Submitted By: Ananya Sharma (Neu Id: 002954987)

Submitted to: Dr. Mary Donhoffner

Predictive Analytics- Module 3

Magazine Subscription Marketing Customer Behaviour

Introduction

Like every company, a magazine company is trying to understand who the regular subscribers of the magazine are. The company's sales have been on the decline, and we are trying to analyze the various parameters at hand. Furthermore, we are using logistic regression and support vector machine methodology in this project.

The dataset has 2240 Rows and 30 columns. The dataset has majorly two categorical variables namely Education and Marital Status. The rest of the data is either numerical or binary. The data set uses 506.6 + KB of space and has variables in object, int, and float data types.

Exploratory Data Analysis and Data Munging

```
#Statistical Analysis
Mark_Camp.describe()
```

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	...	Nur
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	...	
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	...	
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	...	
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	...	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	...	
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	...	
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	...	

8 rows × 26 columns

Figure 1: Description of the Data Set

The summary of the data set shows some outliers and anomalies with the dataset. We explore further and plot more graphs.

Checking for Duplicates

```
[62]:  #Finding any duplicated data
       Mark_Camp.duplicated()

t[62]: 0       False
       1       False
       2       False
       3       False
       4       False
               ...
       2235    False
       2236    False
       2237    False
       2238    False
       2239    False
       Length: 2240, dtype: bool
```

Figure 2: Checking Duplicates

The data set does not show much duplication and we move on to check the null values. From the figure below we see that the variable Income has 24 null values. Out of curiosity I also checked the skew of the dataset.

Checking for Null Values

```
[63]:  1 #Checking for Null Data
       2 Mark_Camp.isnull().sum()
       3 #The result below shows that Income has 24 Null Values

t[63]: 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
```

Figure 3: Null Values

```
1 #Understanding which variable is skewed and which is not
2 Mark_Camp.skew()
```

```
ID                      0.039832
Year_Birth             -0.349944
Income                  6.763487
Kidhome                 0.635288
Teenhome                0.407115
Recency                -0.001987
MntWines                1.175771
MntFruits               2.102063
MntMeatProducts         2.083233
MntFishProducts         1.919769
MntSweetProducts        2.136081
MntGoldProds            1.886106
NumDealsPurchases       2.418569
NumWebPurchases         1.382794
NumCatalogPurchases     1.880989
NumStorePurchases       0.702237
NumWebVisitsMonth       0.207926
AcceptedCmp3            3.291705
AcceptedCmp4            3.241574
AcceptedCmp5            3.291705
AcceptedCmp1            3.555444
AcceptedCmp2            8.472093
Complain               10.188972
Z_CostContact           0.000000
Z_Revenue               0.000000
Response                1.971555
dtype: float64
```

Right Skewed

Figure 4: Skewed Data

Plotting a Distplot for income shows that the data is right-skewed. Additionally, a box plot shows some outliers for the same.

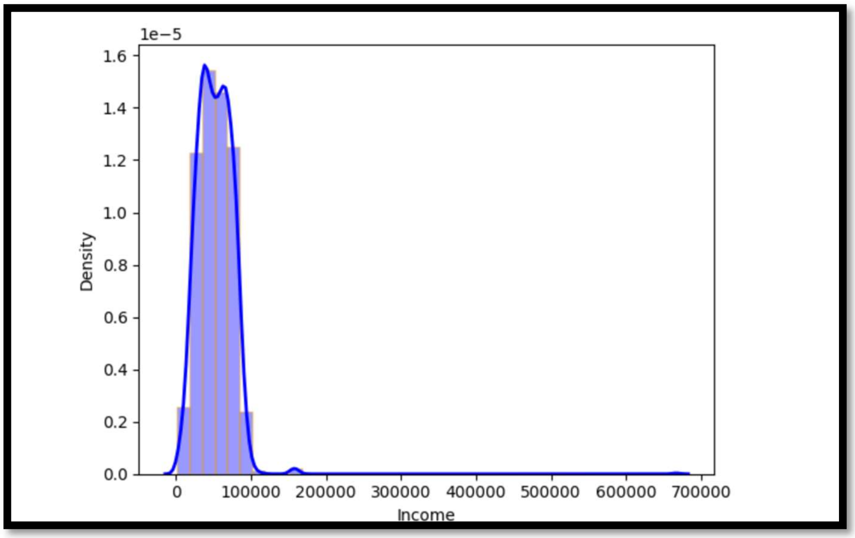

Figure 5: Distplot exploring the Variable Income

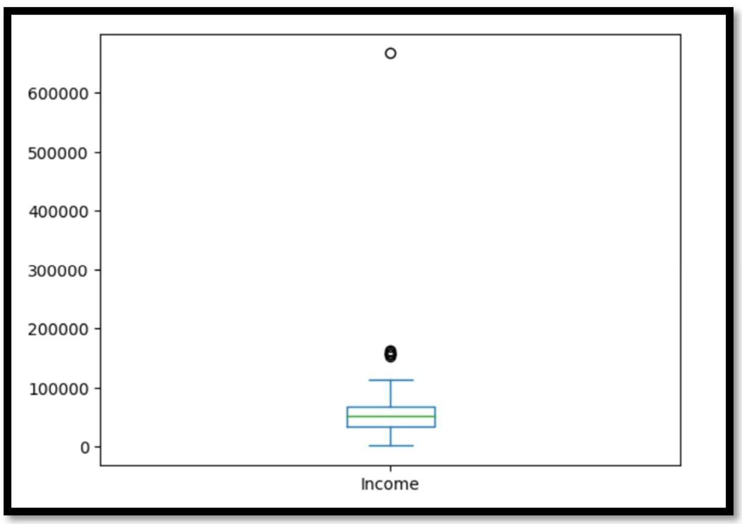

Figure 6: Boxplot with outliers for the Income Variable

After this, we used the IQR formula to trim down the outliers. Plotting a box plot of variable income shows that the outliers have been removed to a certain extent. The IQR method obliterates any value points above and below the whisker points. Furthermore, we remove the null values by applying the mean function and filling the values with the mean values.

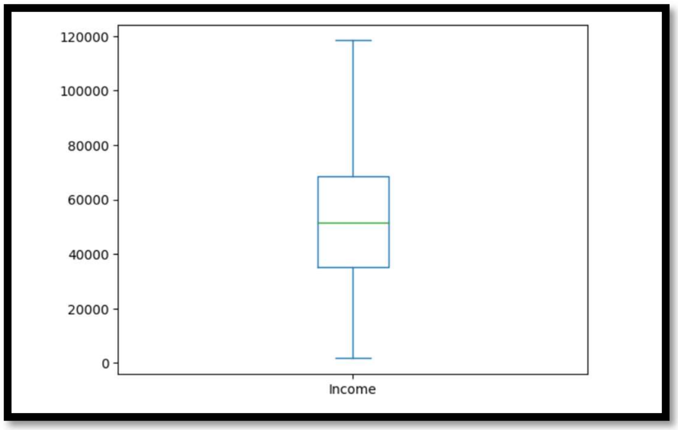

Figure 7: Income after Removal of Outliers

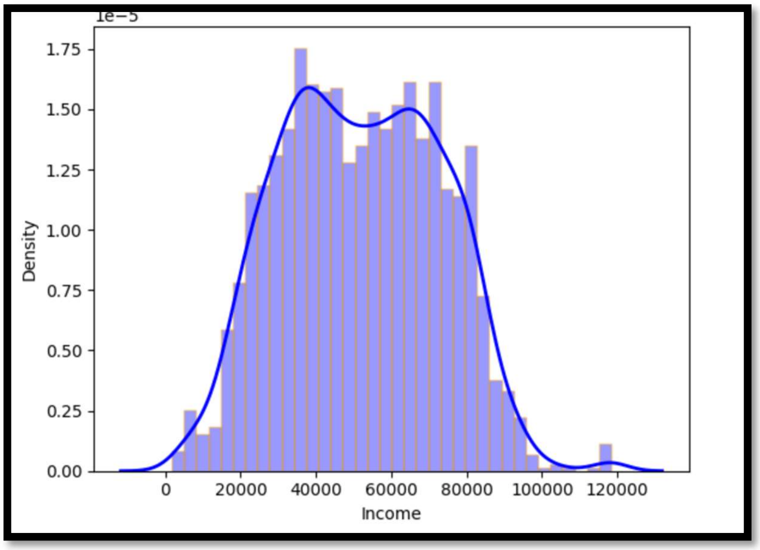

Figure 8: Income Histogram

The distribution of the Income is better after removing the outliers.

Removing the Null Values

```
#Removing the Null Value by filling the space with Null Values
mean_value=Mark_Camp['Income'].mean()
Mark_Camp['Income'].fillna(value=mean_value, inplace=True)
Mark_Camp
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebV
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	...	

Figure 9: Filling the mean values in place of Null Values for the Income Variable

We try to remove the variables from the data set that are redundant and don't contribute much to the analysis.

Dropping Redundant Columns

```
#Dropping the columns ID, Year, Birth, Education, Marital Status
Mark_Camp = Mark_Camp.drop(['ID', 'Year_Birth','Dt_Customer','Z_CostContact','Z_Revenue'], axis=1)
```

```
#Creating a Separate Data frame to study the Box plots for all variables
Categorical_Data_Removal=Mark_Camp.drop(['Education','Marital_Status'],axis=1)
```

```
#Dropping the Accepted Columns (1-5)
Mark_Camp = Mark_Camp.drop(['AcceptedCmp3', 'AcceptedCmp4','AcceptedCmp5','AcceptedCmp1','AcceptedCmp2'], axis=1)
```

Figure 10: Dropping Extra Columns

We have created a separate data frame by the name Categorical_Data_Removal because I was not able to run a for loop to plot the box plot for the rest of the data. I won't be deleting these parameters from the actual data frame Mark Camp.
Apart from this, I have removed variables like Year_Birth, AcceptedCmp3, Z_Revenue, etc. These variables don't contribute significantly to giving business results for increasing the magazine subscription.

```
# #Plotting Box Plot for the Features
for feature in Categorical_Data_Removal:
    sns.boxplot(Categorical_Data_Removal[feature],color='lightblue')
    plt.title(feature)
    plt.figure(figsize=(10,10))
```

Figure 11: Plotting Boxplots for Other Variables

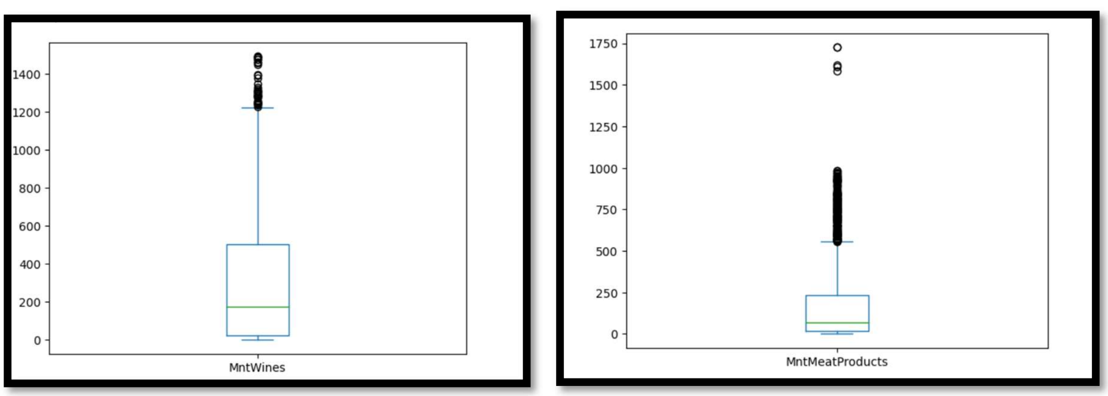

Figure 12: Boxplots with Outliers

After this we use the Inter Quartile Function to remove outliers from variables like MntWines,
MntFruits,MntMeatProducts,MntFishProducts,MntSweetProducts,MntGoldProds,NumDeals Purchases etc. The Next Figure shows that we have been able to remove the outliers.

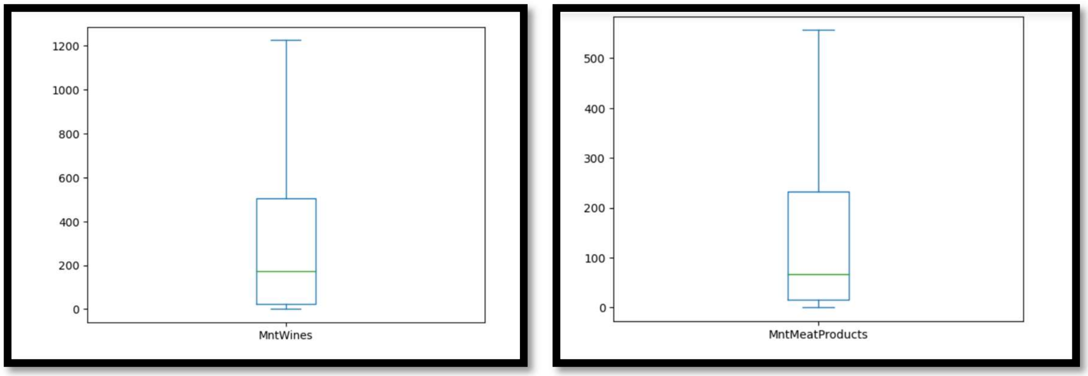

Figure 13: Variables MntWines and MntMeatProducts after Outlier Removal

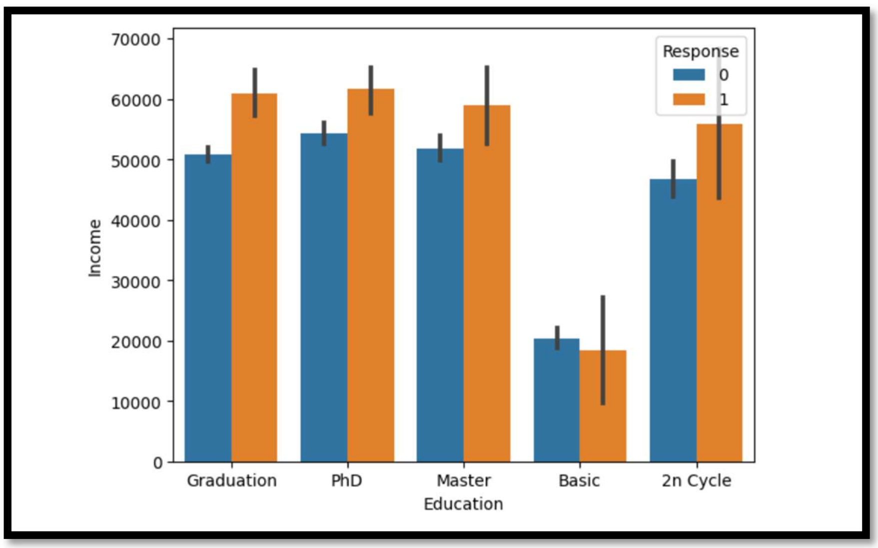

Figure 14: Box Plot Between Education and Income

Finally, after performing the various univariate analysis, we do a multivariate analysis between Education and Income.
As we can see, the graduates and the higher income group have more interest in the campaign versus the people who have had basic schooling.

After this, we perform a corr plot and a heat map analysis to know if the variables are causing multi-collinearity.

```
Response               1.000000
MntWines               0.243489
MntMeatProducts        0.236505
NumCatalogPurchases    0.220810
Income                 0.166115
Name: Response, dtype: float64
```

Figure 15: Corr ()

We see that the various parameters are not heavily correlated with each other. This satisfies one of the assumptions that are needed to perform a logistic regression. (The 6 Assumptions of Logistic Regression, 2020)

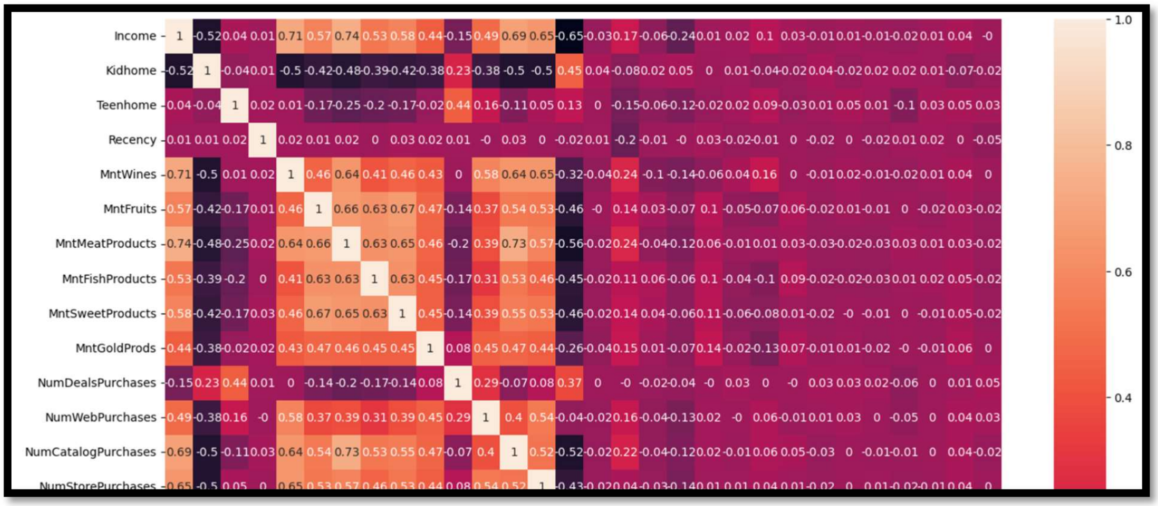

Figure 16: Heat Map

The heat map is too large, so I have put a snapshot of the portion that shows those variables that show a strong correlation. Income shows a good correlation with MntWines and MntMeatProducts to the tune of 0.71 and 0.74 respectively.

Analysis

Finally, we perform one hot encoding on the variables for processing the categorical variables into zeros and ones for easy interpretation by the Model.
We get separate features for each categorical variable. (Dey, 2021)

One Hot Encoding

```
Mark_Camp=pd.get_dummies(Mark_Camp,columns=['Education','Marital_Status'])
Mark_Camp
```

```
(       Income  Kidhome  Teenhome  Recency  MntWines  MntFruits  \
0     58138.0        0         0       58       635         81
1     46344.0        1         1       38        11          1
2     71613.0        0         0       26       426         49
3     26646.0        1         0       26        11          4
4     58293.0        1         0       94       173         43
...       ...      ...       ...      ...       ...        ...
2235  61223.0        0         1       46       709         43
2236  64014.0        2         1       56       406          0
2237  56981.0        0         0       91       908         48
2238  69245.0        0         1        8       428         30
2239  52869.0        1         1       40        84          3

      MntMeatProducts  MntFishProducts  MntSweetProducts  MntGoldProds  ...  \
0                 546              172                81          88.0  ...
1                   6                2                 1           6.0  ...
2                 127              111                21          42.0  ...
```

Figure 17: One Hot Encoding

Train and Test

```python
y = Mark_Camp[['Response']]
x = Mark_Camp.drop('Response',axis=1)
```

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, random_state = 0)
x_train.shape, x_test.shape
```

```
((1568, 29), (672, 29))
```

Figure 18: Test and Train Model

We are testing our target variable, which is the Response variable here. We set the test size to 30% and train the data set with 70% data. We are taking random state as 0 since we don't want any uncertainty or randomness here.

Logistic Regression

```python
# Fitting Logistic Regression Model
import statsmodels.api as sm
XLog = sm.add_constant(x_train)
l_model = sm.Logit(y_train, XLog)
log_fit1 = l_model.fit()
print(log_fit1.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.293197
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:               Response   No. Observations:                 1568
Model:                          Logit   Df Residuals:                     1540
Method:                           MLE   Df Model:                           27
Date:                Thu, 13 Oct 2022   Pseudo R-squ.:                  0.2728
Time:                        13:43:20   Log-Likelihood:                -459.73
converged:                      False   LL-Null:                       -632.22
Covariance Type:            nonrobust   LLR p-value:                 7.070e-57
===========================================================================================
                              coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                      -4.5310        nan        nan        nan         nan         nan
Income                   1.601e-05   9.12e-06      1.756      0.079   -1.86e-06    3.39e-05
Kidhome                    -0.0672      0.241     -0.279      0.780      -0.539       0.405
Teenhome                   -1.3748      0.239     -5.749      0.000      -1.843      -0.906
```

Figure 19: Logistic Regression

From the above figure 19, we understand that our p-value is extremely strong and is less than 0.002. Additionally, significant variables like Recency, MntWines, MntMeatProducts, NumWebPurchases, NumStoresPurchases, and Catalogue Purchase are some variables that indicate that people frequenting online websites are more susceptible to subscribing to magazines, or maybe having more content about wines in the magazine can lead to more people buying from the company.

The p-value is less than 0.05, henceforth our hypothesis stands correct and logistic regression does help to understand that our model is good to go with.

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.5310	nan	nan	nan	nan	nan
Income	1.601e-05	9.12e-06	1.756	0.079	-1.86e-06	3.39e-05
Kidhome	-0.0672	0.241	-0.279	0.780	-0.539	0.405
Teenhome	-1.3748	0.239	-5.749	0.000	-1.843	-0.906
Recency	-0.0272	0.003	-8.306	0.000	-0.034	-0.021
MntWines	0.0015	0.000	3.892	0.000	0.001	0.002
MntFruits	0.0043	0.004	0.985	0.325	-0.004	0.013
MntMeatProducts	0.0026	0.001	3.047	0.002	0.001	0.004
MntFishProducts	-0.0023	0.002	-1.130	0.259	-0.006	0.002
MntSweetProducts	0.0037	0.004	0.874	0.382	-0.005	0.012
MntGoldProds	0.0038	0.003	1.530	0.126	-0.001	0.009
NumDealsPurchases	0.2058	0.077	2.679	0.007	0.055	0.356
NumWebPurchases	0.0216	0.043	0.505	0.613	-0.062	0.106
NumCatalogPurchases	0.1005	0.038	2.630	0.009	0.026	0.175
NumStorePurchases	-0.2467	0.041	-6.082	0.000	-0.326	-0.167
NumWebVisitsMonth	0.3365	0.061	5.479	0.000	0.216	0.457
Complain	0.3871	0.887	0.436	0.663	-1.352	2.126

Figure 20: Extension of the logistic regression model

Accuracy Of Model

```python
#Importing Logistic Regression Libraries from sklearn
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

logreg1 = LogisticRegression()
logreg1.fit(x_train, y_train)

y_pred = logreg.predict(x_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg1.score(x_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.84
```

Figure 21: Accuracy of Logistic Regression Model

The accuracy of the Logistic regression model comes to **84%.**

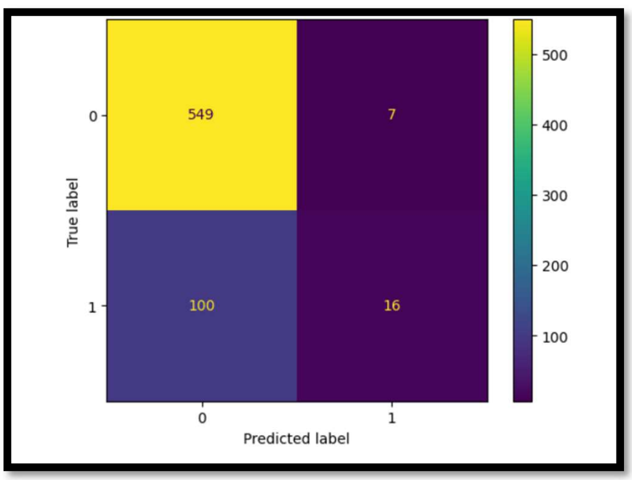

Figure 22: Confusion Matrix

As per the confusion matrix, we see that 549 observations come in True Positive, and 7 observations come in False Positive. 100 observations are in False Negative and 16 are in True Negative.

Precision and Recall

```
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.6956521739130435
Recall: 0.13793103448275862
```

Figure 23: Precision and Recall

The Precision comes to 70%, which means 70% of the people will choose to subscribe to the Magazine subscription and the Recall comes to 0.137, which implies that 14% of people subscribed to the magazine.

Support Vector Machine

```
from sklearn import svm

svm_model = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
svm_model.fit(x_train, y_train)

#Predict the response for test dataset
z_pred = svm_model.predict(x_test)
cnf_matrix = metrics.confusion_matrix(y_test, z_pred)
cnf_matrix
```

Figure 24: Support Vector Machine

In the above figure 24, we are trying to fit the SVM Model. The linear Kernel helps to add more dimension for better classification of the Response variable

```
array([[546,  10],
       [104,  12]], dtype=int64)
```

```
svm_model.score(x_test, y_test)
```

```
0.8303571428571429
```

Figure: Accuracy Score of SVM

Interpreting the confusion Matrix of an SVM Model which has been given above.
<u>The accuracy of the model comes to **83%.**</u>
We find the 10 results were false positives. 546 observations were true positives.
104 results were false negatives and 12 were true negatives.

```
]:  from sklearn import metrics

    # Model Precision
    print("Precision:",metrics.precision_score(y_test, y_pred))

    # Model Recall
    print("Recall:",metrics.recall_score(y_test, y_pred))

    Precision: 0.6956521739130435
    Recall: 0.13793103448275862
```

Figure 25: Precision and Recall for SVM Model

The precision of the model **is 70%** and the Recall is **14%**. This means that 14% of people have subscribed to the magazine, whereas 70% of people will consider taking the magazine.

Conclusion

1)We have removed outliers in a couple of variables as a step to preparing our model for training and testing. Used Histplot, Distplot, Boxplot, and Heatmaps to analyze the dataset.

2) In our project both SVM and Logistic Regression are giving almost similar accuracy, with Logistic Regression slightly higher than SVM. However, SVM is deterministic in nature and Logistic Regression is probabilistic in approach.

Observations	**Logistic Regression**	**Support Vector Machine**
Accuracy	84%	83%
Recall	14%	14%
Precision	70%	70%

3) The major reason why the SVM model's accuracy is close to Logistic Regression is that we use the Linear Kernel and henceforth, we can use both models for this project since the dataset is not very large.

4) A few features like Income, Recency, NumWebVisitsMonth, and MntWines are a few variables that help us to decide that the magazine company should consider these aspects to get more subscriptions. For example, more content on wine ensures higher subscriptions to the magazine. More education means that a person will have good earnings and is likely to subscribe to magazines, as per our bar plot.

References

Koirala, S. (2021, December 7). *Customers Subscription Analysis and Prediction Based on App Behavior Analysis (Logistic Regression)*. Medium. Retrieved October 13, 2022, from https://towardsdatascience.com/customer-subscription-analysis-and-prediction-based-on-app-behavior-analysis-logistic-regression-16a2c0def544

Bassey, P. (2022, April 4). *Logistic Regression Vs Support Vector Machines (SVM)*. Medium. Retrieved October 13, 2022, from https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16

The 6 Assumptions of Logistic Regression (With Examples). (2020, October 13). Statology. Retrieved October 13, 2022, from https://www.statology.org/assumptions-of-logistic-regression/

Dey, V. (2021, November 20). *When to Use One-Hot Encoding in Deep Learning?* Analytics India Magazine. Retrieved October 13, 2022, from https://analyticsindiamag.com/when-to-use-one-hot-encoding-in-deep-learning/\

Narkhede, S. (2021, June 15). *Understanding Confusion Matrix - Towards Data Science*. Medium. Retrieved October 13, 2022, from https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Appendix

Importing libraries

#Importing Libraries

import numpy **as** np

import pandas **as** pd

import matplotlib.pyplot **as** plt

import seaborn **as** sns

Reading the libraries

#Reading the File

Mark_Camp=pd.read_csv("marketing_campaign.csv")

Mark_Camp.shape

Details of the Campaign

 # First Ten Details of the Data Set

Mark_Camp.head(n=10)

Campaign Information

Mark_Camp.info() # *Information on the Data Set*

Statistical Analysis

#Statistical Analysis

Mark_Camp.describe()

Checking for Duplicates

#Finding any duplicated data

Mark_Camp.duplicated()

Checking for Null Values

#Checking for Null Data

Mark_Camp.isnull().sum()

#The result below shows that Income has 24 Null Values

#Understanding which variable is skewed and which is not

Mark_Camp.skew()

Distplot

#Plotting a graph to understand the distribution of Data

```
sns.distplot(Mark_Camp['Income'], hist=True, kde=True,
        bins=int(200/5), color = 'Blue',
        hist_kws={'edgecolor':'orange'},
        kde_kws={'linewidth': 2})
```

#Creating a Dist plot - The figure below shows that the income is slightly left skewed

Creating a Box Plot

#Creating a Box Plot to check any outliers in the Income Parameter

Mark_Camp['Income'].plot(kind='box')

 # The figure below shows that the data has got some outliers

Removing Outliers for Income Parameter

#Function to remove Outliers

```
IQR=Mark_Camp.Income.quantile(0.75)-Mark_Camp.Income.quantile(0.25)
lower_bridge=Mark_Camp.Income.quantile(0.25)-(IQR*1.5)
upper_bridge=Mark_Camp.Income.quantile(0.75)+(IQR*1.5)
print(lower_bridge, upper_bridge)
```

#Replacing the Null values with the lower and the higher bridge values to nullify the outliers

```
Mark_Camp.loc[Mark_Camp['Income']>=118350.5,'Income']=118350.5
Mark_Camp.loc[Mark_Camp['Income']<=-14525.5,'Income']=-14525.5
```

#Plotting a Box Plot to see if there is any outlier which is left

Mark_Camp['Income'].plot(kind='box')

Mark_Camp['MntMeatProducts'].plot(kind='box')

Distplot of Variable Income after removing Outliers

sns.distplot(Mark_Camp['Income'], hist=True, kde=True,

```
bins=int(180/5), color = 'Blue',
hist_kws={'edgecolor':'Orange'},
kde_kws={'linewidth': 2})
```

Removing the Null Values

```
#Removing the Null Value by filling the space with Null Values
mean_value=Mark_Camp['Income'].mean()
Mark_Camp['Income'].fillna(value=mean_value, inplace=True)
Mark_Camp
```

Dropping Redundant Columns

```
#Dropping the columns ID, Year, Birth, Education, Marital Status
Mark_Camp = Mark_Camp.drop(['ID',
'Year_Birth','Dt_Customer','Z_CostContact','Z_Revenue'], axis=1)
#Creating a Separate Data frame to study the Box plots for all variables
Categorical_Data_Removal=Mark_Camp.drop(['Education','Marital_Status'],axis=1)
#Dropping the Accepted Columns (1-5)
Mark_Camp = Mark_Camp.drop(['AcceptedCmp3',
'AcceptedCmp4','AcceptedCmp5','AcceptedCmp1','AcceptedCmp2'], axis=1)
Mark_Camp.head()
# #Plotting Box Plot for the Features
for feature in Categorical_Data_Removal:
    sns.boxplot(Categorical_Data_Removal[feature],color='lightblue')
    plt.title(feature)
    plt.figure(figsize=(10,10))
 #From the plot we see that there are outliers in
MntWines,MntFruits,MntMeatProducts,MntFishProducts,MntSweetProducts,MntGoldProds,
NumDealsPurchases
```

Removing Outliers of Other Variables

```
#Removing the outliers in MntWines
#Calculating the inter quantile range
```

```
IQR1=Mark_Camp.MntWines.quantile(0.75)-Mark_Camp.quantile(0.25)

lower_bridge1=Mark_Camp.MntWines.quantile(0.25)-(IQR1*1.5)

upper_bridge1=Mark_Camp.MntWines.quantile(0.75)+(IQR1*1.5)

print(lower_bridge1,upper_bridge1)

Mark_Camp.loc[Mark_Camp['MntWines']>=1225.000,'MntWines']=1225.000

Mark_Camp.loc[Mark_Camp['MntWines']<=-697.000,'MntWines']=-697.000

Mark_Camp['MntWines'].plot(kind='box')

#Removing the outliers of MntFruits

IQR2=Mark_Camp.MntFruits.quantile(0.75)-Mark_Camp.MntFruits.quantile(0.25)

lower_bridge2=Mark_Camp.MntFruits.quantile(0.25)-(IQR2*1.5)

upper_bridge2=Mark_Camp.MntFruits.quantile(0.75)+(IQR2*1.5)

print(lower_bridge2,upper_bridge2)

Mark_Camp.loc[Mark_Camp['MntFruits']>=81.0,'MntFruits']=81.0

Mark_Camp.loc[Mark_Camp['MntFruits']<-47.0,'MntFruits']=-47.0

#Plot without the outliers

Mark_Camp['MntFruits'].plot(kind='box')

#Removinf outliers for MntMeatProducts

IQR3=Mark_Camp.MntMeatProducts.quantile(0.75)-
Mark_Camp.MntMeatProducts.quantile(0.25)

lower_bridge3=Mark_Camp.MntMeatProducts.quantile(0.25)-(IQR3*1.5)

upper_bridge3=Mark_Camp.MntMeatProducts.quantile(0.75)+(IQR3*1.5)

print(lower_bridge3,upper_bridge3)

Mark_Camp.loc[Mark_Camp['MntMeatProducts']>=556.0,'MntMeatProducts']=556.0

Mark_Camp.loc[Mark_Camp['MntMeatProducts']<-308.0,'MntMeatProducts']=-308.0

#Plot without the outliers

Mark_Camp['MntMeatProducts'].plot(kind='box')

#Removing outliers for MntSweetProducts
```

IQR4=Mark_Camp.MntSweetProducts.quantile(0.75)-Mark_Camp.MntSweetProducts.quantile(0.25)

lower_bridge4=Mark_Camp.MntSweetProducts.quantile(0.25)-(IQR4*1.5)

upper_bridge4=Mark_Camp.MntSweetProducts.quantile(0.75)+(IQR4*1.5)

print(lower_bridge4,upper_bridge4)

#Higher and Lower Limit

Mark_Camp.loc[Mark_Camp['MntSweetProducts']>=81.0,'MntSweetProducts']=81.0

Mark_Camp.loc[Mark_Camp['MntSweetProducts']<-47.0,'MntSweetProducts']=-47.0

#Plot without the outliers

Mark_Camp['MntSweetProducts'].plot(kind='box')

#Removing outliers for MntGoldProds

IQR5=Mark_Camp.MntGoldProds.quantile(0.75)-Mark_Camp.MntGoldProds.quantile(0.25)

lower_bridge5=Mark_Camp.MntGoldProds.quantile(0.25)-(IQR5*1.5)

upper_bridge5=Mark_Camp.MntGoldProds.quantile(0.75)+(IQR5*1.5)

print(lower_bridge5,upper_bridge5)

#Higher and Lower Limit

Mark_Camp.loc[Mark_Camp['MntGoldProds']>=126.5,'MntGoldProds']=126.5

Mark_Camp.loc[Mark_Camp['MntGoldProds']< -61.0,'MntGoldProds']=-61.0

#Plot without the outliers

Mark_Camp['MntGoldProds'].plot(kind='box')

IQR6=Mark_Camp.NumDealsPurchases.quantile(0.75)-Mark_Camp.NumDealsPurchases.quantile(0.25)

lower_bridge6=Mark_Camp.NumDealsPurchases.quantile(0.25)-(IQR6*1.5)

upper_bridge6=Mark_Camp.NumDealsPurchases.quantile(0.75)+(IQR6*1.5)

print(lower_bridge6, upper_bridge6)

Mark_Camp.loc[Mark_Camp['NumDealsPurchases']>=6.0,'NumDealsPurchases']=6.0
Mark_Camp.loc[Mark_Camp['NumDealsPurchases']< -2.0,'NumDealsPurchases']=-2.0

#Plot without the outliers
Mark_Camp['NumDealsPurchases'].plot(kind='box')

```
IQR8=Mark_Camp.NumWebPurchases.quantile(0.75)-
Mark_Camp.NumWebPurchases.quantile(0.25)

lower_bridge8=Mark_Camp.NumWebPurchases.quantile(0.25)-(IQR8*1.5)

upper_bridge8=Mark_Camp.NumWebPurchases.quantile(0.75)+(IQR8*1.5)

print(lower_bridge8, upper_bridge8)

Mark_Camp.loc[Mark_Camp['NumWebPurchases']>=12.0,'NumWebPurchases']=12.0

Mark_Camp.loc[Mark_Camp['NumWebPurchases']<-4.0,'NumWebPurchases']=-4.0


#Plot without the outliers

Mark_Camp['NumWebPurchases'].plot(kind='box')
```

Bar Plot For Education and Income

```
#Making a Bar Plot

sns.barplot(x='Education',y='Income', data=Mark_Camp,

            hue='Response')

plt.show()

#When we try to study the relationship of income the education level while studythe ing
response of the campaigns we understand

#That Graduates are more interested in Magazine Subscription
```

Correlation

```
#Understanding the correlation of the various Variables

corr = Mark_Camp.corr()

corr.Response.sort_values(ascending=False).head(5)
```

Heat Map

```
plt.figure(figsize=[16,16])

matrix = Mark_Camp.corr().round(2)

sns.heatmap(matrix, annot=True)

plt.show()
```

#From the heat Map we understand that there are a lot of variables that are not correlated to each other

One Hot Encoding

Mark_Camp=pd.get_dummies (Mark_Camp,columns=['Education','Marital_Status'])

Mark_Camp

Train and Test

y = Mark_Camp[['Response']]

x = Mark_Camp.drop('Response',axis=1)

from sklearn.model_selection **import** train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, random_state = 0)

x_train.shape, x_test.shape

Logistic Regression

 # Fitting Logistic Regression Model

import statsmodels.api **as** sm

XLog = sm.add_constant(x_train)

l_model = sm.Logit(y_train, XLog)

log_fit1 = l_model.fit()

print(log_fit1.summary())

Accuracy Of Model

#Importing Logistic Regression libraries from sklearn

from sklearn.linear_model **import** LogisticRegression

from sklearn **import** metrics

logreg1 = LogisticRegression()

logreg1.fit(x_train, y_train)

y_pred = logreg.predict(x_test)

print('Accuracy of logistic regression classifier on test set:
{:.2f}'.format(logreg1.score(x_test, y_test)))

Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
logistic_regression= LogisticRegression()
model=logistic_regression.fit(x_train,y_train)
plot_confusion_matrix(logistic_regression, x_test, y_test)
plt.show()
```

Precision and Recall

```
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

Support Vector Machine

```
from sklearn import svm
svm_model = svm.SVC(kernel='linear') # Linear Kernel
#Train the model using the training sets
svm_model.fit(x_train, y_train)
#Predict the response for test dataset
z_pred = svm_model.predict(x_test)
cnf_matrix = metrics.confusion_matrix(y_test, z_pred)
cnf_matrix
print()
svm_model.score(x_test, y_test)
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision
print("Precision:",metrics.precision_score(y_test, y_pred))
# Model Recall
```

```
print("Recall:",metrics.recall_score(y_test, y_pred))
```