Predictive Analytics

ALY 6020, CRN 80405

Professor Vladimir Shapiro

Predictive Analytics - Module 4 Assignment

Submitted By - Richa Umesh Rambhia

Module 4 Assignment - Understanding Magazine Subscription Behavior

Table of Contents

Introduction

Logistic Regression Model

Logistic Regression is a *statistical type of machine learning model* that helps in the **classification and predictive analytics** in order to estimate the probability and likelihood of an event occuring. [1] The difference between a Linear Regression model and a Logistic Regression model is that Linear Regression algorithms are used to identify relationships between a continuous dependent variable and one or more independent variable, whereas Logistic Regression models are used to **predict a category based on categorical type** variable versus the continuous data points.

The different types of Logistic Regression algorithms which are defined based on the categorical data points are as follows.

1. **Binary Logistic Regression**
2. **Multinomial Logistic Regression**
3. **Ordinal Logistic Regression**

SVM Model

A Support Vector Machine (SVM) model is a powerful supervised machine learning algorithm which is used for **classification and regression** based problems. The main idea of the SVM algorithm is to **find the best hyperplane** which *separates the two classes into a high-dimensional feature space*. The algorithm thus tries to find a hyperplane that maximizes the margin between the classes. [2] Hyperplanes are decision boundaries that help classify the data points, and the data points fall on either side of the hyperplane that can be attributed to different classes.

Decision Tree Model

Decision Tree algorithm is a powerful tool for **classification and regression** based problems, which is a **flowchart-like tree structure** where the internal node denotes a test on the attribute, each branch represents the outcome of the test, and each leaf node (terminal node) represents a class label. Decision Tree algorithms are a tree-like model that helps in decision making that is followed by a sequence of if-else questions about the various input data until a prediction is made. [3]

Problem Statement

A magazine company is trying to understand last year’s decline in subscriptions. With people spending more time at home, they thought people would be spending more time reading. Using this dataset, help the company understand what is or isn’t working.

Magazine Subscription Behavior Analysis

The magazine subscription behavior analysis is based on the **marketing dataset** which has over 29 different features in order to understand and analyze the response on the subscriptions of the customer. This will help in recommending the company what parameters are influencing the response variable and what percent of customers have declined the subscriptions.

The marketing dataset has over **2240 rows of data and 29 field values** where the goal is to build **different classification models** in order to classify whether the subscriptions had a response or no from the customer. The various factors that affect the response parameter can be further analyzed in order to understand the parameters affecting and contributing to the target variable. The features present in the marketing dataset are *ID, Education, Marital Status, Income, Kidhome, Teenhome, Number of purchases made, Number of attempts the customer accepted the offer in the campaign, etc.* and that will help in analyzing the various independent variables and classifying the target variable, which in this case is the **Response variable**.

Analysis

Task 1:

Use proper data cleansing techniques to ensure you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data.

Task 2:

Build a logistic model to accurately predict subscription behavior. Discuss which variables are significant, their business impact, and how that may help you learn about the business.

Task 3:

Build an SVM model to accurately predict subscription behavior. Discuss the model's accuracy and how that compares to the logistic model.

Task 4:

Build a decision tree model (at most 4 branches) to accurately predict subscription behavior. Discuss the model's accuracy and how that compares to the other two models (variables and accuracy).

Task 5:

Compare the accuracy of all the models (overall accuracy, precision, recall) and the overall variables that were deemed significant. Discuss which model you would recommend based on these three metrics. Discuss what key variables they should focus on and their business context once you select your final model (that should be the key takeaway).

Installing required packages

In [4]:
```
!pip install pandas_profiling
!pip install featurewiz
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting pandas_profiling
  Downloading pandas_profiling-3.6.6-py2.py3-none-any.whl (324 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 324.4/324.4 kB 11.0 MB/s eta 0:00:00
Collecting ydata-profiling
  Downloading ydata_profiling-4.1.2-py2.py3-none-any.whl (345 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.9/345.9 kB 35.2 MB/s eta 0:00:00
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (0.12.2)
Collecting matplotlib<3.7,>=3.2
  Downloading matplotlib-3.6.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.8/11.8 MB 119.4 MB/s eta 0:00:00
Requirement already satisfied: pandas!=1.4.0,<1.6,>1.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (1.5.3)
Requirement already satisfied: pydantic<1.11,>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (1.10.7)
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (0.13.5)
```

In [ ]:
```
!pip install --upgrade scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.9.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.22.4)
```

Importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import pandas_profiling
import ydata_profiling
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import KNNImputer
from featurewiz import featurewiz
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import Lasso, LogisticRegression
from sklearn.utils.class_weight import compute_class_weight
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_score, recall_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import tree
```

Loading the dataset

In [2]:
```
marketing_data = pd.read_csv("marketing_campaign-1.csv")
marketing_data
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisits
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	11	...	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	426	...	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	11	...	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	173	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-06-13	46	709	...	
2236	4001	1946	PhD	Together	64014.0	2	1	2014-06-10	56	406	...	
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-01-25	91	908	...	
2238	8235	1956	Master	Together	69245.0	0	1	2014-01-24	8	428	...	
2239	9405	1954	PhD	Married	52869.0	1	1	2012-10-15	40	84	...	

2240 rows × 29 columns

Table 1. Marketing Data for Magazine Behavior Analysis

- **ID:** Customer's Unique Identifier
- **Year_Birth:** Customer's Birth Year
- **Education:** Customer's education level
- **Marital_Status:** Customer's marital status
- **Income:** Customer's yearly household income
- **Kidhome:** Number of children in customer's household
- **Tennhome:** Number of teenagers in customer's household
- **Dt_Customer:** Date of customer's enrollment with the company
- **Recency:** Number of days since customer's last purchase
- **MntWines:** Amount spent on wine in the last 2 years
- **MntFruits:** Amount spent on fruits in the last 2 years
- **MntMeatProducts:** Amount spent on meat in the last 2 years
- **MntFishProducts:** Amount spent on fish in the last 2 years
- **MntSweetProducts:** Amount spent on sweets in the last 2 years
- **MntGoldProds:** Amount spent on gold in the last 2 years
- **NumDealsPurchase:** Number of purchases made with a discount
- **NumWebPurchase:** Number of purchases made through the company's website
- **NumCatalogPurchase:** Number of purchases made using a catalog
- **NumStorePurchase:** Number of purchases made directly in stores
- **NumWebVisitsMonth:** Number of visits to company's website in the last month
- **AcceptedCmp3:** 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- **AcceptedCmp4:** 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- **AcceptedCmp5:** 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- **AcceptedCmp1:** 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- **AcceptedCmp2:** 2 if customer accepted the offer in the 2nd campaign, 0 otherwise
- **Response:** 1 if customer accepted the offer in the last campaign, 0 otherwise
- **Complain:** 1 if a customer complained in the last 2 years, 0 otherwise
- **Country:** Customer's Location

Figure 1. Data Dictionary for Marketing Data Analysis

Step 1: Exploratory Data Analysis

EDA is performed on the data in order to analyze various parameters and features of the dataset and to understand the *structure* of the dataset such that various *trends and patterns* between the variables is known. Exploratory Data Analysis helps in understanding the *relationship between the various independent and dependent variables* of the dataset that would further be useful in building the model such as description analysis and statistical analysis.

Descriptive Analysis

In [7]:
```
# displaying number of rows and columns
print("Total number of Rows and Columns:", marketing_data.shape)

print("\n-----------------------------------------------------------")

# displaying field values/column names
print("\nColumn Names:\n")
marketing_data.columns
```

```
Total number of Rows and Columns: (2240, 29)

-----------------------------------------------------------

Column Names:
```

Out[7]:
```
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')
```

In [8]:
```
# displaying data types
print("Data types:\n")
marketing_data.dtypes
```

```
Data types:
```

Out[8]:
```
ID                     int64
Year_Birth             int64
Education             object
Marital_Status        object
Income               float64
Kidhome                int64
Teenhome               int64
Dt_Customer           object
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
Z_CostContact          int64
Z_Revenue              int64
Response               int64
dtype: object
```

Table 2. Data types for Marketing Data

From the *descriptive analysis*, it is observed that there are total **2240 rows of data** and **29 field values** and the data type for each of the field value is displayed in order to understand what data type values are present in the dataset.

Here, there are different types of data points that are present in the dataset which are **numerical data type** having either *'int' or 'float'* values, along with **object data type** which further needs to be corrected to *string or category* type of data and *datetime* datatype.

Statistical Analysis

In [9]:
```
# dataset info
print("Dataset Info:\n")
marketing_data.info()
```

```
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
```

Table 3. Information about the dataset

In [10]:
```
# describing the dataset
print("Describing the dataset:\n")
round(marketing_data.describe(),1)
```

```
Describing the dataset:
```

Out[10]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	...	Num
count	2240.0	2240.0	2216.0	2240.0	2240.0	2240.0	2240.0	2240.0	2240.0	2240.0	...	
mean	5592.2	1968.8	52247.3	0.4	0.5	49.1	303.9	26.3	167.0	37.5	...	
std	3246.7	12.0	25173.1	0.5	0.5	29.0	336.6	39.8	225.7	54.6	...	
min	0.0	1893.0	1730.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
25%	2828.2	1959.0	35303.0	0.0	0.0	24.0	23.8	1.0	16.0	3.0	...	
50%	5458.5	1970.0	51381.5	0.0	0.0	49.0	173.5	8.0	67.0	12.0	...	
75%	8427.8	1977.0	68522.0	1.0	1.0	74.0	504.2	33.0	232.0	50.0	...	
max	11191.0	1996.0	666666.0	2.0	2.0	99.0	1493.0	199.0	1725.0	259.0	...	

8 rows × 26 columns

Table 4. Dataset Description

Statistical Analysis helps in understanding about each of the numerical field type based on the **total count values, minimum value, maximum value, standard deviation**, etc. giving an overall analysis of the field data points about the various rows present in the dataset.

For example, as observed in the dataset, we see that there are multiple field values having the *minimum, maximum values* along with the *total count of values* which is **2240** and *standard deviation* of the column values. It can be observed that the maximum value of *Income* is **666666.0** and the maximum value of *Recency* is **99.0**.

Thus, similarly, other parameters of the dataset can be analyzed based on their statistical values.

Data Profiling

In [11]:
```
marketing_data_report = marketing_data.profile_report(title='Marketing Data Analysis Report', explorative = T
marketing_data_report
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Out[11]:

In [12]:
```
# Saving the profile report
marketing_data_report.to_file(output_file="Marketing Data Analysis Report.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

The data profiling report generated for the dataset helps in understanding various parameters such as the data type of the field values, the missing and duplicate values present in the dataset, the correlation between each of the field value, and the analysis of each of the field value on a individual basis based on correlation plot, histogram, and interaction graphs.

From the profiling report, it is observed that there are **15 numerical variable type and 14 categorical data type** field values present in the dataset of which the numerical data type have **integer and float values**, along with string and datetime objects. Apart from this, there are **missing values** present in the dataset, and the missing values visualization or plot also helps in understanding the same, for which the percent of missing values is less than **0.1% i.e., 24 missing cells** are present in the dataset for one of the field value. For each field value a separate visualization is also displayed in order to specifially analyze a particular field value, and there are **no duplicate values present in the dataset.**

It can also be observed that there are two constant data values present in the dataset; **'Z_CostContact' and 'Z_Revenue'**, and each of the field value displays the *correlation with each other indicating overall correlation* along with any *imbalance data present in the column data.*

Further cleaning of the data is implemented in the below steps.

Step 2: Data Cleaning

Task 1: Use proper data cleansing techniques to ensure you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data.

---

1. Checking for null values in each column of the dataset, i.e., missing values
2. Replacing missing values using various imputation methods
3. Checking for incorrect data types in field values and correcting the data type of the column
4. Checking for outliers in the dataset

    a. *Boxplot*

    b. *Distribution Plot*

---

1. Checking for null values in each column of the dataset, i.e., missing values

In [ ]:
```
for x in range(29):
    print("%-45s %10d" % (marketing_data.columns.values[x], marketing_data.iloc[:,x].isna().sum()))
```

```
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
```

Table 5. Missing or null values

As observed from the table above, there are missing values present in the **Income** column that needs to be addressed by either imputation methods or dropping the rows of data if there are less than 20% of missing values. Hence, the data is analyzed in order to understand which imputation method needs to be implemented.

2. Replacing missing values using various imputation methods

In [ ]:
```
# displaying unique data

print("Displaying the unique data present in columns\n")
marketing_data.nunique()
```

```
Displaying the unique data present in columns
```

Out[14]:
```
ID                     2240
Year_Birth               59
Education                 5
Marital_Status            8
Income                 1974
Kidhome                   3
Teenhome                  3
Dt_Customer             663
Recency                 100
MntWines                776
MntFruits               158
MntMeatProducts         558
MntFishProducts         182
MntSweetProducts        177
MntGoldProds            213
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      14
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Z_CostContact             1
Z_Revenue                 1
Response                  2
dtype: int64
```

Table 6. Unique Data Count

In [ ]:
```
# checking for data distribution of field values 'Income' as it has missing values


# distribution plot for Income column

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(marketing_data['Income'])
plt.title("Distribution Plot for Income")
plt.show()
```

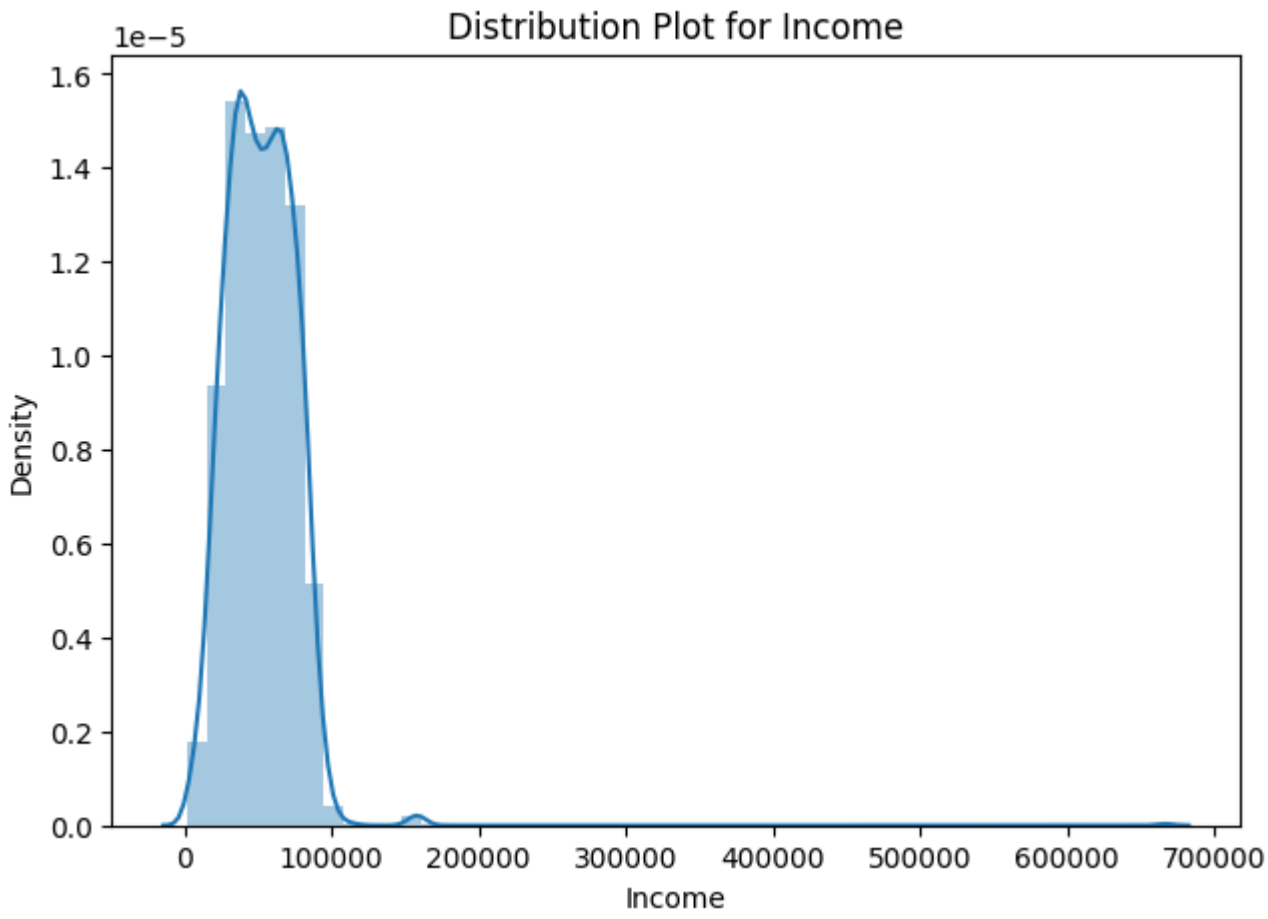

Figure 2. Distribution Plot for Income variable

The distribution plot for the Income variable above shows that the data points are right skewed and hence if applying a mean-median imputation, the median method needs to be applied to replace the missing values present in the dataset.

In [ ]:
```
# checking for missing values in Income column

missing_values = pd.isnull(marketing_data['Income'])
missing_values
```

Out[16]:
```
0       False
1       False
2       False
3       False
4       False
        ...
2235    False
2236    False
2237    False
2238    False
2239    False
Name: Income, Length: 2240, dtype: bool
```

KNN Imputation method for replacing missing values in Income column

In [3]:
```
# KNN imputation method for replacing missing values in Income column

imputer = KNNImputer(n_neighbors=3)
marketing_data['Income'] = imputer.fit_transform(marketing_data[['Income']])
print("Imputation successful.")
```

```
Imputation successful.
```

Since the missing values are present in the **Income** column, which indicates the income of a person, it **cannot be randomly imputed using mean or median** method as the approach would be inefficient. Thus, using a **KNN imputation** or **regression imputation** method the missing values can be replaced or imputed.

The above code successful imputed the missing values for the Income column using the **KNN imputation method** which is more efficient as compared to other imputation methods in this scenario.

In [ ]:
```
# checking for dataframe after imputation

marketing_data.head(20)
```

Out[18]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVis
0	5524	1957	Graduation	Single	58138.000000	0	0	2012-09-04	58	635	...	
1	2174	1954	Graduation	Single	46344.000000	1	1	2014-03-08	38	11	...	
2	4141	1965	Graduation	Together	71613.000000	0	0	2013-08-21	26	426	...	
3	6182	1984	Graduation	Together	26646.000000	1	0	2014-02-10	26	11	...	
4	5324	1981	PhD	Married	58293.000000	1	0	2014-01-19	94	173	...	
5	7446	1967	Master	Together	62513.000000	0	1	2013-09-09	16	520	...	
6	965	1971	Graduation	Divorced	55635.000000	0	1	2012-11-13	34	235	...	
7	6177	1985	PhD	Married	33454.000000	1	0	2013-05-08	32	76	...	
8	4855	1974	PhD	Together	30351.000000	1	0	2013-06-06	19	14	...	
9	5899	1950	PhD	Together	5648.000000	1	1	2014-03-13	68	28	...	
10	1994	1983	Graduation	Married	52247.251354	1	0	2013-11-15	11	5	...	
11	387	1976	Basic	Married	7500.000000	0	0	2012-11-13	59	6	...	
12	2125	1959	Graduation	Divorced	63033.000000	0	0	2013-11-15	82	194	...	
13	8180	1952	Master	Divorced	59354.000000	1	1	2013-11-15	53	233	...	
14	2569	1987	Graduation	Married	17323.000000	0	0	2012-10-10	38	3	...	
15	2114	1946	PhD	Single	82800.000000	0	0	2012-11-24	23	1006	...	
16	9736	1980	Graduation	Married	41850.000000	1	1	2012-12-24	51	53	...	
17	4939	1946	Graduation	Together	37760.000000	0	0	2012-08-31	20	84	...	
18	6565	1949	Master	Married	76995.000000	0	1	2013-03-28	91	1012	...	
19	2278	1985	2n Cycle	Single	33812.000000	1	0	2012-11-03	86	4	...	

20 rows × 29 columns

Table 7. Marketing Data after data imputation

In [ ]:
```
# checking for any missing values after data cleaning & imputations

for x in range(29):
    print("%-45s %10d" % (marketing_data.columns.values[x], marketing_data.iloc[:,x].isna().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
```

3. Checking for incorrect data types in field values and correcting the data type of the column

In [4]:
```
# correcting the data types for the variables of the dataset which are of object type to string/category and

marketing_data['Education'] = marketing_data['Education'].astype('category')
marketing_data['Marital_Status'] = marketing_data['Marital_Status'].astype('category')
marketing_data['Dt_Customer'] = marketing_data['Dt_Customer'].astype('datetime64[ns]')
marketing_data['Income'] = marketing_data['Income'].astype('int')
print("Data Type conversion successful.")
```

```
Data Type conversion successful.
```

In [15]:
```
# checking for the correct data type of the variable

marketing_data.dtypes
```

Out[15]:
```
ID                              int64
Year_Birth                      int64
Education                    category
Marital_Status               category
Income                          int64
Kidhome                         int64
Teenhome                        int64
Dt_Customer           datetime64[ns]
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
Z_CostContact                   int64
Z_Revenue                       int64
Response                        int64
dtype: object
```

In [ ]:
```
# checking for data values after data type conversion

marketing_data
```

Out[22]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisits
0	5524	1957	Graduation	Single	58138	0	0	2012-09-04	58	635	...	
1	2174	1954	Graduation	Single	46344	1	1	2014-03-08	38	11	...	
2	4141	1965	Graduation	Together	71613	0	0	2013-08-21	26	426	...	
3	6182	1984	Graduation	Together	26646	1	0	2014-02-10	26	11	...	
4	5324	1981	PhD	Married	58293	1	0	2014-01-19	94	173	...	
...	...	...	...	...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223	0	1	2013-06-13	46	709	...	
2236	4001	1946	PhD	Together	64014	2	1	2014-06-10	56	406	...	
2237	7270	1981	Graduation	Divorced	56981	0	0	2014-01-25	91	908	...	
2238	8235	1956	Master	Together	69245	0	1	2014-01-24	8	428	...	
2239	9405	1954	PhD	Married	52869	1	1	2012-10-15	40	84	...	

2240 rows × 29 columns

Table 8. Marketing Data after data type conversion

4. Checking for outliers in the dataset

a. Boxplot

The below code creates **boxplots** for the various field values of the marketing dataset in order to check for outliers present in the dataset. Here, the boxplots are implemented for the variables **Kidhome**, **Teenhome**, **Recency**, and **MntWines** as shown in the figures below.

There are outliers present in the variables as observed in the boxplot below, which will not be removed as each of the data point is important for analysis and model building.

In [ ]:
```
# creating boxplot for 'Kidhome' and 'Teenhome' variable

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].boxplot(marketing_data['Kidhome'])
axs[1].boxplot(marketing_data['Teenhome'])
axs[0].set_title('Boxplot for column, Kidhome')
axs[1].set_title('Boxplot for column, Teenhome')
axs[0].set_ylabel('Data Values')
axs[1].set_ylabel('Data Values')

plt.show()
```

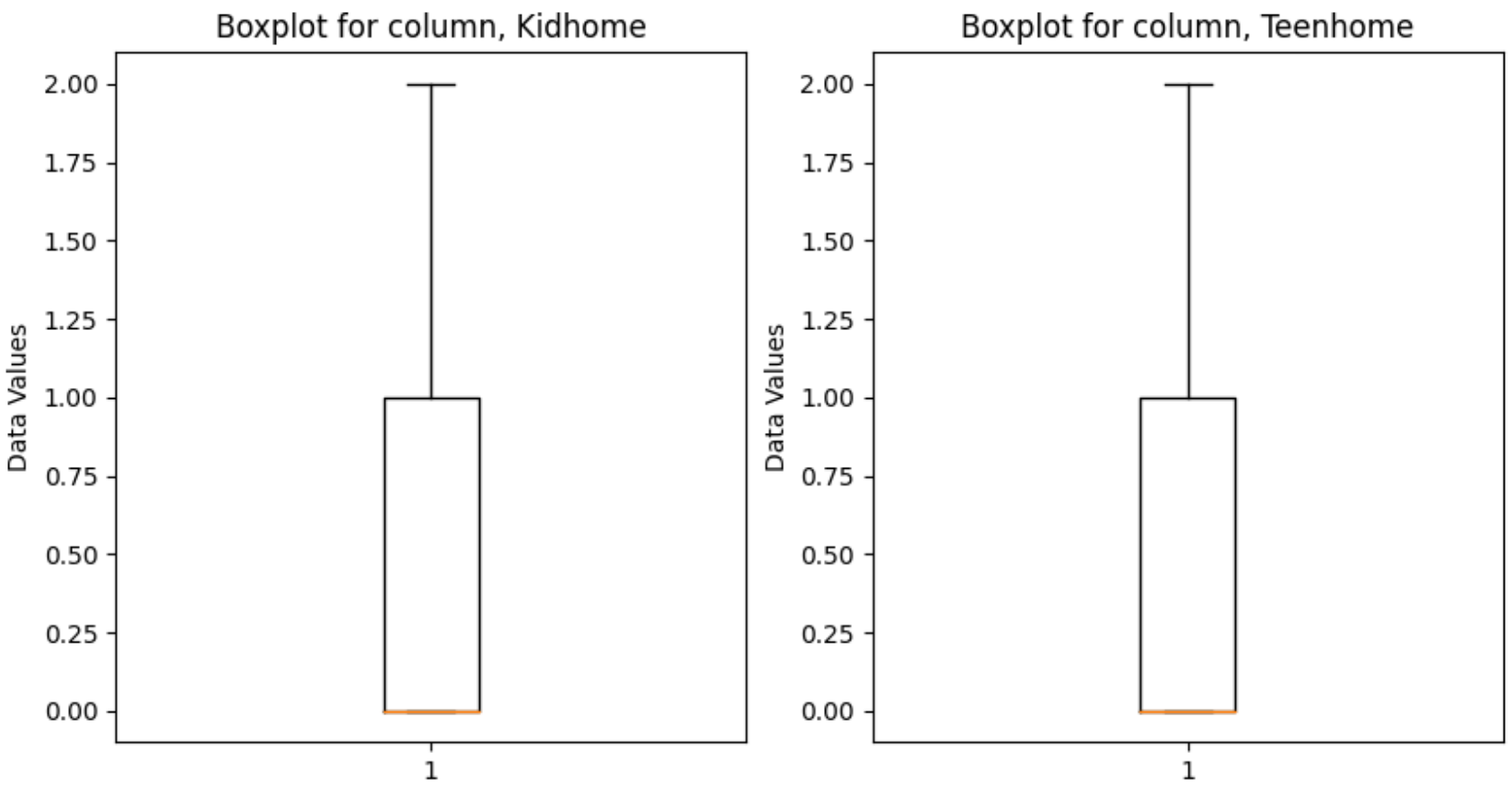

Figure 3. Boxplot for Kidhome & Teenhome column

In [ ]:
```
# creating boxplot for 'Recency' and 'MntWines' variable

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].boxplot(marketing_data['Recency'])
axs[1].boxplot(marketing_data['MntWines'])
axs[0].set_title('Boxplot for column, Recency')
axs[1].set_title('Boxplot for column, MntWines')
axs[0].set_ylabel('Data Values')
axs[1].set_ylabel('Data Values')

plt.show()
```

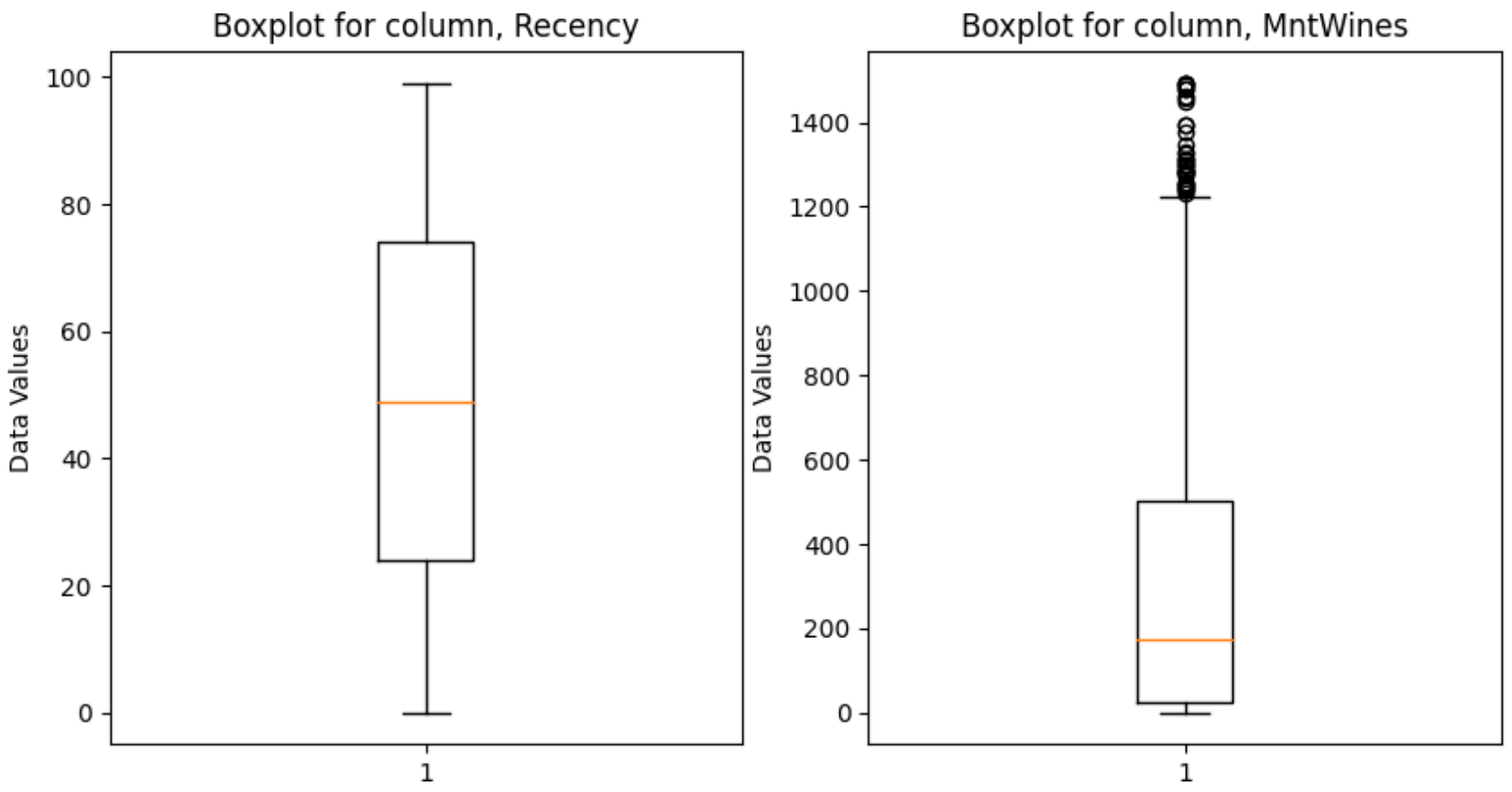

Figure 4. Boxplot for Recency & MntWines column

b. Distribution Plot

The distribution plot for the various parameters of the dataset values gives an overview of the outliers that are present and the distribution of the data points across present in the dataset.

The plot below for *NumStorePurchases* shows that the data is **normally distributed** across the data points, meaning that the data points are evenly distributed around the mean value. However, the plot for the *MntFruits* and *MntMeatProducts* variable indicates that the data is **right skewed**, i.e., the data is concentrated towards a certain range of values and is not equally distributed.

In [ ]:
```
# distribution plot for MntFruits & MntMeatProducts

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(marketing_data['MntFruits'])
plt.subplot(1,2,2)
sns.distplot(marketing_data['MntMeatProducts'])
plt.show()
```

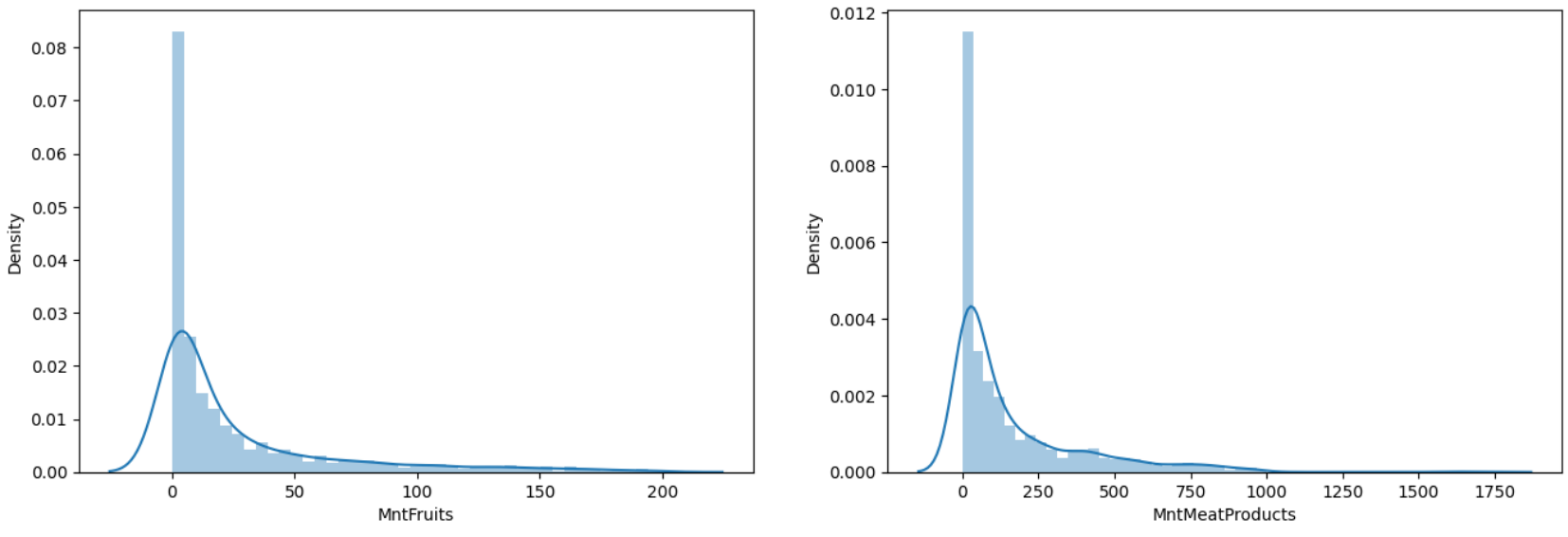

Figure 5. Distribution Plot for MntFruits & MntMeatProducts

In [ ]:
```
# distribution plot for NumStorePurchases & NumWebPurchases

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(marketing_data['NumStorePurchases'])
plt.subplot(1,2,2)
sns.distplot(marketing_data['NumWebPurchases'])
plt.show()
```

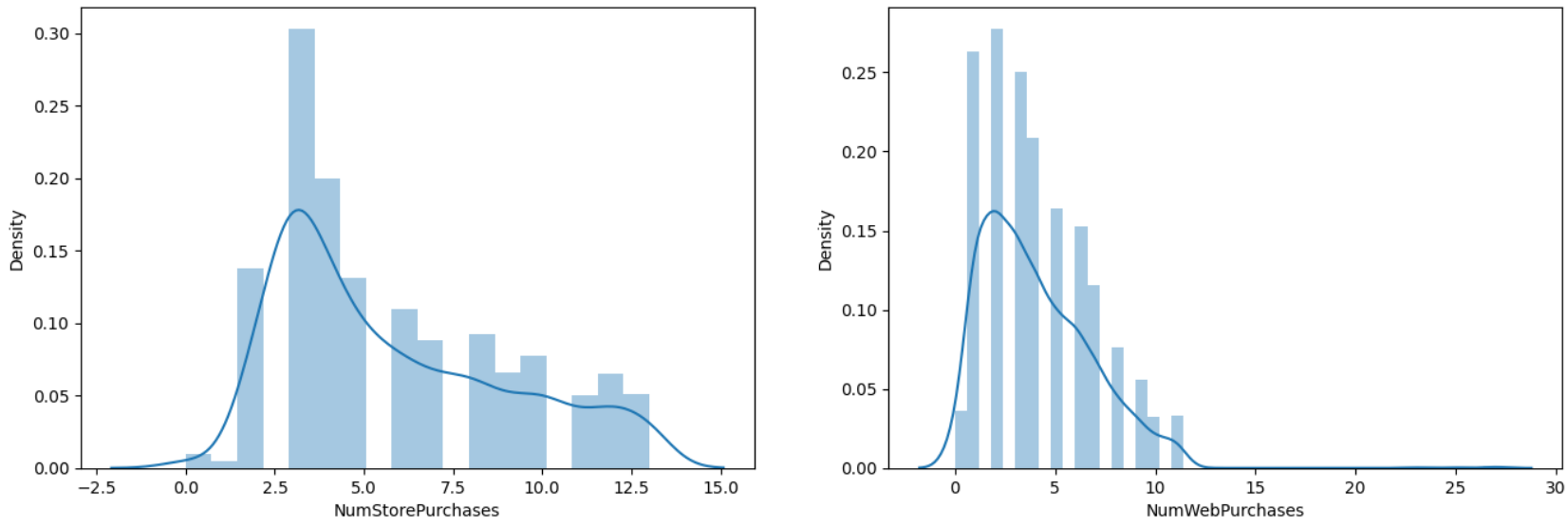

Figure 6. Distribution Plot for NumStorePurchases & NumWebPurchases

In [ ]:
```
# distribution plot for NumDealsPurchases & NumCatalogPurchases

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(marketing_data['NumDealsPurchases'])
plt.subplot(1,2,2)
sns.distplot(marketing_data['NumCatalogPurchases'])
plt.show()
```

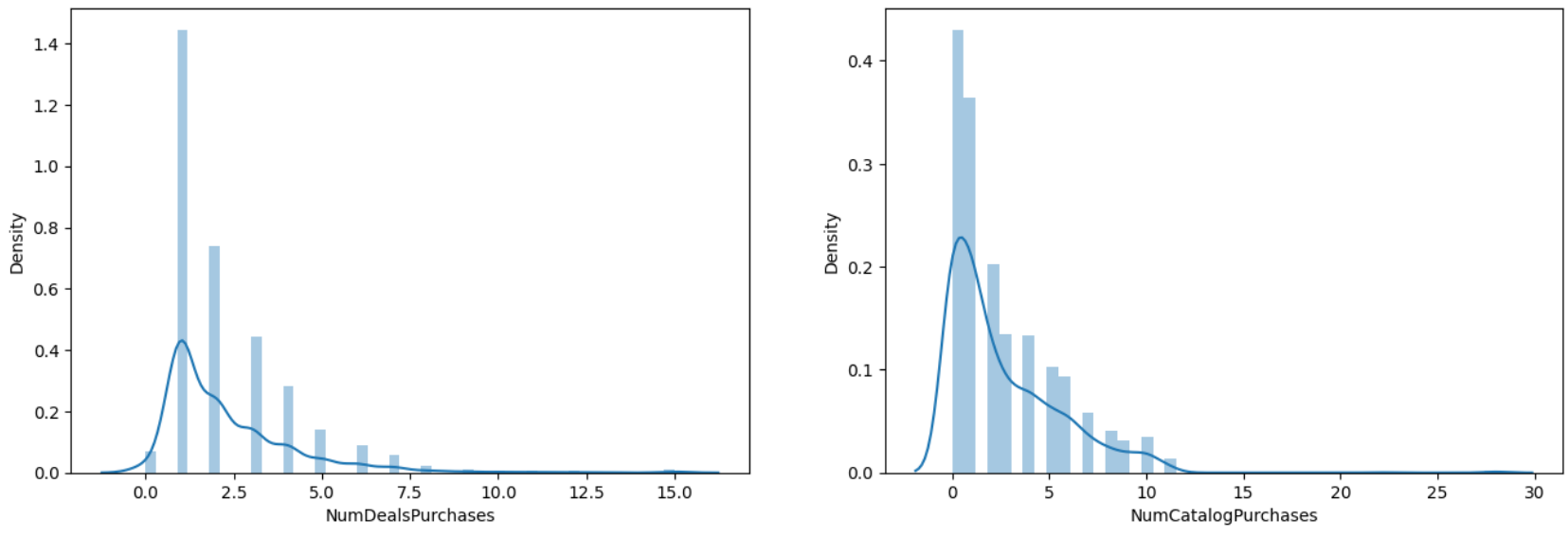

Figure 7. Distribution Plot for NumDealsPurchases & NumCatalogPurchases

Here, the data points for *NumDealsPurchase* and *NumCatalogPurchases* are right skewed and the data is concentrated towards a range of values and not normally distributed across the mean values.

Step 3: Data Visualization

In [ ]:
```
# 1. Response Count Analysis

plt.figure(figsize=(6,5))
sns.countplot(x='Response', data=marketing_data, palette="pastel")
plt.title('\nResponse Count Analysis')
plt.show()
```

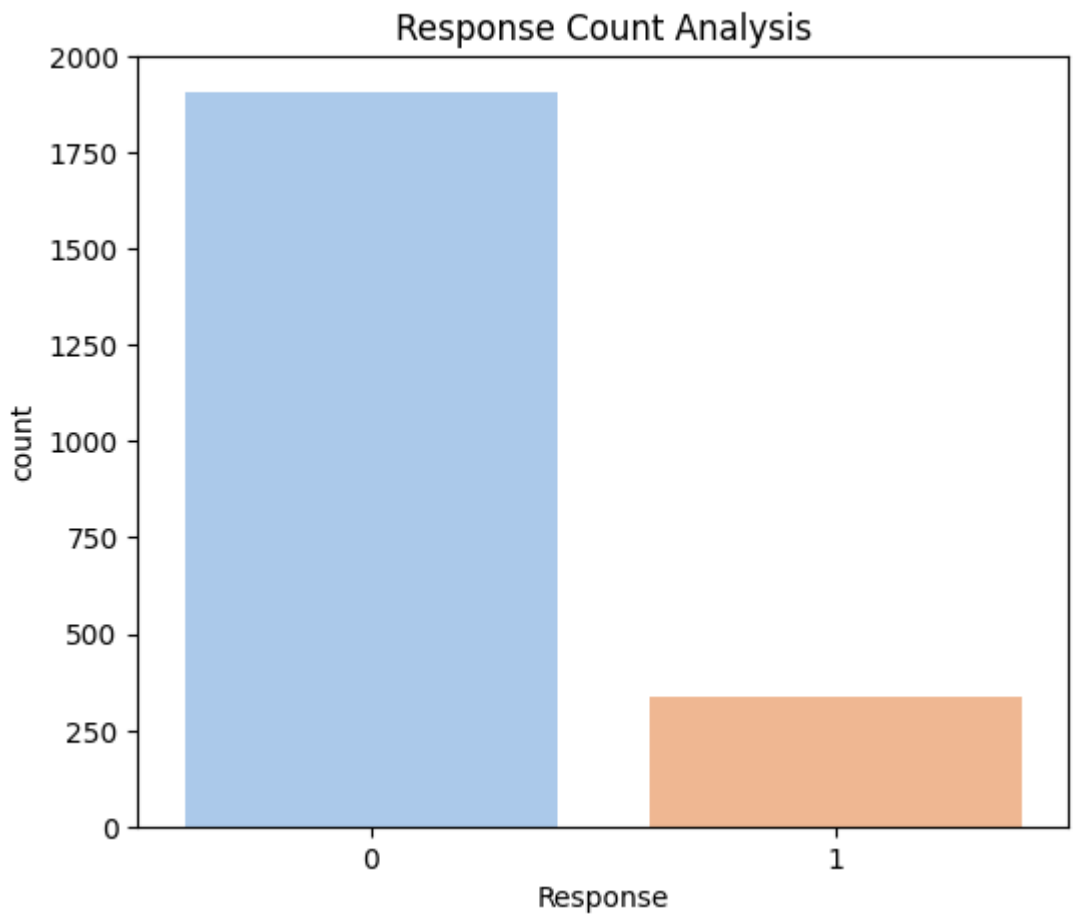

Figure 8. Response Count Analysis

From the above graph of *Response Count Analysis*, it is observed that the data is **biased towards one class** of no reponse, and hence there is a possibility that the model will be biased to this class which will classify majority of the data points into no response from the customers.

In [ ]:
```
# 2. Count Analysis of Number of Deal and Web Purchases

fig, axs = plt.subplots(1, 2, figsize=(12, 5))
sns.countplot(x='NumDealsPurchases', data=marketing_data, palette="pastel", ax=axs[0])
axs[0].set_title('\nAnalysis of Number of Deal Purchases')
sns.countplot(x='NumWebPurchases', data=marketing_data, palette="pastel", ax=axs[1])
axs[1].set_title('\nAnalysis of Number of Web Purchases')
plt.show()
```

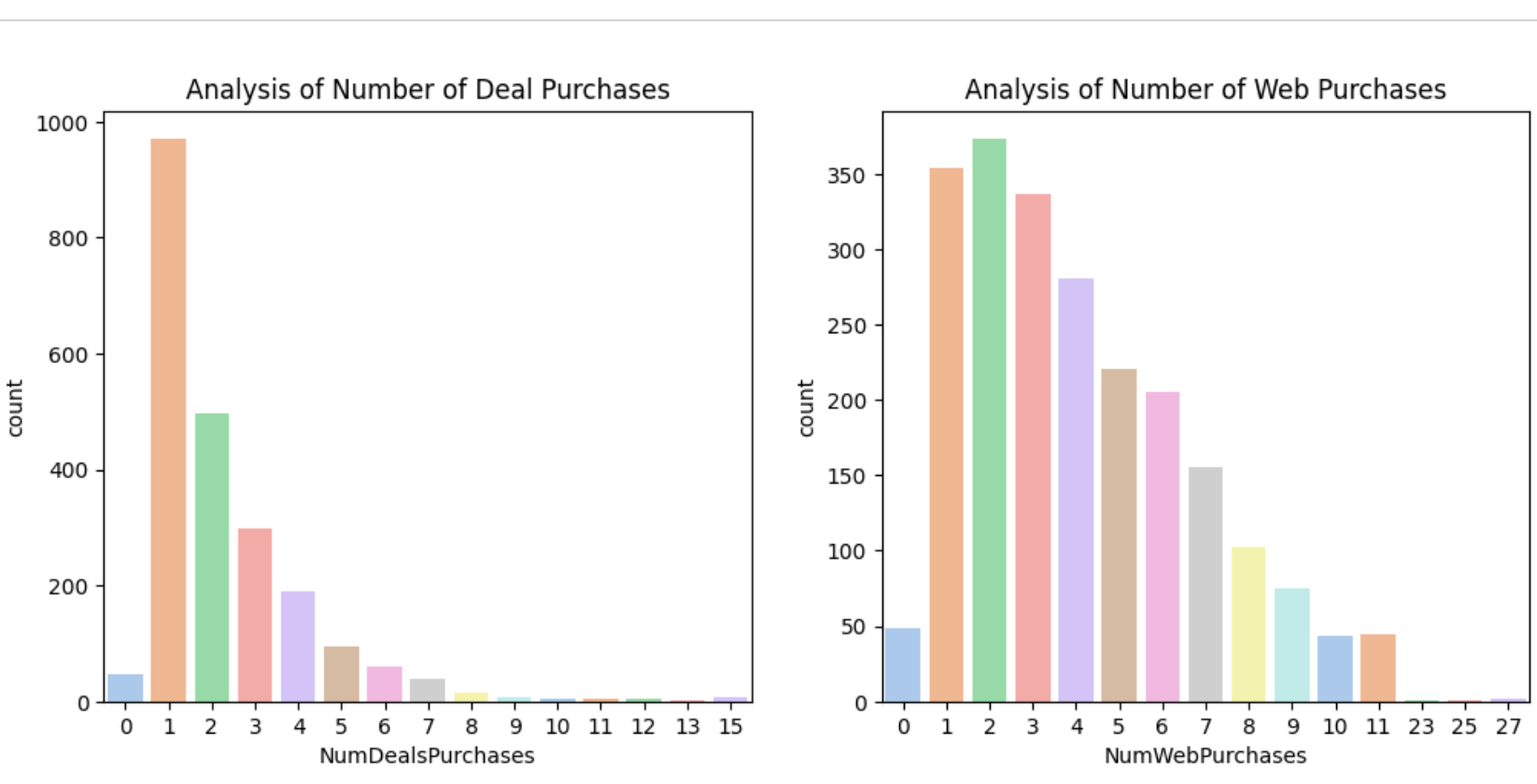

Figure 9. Analysis of Number of Deal Purchases & Number of Web Purchases

The above visualization plots for the **NumDealsPurchases graph** and **NumWebPurchases** helps in analyzing the count of purchases where the various count values of the percentage are displayed that can show which percent value has a higher count. This helps in understanding the count of number of deal purchases and number of web purchases of a customer.

In [ ]:
```
# 3. Count Analysis of Number of Catalog and Store Purchases

fig, axs = plt.subplots(1, 2, figsize=(12, 5))
sns.countplot(x='NumCatalogPurchases', data=marketing_data, palette="pastel", ax=axs[0])
axs[0].set_title('\nAnalysis of Number of Catalog Purchases')
sns.countplot(x='NumStorePurchases', data=marketing_data, palette="pastel", ax=axs[1])
axs[1].set_title('\nAnalysis of Number of Store Purchases')
plt.show()
```

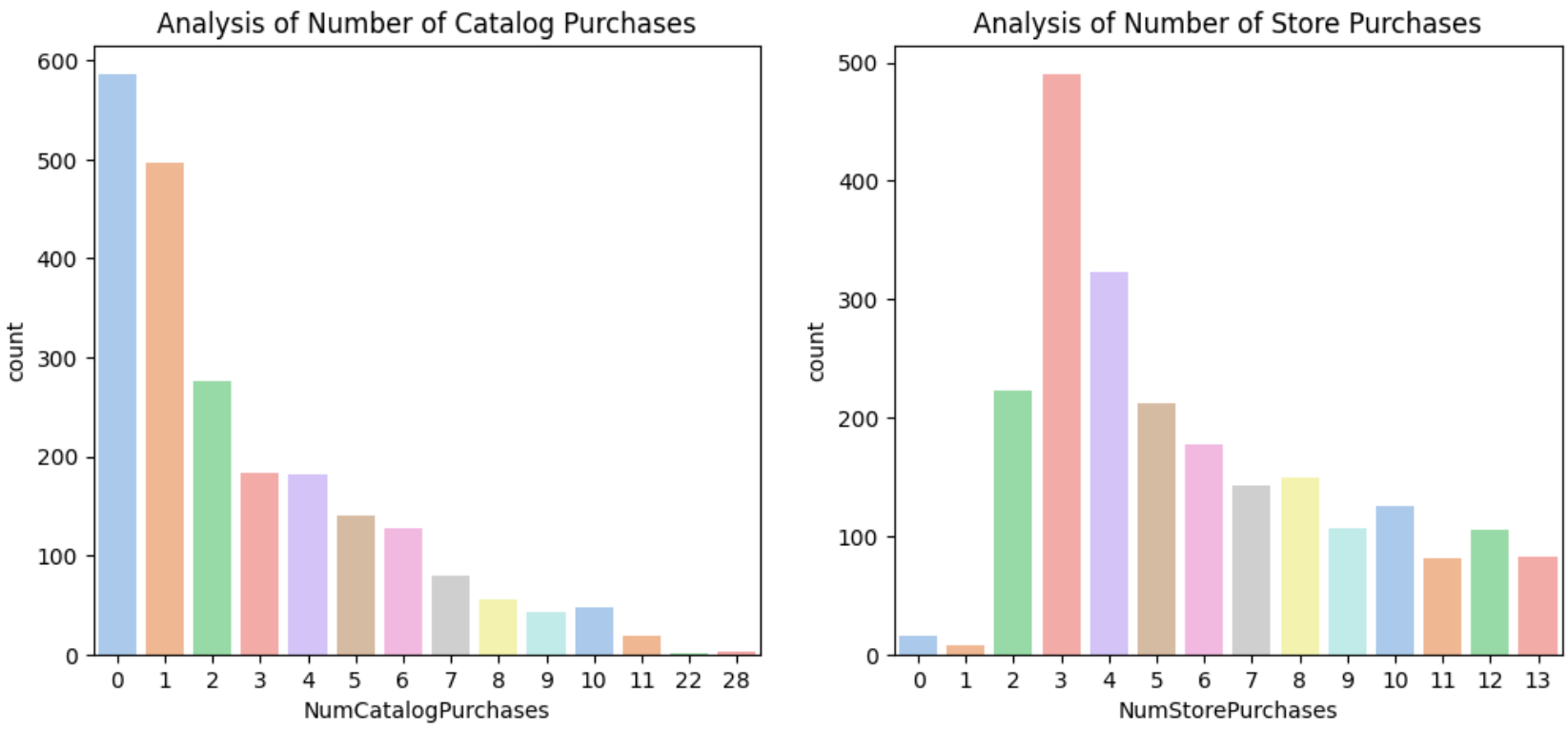

Figure 10. Analysis of NumCatalogPurchases & NumStorePurchases

The above visualization plots for the **NumCatalogPurchases graph** and **NumStorePurchases** helps in analyzing the count of purchases where the various count values of the percentage are displayed that can show which percent value has a higher count. This helps in understanding the count of number of catalog purchases and number of store purchases of a customer.

In [ ]:
```
# 4. Presence of Complain (Yes or No)

plt.figure(figsize=(5,5))
sns.countplot(x='Complain', data=marketing_data, palette="pastel")
plt.title('\nPresence of Complain from Customer in last 2 years')
plt.show()
```

Figure 11. Count of Presence of Complain from Customer

This graph above displays the count of presence of complain received from the customer. Again, the visualization helps to understand that the class is biased towards one category and hence requires to be addressed further to avoid class imbalance and inefficient model performance.

Step 4: Pre-Modeling Steps

1. Label Encoding
2. Correlation Plot
3. Defining the features for model training
4. Spliting the dataset into train & test set
5. Class Bias
6. Standardization

1. Label Encoding

Since some of the selected features for prediction are categorical data and most of the ML models require the data to be numerical or binary, it is important that these features are converted into **binary** or **numerical** type data in order for the model to be able to classify the classes.

Label Encoding is a method which helps to **convert the categorical variables** into **numerical values**, thus helping to transform the data point into a format where the algorithm is able to process the data for classification. *LabelEncoder()* function is used to encode the categorical type data to numerical type, where new columns of data are created for the categorical field value in the dataset which will be used in the training of the model.

In [5]:
```
labelencoder = LabelEncoder()

marketing_data['Education_Label'] = labelencoder.fit_transform(marketing_data["Education"])
marketing_data['Marital_Status_Label'] = labelencoder.fit_transform(marketing_data["Marital_Status"])
print("Label Encoding successful.")
```

```
Label Encoding successful.
```

In [ ]:
```
# checking for data values and field names after Label Encoding

marketing_data
```

Out[33]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	AcceptedCmp4
0	5524	1957	Graduation	Single	58138	0	0	2012-09-04	58	635	...	0
1	2174	1954	Graduation	Single	46344	1	1	2014-03-08	38	11	...	0
2	4141	1965	Graduation	Together	71613	0	0	2013-08-21	26	426	...	0
3	6182	1984	Graduation	Together	26646	1	0	2014-02-10	26	11	...	0
4	5324	1981	PhD	Married	58293	1	0	2014-01-19	94	173	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
2235	10870	1967	Graduation	Married	61223	0	1	2013-06-13	46	709	...	0
2236	4001	1946	PhD	Together	64014	2	1	2014-06-10	56	406	...	0
2237	7270	1981	Graduation	Divorced	56981	0	0	2014-01-25	91	908	...	1
2238	8235	1956	Master	Together	69245	0	1	2014-01-24	8	428	...	0
2239	9405	1954	PhD	Married	52869	1	1	2012-10-15	40	84	...	0

2240 rows × 31 columns

Table 9. Marketing Data after Label Encoding

In [ ]:
```
marketing_data.columns
```

Out[34]:
```
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',
       'Education_Label', 'Marital_Status_Label'],
      dtype='object')
```

2. Corelation Plot

A **correlation plot** or matrix is a *visual representation of the variables* present in the dataset which helps in understanding the *relationship* between the different variables and how highly the variables are corelated to each other.

The values of the correlation plot range from **-1 to 1**, where -1 indicates a **negative correlation** between the variables, 0 indicates **no correlation**, and 1 indicates a **positive correlation**.

The variables that have positive correlation are said to be highly correlated to each and hence either of the two variables must be removed for the model building as it may lead to **multicollinearity** where the efficiency of the model may reduce.

In [ ]:
```
# creating a new dataframe for correlation matrix

new_marketingdata = pd.DataFrame()
new_marketingdata = marketing_data.drop(columns=['ID', 'Z_Revenue', 'Z_CostContact'])
print("Dataframe created.")
```

```
Dataframe created.
```

In [ ]:
```
new_marketingdata.columns
```

Out[36]:
```
Index(['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', 'Response', 'Education_Label',
       'Marital_Status_Label'],
      dtype='object')
```

In [ ]:
```
# plotting correlation matrix

plt.figure(figsize = (15,8))
ax = plt.subplot()
sns.heatmap(new_marketingdata.corr(),annot=True, fmt='.1f', ax=ax, cmap="Blues")
ax.set_title('Correlation Plot');
```

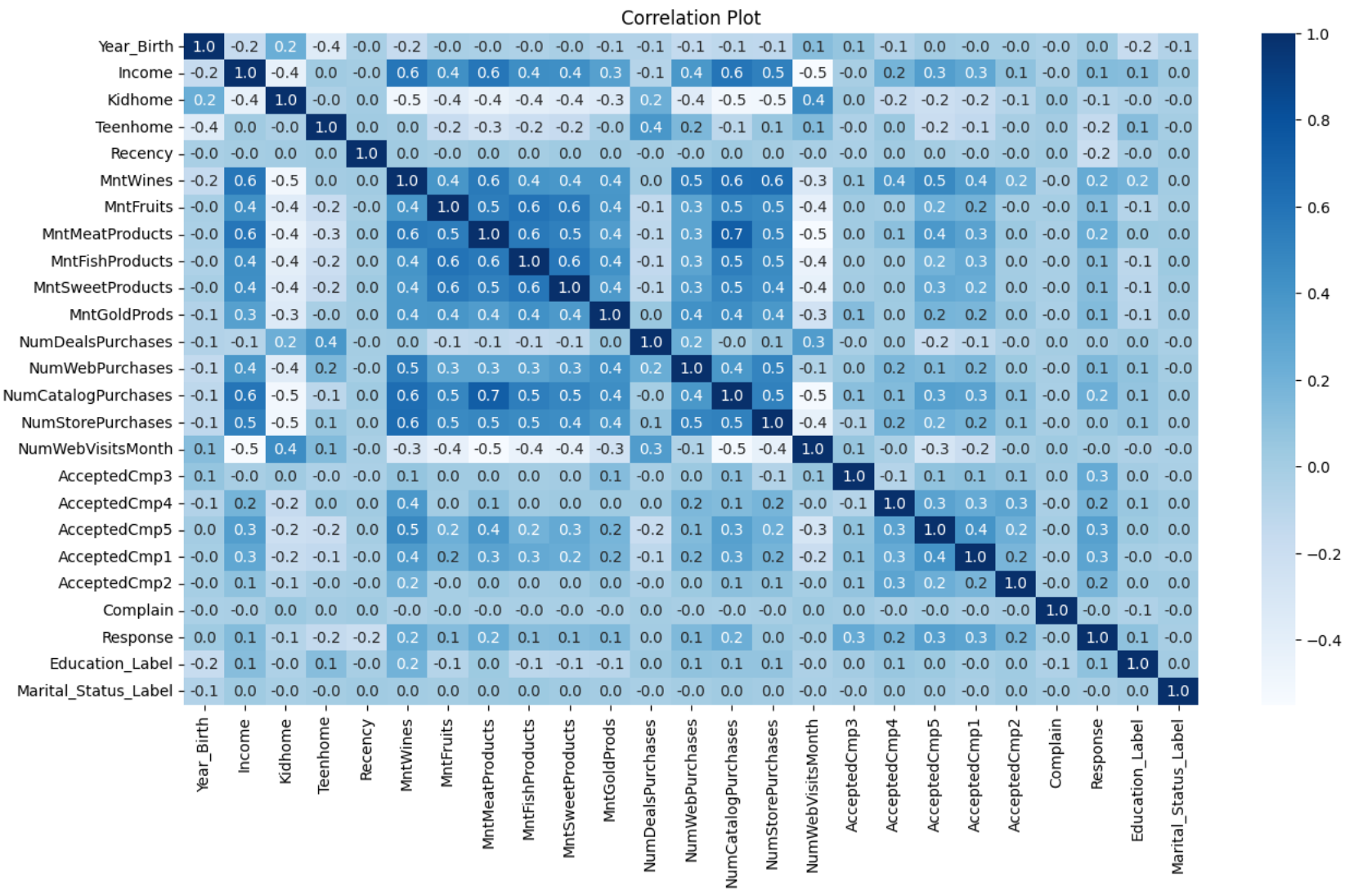

Figure 12. Correlation Matrix

Looking for strong corelations between independent & dependent variables

---

As observed in the correlation matrix above, we see that there are many variables or features that are *highly correlated* to each other and hence we need to analyze the features that are strongly correlated such that these features are excluded from the training of the model in order to avoid **multicollinearity** and *improve the efficiency of the model*. The following features are strongly correlated with the other features in the dataset and can be excluded from model building to avoid multicollinearity.

Correlation among the variables:

1. **Income** is highly correlated with majority of the independent variables like *MntWines, MntFruits, MntGoldProds, NumWebPurchases, etc.*
2. **MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProds** are all highly correlated to each other
3. **NumWebPurchases, NumCatalogPurchases, NumWebVisitsMonth** are highly correlated to each other
4. **MntWines** is correlated with variables *AcceptedCmp1, AcceptedCmp2, AcceptedCmp4, AcceptedCmp5*

Checking VIF score for Multicollinearity

VIF score i.e., **Variance Inflation Factor** is a *measure of multicollinearity* between the independent variables in the regression analysis. This calculates the variance of the variables which helps in understanding the coefficient value and how much the variable is inflated due to collinearity in the model. The VIF score from **range 0 to 5** can be accepted to be considered for the training of the model, while values above 5 are considered to have high multicollinearity which would affect the accuracy and performance of the model, hence should be excluded.

In [ ]:
```
# checking VIF score for field values

numerical_marketing_data = marketing_data.select_dtypes(include=[np.number])
vif_score1 = pd.DataFrame()
vif_score1["Feature"] = numerical_marketing_data.columns
vif_score1["VIF Score"] = [variance_inflation_factor(numerical_marketing_data.values, i) for i in range(numer
print(vif_score1)
```

	Feature	VIF Score
0	ID	1.010835
1	Year_Birth	1.259204
2	Income	2.190397
3	Kidhome	1.884153
4	Teenhome	1.618260
5	Recency	1.063283
6	MntWines	3.494357
7	MntFruits	1.954082
8	MntMeatProducts	2.949711
9	MntFishProducts	2.144942
10	MntSweetProducts	1.934484
11	MntGoldProds	1.540669
12	NumDealsPurchases	1.683694
13	NumWebPurchases	1.947963
14	NumCatalogPurchases	3.063649
15	NumStorePurchases	2.424874
16	NumWebVisitsMonth	2.347542
17	AcceptedCmp3	1.159159
18	AcceptedCmp4	1.377840
19	AcceptedCmp5	1.663838
20	AcceptedCmp1	1.360058
21	AcceptedCmp2	1.159884
22	Complain	1.011329
23	Z_CostContact	0.000000
24	Z_Revenue	0.000000
25	Response	1.410645
26	Education_Label	1.187969
27	Marital_Status_Label	1.010188

Table 10. VIF Score Table

From the above VIF score table it is observed that the VIF score **is in the range from 0 to 5**, thus there is **no issue of multicollinearity** and that the variables are not correlated to each. The features of the dataset thus can be used for training the model avoiding the issue of multicollinearity where the performance of the model can be improved.

a. Understanding the top features selected by Correlation Matrix

---

In [ ]:
```
corr_result = marketing_data.corr()
correlation_response = corr_result['Response'].sort_values(ascending=False)
topfeatures = correlation_response[1:6]
print("The top features selected by correlation matrix are:\n")
print(topfeatures)
```

```
The top features selected by correlation matrix are:

AcceptedCmp5       0.326634
AcceptedCmp1       0.293982
AcceptedCmp3       0.254258
MntWines           0.247254
MntMeatProducts    0.236335
Name: Response, dtype: float64
```

b. Lasso Regression to select the most important features for model training

---

In [ ]:
```
# creating a new dataframe excluding categorical variable

new_marketingdata_lasso = pd.DataFrame()
new_marketingdata_lasso = marketing_data.drop(columns=['Education', 'Marital_Status','Dt_Customer'])
print("Dataframe created.")
```

```
Dataframe created.
```

In [ ]:
```
A = new_marketingdata_lasso.drop(['Response'], axis=1)
B = new_marketingdata_lasso['Response']
lasso_result = Lasso(alpha=0.1)
lasso_result.fit(A, B)
coef = pd.Series(lasso_result.coef_, index=A.columns)
features_lasso = coef.abs().sort_values(ascending=False).head(5).index
print("The top features selected by Lasso regression:\n")
print(features_lasso)
```

```
The top features selected by Lasso regression:

Index(['NumStorePurchases', 'Recency', 'Year_Birth', 'MntGoldProds',
       'MntMeatProducts'],
      dtype='object')
```

c. Features selected for model building

The features that are selected for the model building based on the Correlation Plot values, Lasso Regression, and VIF Score are as follows:

Correlation Plot Values	Lasso Regression	Features with VIF Scores in range 0 to 5
AcceptedCmp5	NumStorePurchases	ID
AcceptedCmp1	Recency	Year_Birth
AcceptedCmp3	Year_Birth	Income
MntWines	MntGoldProds	Kidhome
MntMeatProducts	MntMeatProducts	Teenhome
		Recency
		MntWines
		MntFruits
		MntMeatProducts
		MntFishProducts
		MntSweetProducts
		MntGoldProds
		NumDealsPurchases
		NumWebPurchases
		NumCatalogPurchases
		NumStorePurchases
		NumWebVisitsMonth
		AcceptedCmp3
		AcceptedCmp4
		AcceptedCmp5
		AcceptedCmp1

Correlation Plot Values	Lasso Regression	Features with VIF Scores in range 0 to 5
		AcceptedCmp2
		Complain
		Z_CostContact
		Z_Revenue
		Education

Table 11. Feature Selection & Extraction Table

3. Defining the features for model training (Dimensionality Reduction)

The model is trained & built on the below mentioned features that are selected from the analysis of the Feature selection and extraction, Correlation matrix, Lasso Regression, and VIF score for multicollinearity.

1. **ID, Education, Marital_Status, Dt_Customer, Year_Birth** are considered irrelevant for the training and prediction of the model for the response variable, hence is excluded from the model building.
2. **Z_CostContact and Z_Revenue** are constant values in the dataset, thus are excluded from the training of the model.
3. Since **VIF Score of the variables are in the range of 0 to 5** it indicates that the variables are not correlated to each and hence there is no issue of multicollinearity.

In [6]:
```
X = marketing_data.drop(columns=['ID', 'Education', 'Marital_Status', 'Dt_Customer', 'Year_Birth', 'Z_CostCon
y = marketing_data['Response']
```

4. Splitting the dataset into train & test set

The dataset is split into training and testing data with a random split of **80%** train set and **20%** for test data.

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42)
```

5. Handling imbalance data (Class Bias)

Class Bias in a dataset occurs when **distribution of data points in a dataset is uneven**, with one or more classes being overrepresented or underrepresented, which means that either of the category has majority of the data points and thus while training of the model, the prediction will be biased to that category. Hence, it is important to **handle class imbalance** in the dataset which can be performed using various methods, and one such method that is implemented below is the **class weights**.

In [8]:
```
class_weights = compute_class_weight(
                                        class_weight = "balanced",
                                        classes = np.unique(y_train),
                                        y = y_train
                                    )
class_weights = dict(zip(np.unique(y_train), class_weights))
print(class_weights)
```

```
{0: 0.5867714472822528, 1: 3.3811320754716983}
```

6. Standardization

Standardization is performed on the split dataset in order to make the features selected comparable to a standardized scale.

In [9]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Standardization successful")
```

```
Standardization successful
```

Label Count in train & test dataset

In [10]:
```
print('Labels count in y:', np.bincount(y))
print('Labels count in y_train:', np.bincount(y_train))
print('Labels count in y_test:', np.bincount(y_test))
```

```
Labels count in y: [1906  334]
Labels count in y_train: [1527  265]
Labels count in y_test: [379  69]
```

Step 5: Model Building

Task 2: Build a logistic model to accurately predict subscription behavior. Discuss which variables are significant, their business impact, and how that may help you learn about the business.

Building a Logistic Regression model to predict subscription behavior.

Fitting the Logistic Regression model

The LR model is fit with a balanced class weight method in order to avoid class imbalance or bias data present in the training dataset.

In [11]:
```
logisticreg_model = LogisticRegression(solver='newton-cg', class_weight='balanced')
logisticreg_model.fit(X_train_scaled, y_train)
```

Out[11]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', solver='newton-cg')
```

Displaying the coefficients & intercepts after fitting the model

As observed from the below code, the coefficient values of the variables are displayed which are either positive or negative, which indicates that the variables with **positive** value have a **positive relationship with the target variable** whereas values having a **negative sign** indicate that there is a **negative relationship between the independent variable and the target variable.**

In [ ]:
```
coefficient_values = pd.DataFrame({'Feature': X.columns, 'Coefficient': logisticreg_model.coef_[0]})
print('Coefficients:')
print(coefficient_values)
```

```
Coefficients:
                 Feature  Coefficient
0                 Income    -0.118962
1                Kidhome    -0.073578
2               Teenhome    -0.556644
3                Recency    -0.838319
4              MntWines    -0.205099
5             MntFruits     0.001428
6        MntMeatProducts     0.617741
7        MntFishProducts    -0.008853
8       MntSweetProducts     0.235296
9           MntGoldProds     0.200380
10    NumDealsPurchases     0.368254
11      NumWebPurchases     0.272962
12  NumCatalogPurchases     0.421981
13    NumStorePurchases    -0.435297
14    NumWebVisitsMonth     0.753439
15         AcceptedCmp3     0.499251
16         AcceptedCmp4     0.322702
17         AcceptedCmp5     0.554986
18         AcceptedCmp1     0.280563
19         AcceptedCmp2     0.125874
20             Complain     0.016911
21       Education_Label     0.595678
22  Marital_Status_Label    -0.081501
```

Table 12. Coefficients Table

Summary Report of the Logistic Regression model

Summary report of the Logistic Regression model provides an overview of the model build and how accurately the model fits the data for each independent variable to predict or classify the target variable or dependent variable. The report is used to evaluate the overall fit of the model, identify which independent variables are most important in predicting the dependent variable, and analyze the statistical significance of each coefficients.

- From the summary report for the Logistic Regression model below, it is observed that the p-value for some of the features of the dataset is **greater that the significance value of 0.05**, hence these variables or features are considered **statistically insignificant**. The variables that are statistically insignificant are **Kidhome, MntFruits, MntFishProducts, MntSweetProducts, MntGoldProds, NumWebVisitsMonth, and Complain.** Thus, these variables are not contributing in the classification of the response variable.
- The remaining features have **p-value less than the significance value of 0.05**, hence are consider as **statistically significant variables**, indicating that these features are contributing in the prediction of the target variable.
- Of the statistically significant variables, features having highest coefficient values are **AcceptedCmp3 and AcceptedCmp5**, indicating that they have the **highest positive influence on the target variable**, whereas **Income** has the **highest negative impact on the target variable**, which means that if the income is higher, there is no response for the magazine subscription.

- **Hence, the variables that are significant which have an impact on the business are AcceptedCmp3 and AcceptedCmp5, and the variable that has the highest negative impact is the Income feature.**

In [ ]:
```
model_summary = sm.Logit(endog=y, exog=X)
summary_result = model_summary.fit()
print(summary_result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.294728
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:               Response   No. Observations:                 2240
Model:                          Logit   Df Residuals:                     2217
Method:                           MLE   Df Model:                           22
Date:                Sat, 06 May 2023   Pseudo R-squ.:                  0.3002
Time:                        22:11:10   Log-Likelihood:                -660.19
converged:                       True   LL-Null:                       -943.39
Covariance Type:            nonrobust   LLR p-value:                9.636e-106
==============================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Income               -2.874e-05   6.42e-06     -4.480      0.000   -4.13e-05   -1.62e-05
Kidhome                 -0.2396      0.189     -1.270      0.204      -0.609       0.130
Teenhome                -0.8331      0.184     -4.521      0.000      -1.194      -0.472
Recency                 -0.0301      0.003    -11.011      0.000      -0.036      -0.025
MntWines                 0.0007      0.000      2.069      0.039    3.64e-05       0.001
MntFruits                0.0020      0.002      0.864      0.387      -0.003       0.006
MntMeatProducts          0.0022      0.000      4.613      0.000       0.001       0.003
MntFishProducts         -0.0002      0.002     -0.121      0.904      -0.004       0.003
MntSweetProducts         0.0008      0.002      0.380      0.704      -0.003       0.005
MntGoldProds             0.0018      0.002      1.178      0.239      -0.001       0.005
NumDealsPurchases        0.1455      0.044      3.295      0.001       0.059       0.232
NumWebPurchases          0.1045      0.031      3.350      0.001       0.043       0.166
NumCatalogPurchases      0.0673      0.039      1.731      0.083      -0.009       0.143
NumStorePurchases       -0.1697      0.033     -5.203      0.000      -0.234      -0.106
NumWebVisitsMonth        0.0313      0.033      0.942      0.346      -0.034       0.097
AcceptedCmp3             1.6615      0.212      7.827      0.000       1.245       2.078
AcceptedCmp4             1.0475      0.269      3.887      0.000       0.519       1.576
AcceptedCmp5             1.5294      0.271      5.637      0.000       0.998       2.061
AcceptedCmp1             1.1352      0.262      4.336      0.000       0.622       1.648
AcceptedCmp2             1.4602      0.524      2.787      0.005       0.433       2.487
Complain                 0.4325      0.751      0.576      0.565      -1.040       1.905
Education_Label          0.2601      0.068      3.824      0.000       0.127       0.393
Marital_Status_Label    -0.2130      0.060     -3.526      0.000      -0.331      -0.095
==============================================================================
```

Ranking the top three variables by the highest coefficient (by absolute value).

In [ ]:
```
coef_sort = abs(summary_result.params).sort_values(ascending=False).head(3)
table2 = pd.DataFrame({'Coefficient (abs)': coef_sort})
table2 = table2.loc[coef_sort.index]
print(table2)
```

```
              Coefficient (abs)
AcceptedCmp3           1.661463
AcceptedCmp5           1.529384
AcceptedCmp2           1.460207
```

Model Testing

In [ ]:
```
y_pred = logisticreg_model.predict(X_test_scaled)
```

Evaluating the performance of the model

1. Accuracy of the model on training and testing dataset
2. Confusion Matrix
3. Classification Report
4. AUC-ROC curve

In [ ]:
```
# Accuracy of the model on training and testing set

print('Accuracy of Logistic Regressor model on training set: {:.3f}'.format(logisticreg_model.score(X_train_s
print('Accuracy of Logistic Regressor model on test set:     {:.3f}'.format(logisticreg_model.score(X_test_sc

model_result1 = logisticreg_model.score(X_test_scaled, y_test)
model_result1 = round(model_result1,4)
print("Overall Accuracy of the model is ", model_result1)
```

```
Accuracy of Logistic Regressor model on training set: 0.833
Accuracy of Logistic Regressor model on test set:     0.783
Overall Accuracy of the model is  0.7835
```

In [ ]:
```
# Confusion Matrix

confusionmatrix_LR = confusion_matrix(y_test, y_pred)

fig = sns.heatmap(confusionmatrix_LR, annot=True,  annot_kws={"size": 15}, cmap = 'Blues', fmt='g')
fig.xaxis.set_ticklabels(['No Response','Response'])
fig.yaxis.set_ticklabels(['No Response','Response'])
fig.set_xlabel('Predicted Values')
fig.set_ylabel('Actual Values ')
fig.set_title('Confusion Matrix for the Logistic Regression Model')
sns.set(font_scale=1.0)
```

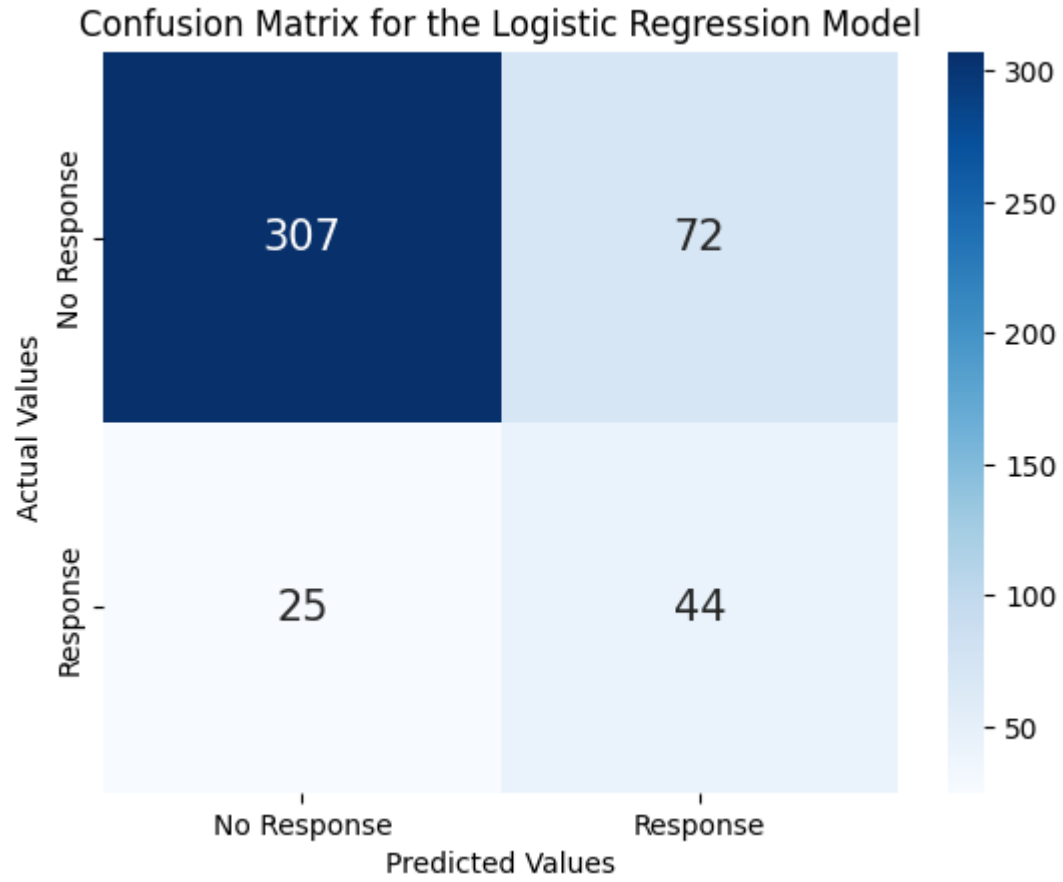

Figure 13. Confusion Matrix for Logistic Regression Model

In [ ]:
```
# Classification Report

print("\n Classification report %s:\n%s\n" % (logisticreg_model, metrics.classification_report(y_test, y_pred
```

```
 Classification report LogisticRegression(class_weight='balanced', solver='newton-cg'):
              precision    recall  f1-score   support

           0       0.92      0.81      0.86       379
           1       0.38      0.64      0.48        69

    accuracy                           0.78       448
   macro avg       0.65      0.72      0.67       448
weighted avg       0.84      0.78      0.80       448
```

In [ ]:
```
# AUC-ROC Curve

ypred_prob = logisticreg_model.predict_proba(X_test_scaled)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, ypred_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8,5))
plt.title('ROC Curve')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.xlabel('False Positive Rate')
plt.show()
```

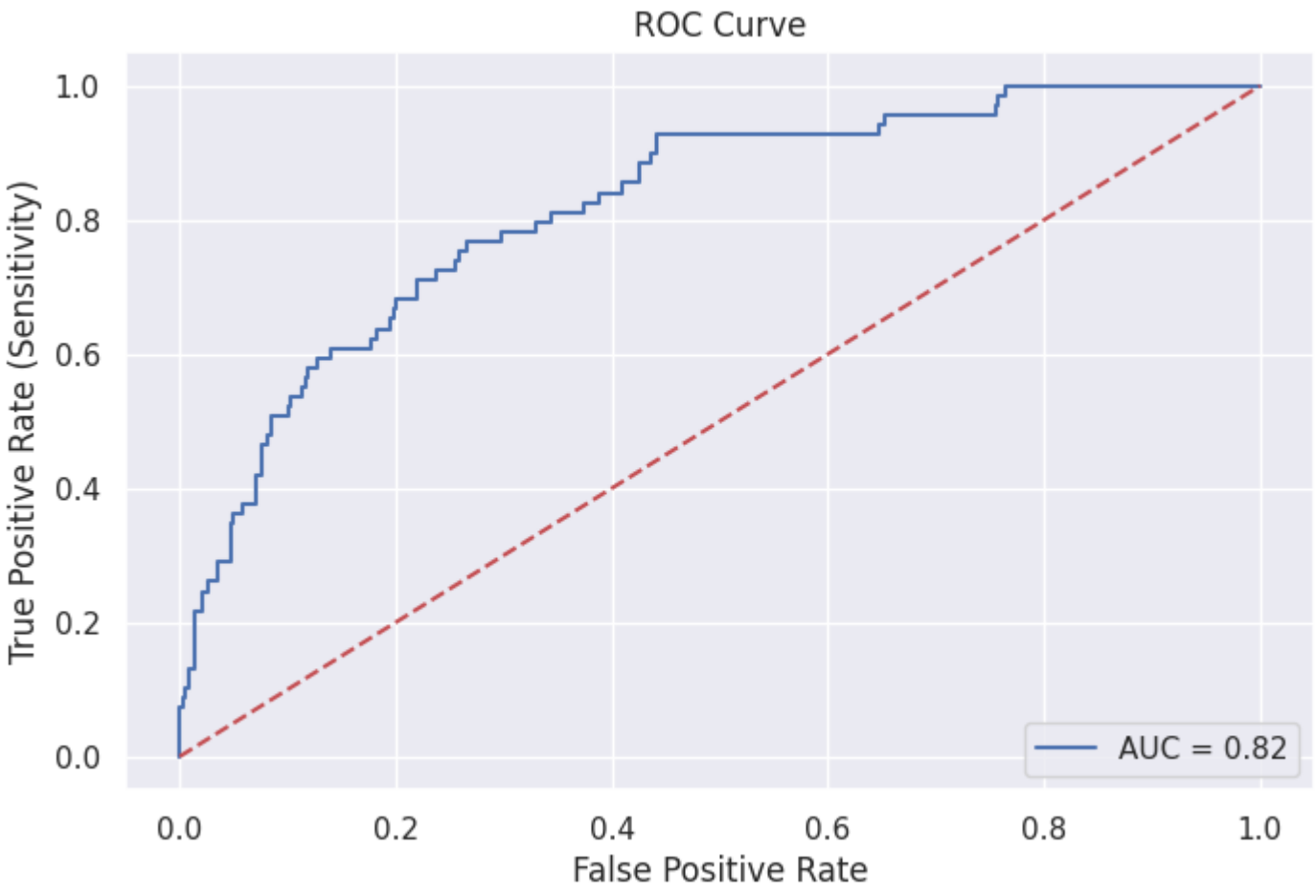

Figure 14. ROC Curve

Accuracy metric:

From the above evaluation metrics it is observed that the Logistic Regression model performed well in classifying the magazine behavior with an accuracy of **83.3% for training data and 78.3% for testing data,** which indicates that the model has good accuracy in classification. However, the accuracy difference between the training and testing data is more which may occur in overfitting of the data, indicating that the model is able to make predictions on the training data but cannot accurately predict on the new or unseen data.

Confusion matrix:

The confusion matrix of the Logistic Regression model indicates that the **no response category is correctly classified 307 times** whereas the **response class is correctly classified 44 times**, which is a good percent where the data is been correctly classified. However, the **no response category is wrongly classified times as response class 72 times** and **response class is classified as no response category 25 times**. Hence, the false positive and false negative values need to addressed to avoid inefficiency of the category classification.

Classification Report:

From the classification report we understand the model performance in terms of training and testing data based on the precision and recall values. The precision score for **no response and response is 92% and 38% respectively**, and as it can be observed the precision score is less for the response category. Similarly, the recall score for the **no response class is 81%** and for **response class is 64%** indicating a good score for the two classes.

ROC Curve:

The AUC score for the Logistic Regression model built to classify the magazine behavior is **82%** which is a good AUC score indicating that the model is a best-fit model.

Step 5: Model Building

Task 3: Build an SVM model to accurately predict subscription behavior. Discuss the model's accuracy and how that compares to the logistic model.

Building a SVM model to predict subscription behavior.

Fitting the SVM model

The SVM model is fit with a linear kernel and balanced class weight method in order to avoid class imbalance or bias data present in the training dataset.

In [12]:
```
svm_model = SVC(kernel='linear', C=1, random_state=42, class_weight='balanced')
svm_model.fit(X_train_scaled, y_train)
```

Out[12]:
```
SVC(C=1, class_weight='balanced', kernel='linear', random_state=42)
```

Model Testing

In [ ]:
```
y_pred = svm_model.predict(X_test_scaled)
```

Evaluating the performance of the model

1. Accuracy of the model on training and testing dataset
2. Confusion Matrix
3. Classification Report
4. Feature Importance

In [ ]:
```
# Accuracy of the model on training and testing set

print('Accuracy of SVM model on training set: {:.3f}'.format(svm_model.score(X_train_scaled, y_train)))
print('Accuracy of SVM model on test set:     {:.3f}'.format(svm_model.score(X_test_scaled, y_test)))

model_result2 = svm_model.score(X_test_scaled, y_test)
model_result2 = round(model_result2,4)
print("Overall Accuracy of the model is ", model_result2)
```

```
Accuracy of SVM model on training set: 0.832
Accuracy of SVM model on test set:     0.799
Overall Accuracy of the model is  0.7991
```

In [ ]:
```
# Confusion Matrix

confusionmatrix_LR = confusion_matrix(y_test, y_pred)

fig = sns.heatmap(confusionmatrix_LR, annot=True,  annot_kws={"size": 15}, cmap = 'Blues', fmt='g')
fig.xaxis.set_ticklabels(['No Response','Response'])
fig.yaxis.set_ticklabels(['No Response','Response'])
fig.set_xlabel('Predicted Values')
fig.set_ylabel('Actual Values ')
fig.set_title('Confusion Matrix for the SVM Model')
sns.set(font_scale=1.0)
```

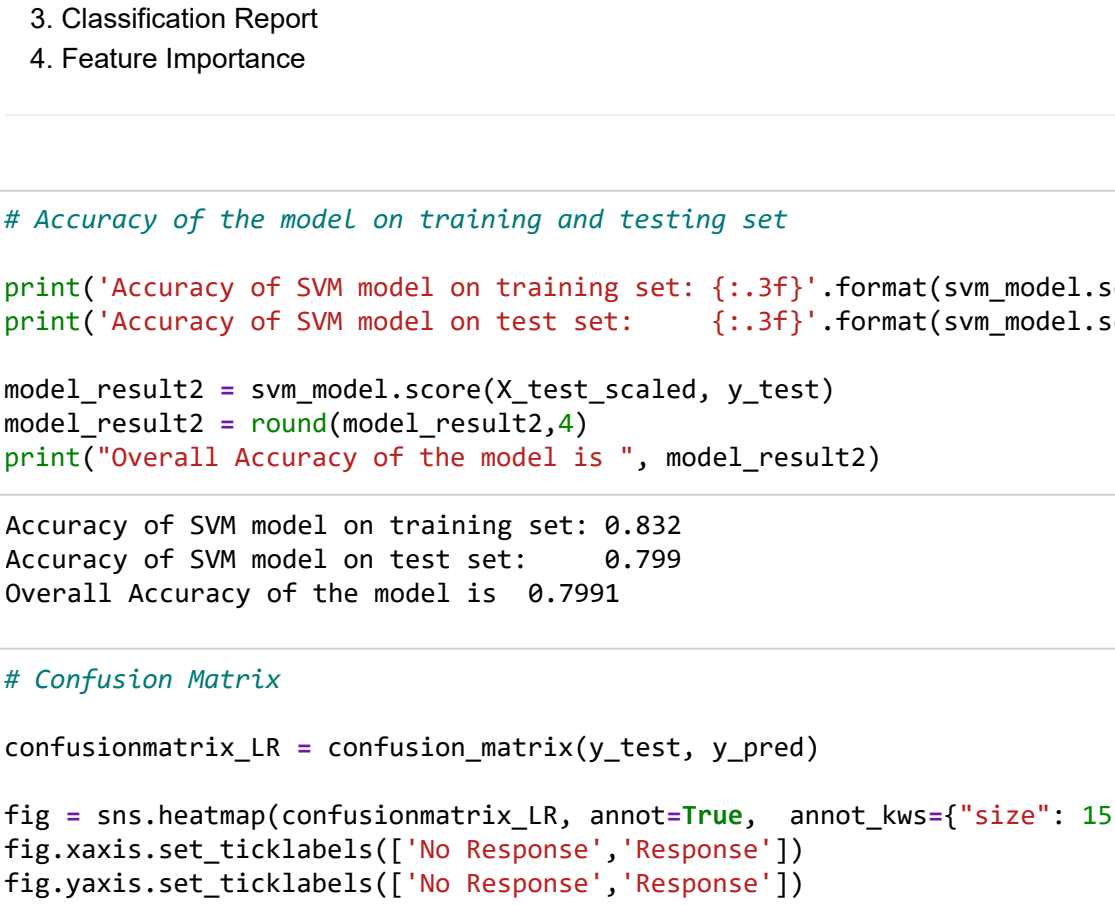

Figure 15. Confusion Matrix for SVM Model

In [ ]:
```
# Classification Report

print("\n Classification report %s:\n%s\n" % (svm_model, metrics.classification_report(y_test, y_pred)))
```

```
 Classification report SVC(C=1, class_weight='balanced', kernel='linear', random_state=42):
              precision    recall  f1-score   support

           0       0.93      0.83      0.87       379
           1       0.40      0.64      0.49        69

    accuracy                           0.80       448
   macro avg       0.66      0.73      0.68       448
weighted avg       0.85      0.80      0.82       448
```

Feature Importance Visualization for SVM Model

In [ ]:
```
feature_importances = pd.Series(abs(svm_model.coef_[0]), index=X.columns)
feature_importances.nlargest(10).plot(kind='barh', color="darkblue", title="Feature Importance for SVM model"
```

Out[66]: <Axes: title={'center': 'Feature Importance for SVM model'}>

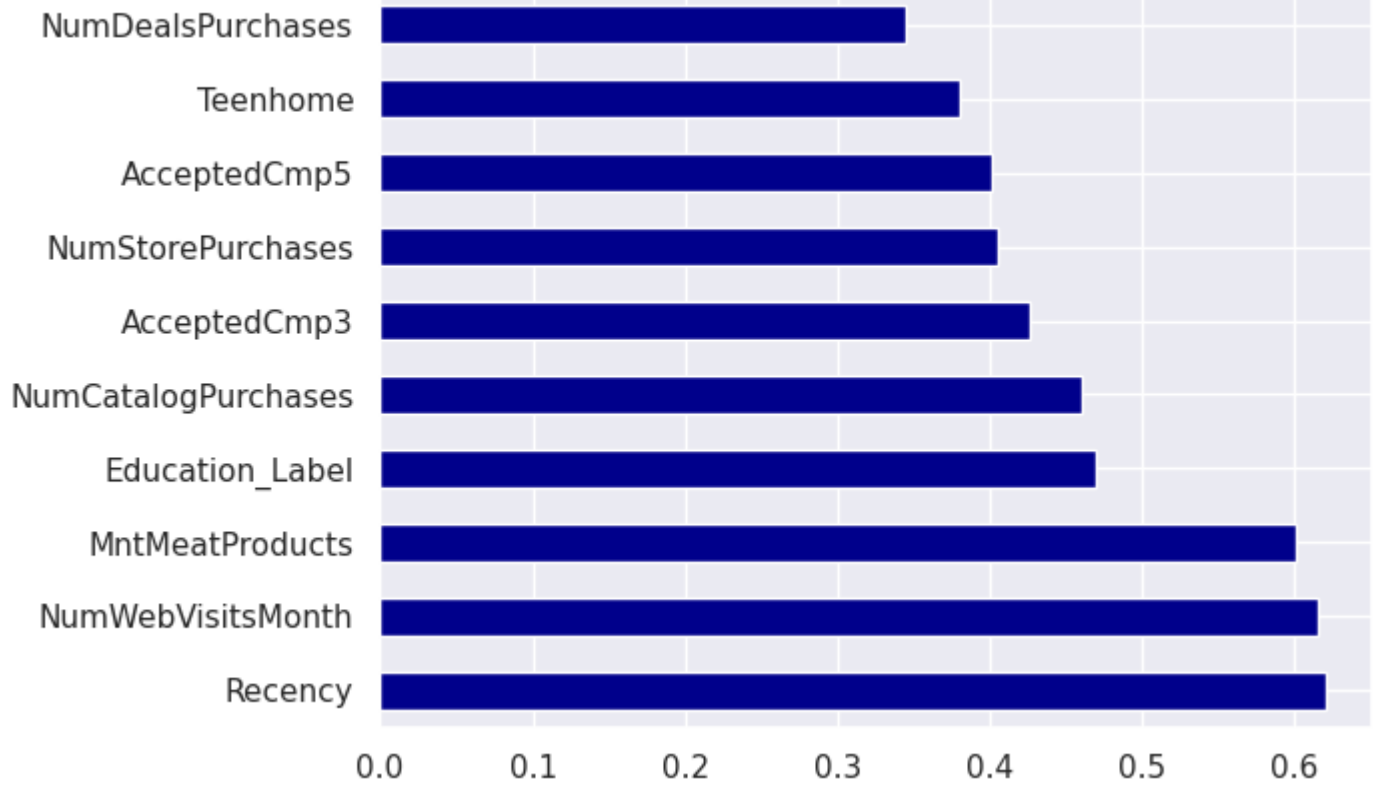

Figure 16. Feature Importance for SVM Model

Accuracy metric:

From the above evaluation metrics it is observed that the SVM model performed well in classifying the magazine behavior with an accuracy of **83.2% for training data and 79.9% for testing data,** which indicates that the model has good accuracy in classification. However, the accuracy difference between the training and testing data is slighlty more in this case which may occur in overfitting of the data, indicating that the model is able to make predictions on the training data but cannot accurately predict on the new or unseen data.

Confusion matrix:

The confusion matrix of the SVM model indicates that the **no response category is correctly classified 314 times** whereas the **response class is correctly classified 44 times**, which is a good percent where the data is been correctly classified. However, the **no response category is wrongly classified times as response class 65 times** and **response class is classified as no response category 25 times**. Hence, the false positive and false negative values need to addressed to avoid inefficiency of the category classification.

Classification Report:

From the classification report we understand the model performance in terms of training and testing data based on the precision and recall values. The precision score for **no response and response is 93% and 40% respectively**, and as it can be observed the precision score is less for the response category. Similarly, the recall score for the **no response class is 83%** and for **response class is 64%** indicating a good score for the two classes.

Feature Importance Score:

The feature importance graph shows that variables **Recency, NumWebVisitsMonth, and MntMeatProducts** have the highest feature importance with **score of 0.6** indicating that the model classification for magazine behavior is based on these features and that they are highly contributing in the prediction of the target variable. Thus, **it is recommended that the company should focus on these**

parameters and features while analyzing the magazine subscription behavior of the customer.

SVM model comparison with Logistic Regression model

- As compared to the goodness of fit metrics and evaluation metrics for the Logistic Regression and SVM model, it is observed that **SVM model performed better in classifying the magazine behavior of the customers**.
- This is because the **accuracy of the testing data is slightly high** as compared to that of the Logistic Regression model.
- Apart from that, when analyzed the **confusion matrix we see that high number of classes are correctly classified** using the SVM model as compared to the LR model, which is important because if the less number of false positive and false negative values are there, it will be less of a job to manually address that and hence would be effective in analyzing the data.
- Also, the **precision and recall score is slightly better for the SVM model** in comparison with the Logistic Regression model, indicating that the prediction of classes is better performed in SVM model.

Step 5: Model Building

Task 4: Build a decision tree model (at most 4 branches) to accurately predict subscription behavior. Discuss the model's accuracy and how that compares to the other two models (variables and accuracy).

Building a Decision Tree model to predict subscription behavior.

Fitting the Decision Tree model

The Decision Tree model is fit with a max depth of 4 and balanced class weight method in order to avoid class imbalance or bias data present in the training dataset.

In [13]:
```
decisiontree_model = DecisionTreeClassifier(max_depth=4, random_state=42, class_weight="balanced")
decisiontree_model.fit(X_train_scaled, y_train)
```

Out[13]:
```
DecisionTreeClassifier

DecisionTreeClassifier(class_weight='balanced', max_depth=4, random_state=42)
```

Plotting the Decision Tree

In [ ]:
```
# plotting the decision tree

plt.figure(figsize=(18,8))
plot_tree(decisiontree_model, filled=True, rounded=True, feature_names=X_train.columns, class_names=["No Resp
plt.show()
```

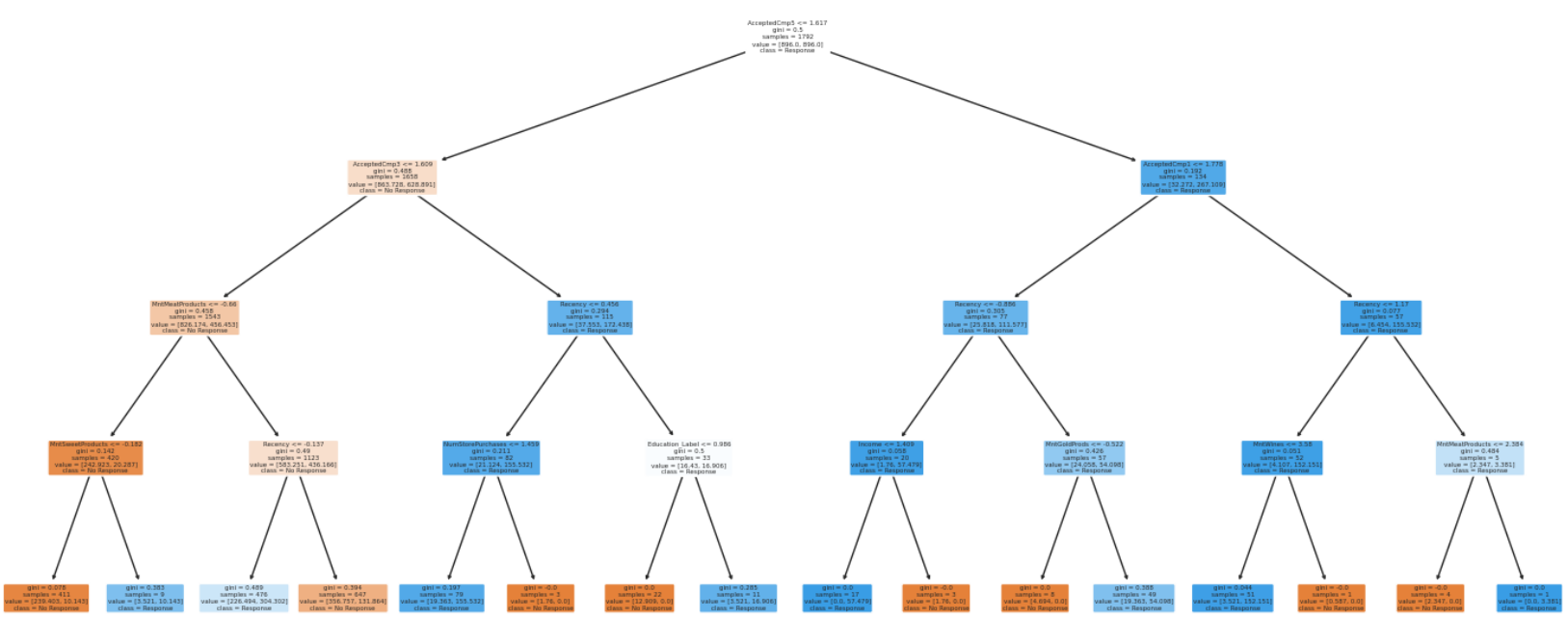

Figure 17. Decision Tree

Model Testing

In [ ]:
```
y_pred = decisiontree_model.predict(X_test_scaled)
```

Evaluating the performance of the model

1. Accuracy of the model on training and testing dataset
2. Confusion Matrix
3. Classification Report
4. Feature Importance

In [ ]:
```
# Accuracy of the model on training and testing set

print('Accuracy of Decision Tree model on training set: {:.3f}'.format(decisiontree_model.score(X_train_scale
print('Accuracy of Decision Tree model on test set:     {:.3f}'.format(decisiontree_model.score(X_test_scaled

model_result3 = decisiontree_model.score(X_test_scaled, y_test)
model_result3 = round(model_result3,4)
print("Overall Accuracy of the model is ", model_result3)
```

```
Accuracy of Decision Tree model on training set: 0.714
Accuracy of Decision Tree model on test set:     0.672
Overall Accuracy of the model is  0.6719
```

In [ ]:
```
# Confusion Matrix

confusionmatrix_LR = confusion_matrix(y_test, y_pred)

fig = sns.heatmap(confusionmatrix_LR, annot=True,  annot_kws={"size": 15}, cmap = 'Blues', fmt='g')
fig.xaxis.set_ticklabels(['No Response','Response'])
fig.yaxis.set_ticklabels(['No Response','Response'])
fig.set_xlabel('Predicted Values')
fig.set_ylabel('Actual Values ')
fig.set_title('Confusion Matrix for the Decision Tree Model')
sns.set(font_scale=1.0)
```

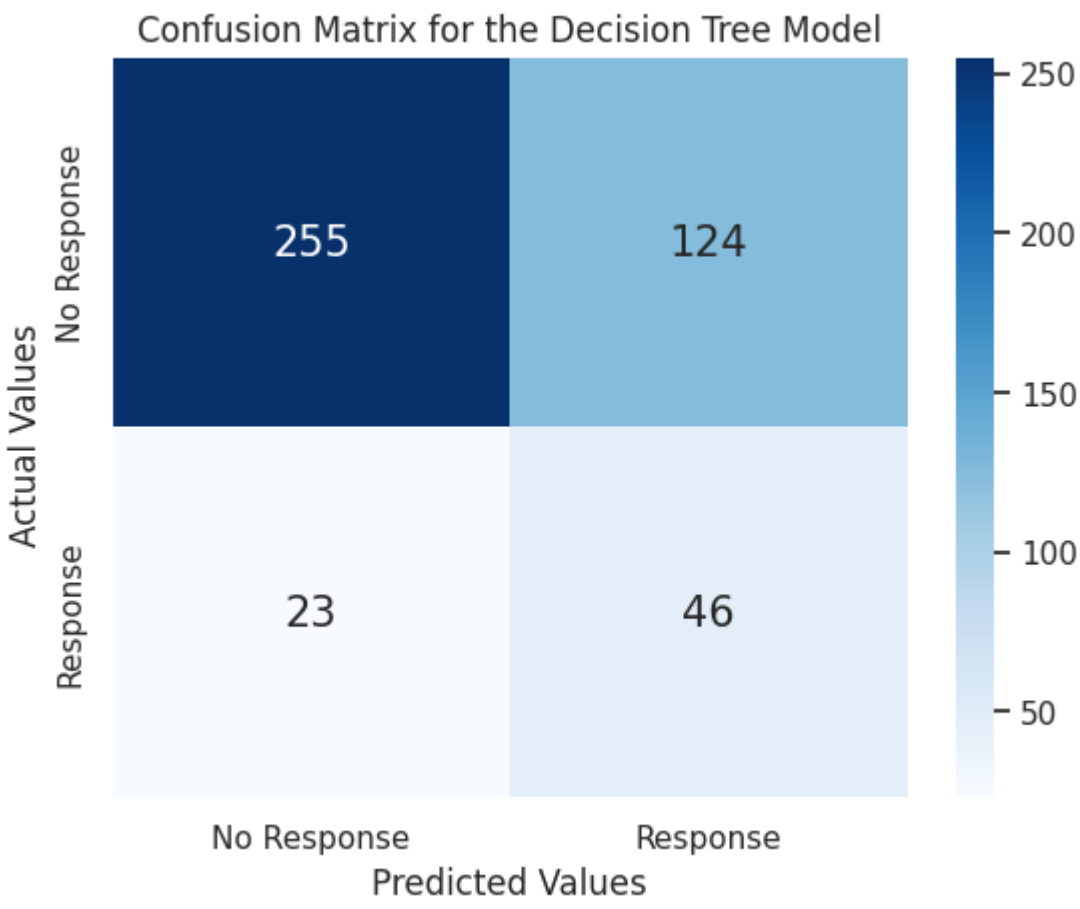

Figure 18. Confusion Matrix for Decision Tree model

In [ ]:
```
# Classification Report

print("\n Classification report %s:\n%s\n" % (decisiontree_model, metrics.classification_report(y_test, y_pre
```

```
 Classification report DecisionTreeClassifier(class_weight='balanced', max_depth=4, random_state=42):
              precision    recall  f1-score   support

           0       0.92      0.67      0.78       379
           1       0.27      0.67      0.38        69

    accuracy                           0.67       448
   macro avg       0.59      0.67      0.58       448
weighted avg       0.82      0.67      0.72       448
```

Feature Importance for Decision Tree model

In [ ]:
```
feature_importances = pd.Series(decisiontree_model.feature_importances_, index=X.columns)
feature_importances.nlargest(10).plot(kind='barh', color="darkblue", title = "Feature Importance for Decision
```

Out[73]: <Axes: title={'center': 'Feature Importance for Decision Tree model'}>

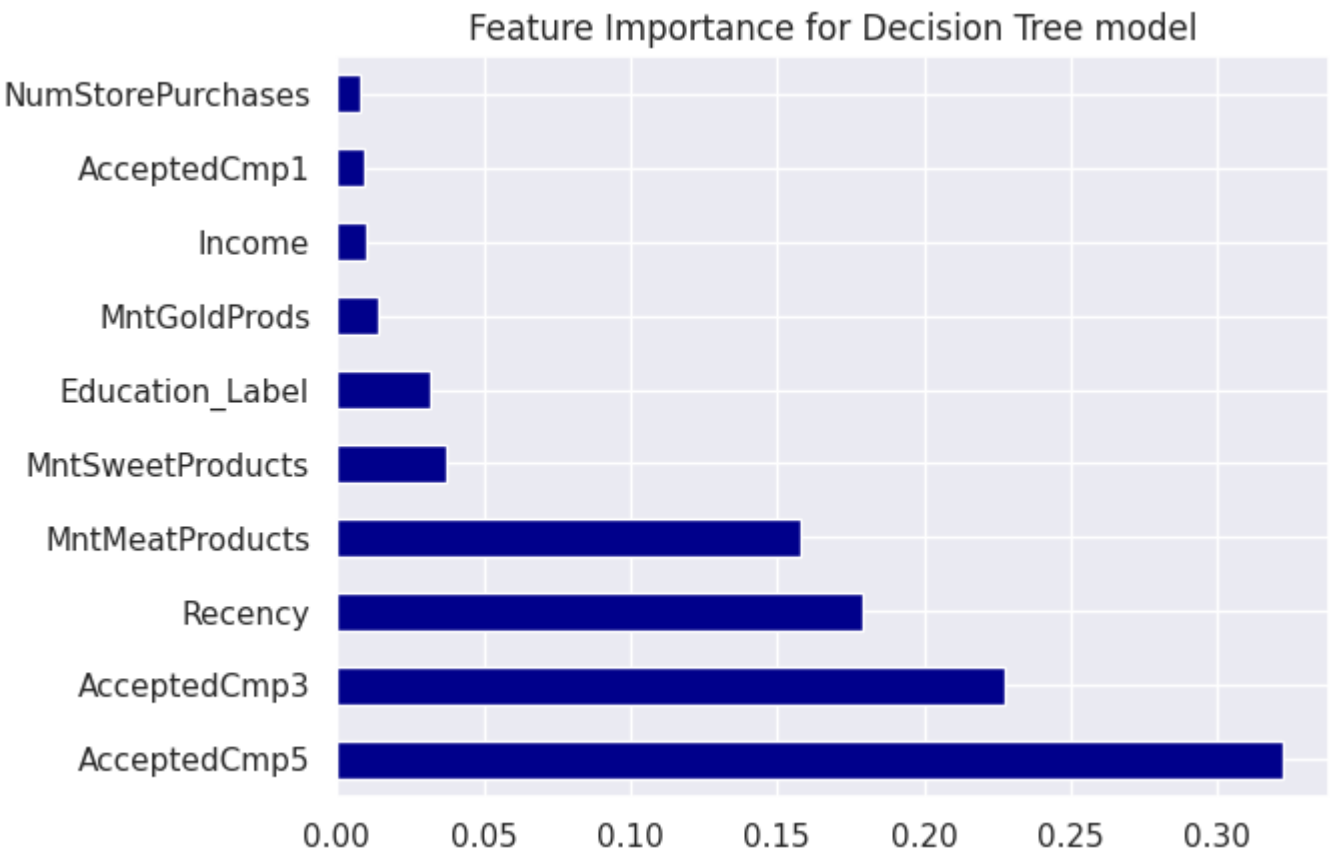

Figure 19. Feature Importance for Decision Tree model

Feature Importance Score

In [ ]:
```
# extracting feature importance

feature_importance = pd.DataFrame({'Feature': X_train.columns, 'Importance Score' : np.round(decisiontree_mod
feature_importance.sort_values('Importance Score', ascending=False, inplace = True)
print(feature_importance)
```

```
                Feature  Importance Score
17          AcceptedCmp5             0.322
15          AcceptedCmp3             0.228
3                Recency             0.179
6          MntMeatProducts             0.158
8         MntSweetProducts             0.037
21         Education_Label             0.032
9             MntGoldProds             0.014
0                   Income             0.010
18          AcceptedCmp1             0.009
13     NumStorePurchases             0.008
4                MntWines             0.003
20               Complain             0.000
19          AcceptedCmp2             0.000
16          AcceptedCmp4             0.000
11       NumWebPurchases             0.000
14     NumWebVisitsMonth             0.000
12   NumCatalogPurchases             0.000
1                 Kidhome             0.000
10     NumDealsPurchases             0.000
7         MntFishProducts             0.000
5               MntFruits             0.000
2                Teenhome             0.000
22  Marital_Status_Label             0.000
```

Table 13. Feature Importance for Decision Tree model

Accuracy metric:

From the above evaluation metrics it is observed that the Decision Tree model performed well in classifying the magazine behavior with an accuracy of **71.4% for training data and 67.2% for testing data,** which indicates that the model has somewhat a good accuracy in classification. However, the accuracy difference between the training and testing data is slighlty more in this case which may occur in overfitting of the data, indicating that the model is able to make predictions on the training data but cannot accurately predict on the new or unseen data.

Confusion matrix:

The confusion matrix of the SVM model indicates that the **no response category is correctly classified 255 times** whereas the **response class is correctly classified 46 times**, which is a good percent where the data is been correctly classified. However, the **no response category is wrongly classified times as response class 124 times** and **response class is classified as no response category 23 times**. Hence, the false positive and false negative values need to addressed to avoid inefficiency of the category classification.

Classification Report:

From the classification report we understand the model performance in terms of training and testing data based on the precision and recall values. The precision score for **no response and response is 92% and 27% respectively**, and as it can be observed the precision score is less for the response category. Similarly, the recall score for the **no response class is 87%** and for **response class is 67%** indicating a good score for the two classes.

Feature Importance Score:

The feature importance graph shows that variables **AcceptedCmp5, AcceptedCmp3, and Recency** have the highest feature importance with a **score of 0.3** indicating that the model classification for magazine behavior is based on these features and that they are highly contributing in the prediction of the target variable. Thus, **it is recommended that the company should focus on these parameters and features while analyzing the magazine subscription behavior of the customer.**

Decision Tree model comparison with Logistic Regression & SVM model

---

- As compared to the goodness of fit metrics and evaluation metrics for the Logistic Regression, SVM model and Decision Tree model, it is observed that **SVM model performed better in classifying the magazine behavior of the customers** as compared to the Logistic Regression and Decision Tree model.
- Decision Tree model **performed poor** as compared to both Logistic Regression and SVM model.
- This is because the **accuracy of the training and testing data dropped down to high percent** as compared to that of the SVM and Logistic Regression model.
- Apart from that, when analyzed the **confusion matrix we see that high number of classes are correctly classified** using the SVM model as compared to the LR model and Decision Tree model, which is important because if the less number of false positive and false negative values are there, it will be less of a job to manually address that and hence would be effective in analyzing the data.
- Also, the **precision and recall score is relatively low for the Decision Tree model** in comparison with the Logistic Regression model and SVM model, indicating that the prediction of classes is better performed in SVM model and Logistic Regression model.

Results

Task 5: Compare the accuracy of all the models (overall accuracy, precision, recall) and the overall variables that were deemed significant. Discuss which model you would recommend based on these three metrics. Discuss what key variables they should focus on and their business context once you select your final model (that should be the key takeaway).

---

1. Comparing the accuracy of the models built (accuracy, precision, recall)

In [ ]:
```
metrics_data = []
metrics_data.append(['Logistic Regression Model', model_result1, '38%', '64%', 'AcceptedCmp3, AcceptedCmp5, A
metrics_data.append(['SVM Model', model_result2, '40%', '64%', 'Recency, NumWebVisitsMonth, MntMeatProducts']
metrics_data.append(['Decision Tree Model', model_result3, '27%', '67%', 'AcceptedCmp5, AcceptedCmp3, Recency
metrics_df = pd.DataFrame(metrics_data, columns=['Model', 'Accuracy', 'Precision', 'Recall', 'Significant Var
metrics_df
```

Out[79]:

	Model	Accuracy	Precision	Recall	Significant Variables
0	Logistic Regression Model	0.7835	38%	64%	AcceptedCmp3, AcceptedCmp5, AcceptedCmp2
1	SVM Model	0.7991	40%	64%	Recency, NumWebVisitsMonth, MntMeatProducts
2	Decision Tree Model	0.6719	27%	67%	AcceptedCmp5, AcceptedCmp3, Recency

Table 13. Model Comparison Table

2. Discussing the models to be recommended based on the evaluation metrics

---

Based on the evaluation metrics as shown in the table above, it is observed that **SVM model has performed the best** as compared to the Logistic Regression model and Decision Tree model for the classification of the magazine behavior for customers. This is because the accuracy and precision - recall score for the SVM model is higher as compared to the other models and **hence the model that would be recommended based on the evaluation metrics is the SVM model.**

3. Discussing the key variables they should focus on their business context

---

The key variables based on the model selected which is the SVM model are **'Recency', 'NumWebVisitsMonth', 'MntMeatProducts'.** The feature importance score for the SVM model for the three variables is 0.6 and hence they are the key variables that the company should focus on based on the business context.

Conclusion

Recommendations

- Based on the analysis and results, we conclude that SVM model is recommended to be implemented for the classification of the magazine behavior for the customer. The SVM model is **overfitted as the accuracy values for training and testing data differ by a certain percent value**, and hence the model requires further improvement such that it can be used in future for further prediction and classification, which can be done by updating new features and data points that will increase the efficiency and performance of the model, implying a best-fit model for training.
- The features that should be focused upon are Recency, NumWebVisitsMonth, and MntMeatProducts based on the feature importance score of the SVM model. Hence, the company should focus on these features for analyzing the magazine subscription behavior of the customer.

Future Scope

The SVM model selected is able to classify the target variable and also extract the features that contribute to the prediction, but the performance and efficiency of the model can be improved and thus requires reevaluating the performance of the model by adding new features to the model and increasing the training data. Based on the results, it is observed that the data is overfitted and hence it is important that both the quantity and quality of data is improved to increase the efficiency of the model.

Thus, the model can be improved and updated based on new features being added to the dataset that can help to better analyze the target variable.

References

[1] What is Logistic regression? | IBM. (n.d.). https://www.ibm.com/topics/logistic-regression (https://www.ibm.com/topics/logistic-regression)

[2] Gandhi, R. (2022, November 14). Support Vector Machine — Introduction to Machine Learning Algorithms. Medium. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47)

[3] GeeksforGeeks. (2023). Decision Tree. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/ (https://www.geeksforgeeks.org/decision-tree/)

[4] Singh, K. (2022). How to Improve Class Imbalance using Class Weights in Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/ (https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/)

[5] Brownlee, J. (2020b). Cost-Sensitive SVM for Imbalanced Classification. MachineLearningMastery.com. https://machinelearningmastery.com/cost-sensitive-svm-for-imbalanced-classification/#:~:text=Perhaps%20the%20simplest%20and%20most,level%20weighted%20modification%20was%20proposed (https://machinelearningmastery.com/cost-sensitive-svm-for-imbalanced-classification/#:~:text=Perhaps%20the%20simplest%20and%20most,level%20weighted%20modification%20was%20proposed).