analysis

May 7, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from datetime import date
import seaborn as sns
```

1 Importing the Dataset

[2]:
```
df = pd.read_csv('marketing_campaign.csv', sep=';')
```

[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	2012-09-04	58	635	…	7	0	
1	2014-03-08	38	11	…	5	0	
2	2013-08-21	26	426	…	4	0	
3	2014-02-10	26	11	…	6	0	
4	2014-01-19	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]
```

2 Understanding the Dataset

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

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

There are 2240 individuals in this dataset.

[5]:
```
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
```

[6]:
```
df.describe()
```

[6]:

	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]
```

The statistics of the dataset.

3 Data Cleaning and Preparation

Removal of all missing values.

[7]:
```
df.dropna(inplace=True)
```

[8]:
```
df
```

[8]:
```
          ID  Year_Birth   Education Marital_Status   Income  Kidhome  \
0       5524        1957  Graduation         Single  58138.0        0
1       2174        1954  Graduation         Single  46344.0        1
2       4141        1965  Graduation       Together  71613.0        0
3       6182        1984  Graduation       Together  26646.0        1
4       5324        1981         PhD        Married  58293.0        1
…       …           …           …              …       …        …
2235   10870        1967  Graduation        Married  61223.0        0
2236    4001        1946         PhD       Together  64014.0        2
2237    7270        1981  Graduation       Divorced  56981.0        0
2238    8235        1956      Master       Together  69245.0        0
2239    9405        1954         PhD        Married  52869.0        1

      Teenhome Dt_Customer  Recency  MntWines  …  NumWebVisitsMonth  \
0            0  2012-09-04       58       635  …                  7
1            1  2014-03-08       38        11  …                  5
2            0  2013-08-21       26       426  …                  4
3            0  2014-02-10       26        11  …                  6
4            0  2014-01-19       94       173  …                  5
…          …          …       …         …  …                  …
2235         1  2013-06-13       46       709  …                  5
2236         1  2014-06-10       56       406  …                  7
2237         0  2014-01-25       91       908  …                  6
2238         1  2014-01-24        8       428  …                  3
2239         1  2012-10-15       40        84  …                  7
```

```
      AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
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
…              …             …             …             …             …
2235             0             0             0             0             0
2236             0             0             0             1             0
2237             0             1             0             0             0
2238             0             0             0             0             0
2239             0             0             0             0             0

      Complain  Z_CostContact  Z_Revenue  Response
0            0              3         11         1
1            0              3         11         0
2            0              3         11         0
3            0              3         11         0
4            0              3         11         0
…          …              …         …          …
2235         0              3         11         0
2236         0              3         11         0
2237         0              3         11         0
2238         0              3         11         0
2239         0              3         11         1

[2216 rows x 29 columns]
```

With the missing values removed, there are 2216 individuals left in the dataset.

Looking for unique values in the Education and Marital_Status columns:

[9]:
```
print(df['Education'].unique())
print(df['Marital_Status'].unique())
```

```
['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']
```

Changing all '2n Cycle' inputs to 'Master' because they represent the same level of education. But first, copying the df into another variable to avoid accidentally changing the original dataset.

[10]:
```
new_df = df.copy()

new_df['Education'] = new_df['Education'].replace('2n Cycle', 'Master')
new_df['Education'].unique()
```

[10]:
```
array(['Graduation', 'PhD', 'Master', 'Basic'], dtype=object)
```

Repeating the above for the Marital_Status column:

[11]:
```
new_df['Marital_Status'] = new_df['Marital_Status'].replace('Alone', 'Single')
new_df['Marital_Status'].unique()
```

[11]:
```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Absurd',
       'YOLO'], dtype=object)
```

Removing rows that have nonsensical values such as Absurd and YOLO:

[12]:
```
new_df.drop(new_df[new_df['Marital_Status'] == 'Absurd'].index, inplace = True)
new_df.drop(new_df[new_df['Marital_Status'] == 'YOLO'].index, inplace = True)
new_df['Marital_Status'].unique()
```

[12]:
```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow'], dtype=object)
```

The Year_Birth column isn't useful by itself, the individuals' actual age needs to be calculated in order to sort the data by age range in the future.

[13]:
```
current_year = date.today().year
new_df['Age'] = current_year - new_df['Year_Birth']
new_df['Age']
```

[13]:
```
0       67
1       70
2       59
3       40
4       43
        ..
2235    57
2236    78
2237    43
2238    68
2239    70
Name: Age, Length: 2212, dtype: int64
```

I've created a new column called Age with the actual ages of the individuals, now I will create a column with age ranges.

[14]:
```
new_df['Age_Range'] = new_df['Age'].apply(lambda x:
    'under_18' if x <= 17 else (
    '18-35' if 18 <= x <= 35 else (
    '36-50' if 36 <= x <= 50 else '50+'
    ))
)
new_df['Age_Range']
```

[14]:
```
0         50+
1         50+
2         50+
3       36-50
```

```
4       36-50
        …
2235      50+
2236      50+
2237    36-50
2238      50+
2239      50+
Name: Age_Range, Length: 2212, dtype: object
```

Dropping all columns that won't be needed in further calculations:

[15]:
```
new_df = new_df.drop(['ID', 'Year_Birth', 'Z_CostContact', 'Z_Revenue'], axis=1)
new_df.head()
```

[15]:

	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	\
0	Graduation	Single	58138.0	0	0	2012-09-04	58	
1	Graduation	Single	46344.0	1	1	2014-03-08	38	
2	Graduation	Together	71613.0	0	0	2013-08-21	26	
3	Graduation	Together	26646.0	1	0	2014-02-10	26	
4	PhD	Married	58293.0	1	0	2014-01-19	94	

	MntWines	MntFruits	MntMeatProducts	…	NumWebVisitsMonth	AcceptedCmp3	\
0	635	88	546	…	7	0	
1	11	1	6	…	5	0	
2	426	49	127	…	4	0	
3	11	4	20	…	6	0	
4	173	43	118	…	5	0	

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

	Age	Age_Range
0	67	50+
1	70	50+
2	59	50+
3	40	36-50
4	43	36-50

[5 rows x 27 columns]

4 Feature Engineering

To gather useful insights and potentially improve the models' performance, I create new features.

Creating a column showing the total amount spent on different product categories:

[16]:
```
new_df['Total_Spent'] = new_df['MntWines'] + new_df['MntFruits'] +␣
↪new_df['MntMeatProducts'] + new_df['MntFishProducts'] +␣
↪new_df['MntSweetProducts'] + new_df['MntGoldProds']
new_df.head()
```

[16]:

	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	\
0	Graduation	Single	58138.0	0	0	2012-09-04	58	
1	Graduation	Single	46344.0	1	1	2014-03-08	38	
2	Graduation	Together	71613.0	0	0	2013-08-21	26	
3	Graduation	Together	26646.0	1	0	2014-02-10	26	
4	PhD	Married	58293.0	1	0	2014-01-19	94	

	MntWines	MntFruits	MntMeatProducts	…	AcceptedCmp3	AcceptedCmp4	\
0	635	88	546	…	0	0	
1	11	1	6	…	0	0	
2	426	49	127	…	0	0	
3	11	4	20	…	0	0	
4	173	43	118	…	0	0	

	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Response	Age	\
0	0	0	0	0	1	67	
1	0	0	0	0	0	70	
2	0	0	0	0	0	59	
3	0	0	0	0	0	40	
4	0	0	0	0	0	43	

	Age_Range	Total_Spent
0	50+	1617
1	50+	27
2	50+	776
3	36-50	53
4	36-50	422

[5 rows x 28 columns]

Creating columns showing total number of accepted offers, total number of purchases made, and the enrollement year:

[17]:
```
new_df['Total_Offers'] = new_df['AcceptedCmp1'] + new_df['AcceptedCmp2'] +␣
↪new_df['AcceptedCmp3'] + new_df['AcceptedCmp4'] + new_df['AcceptedCmp5']

new_df['Total_Purchases'] = new_df['NumDealsPurchases' ] +␣
↪new_df['NumWebPurchases'] + new_df['NumCatalogPurchases'] +␣
↪new_df['NumStorePurchases']

new_df['Dt_Customer'] = pd.to_datetime(new_df['Dt_Customer'], errors='coerce')
```

```
new_df['Enrollment_Year'] = new_df['Dt_Customer'].dt.year
new_df['Seniority'] = current_year - new_df['Enrollment_Year']

new_df.head()
```

[17]:

	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	\
0	Graduation	Single	58138.0	0	0	2012-09-04	58	
1	Graduation	Single	46344.0	1	1	2014-03-08	38	
2	Graduation	Together	71613.0	0	0	2013-08-21	26	
3	Graduation	Together	26646.0	1	0	2014-02-10	26	
4	PhD	Married	58293.0	1	0	2014-01-19	94	

	MntWines	MntFruits	MntMeatProducts	…	AcceptedCmp2	Complain	\
0	635	88	546	…	0	0	
1	11	1	6	…	0	0	
2	426	49	127	…	0	0	
3	11	4	20	…	0	0	
4	173	43	118	…	0	0	

	Response	Age	Age_Range	Total_Spent	Total_Offers	Total_Purchases	\
0	1	67	50+	1617	0	25	
1	0	70	50+	27	0	6	
2	0	59	50+	776	0	21	
3	0	40	36-50	53	0	8	
4	0	43	36-50	422	0	19	

	Enrollment_Year	Seniority
0	2012	12
1	2014	10
2	2013	11
3	2014	10
4	2014	10

[5 rows x 32 columns]

5 Exploratory Data Analysis (EDA)

Performing exploratory data analysis (EDA) is crucial for understanding variable distributions, identifying patterns, and visualizing relationships between variables to gain valuable insights.

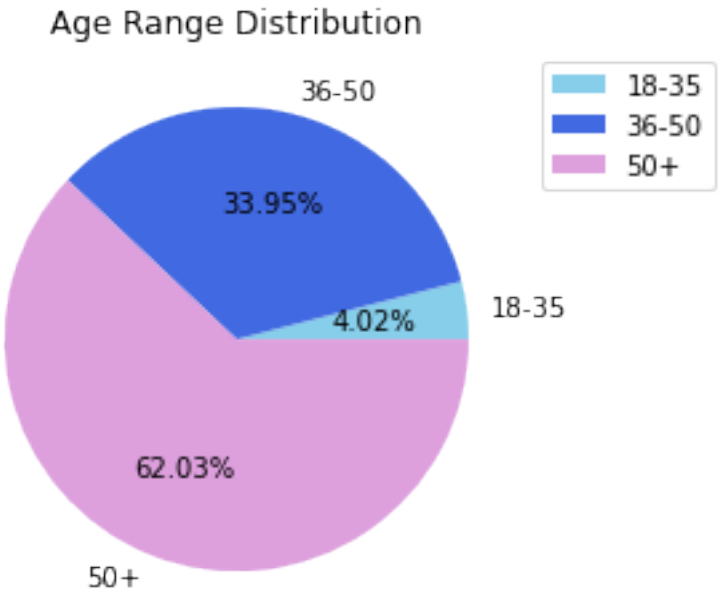
Age distribution:

[18]:
```
age_range = new_df.groupby('Age_Range').size().reset_index(name='Count')
age_range['Percentage'] = (age_range['Count'] * 100 / age_range['Count'].sum()).
 ↪round(2)
age_range
```

[18]:

	Age_Range	Count	Percentage
0	18-35	89	4.02
1	36-50	751	33.95
2	50+	1372	62.03

[19]:
```
labels = age_range['Age_Range']
count = age_range['Count']

plt.pie(count, labels=labels, autopct='%1.2f%%', colors=['skyblue',␣
 ↪'royalblue', 'plum'])
plt.title('Age Range Distribution')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
```



The majority (62.03%) of the individuals are aged above 50. There are no individuals younger than 18. The second largest group (33.95%) are individuals between the ages of 36 and 50. Those between 18 and 35 account for only 4.02%.

Income distribution:

[20]:
```
new_df['Income'].describe()
```

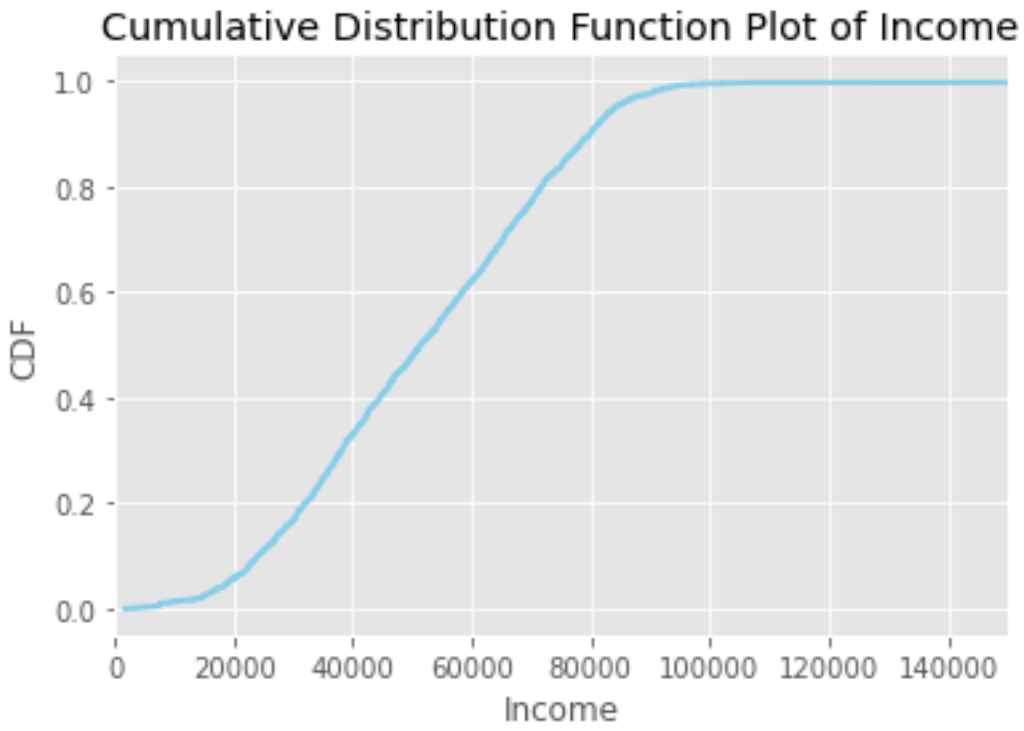
[20]:
```
count      2212.000000
mean      52232.510850
std       25187.455359
min        1730.000000
25%       35233.500000
```

```
50%       51381.500000
75%       68522.000000
max      666666.000000
Name: Income, dtype: float64
```

[21]:
```
income_sorted = np.sort(new_df['Income'])
income_sorted
```

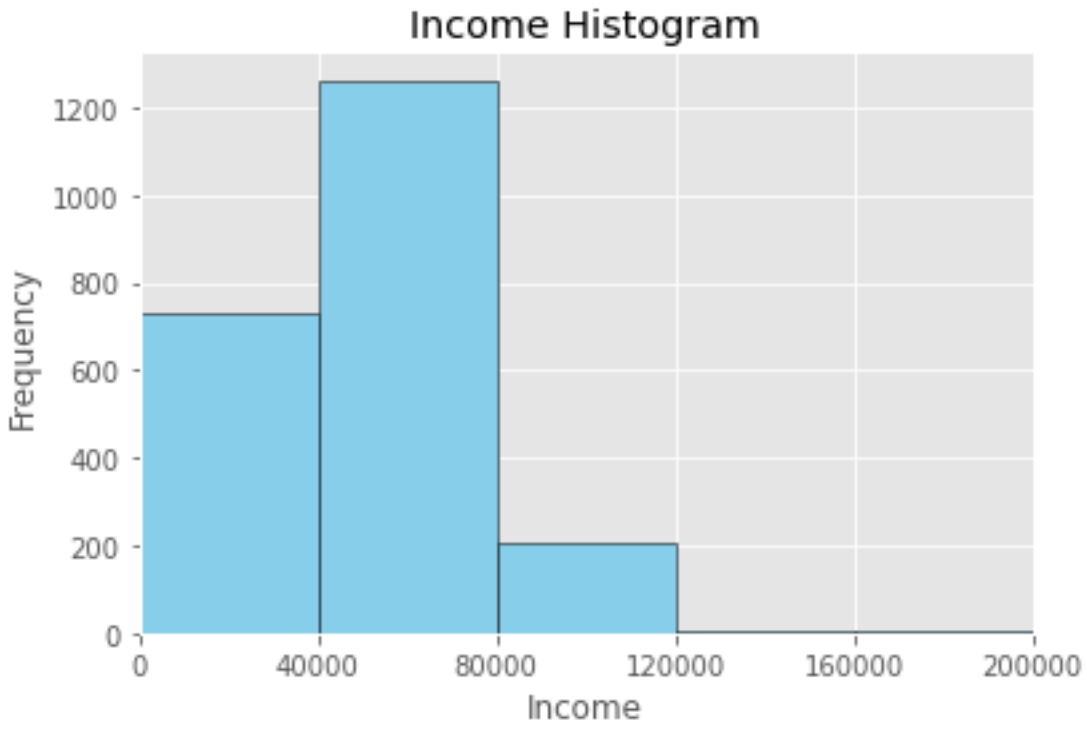
[21]:
```
array([  1730.,   2447.,   3502., …, 160803., 162397., 666666.])
```

[22]:
```
y = np.arange(1, len(income_sorted) + 1) / len(income_sorted)

plt.style.use('ggplot')
plt.step(income_sorted, y, linewidth=2, color='skyblue')

plt.xticks(np.arange(0,200000, step=20000))
plt.xlim(0, 150000)

plt.xlabel('Income')
plt.ylabel('CDF')
plt.title('Cumulative Distribution Function Plot of Income')

plt.show()
```



More than 95% of individuals have incomes below $100,000.

[23]:
```
plt.hist(new_df['Income'],
         range=(0,200000), bins=5, color='skyblue', edgecolor='black')

plt.xlim(0, 200000)
plt.xticks(np.arange(0, 200001, 40000))

plt.xlabel('Income')
plt.ylabel('Frequency')
plt.title('Income Histogram')

plt.show()
```



More than 55% of the individuals in the data set have incomes between 40,000 and 80,000 dollars.

Less than 10% of individuals earn more than $80,000.

Education Distribution:

[24]:
```
education = new_df.groupby('Education').size().reset_index(name='Count')
education['Percentage'] = (education['Count'] * 100 / education['Count'].sum()).
 ↪round(2)
education
```

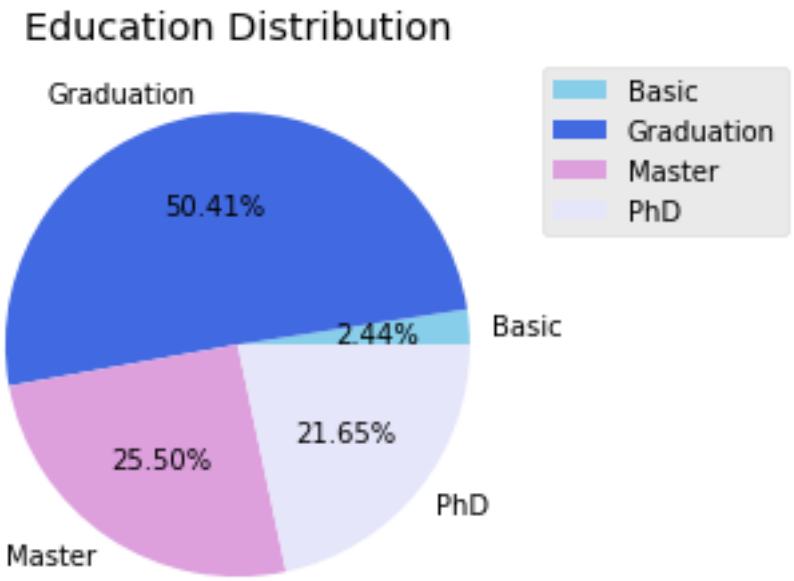
[24]:

	Education	Count	Percentage
0	Basic	54	2.44
1	Graduation	1115	50.41
2	Master	564	25.50
3	PhD	479	21.65

Around 50% of the individuals in the dataset have a college degree.

Around 47% of individuals have either a Master's Degree or a PhD.

[25]:
```
labels = education['Education']
count = education['Count']

plt.pie(count, labels=labels, autopct='%1.2f%%', colors=['skyblue',␣
 ↪'royalblue', 'plum', 'lavender'])
plt.title('Education Distribution')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
```

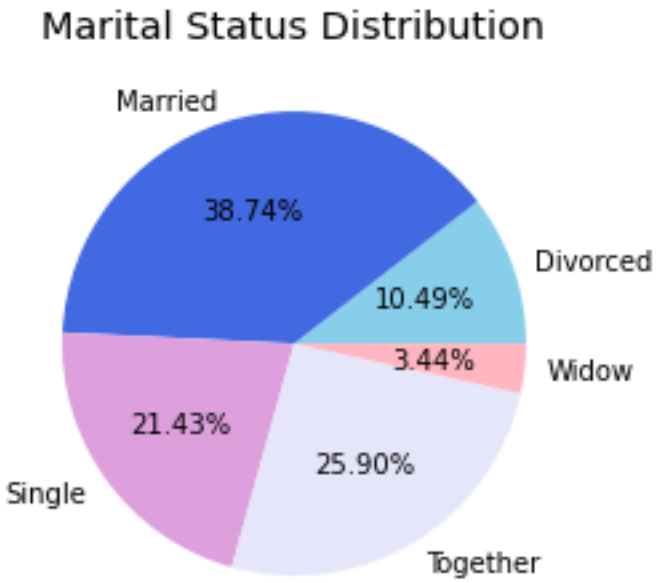


Marital Status Distribution:

[26]:
```
marital_status = new_df.groupby('Marital_Status').size().
 ↪reset_index(name='Count')
marital_status['Percentage'] = (marital_status['Count'] * 100 /␣
 ↪marital_status['Count'].sum()).round(2)
print(marital_status)
```

```
labels = marital_status['Marital_Status']
count = marital_status['Count']

plt.pie(count, labels=labels, autopct='%1.2f%%', colors=['skyblue',␣
 ↪'royalblue', 'plum', 'lavender', 'lightpink'])
plt.title('Marital Status Distribution')
plt.show()
```

	Marital_Status	Count	Percentage
0	Divorced	232	10.49
1	Married	857	38.74
2	Single	474	21.43
3	Together	573	25.90
4	Widow	76	3.44



Over 62% of individuals in the dataset are in a relationship (Married or Together).

More than 34% of individuals are single (Single, Divorced or Widow)

Looking at how many individuals in the dataset are parents or not by creating a new column that combines both the Kidhome and Teenhome columns:

[27]:
```
new_df['Parent'] = new_df['Kidhome'] | new_df['Teenhome']
new_df['Parent'].head()
```

[27]:
```
0    0
1    1
```

```
2    0
3    1
4    1
Name: Parent, dtype: int64
```

[28]:
```
new_df['Parent'] = new_df['Parent'].astype('bool')
new_df['Parent'].head()
```

[28]:
```
0    False
1     True
2    False
3     True
4     True
Name: Parent, dtype: bool
```

I created a new column called Parent and turned all the values into boolean values in order to have a simple way to check if someone is a parent or not.

[29]:
```
parent_status = new_df.groupby('Parent').size().reset_index(name='Count')
parent_status['Percentage'] = (parent_status['Count'] * 100 /␣
 ↪parent_status['Count'].sum()).round(2)
parent_status
```

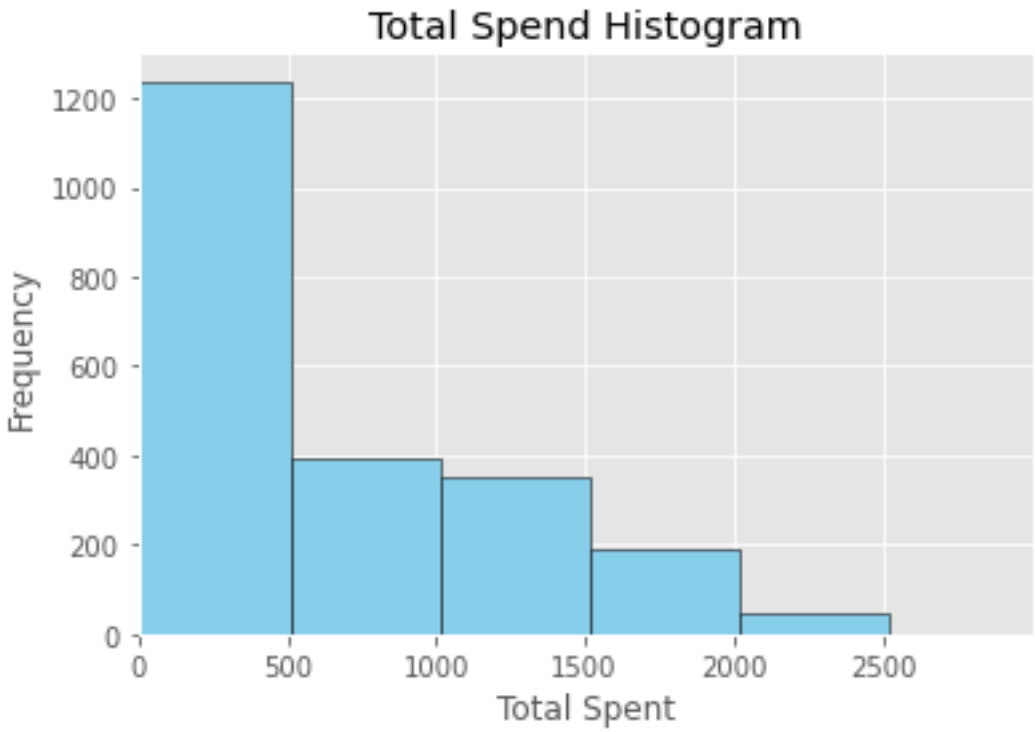
[29]:

	Parent	Count	Percentage
0	False	631	28.53
1	True	1581	71.47

28.53% of the individuals are not parents. 71.47% of the individuals are parents.

Now, I look at the Total_Spent variable.

[30]:
```
plt.hist(new_df['Total_Spent'], bins = 5, color='skyblue', edgecolor='black')

plt.xlim(0, 3000)
plt.xticks(np.arange(0,3000,500))

plt.xlabel('Total Spent')
plt.ylabel('Frequency')
plt.title('Total Spend Histogram')

plt.show()
```



More than 54% of individuals spent less than 500 dollars.

18% of individuals exhibit spending levels between 500 and 1000 dollars, while 16% of individuals fall within the 1000 to 1500 dollar spending range.

Instances of expenditure exceeding 2000 dollars appear to be infrequent.

Next, I examine the preferred purchase method, including in-store, deal-based, web-based, or catalog purchases.

[31]:
```
total_store = new_df['NumStorePurchases'].sum()
print(total_store)
```

```
12830
```

[32]:
```
total_web = new_df['NumWebPurchases'].sum()
print(total_web)
```

```
9032
```

[33]:
```
total_deals = new_df['NumDealsPurchases'].sum()
print(total_deals)
```

```
5135
```

[34]:
```
total_catalog = new_df['NumCatalogPurchases'].sum()
print(total_catalog)
```

```
5902
```

The individuals favor in-store purchases which constitute approximately 38% of total purchases.

Web purchases come in second, constituting approximately 27% of total purchases.

We now find the product categories on which individuals spend the most money.

[35]:
```
total_meat = new_df['MntMeatProducts'].sum()
print(total_meat)
```

```
369338
```

[36]:
```
total_fish = new_df['MntFishProducts'].sum()
print(total_fish)
```

```
82986
```

[37]:
```
total_fruits = new_df['MntFruits'].sum()
print(total_fruits)
```

```
58230
```

[38]:
```
total_sweet = new_df['MntSweetProducts'].sum()
print(total_sweet)
```

```
59829
```

[39]:
```
total_wine = new_df['MntWines'].sum()
print(total_wine)
```

```
674728
```

[40]:
```
total_gold = new_df['MntGoldProds'].sum()
print(total_gold)
```

```
96935
```

[41]:
```
total_spent_products = total_meat + total_fish + total_fruits + total_sweet +␣
↪total_wine + total_gold
total_spent_products
```

[41]:
```
1342046
```

Wine products constitute approximately 50% of total amount spent.

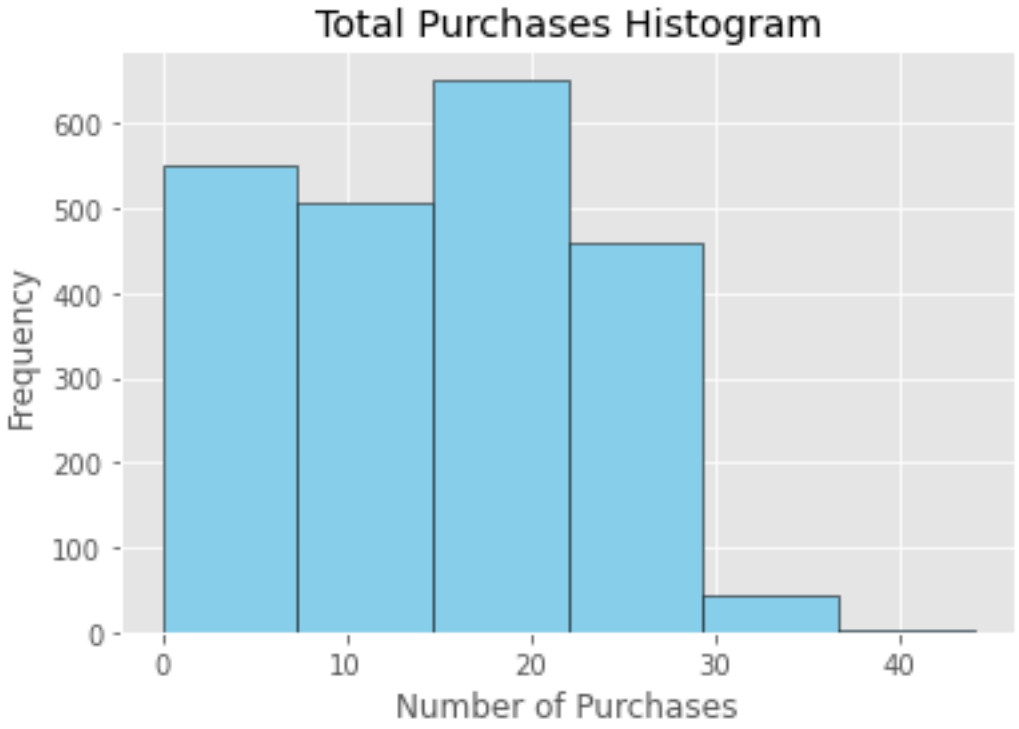
Meet products come in second, constituting approximately 28% of total amount spent.

Fruits only constitute approximately 4% of total amount spent.

We also want to look at the distribution of the total amount of purchases.

[42]:
```
plt.hist(new_df['Total_Purchases'], bins = 6, color='skyblue',␣
↪edgecolor='black')
```

```
plt.xlabel('Number of Purchases')
plt.ylabel('Frequency')
plt.title('Total Purchases Histogram')

plt.show()
```



Around 29% percent of individuals made between 15 and 22 purchases, while 25% made fewer than 7.

Approximately 23% of individuals made between 7 and 15 purchases, and 21% made between 22 and 29 purchases.

Now, I look at the performance of the past 5 marketing campaigns.

[43]:
```
offers = new_df.groupby('Total_Offers').size().reset_index(name='Count')
offers['Percentage'] = (offers['Count'] * 100 / offers['Count'].sum()).round(2)
offers
```

[43]:

	Total_Offers	Count	Percentage
0	0	1754	79.29
1	1	323	14.60
2	2	80	3.62
3	3	44	1.99

```
4             4     11         0.50
```

79% of individuals did not accept any offers during any of the previous 5 marketing campaigns.

15% of individuals accepted only one offer.

7% of individuals accepted between 2 and 4 offers.

There were no customers who accepted all five offers from the marketing campaigns.

Next, I want to find if there is any correlation between the variables.

[44]:
```
new_df.head()
```

[44]:
```
     Education Marital_Status   Income  Kidhome  Teenhome Dt_Customer  Recency  \
0  Graduation         Single  58138.0        0         0  2012-09-04       58
1  Graduation         Single  46344.0        1         1  2014-03-08       38
2  Graduation       Together  71613.0        0         0  2013-08-21       26
3  Graduation       Together  26646.0        1         0  2014-02-10       26
4         PhD        Married  58293.0        1         0  2014-01-19       94

   MntWines  MntFruits  MntMeatProducts  …  Complain  Response  Age  \
0       635         88              546  …         0         1   67
1        11          1                6  …         0         0   70
2       426         49              127  …         0         0   59
3        11          4               20  …         0         0   40
4       173         43              118  …         0         0   43

   Age_Range  Total_Spent  Total_Offers  Total_Purchases  Enrollment_Year  \
0        50+         1617             0               25             2012
1        50+           27             0                6             2014
2        50+          776             0               21             2013
3      36-50           53             0                8             2014
4      36-50          422             0               19             2014

   Seniority  Parent
0         12   False
1         10    True
2         11   False
3         10    True
4         10    True

[5 rows x 33 columns]
```

[45]:
```
corr_df = new_df.drop(['Kidhome', 'Teenhome', 'Dt_Customer', 'Age',␣
 ↪'Enrollment_Year'], axis=1)
corr_df.head()
```

[45]:
```
     Education Marital_Status   Income  Recency  MntWines  MntFruits  \
0  Graduation         Single  58138.0       58       635         88
```

```
1  Graduation         Single  46344.0       38        11          1
2  Graduation       Together  71613.0       26       426         49
3  Graduation       Together  26646.0       26        11          4
4         PhD        Married  58293.0       94       173         43

   MntMeatProducts  MntFishProducts  MntSweetProducts  MntGoldProds  …  \
0              546              172                88            88  …
1                6                2                 1             6  …
2              127              111                21            42  …
3               20               10                 3             5  …
4              118               46                27            15  …

   AcceptedCmp1  AcceptedCmp2  Complain  Response  Age_Range  Total_Spent  \
0             0             0         0         1        50+         1617
1             0             0         0         0        50+           27
2             0             0         0         0        50+          776
3             0             0         0         0      36-50           53
4             0             0         0         0      36-50          422

   Total_Offers  Total_Purchases  Seniority  Parent
0             0               25         12   False
1             0                6         10    True
2             0               21         11   False
3             0                8         10    True
4             0               19         10    True

[5 rows x 28 columns]
```

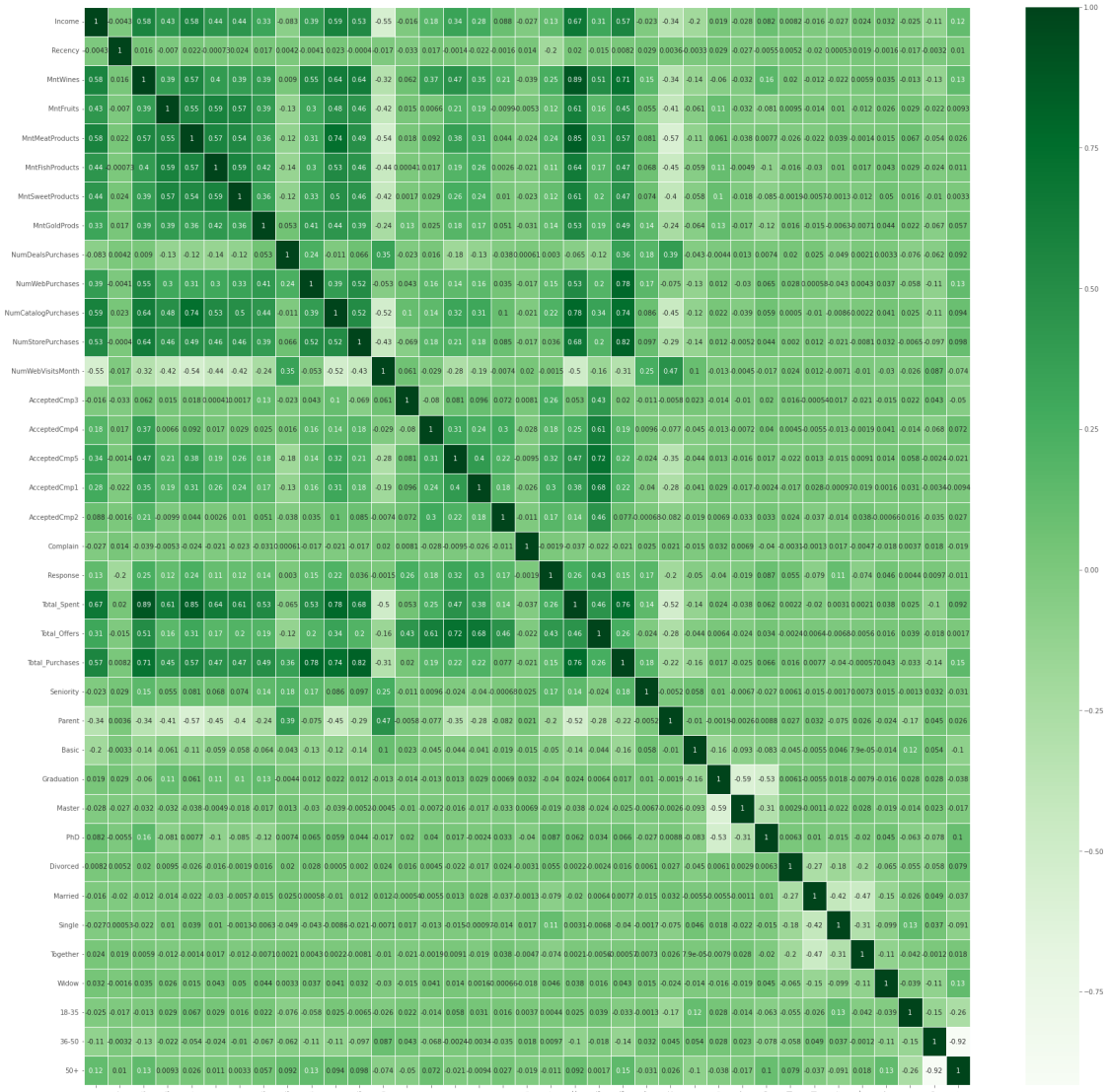
First, I dropped unnecessary variables, and now I will create dummy variables for non-numerical values.

[46]:
```
dummies = pd.get_dummies(corr_df.Education)
dummies_2 = pd.get_dummies(corr_df.Marital_Status)
dummies_3 = pd.get_dummies(corr_df.Age_Range)

merged = pd.concat([corr_df, dummies, dummies_2, dummies_3], axis='columns')

merged = merged.drop(['Education', 'Marital_Status', 'Age_Range'],␣
 ↪axis='columns')
```

[47]:
```
matrix = merged.corr()
```

[48]:
```
fig, ax = plt.subplots(figsize=(30,30))

sns.heatmap(matrix, cmap="Greens", annot=True, linewidths=.5, ax=ax)
```

[48]:
```
<Axes: >
```



The correlation coefficient between the **Income** and **Total_Spent** variables is **0.67**. This indicates a **strong positive** correlation between income and total spent on purchases.

The correlation coefficient between the **Income** and the **Total_Purchases** variables is **0.57**. This indicates a **moderate positive** correlation between income and total purchases.

The correlation coefficient between the **Parent** and **Total_Spent** variables is **-0.52**. This indicates a **moderate negative** correlation between being a parent and total spent on purchases.

There is a **weak positive** correlation (**0.31**) between **total offers** accepted and **income**.

There is a **weak positive** correlation (**0.34**) between **total offers** accepted and the **number of catalog purchases**.

The **Response** variable is **moderately positively** correlated with the **Total_Offers** variable (**0.43**)

There is a **weak positive** correlation between the **Response** variable and variables **Total_Spent** and **NumCatalogPurchases**

6 Model Selection and Training

The main goal of this project is to predict who will respond to an offer for a product or service, thus providing a significant boost to the efficiency of future marketing campaigns.

To accomplish this, I trained several predictive models and analyzed the variables that had the most significant impact on the model's performance.

[49]:
```
corr_df['Marital_Status'] = corr_df['Marital_Status'].replace(['Divorced',␣
↪'Widow'], 'Single')
corr_df['Marital_Status'] = corr_df['Marital_Status'].replace(['Together',␣
↪'Married'], 'In Relationship')

corr_df['Marital_Status'].unique()
```

[49]:
```
array(['Single', 'In Relationship'], dtype=object)
```

[50]:
```
corr_df.head()
```

[50]:

	Education	Marital_Status	Income	Recency	MntWines	MntFruits	\
0	Graduation	Single	58138.0	58	635	88	
1	Graduation	Single	46344.0	38	11	1	
2	Graduation	In Relationship	71613.0	26	426	49	
3	Graduation	In Relationship	26646.0	26	11	4	
4	PhD	In Relationship	58293.0	94	173	43	

	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	…	\
0	546	172	88	88	…	
1	6	2	1	6	…	
2	127	111	21	42	…	
3	20	10	3	5	…	
4	118	46	27	15	…	

	AcceptedCmp1	AcceptedCmp2	Complain	Response	Age_Range	Total_Spent	\
0	0	0	0	1	50+	1617	
1	0	0	0	0	50+	27	
2	0	0	0	0	50+	776	
3	0	0	0	0	36-50	53	
4	0	0	0	0	36-50	422	

	Total_Offers	Total_Purchases	Seniority	Parent
0	0	25	12	False

```
1              0                6          10    True
2              0               21          11   False
3              0                8          10    True
4              0               19          10    True

[5 rows x 28 columns]
```

[51]:
```
corr_df = corr_df.drop(['Education'], axis=1)
```

[52]:
```
dummy = pd.get_dummies(corr_df.Marital_Status)
dummy_2 = pd.get_dummies(corr_df.Age_Range)

model_df = pd.concat([corr_df, dummy, dummy_2], axis='columns')

model_df = model_df.drop(['Marital_Status', 'Age_Range'], axis='columns')

model_df.head()
```

[52]:
```
     Income  Recency  MntWines  MntFruits  MntMeatProducts  MntFishProducts  \
0  58138.0       58       635         88              546              172
1  46344.0       38        11          1                6                2
2  71613.0       26       426         49              127              111
3  26646.0       26        11          4               20               10
4  58293.0       94       173         43              118               46

   MntSweetProducts  MntGoldProds  NumDealsPurchases  NumWebPurchases  …  \
0                88            88                  3                8  …
1                 1             6                  2                1  …
2                21            42                  1                8  …
3                 3             5                  2                2  …
4                27            15                  5                5  …

   Total_Spent  Total_Offers  Total_Purchases  Seniority  Parent  \
0         1617             0               25         12   False
1           27             0                6         10    True
2          776             0               21         11   False
3           53             0                8         10    True
4          422             0               19         10    True

   In Relationship  Single  18-35  36-50    50+
0            False    True  False  False   True
1            False    True  False  False   True
2             True   False  False  False   True
3             True   False  False   True  False
4             True   False  False   True  False

[5 rows x 30 columns]
```

[53]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
```

7 Logistic Regression

[54]:
```
X = model_df.drop('Response',axis= 1)
y = model_df['Response']
```

[55]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=101)
```

[56]:
```
from sklearn import preprocessing as pp
from sklearn.preprocessing import StandardScaler
```

[57]:
```
scaler = StandardScaler()

scaler.fit(X_train)
X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)
```

[58]:
```
model = LogisticRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

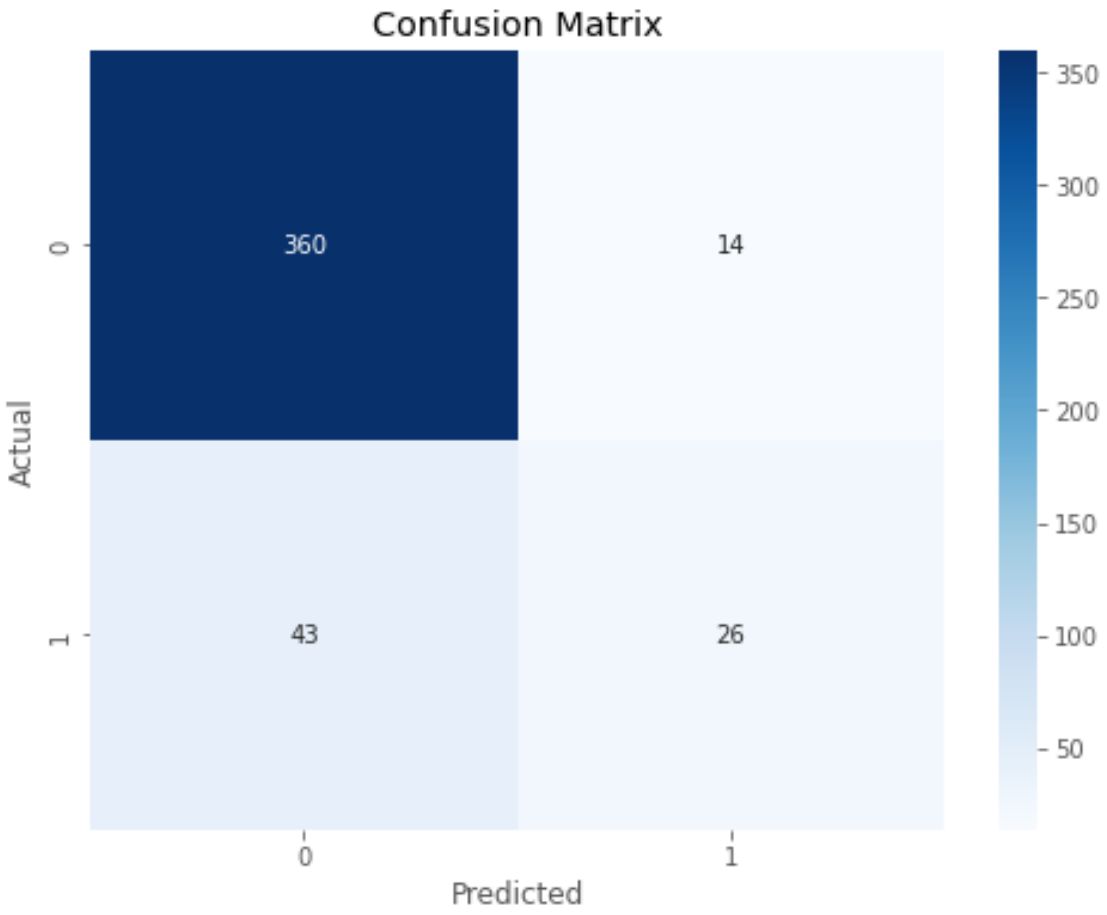
[59]:
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

```
Accuracy: 0.871331828442438
Precision: 0.65
Recall: 0.37681159420289856
F1 Score: 0.47706422018348627
```

[60]:
```
from sklearn.metrics import confusion_matrix
```

[61]:
```
confusion_mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='d')
```

```
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



[62]:
```
reg_summary = pd.DataFrame(data = X.columns.values, columns = ['Feature'])
reg_summary['Coefficient'] = np.transpose(model.coef_)
reg_summary.sort_values(by = 'Coefficient', ascending = True)
```

[62]:

	Feature	Coefficient
1	Recency	-0.919862
11	NumStorePurchases	-0.581430
23	Parent	-0.509488
24	In Relationship	-0.324936
26	18-35	-0.213593
5	MntFishProducts	-0.115244
21	Total_Purchases	-0.096470
6	MntSweetProducts	-0.035780

```
0                 Income     0.011401
18              Complain     0.015646
2               MntWines     0.034881
28                   50+     0.040262
27                 36-50     0.049594
7           MntGoldProds     0.051460
10  NumCatalogPurchases     0.057554
14          AcceptedCmp4     0.091059
17          AcceptedCmp2     0.137069
3              MntFruits     0.145837
19           Total_Spent     0.159522
9        NumWebPurchases     0.200533
8      NumDealsPurchases     0.213546
16          AcceptedCmp1     0.247918
15          AcceptedCmp5     0.324051
25                Single     0.324936
4        MntMeatProducts     0.366553
13          AcceptedCmp3     0.377329
20          Total_Offers     0.413514
12     NumWebVisitsMonth     0.455111
22              Seniority     0.706108
```

8 Decision Tree

[63]:
```
from sklearn import tree
```

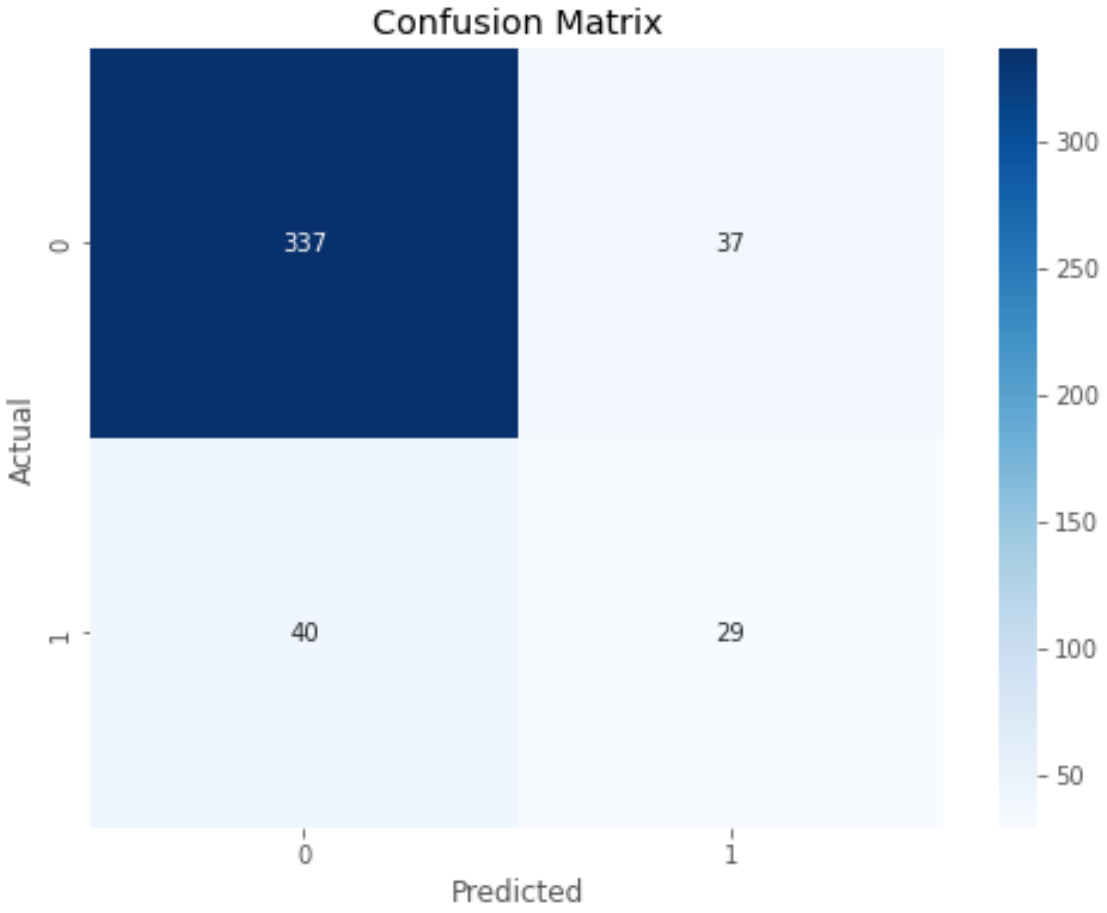
[64]:
```
model = tree.DecisionTreeClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

[65]:
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

```
Accuracy: 0.8261851015801355
Precision: 0.4393939393939394
Recall: 0.42028985507246375
F1 Score: 0.42962962962962964
```

[66]:
```
confusion_mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



[67]:
```
tree_summary = pd.DataFrame(data = X.columns.values, columns = ['Feature'])
tree_summary['Importances'] = np.transpose(model.feature_importances_)
tree_summary.sort_values(by = 'Importances', ascending = False)
```

[67]:

	Feature	Importances
20	Total_Offers	0.161304
1	Recency	0.125591
7	MntGoldProds	0.078045
4	MntMeatProducts	0.075640
19	Total_Spent	0.066754
3	MntFruits	0.058012
13	AcceptedCmp3	0.050859

```
12    NumWebVisitsMonth     0.050147
22            Seniority     0.042531
21      Total_Purchases     0.040039
0                Income     0.033601
2              MntWines     0.032406
11   NumStorePurchases     0.031058
10  NumCatalogPurchases     0.026794
6      MntSweetProducts     0.026168
8     NumDealsPurchases     0.025080
5       MntFishProducts     0.021109
9       NumWebPurchases     0.015299
24      In Relationship     0.012393
25               Single     0.009036
23               Parent     0.007262
27                36-50     0.004590
28                  50+     0.003672
16         AcceptedCmp1     0.001913
14         AcceptedCmp4     0.000698
18             Complain     0.000000
17         AcceptedCmp2     0.000000
26                18-35     0.000000
15         AcceptedCmp5     0.000000
```

9 XGBOOST

[76]:
```
import xgboost as xgb
```

[77]:
```
model = xgb.XGBClassifier(max_depth = 2)
model.fit(X_train, y_train)
```

[77]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=2, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, …)
```

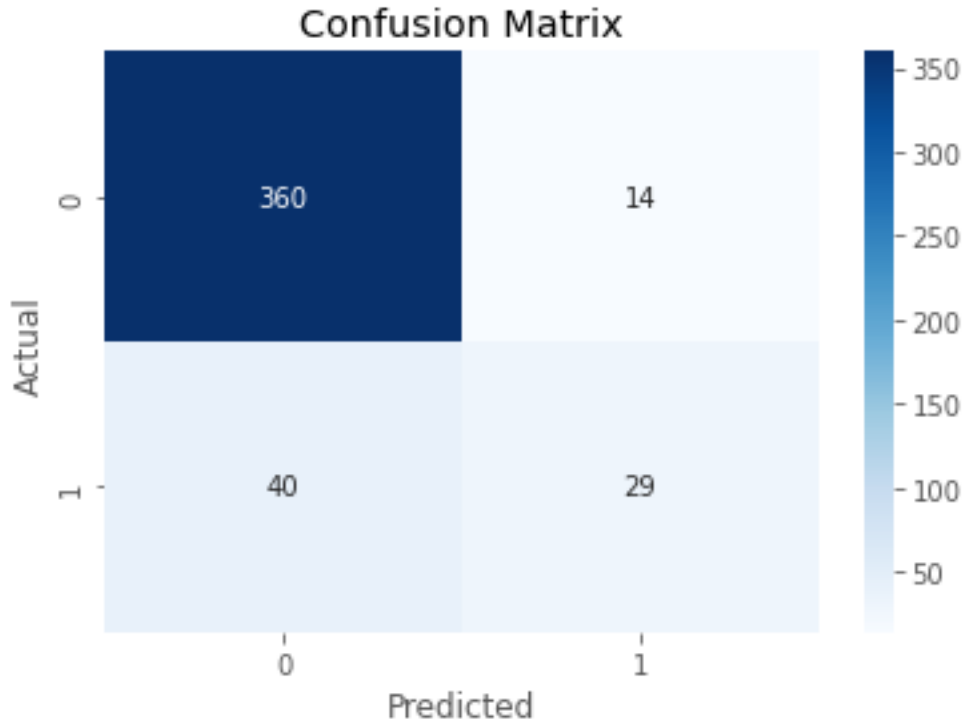
[78]:
```
y_pred = model.predict(X_test)
```

[79]:
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

```
Accuracy: 0.8781038374717833
Precision: 0.6744186046511628
Recall: 0.42028985507246375
F1 Score: 0.5178571428571429
```

[80]:
```
confusion_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



[81]:
```
xgb_summary = pd.DataFrame(data = X.columns.values, columns = ['Feature'])
xgb_summary['Importances'] = model.feature_importances_

xgb_summary.sort_values(by = 'Importances', ascending = False)
```

[81]:

	Feature	Importances
20	Total_Offers	0.237723
22	Seniority	0.081015
24	In Relationship	0.070212
13	AcceptedCmp3	0.066753
1	Recency	0.062922
12	NumWebVisitsMonth	0.050453
23	Parent	0.046253
0	Income	0.044185
4	MntMeatProducts	0.042926
10	NumCatalogPurchases	0.040396
11	NumStorePurchases	0.030488
19	Total_Spent	0.028964
7	MntGoldProds	0.026344
9	NumWebPurchases	0.025987
2	MntWines	0.021333
8	NumDealsPurchases	0.020960
6	MntSweetProducts	0.019538
5	MntFishProducts	0.015609
21	Total_Purchases	0.013355
17	AcceptedCmp2	0.013115
16	AcceptedCmp1	0.012564
3	MntFruits	0.011579
26	18-35	0.011264
15	AcceptedCmp5	0.006063
27	36-50	0.000000
14	AcceptedCmp4	0.000000
25	Single	0.000000
18	Complain	0.000000
28	50+	0.000000

After constructing and evaluating three distinct models (Logistic Regression, Decision Tree, and XGBoost), the XGBoost model demonstrated the highest accuracy, precision, recall, and F1-score.

According to this model, the most significant variables to consider are Total_Offers, AcceptedCmp3, In Relationship, Seniority, NumWebVisitsMonth, and Recency.

In conclusion, based on the comprehensive analysis of the data, the upcoming marketing campaign should focus on individuals who have previously accepted offers, have longer enrollment tenure with the company, recently made purchases, are single, and have demonstrated a high number of visits to the company website in the past month.