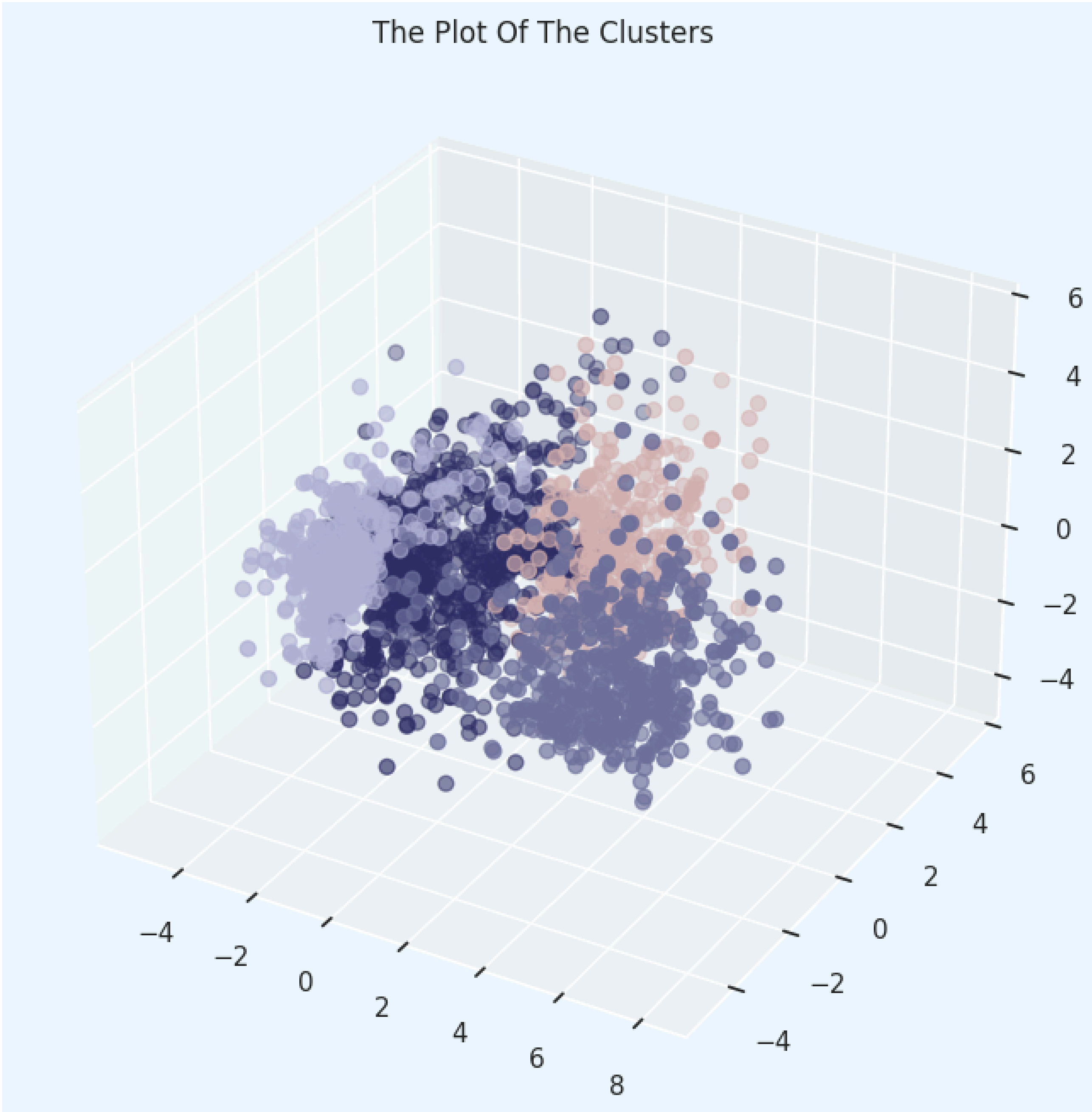
```
Elbow Method to determine the number of clusters to be formed:
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
```




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

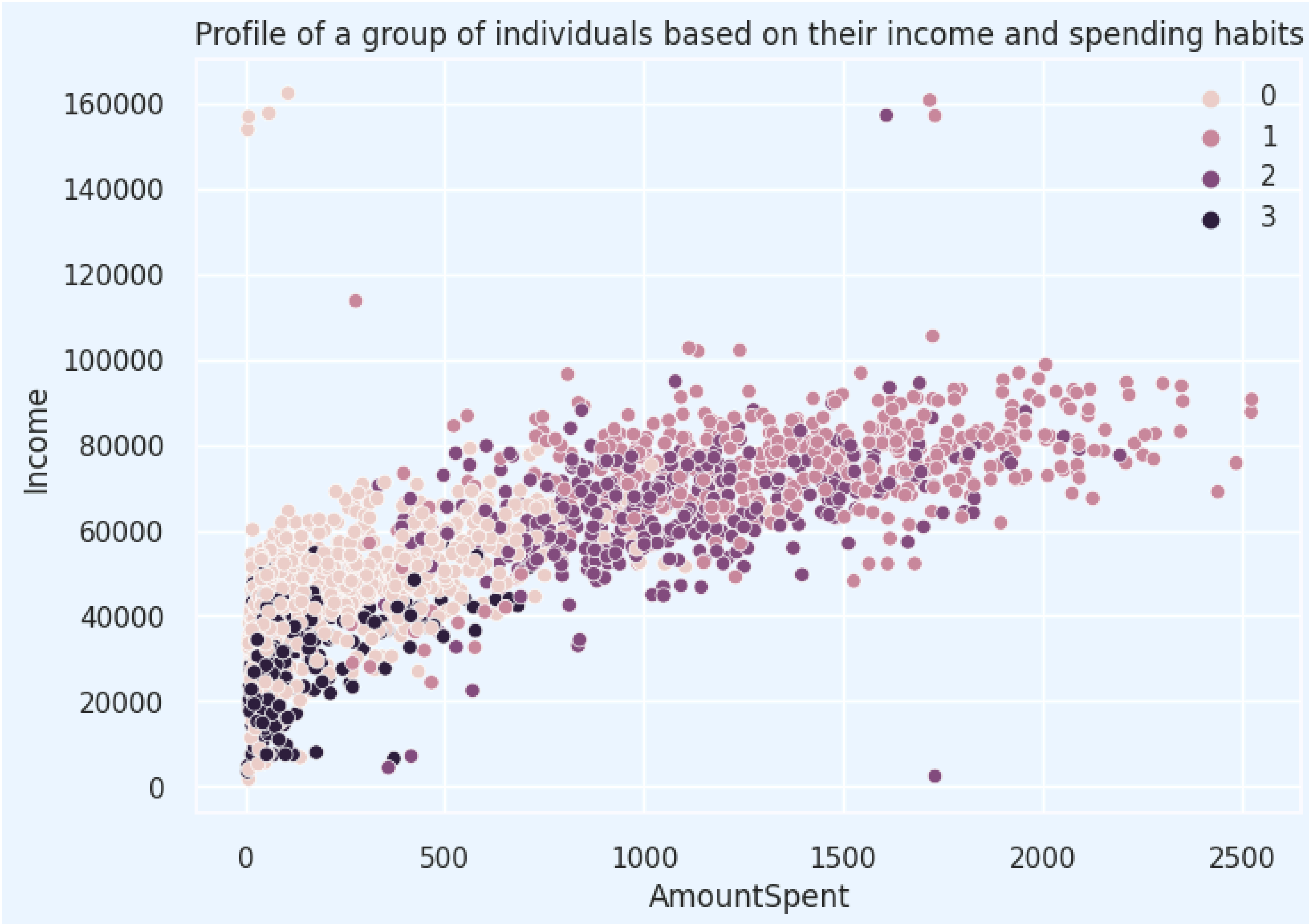
```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(pca_ds)
pca_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
dataset["Clusters"] = yhat_AC
```

```
#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 Clusters")
plt.show()
```


The Plot Of The Clusters
6
4
2
0
−2
−4
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8

```
fig = sns.countplot(x=dataset["Clusters"], palette=pal)
fig.set_title("Distribution Of The Clusters")
plt.show()
```


```
fig = sns.scatterplot(data=dataset, x=dataset["AmountSpent"], y=dataset["Income"], hue=dataset["Clusters"])
fig.set_title("Profile of a group of individuals based on their income and spending habits")
plt.legend()
plt.show()
```




```
plt.figure()
fig = sns.swarmplot(x=dataset["Clusters"], y=dataset["AmountSpent"], color="#CBEDDD",)
fig = sns.boxenplot(x=dataset["Clusters"], y=dataset["AmountSpent"], palette=pal)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 56.9% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 9.2% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 70.3% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 62.8% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 7.8% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 15.0% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3544: UserWarning: 73.1% of the points cannot be placed;
  warnings.warn(msg, UserWarning)
```