Customer Segmentation - Personality Analysis

Kyle Reedy

Customer Segmentation: An Overview

-Each Customer is Different

-Can one single approach to dealing with customers work? Most likely no.

-Customer segmentation is the process of categorizing both current and potential customers into distinct groups based on common attributes or characteristics they share.

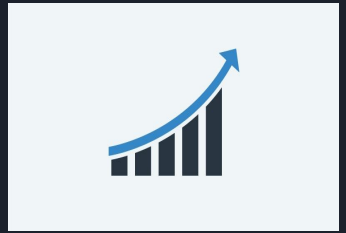
Customer Segmentation: An Overview

Google

Meta

Tech companies such as Google and Meta have built their business models around targeted advertising, thanks to customer segmentation.

- Enhances customer relationship and brand loyalty
- Enhances customer experiences and sales

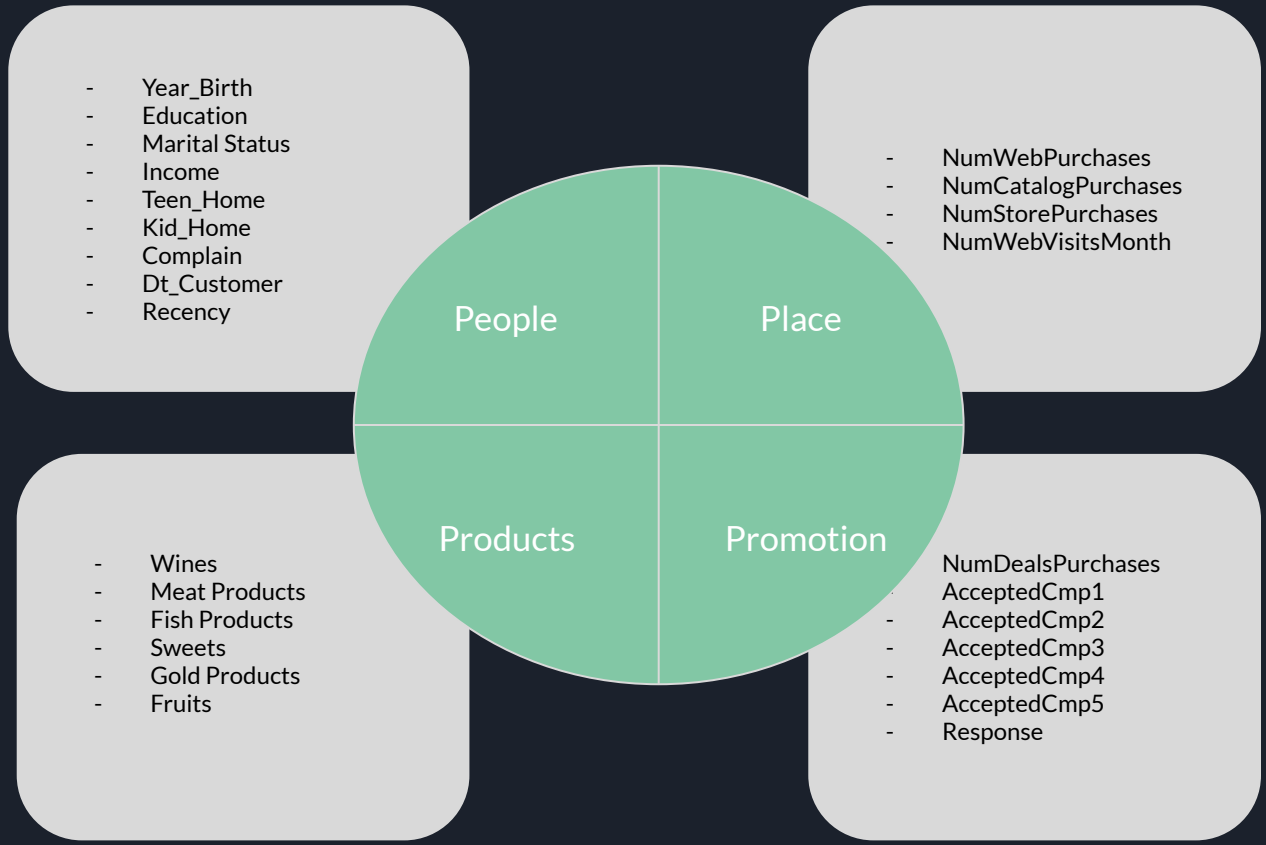


Objectives and Steps Involved

In this project, we will perform a customer personality analysis based on retail stores. This project aims to analyze a dataset provided in CSV format, comprising 29 key customer attributes including education level, marital status, income, and expenditure on various products such as wine, fruits, meat, and gold. Additionally, the dataset includes information on how customers respond to promotions or discounts, as well as the distribution of purchases across different sources such as catalogs, stores, and websites.

Cleaning/Wrangling → EDA → Preprocessing → Modeling

Project Dataset Attributes



Cleaning and Wrangling

```
df2['Income'] = df2['Income'].fillna(df2['Income'].median())
df2.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
```

- The 'Income' column is the only feature where there are null values
- Missing values are filled with median values

- "Customer_For" Feature Created

```
df2["Date_Customer"] = pd.to_datetime(df2["Dt_Customer"])
newest_date = df2["Date_Customer"].max().date()
oldest_date = df2["Date_Customer"].min().date()

print("The newest customer's enrollment date in the records:", newest_date)
print("The oldest customer's enrollment date in the records:", oldest_date)
```

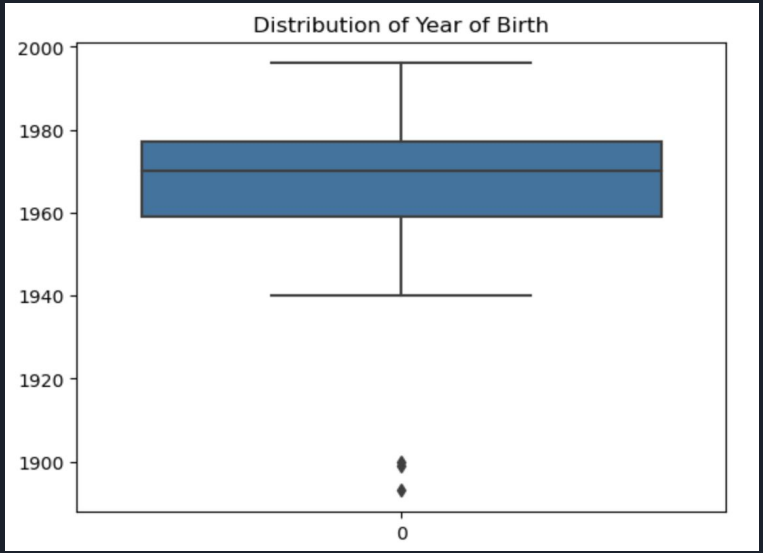
```
The newest customer's enrollment date in the records: 2014-12-06
The oldest customer's enrollment date in the records: 2012-01-08
```

```
/var/folders/2r/qvwb_49162s3q75v3pr4d1x00000gn/T/ipykernel_97584/2632591379.py:1: UserWarning: Parsing dates in DD/M
M/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify
a format to ensure consistent parsing.
  df2["Date_Customer"] = pd.to_datetime(df2["Dt_Customer"])
```

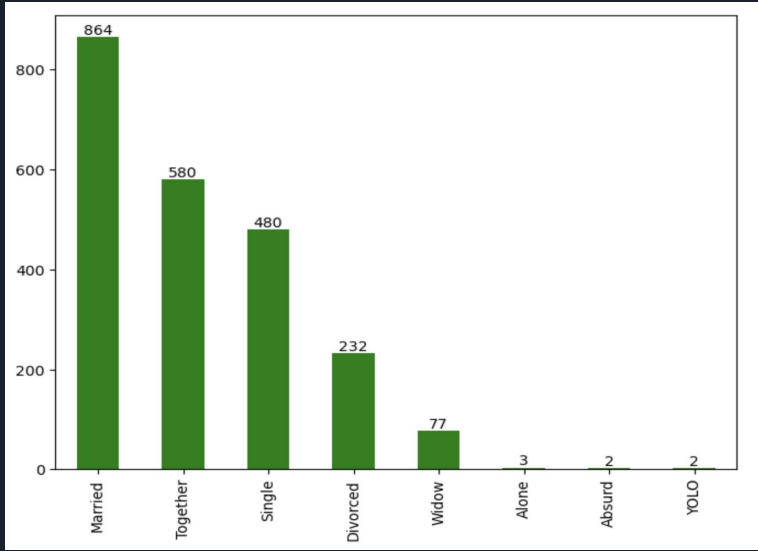
Next, I'll create a new feature ("Customer_For") indicating the number of days since customers first started shopping relative to the last recorded date.

```
days_since_registration = (newest_date - df2["Date_Customer"].dt.date).dt.days
df2["Customer_For"] = days_since_registration
```

Customer Demographics



Average Year of Birth: 1968



Married: 864

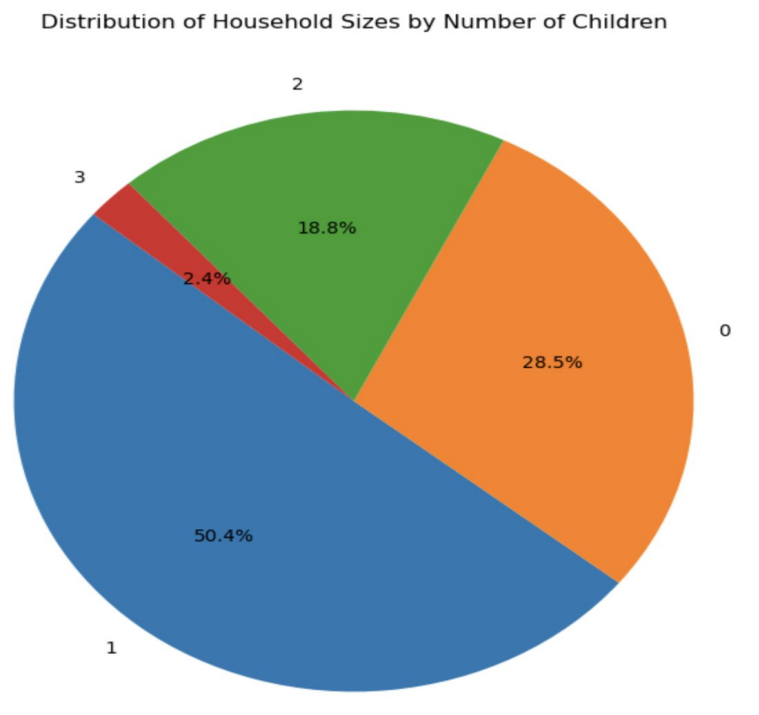
Together: 580

Single: 480

Divorced: 232

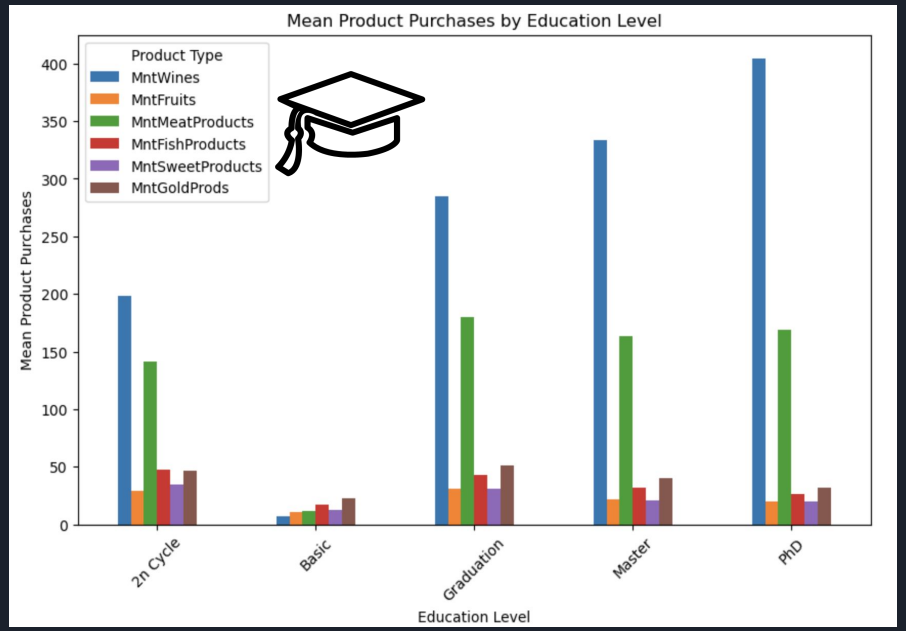
Widow: 77

Demographics

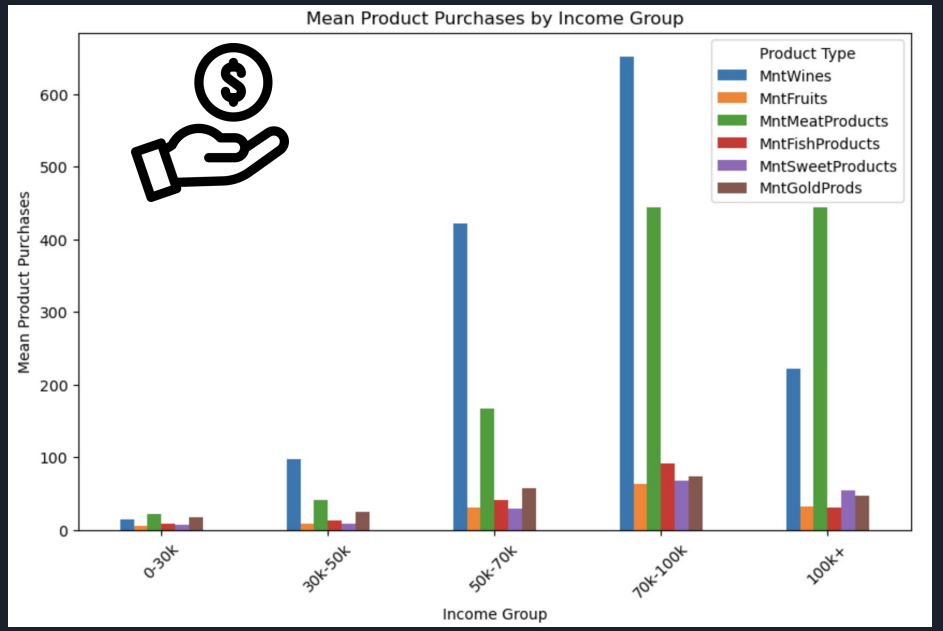


- Half of the dataset includes households with one child at 50.4%
- 0 children: 28.5%
- 2 children: 18.8%
- 3 children: 2.4%

Products Purchased Based on Education and Income



- The average amount purchased for wine products increase the higher the educational degrees are.

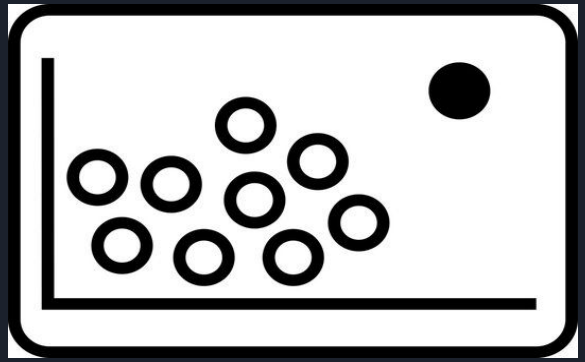


- So few purchases are made in the 0-30k income bracket.
- 50-70k and 70-100k income bracket have high total wine purchases.
- 70-100k and 100k income bracket have high total meat purchases.

Preprocessing and Feature Engineering

```
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 33 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
 29  Date_Customer        2240 non-null   object
 30  Customer_For         2240 non-null   int64
 31  Children             2240 non-null   int64
 32  Income_Group         2240 non-null   object
dtypes: float64(1), int64(27), object(5)
```

Current dataset and dtypes



- Feature Selection, Engineering
- Address outliers
- Encoding Categorical Variables

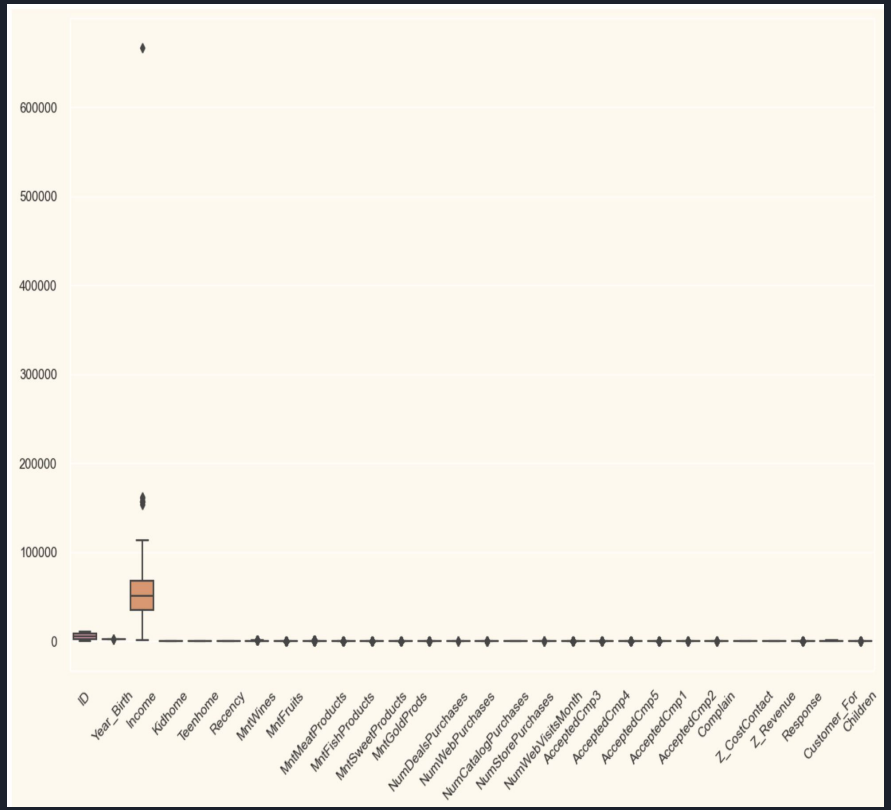
Outliers

Income Feature

- One observation with an income of over 600,000

```
# Filtering out records with income greater than $600,000
df = df[df['Income'] < 600000]
```

- Filtering so that we get rid of this outlier



Outliers

Age Feature

- Three observations of age where it is over 115.

	ID	Age	Education	Marital Status	Parental Status	Children	Kidhome	Teenhome	Income	Spending	...	Meat	Fish	Sweets	Gold	Web	Catalog	Store	Dis Purc
192	7829	115	Undergraduate	Alone	1	1	1	0	36640.0	65	...	8	7	4	25	2	1	2	
239	11004	122	Undergraduate	Alone	1	1	0	1	60182.0	22	...	5	7	0	2	1	0	2	
339	1150	116	Postgraduate	Couple	0	0	0	0	83532.0	1853	...	562	104	64	224	4	6	4	

- Filtering so that we get rid of this outlier

```
1 df = df[(df["Age"]<95)]
2 df[df["Age"]>95]
```

ID	Age	Education	Marital Status	Parental Status	Children	Kidhome	Teenhome	Income	Spending	...	Meat	Fish	Sweets	Gold	Web	Catalog	Store	Discount Purchases	T Pro

0 rows × 24 columns

Label Encoding, Scaling

```
data_types = df.dtypes

# Filtering columns with data type 'object' or 'category'
categorical_variables = data_types[data_types == 'object'].index.tolist() + data_types[data_types == 'category'].ir

# Display the list of categorical variables
print("Categorical variables:", categorical_variables)
```

Categorical Variables

‘Education’

‘Marital Status’

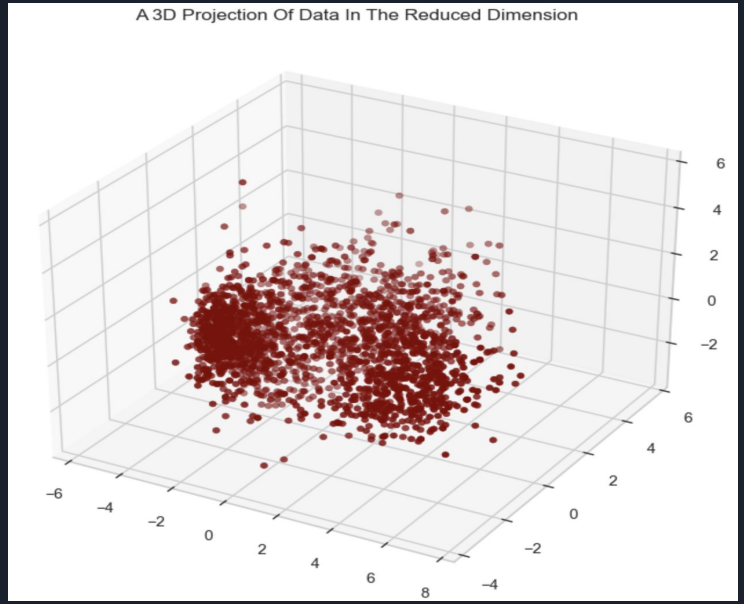
```
# Label Encoding the object dtypes.
LE = LabelEncoder()
for col in categorical_variables:
    df[col] = LE.fit_transform(df[col])
```

```
# Creating a copy of the original DataFrame excluding specified columns
cols_to_keep = [col for col in df.columns if col not in ['ID', 'Total Promo', 'Complain', 'Response']]
ds = df[cols_to_keep].copy()

# Scaling
scaler = StandardScaler()
scaled_ds = pd.DataFrame(scaler.fit_transform(ds), columns=ds.columns)
```

Modeling - Dimensionality Reduction (PCA)

Dimensionality reduction involves decreasing the number of variables under consideration by extracting a set of principal variables.

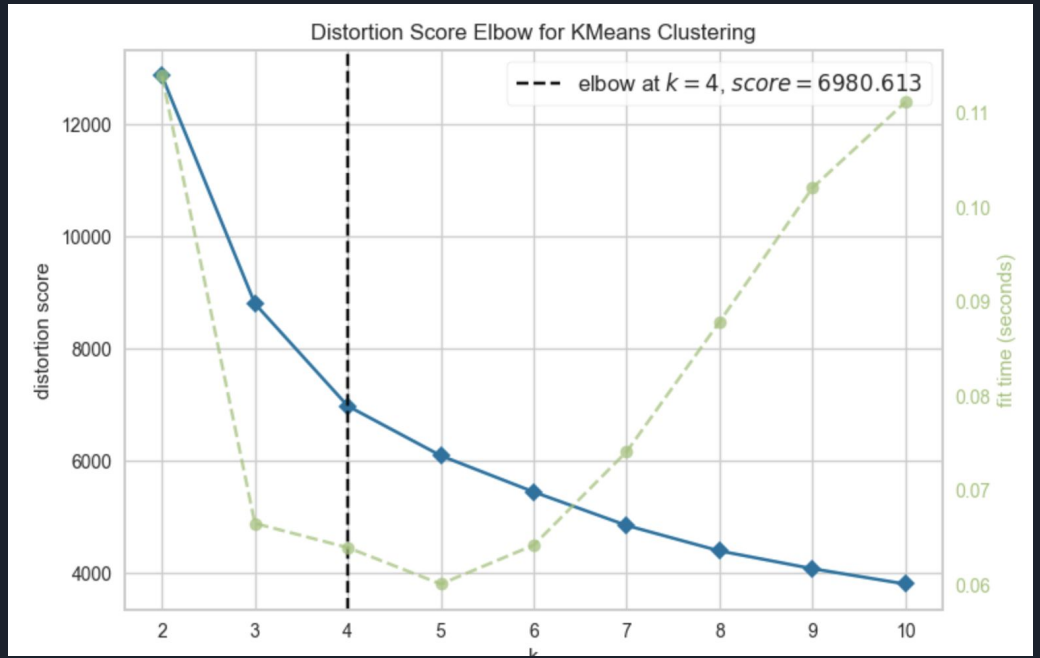
- PCA -> Dimensions to 3



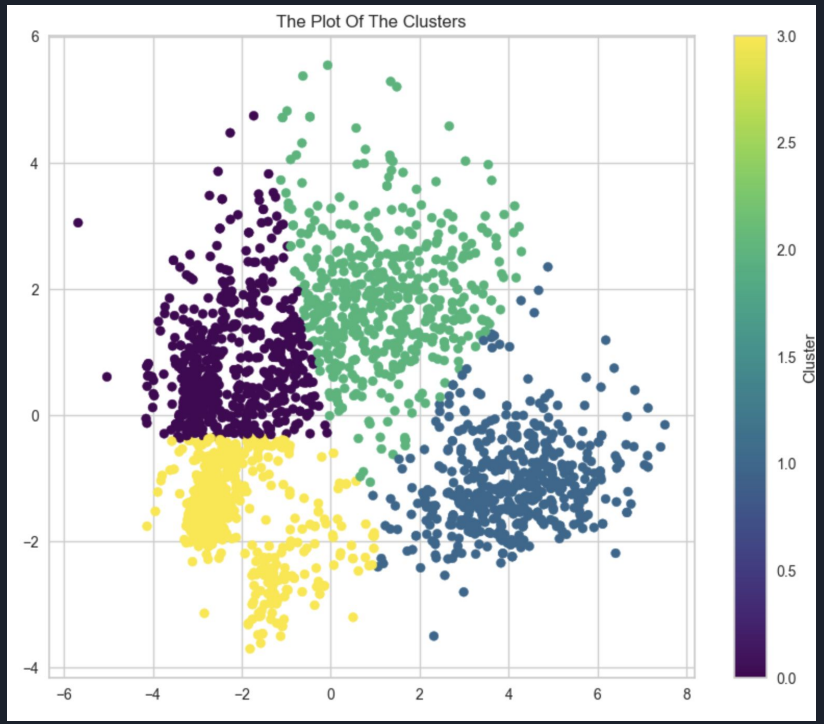

```
x = PCA_sdf["col1"]
y = PCA_sdf["col2"]
z = PCA_sdf["col3"]

# Creating a 3D projection 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()
```

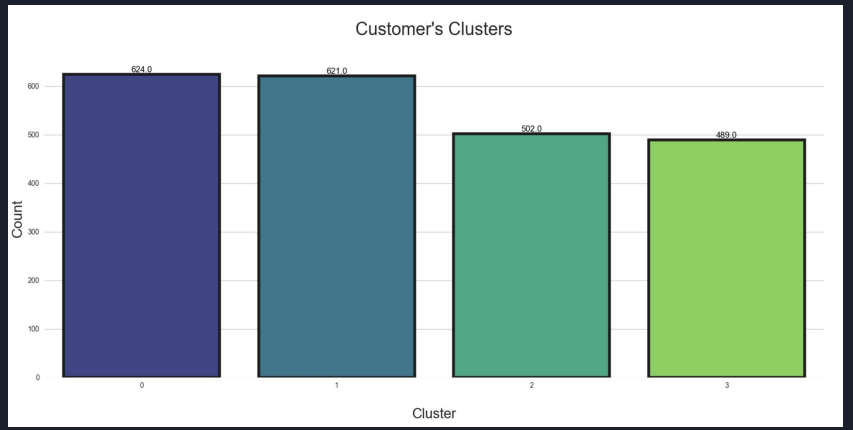
Distortion Score Elbow for K-Means Clustering

Based on the analysis conducted in the preceding cell, it appears that the dataset is best segmented into 4 distinct clusters.



Count per Cluster





Cluster 0: 624

Cluster 1: 621

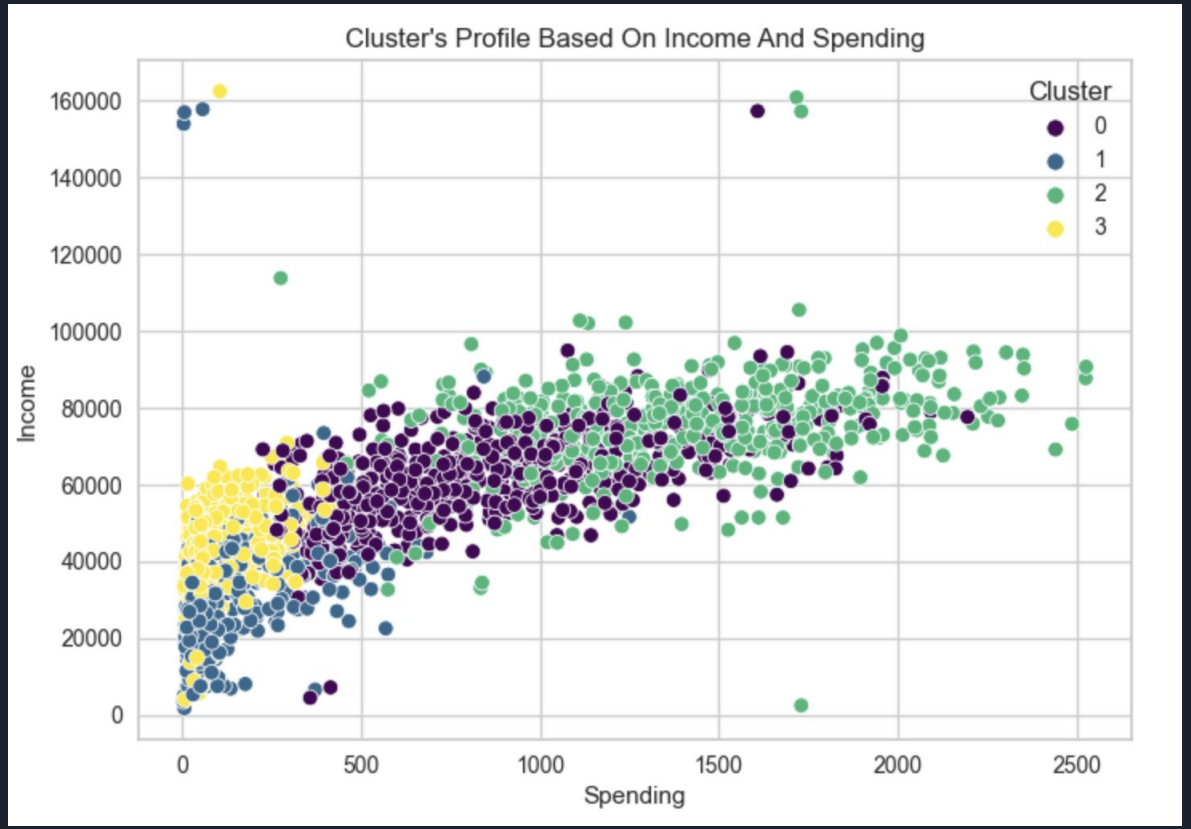
Cluster 2: 502

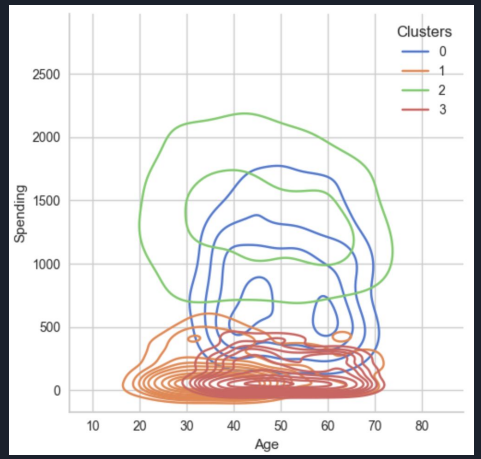
Cluster 3: 489

The clusters are closely aligned in count, with only a 135 count difference from cluster 0 and 3.

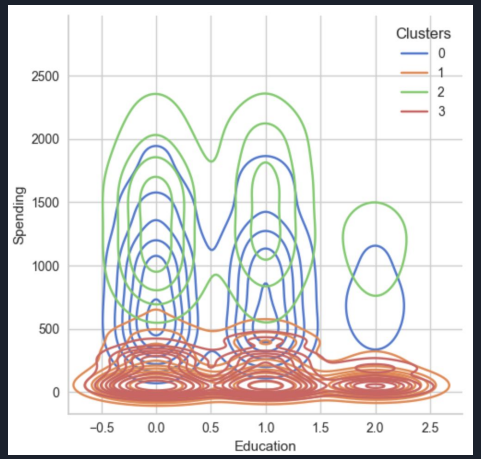
Income vs Spending

- Cluster 0: average income | high spending
- Cluster 1: low income | low spending
- Cluster 2: high income | high spending
- Cluster 3: average income | low spending

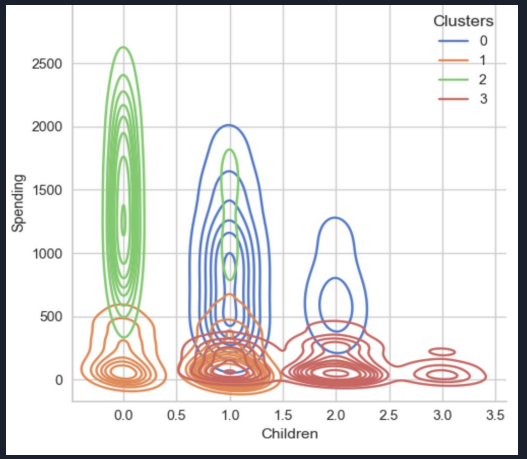




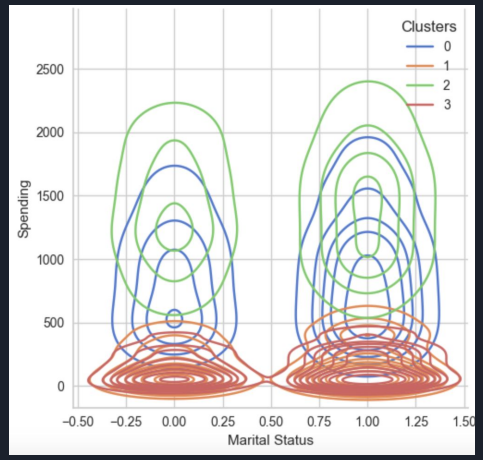
Age



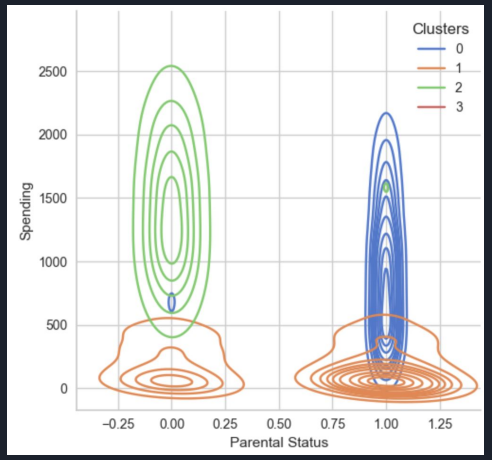
Education



Children



Marital Status



Parental Status

Traits vs Spending

Cluster 0:
Mostly parents
1/3 used promos
Age: 35-65
Spending: $100-$1750
Majority spend $500-$1000
Educated: Graduate or undergrad
Mostly married with 1-2 kids
Many made 2-4 discounted purchases

Cluster 2:
Mostly non-parents
Many used promos
Age: 30-65
Spending: $750-$2300
Majority spend over $1000
Mostly married, no kids
Majority made 1 discounted purchase

Cluster 1:
Some parents
Few used promos
Age: 30-50
Spending: $0-$600
Educated: Graduate
Mixed marital status, few kids
Majority made 1 discounted purchase

Cluster 3:
Mostly parents
Few used promos
Age: 40-65
Spending: $0-$500
Educated: Graduate
Mixed marital status, 1-3 kids
Majority made 1-3 discounted purchases

Cluster Insights

= Clusters to Focus

Recommendations

Target Marketing

Cluster 0: Family oriented bundles

Cluster 1: Value-driven Promotions

Cluster 2: Exclusive Discounts for frequent shoppers

Product Assortment

Cluster 1: Budget friendly deals

Cluster 2: Exclusive, premium specials for these frequent shoppers

Promotion Refinement

Refined targeted promotions
+
Loyalty Programs

Engagement ⬆
Conversion Rates ⬆