Lead Scoring Case Study

In [1]:

```
# Importing all necessary libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Reading the dataset and to get the idea of how the table looks

Leads=pd.read_csv('Leads.csv')

Leads.iloc[np.r_[0:5, -5:0]]
```

Out[2]:

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	305
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.0	1428
9235	19d6451e-fcd6-407c-b83b-48e1af805ea9	579564	Landing Page Submission	Direct Traffic	Yes	No	1	8.0	1845
9236	82a7005b-7196-4d56-95ce-a79f937a158d	579546	Landing Page Submission	Direct Traffic	No	No	0	2.0	238
9237	aac550fe-a586-452d-8d3c-f1b62c94e02c	579545	Landing Page Submission	Direct Traffic	Yes	No	0	2.0	199

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website
9238	5330a7d1-2f2b-4df4-85d6-64ca2f6b95b9	579538	Landing Page Submission	Google	No	No	1	3.0	499
9239	571b5c8e-a5b2-4d57-8574-f2ffb06fdeff	579533	Landing Page Submission	Direct Traffic	No	No	1	6.0	1279

10 rows × 37 columns

Data Understanding

In [3]:

```
# Shape of the dataset

Leads.shape
```

Out[3]:

```
(9240, 37)
```

In [4]:

```
# let's view the data information like what are thew datatypes of the variabl

Leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 37 columns):
 #   Column                                         Non-Null Count  Dtype  
---  ------                                         --------------  -----  
 0   Prospect ID                                    9240 non-null   object 
 1   Lead Number                                    9240 non-null   int64  
 2   Lead Origin                                    9240 non-null   object 
 3   Lead Source                                    9204 non-null   object 
 4   Do Not Email                                   9240 non-null   object 
 5   Do Not Call                                    9240 non-null   object 
 6   Converted                                      9240 non-null   int64  
 7   TotalVisits                                    9103 non-null   float64
 8   Total Time Spent on Website                    9240 non-null   int64  
 9   Page Views Per Visit                           9103 non-null   float64
 10  Last Activity                                  9137 non-null   object 
 11  Country                                        6779 non-null   object 
 12  Specialization                                 7802 non-null   object 
 13  How did you hear about X Education             7033 non-null   object 
 14  What is your current occupation                6550 non-null   object 
 15  What matters most to you in choosing a course  6531 non-null   object 
 16  Search                                         9240 non-null   object 
 17  Magazine                                       9240 non-null   object 
 18  Newspaper Article                              9240 non-null   object 
 19  X Education Forums                             9240 non-null   object 
 20  Newspaper                                      9240 non-null   object 
 21  Digital Advertisement                          9240 non-null   object 
 22  Through Recommendations                        9240 non-null   object 
 23  Receive More Updates About Our Courses         9240 non-null   object 
 24  Tags                                           5887 non-null   object 
```

```
 25  Lead Quality                               4473 non-null   object
 26  Update me on Supply Chain Content          9240 non-null   object
 27  Get updates on DM Content                  9240 non-null   object
 28  Lead Profile                               6531 non-null   object
 29  City                                       7820 non-null   object
 30  Asymmetrique Activity Index                5022 non-null   object
 31  Asymmetrique Profile Index                 5022 non-null   object
 32  Asymmetrique Activity Score                5022 non-null   float64
 33  Asymmetrique Profile Score                 5022 non-null   float64
 34  I agree to pay the amount through cheque   9240 non-null   object
 35  A free copy of Mastering The Interview     9240 non-null   object
 36  Last Notable Activity                      9240 non-null   object
dtypes: float64(4), int64(3), object(30)
memory usage: 2.6+ MB
```

As we can see from the above, there are 7 numerical variables columns and remaining 30 columns are having categorical variables.

In [5]:

```
# Let's describe the data and have some statistical idea about the dataset li

Leads.describe()
```

Out[5]:

	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Score	Asymr Profi
count	9240.000000	9240.000000	9103.000000	9240.000000	9103.000000	5022.000000	502:
mean	617188.435606	0.385390	3.445238	487.698268	2.362820	14.306252	1(
std	23405.995698	0.486714	4.854853	548.021466	2.161418	1.386694	:
min	579533.000000	0.000000	0.000000	0.000000	0.000000	7.000000	1:
25%	596484.500000	0.000000	1.000000	12.000000	1.000000	14.000000	1!
50%	615479.000000	0.000000	3.000000	248.000000	2.000000	14.000000	1(
75%	637387.250000	1.000000	5.000000	936.000000	3.000000	15.000000	1(
max	660737.000000	1.000000	251.000000	2272.000000	55.000000	18.000000	2(

As we can see from the above table, some variable columns like **'totalVisits','Total Time Spent on Website' and 'Page Views Per Visit'** are having outliers while others not so much.

Now, from our above observations from two tables we can see that there are some count mismatch and also some columns are redundant. Hence, first we will try to remove those redundant columns and after that we will check the missing values in the dataset.

Cleaning the dataset

In [6]:

```
# Dropping redundant columns like 'Prospect ID','Lead Number','Country',
# 'I agree to pay the amount through cheque' and 'a free copy of Mastering Th

red_cols=['Prospect ID','Lead Number','Country','I agree to pay the amount th
          'A free copy of Mastering The Interview','City']

Leads_df=Leads.drop(red_cols,1)
```

In [7]:

```
Leads_df.head()
```

Out[7]:

	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Speciali
0	API	Olark Chat	No	No	0	0.0	0	0.0	Page Visited on Website	
1	API	Organic Search	No	No	0	5.0	674	2.5	Email Opened	
2	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	Email Opened	Bu Adminis
3	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	Unreachable	Med Adve
4	Landing Page Submission	Google	No	No	1	2.0	1428	1.0	Converted to Lead	

5 rows × 31 columns

Now, there are some columns/categorical variables having label as 'Select' which means the customer was not selected any option hence it is better to put it as null value - Because there was no suitable option present to select for the customer searching for.

In [8]:

```
# Replacing 'Select' and 'select' with NaN (Since it means no option is selec
Leads_df = Leads_df.replace('Select',np.nan)
```

In [9]:

```
# Check for missing values

round(Leads_df.isnull().sum()/len(Leads_df)*100,2)
```

Out[9]:

```
Lead Origin                                       0.00
Lead Source                                       0.39
Do Not Email                                      0.00
Do Not Call                                       0.00
Converted                                         0.00
TotalVisits                                       1.48
Total Time Spent on Website                       0.00
Page Views Per Visit                              1.48
Last Activity                                     1.11
Specialization                                   36.58
How did you hear about X Education               78.46
What is your current occupation                  29.11
What matters most to you in choosing a course    29.32
Search                                            0.00
Magazine                                          0.00
Newspaper Article                                 0.00
X Education Forums                                0.00
Newspaper                                         0.00
Digital Advertisement                             0.00
Through Recommendations                           0.00
Receive More Updates About Our Courses            0.00
Tags                                             36.29
Lead Quality                                     51.59
```

```
Update me on Supply Chain Content                  0.00
Get updates on DM Content                          0.00
Lead Profile                                      74.19
Asymmetrique Activity Index                       45.65
Asymmetrique Profile Index                        45.65
Asymmetrique Activity Score                       45.65
Asymmetrique Profile Score                        45.65
Last Notable Activity                              0.00
dtype: float64
```

From above percentage of columns shows that some columns are having **more than 30% of missing values**, so it is better to remove these columns because it is **not a great move** if we are imputing more than approx. 30% of data based on **predictions** and **assumptions**.

In [10]:

```
# Droping Columns having more than 30% of missing values

drop_cols= Leads_df.isnull().sum()
drop_cols=drop_cols[drop_cols.values/len(Leads_df)>0.30]
drop_cols
```

Out[10]:

```
Specialization                           3380
How did you hear about X Education       7250
Tags                                     3353
Lead Quality                             4767
Lead Profile                             6855
Asymmetrique Activity Index              4218
Asymmetrique Profile Index               4218
Asymmetrique Activity Score              4218
Asymmetrique Profile Score               4218
dtype: int64
```

Columns we found that are having 30% of missing values in the dataset, so let's get rid of them.

In [11]:

```
# Dropping 9 columns and checking the remaining columns for missing values

drop_columns=list(drop_cols.keys())
Leads_df=Leads_df.drop(drop_columns,1)
round(Leads_df.isnull().sum()/len(Leads_df)*100,2)
```

Out[11]:

```
Lead Origin                                        0.00
Lead Source                                        0.39
Do Not Email                                       0.00
Do Not Call                                        0.00
Converted                                          0.00
TotalVisits                                        1.48
Total Time Spent on Website                        0.00
Page Views Per Visit                               1.48
Last Activity                                      1.11
What is your current occupation                   29.11
What matters most to you in choosing a course     29.32
Search                                             0.00
Magazine                                           0.00
Newspaper Article                                  0.00
X Education Forums                                 0.00
Newspaper                                          0.00
Digital Advertisement                              0.00
Through Recommendations                            0.00
Receive More Updates About Our Courses             0.00
Update me on Supply Chain Content                  0.00
Get updates on DM Content                          0.00
Last Notable Activity                              0.00
dtype: float64
```

Now, for columns having **below 30% missing values** - let's **impute maximum number of occurences** for a particluar column where missing values are found.

In [12]:
```
# Let's start with first columns of missing values

Leads_df['Lead Source'].value_counts()    # Lead Source column
```

Out[12]:
```
Google               2868
Direct Traffic       2543
Olark Chat           1755
Organic Search       1154
Reference             534
Welingak Website      142
Referral Sites        125
Facebook               55
bing                    6
google                  5
Click2call              4
Social Media            2
Live Chat               2
Press_Release           2
WeLearn                 1
NC_EDM                  1
welearnblog_Home        1
blog                    1
Pay per Click Ads       1
testone                 1
youtubechannel          1
Name: Lead Source, dtype: int64
```

Google is having highest number of occurences, hence we will impute the missing values with label 'Google'

In [13]:
```
# TotalVisits column

Leads_df['TotalVisits'].value_counts()
```

Out[13]:
```
0.0      2189
2.0      1680
3.0      1306
4.0      1120
5.0       783
6.0       466
1.0       395
7.0       309
8.0       224
9.0       164
10.0      114
11.0       86
13.0       48
12.0       45
14.0       36
16.0       21
15.0       18
17.0       16
18.0       15
20.0       12
19.0        9
21.0        6
23.0        6
25.0        5
24.0        5
```

```
27.0        5
22.0        3
28.0        2
29.0        2
26.0        2
115.0       1
41.0        1
55.0        1
251.0       1
141.0       1
32.0        1
42.0        1
74.0        1
43.0        1
30.0        1
54.0        1
Name: TotalVisits, dtype: int64
```

0.0 is having highest number of occurences, hence we will impute the missing values with label '0.0'

In [14]:
```
Leads_df['Page Views Per Visit'].value_counts()
```

Out[14]:
```
0.00    2189
2.00    1795
3.00    1196
4.00     896
1.00     651
        ...
2.13       1
4.40       1
6.67       1
8.33       1
2.45       1
Name: Page Views Per Visit, Length: 114, dtype: int64
```

0.0 is having highest number of occurences, hence we will impute the missing values with label '0.0'

In [15]:
```
Leads_df['Last Activity'].value_counts()
```

Out[15]:
```
Email Opened                    3437
SMS Sent                        2745
Olark Chat Conversation          973
Page Visited on Website          640
Converted to Lead                428
Email Bounced                    326
Email Link Clicked               267
Form Submitted on Website        116
Unreachable                       93
Unsubscribed                      61
Had a Phone Conversation          30
Approached upfront                 9
View in browser link Clicked       6
Email Received                     2
Email Marked Spam                  2
Visited Booth in Tradeshow         1
Resubscribed to emails             1
Name: Last Activity, dtype: int64
```

Email Opened is having highest number of occurences, hence we will impute the missing values with label 'Email Opened'

In [16]:
```
Leads_df['What is your current occupation'].value_counts()
```

Out[16]:
```
Unemployed              5600
Working Professional     706
Student                  210
Other                     16
Housewife                 10
Businessman                8
Name: What is your current occupation, dtype: int64
```

Unemployed is having highest number of occurences, hence we will impute the missing values with label 'Unemployed'

In [17]:
```
Leads_df['What matters most to you in choosing a course'].value_counts()
```

Out[17]:
```
Better Career Prospects      6528
Flexibility & Convenience       2
Other                           1
Name: What matters most to you in choosing a course, dtype: int64
```

Better Career Prospects is having highest number of occurences, hence we will impute the missing values with label 'Better Career Prospects'

In [18]:
```
# Now, imputing these values in our missing values dataset for respective cat

missing_values={'Lead Source':'Google','TotalVisits':'0.0','Page Views Per Vi
                'Last Activity':'Email Opened','What is your current occupati
                'What matters most to you in choosing a course':'Better Caree

Leads_df=Leads_df.fillna(value=missing_values)
```

In [19]:
```
Leads_df.isnull().sum()  # chekcing for missing values after imputing values
```

Out[19]:
```
Lead Origin                                      0
Lead Source                                      0
Do Not Email                                     0
Do Not Call                                      0
Converted                                        0
TotalVisits                                      0
Total Time Spent on Website                      0
Page Views Per Visit                             0
Last Activity                                    0
What is your current occupation                  0
What matters most to you in choosing a course    0
Search                                           0
Magazine                                         0
Newspaper Article                                0
X Education Forums                               0
Newspaper                                        0
Digital Advertisement                            0
Through Recommendations                          0
Receive More Updates About Our Courses           0
Update me on Supply Chain Content                0
Get updates on DM Content                        0
Last Notable Activity                            0
dtype: int64
```

Now all columns are having no missing values, we are good to go for our next analysis

In [20]:
```
Leads_df['Lead Source'].value_counts()
```

Out[20]:
```
Google              2904
Direct Traffic      2543
Olark Chat          1755
Organic Search      1154
Reference            534
Welingak Website     142
Referral Sites       125
Facebook              55
bing                   6
google                 5
Click2call             4
Social Media           2
Live Chat              2
Press_Release          2
WeLearn                1
NC_EDM                 1
welearnblog_Home       1
blog                   1
Pay per Click Ads      1
testone                1
youtubechannel         1
Name: Lead Source, dtype: int64
```

We found one column **'Lead Source'** is having same label name **'Google'** but in different format**('google')** so we need to make them in a same format hence using below commands.

In [21]:
```
# Applying lambda to captilize the first character of the column 'Lead Source

Leads_df['Lead Source']=Leads_df['Lead Source'].apply(lambda x:x.capitalize()

Leads_df['Lead Source'].value_counts()
```

Out[21]:
```
Google              2909
Direct traffic      2543
Olark chat          1755
Organic search      1154
Reference            534
Welingak website     142
Referral sites       125
Facebook              55
Bing                   6
Click2call             4
Live chat              2
Press_release          2
Social media           2
Nc_edm                 1
Blog                   1
Welearn                1
Pay per click ads      1
Testone                1
Welearnblog_home       1
Youtubechannel         1
Name: Lead Source, dtype: int64
```

Now, all data labels are in good shape and this is our final cleaning step of the dataset, we will proceed to our next step which is **Data Transformation**.

Data Transformation

Assigning numerical variables to categories with **'Yes' to 1 and 'No' to 0** or **converting binary variables (Yes/No) to (1/0)**

In [22]:
```
# Yes : 1 , No : 0

category={"No":0,"Yes":1}    # creating dictionary for two categories

# Column 'Do Not Email'

Leads_df['Do Not Email']=Leads_df['Do Not Email'].map(category)

# Column 'Do Not Call'

Leads_df['Do Not Call']=Leads_df['Do Not Call'].map(category)

# Column 'Search'

Leads_df['Search']=Leads_df['Search'].map(category)

# Column 'Magazine'

Leads_df['Magazine']=Leads_df['Magazine'].map(category)

# Column 'Newspaper Article'

Leads_df['Newspaper Article']=Leads_df['Newspaper Article'].map(category)

# Column 'X Education Forums'

Leads_df['X Education Forums']=Leads_df['X Education Forums'].map(category)

# Column 'Newspaper'

Leads_df['Newspaper']=Leads_df['Newspaper'].map(category)

# Column 'Digital Advertisement'

Leads_df['Digital Advertisement']=Leads_df['Digital Advertisement'].map(categ

# Column 'Through Recommendations'

Leads_df['Through Recommendations']=Leads_df['Through Recommendations'].map(c

# Column 'Receive More Updates About Our Courses'

Leads_df['Receive More Updates About Our Courses']=Leads_df['Receive More Upd

# Column 'Update me on Supply Chain Content'

Leads_df['Update me on Supply Chain Content']=Leads_df['Update me on Supply C

# Column 'Get updates on DM Content'

Leads_df['Get updates on DM Content']=Leads_df['Get updates on DM Content'].m
```

After converting the binary categories from 'Yes' to 1 and 'No' to 0, **we will use now dummy variables for mutiple levels of categories.**

In [23]:
```
Leads_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 22 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Lead Origin                                     9240 non-null   object
```

```
 1   Lead Source                                    9240 non-null   object
 2   Do Not Email                                   9240 non-null   int64
 3   Do Not Call                                    9240 non-null   int64
 4   Converted                                      9240 non-null   int64
 5   TotalVisits                                    9240 non-null   object
 6   Total Time Spent on Website                    9240 non-null   int64
 7   Page Views Per Visit                           9240 non-null   object
 8   Last Activity                                  9240 non-null   object
 9   What is your current occupation                9240 non-null   object
 10  What matters most to you in choosing a course  9240 non-null   object
 11  Search                                         9240 non-null   int64
 12  Magazine                                       9240 non-null   int64
 13  Newspaper Article                              9240 non-null   int64
 14  X Education Forums                             9240 non-null   int64
 15  Newspaper                                      9240 non-null   int64
 16  Digital Advertisement                          9240 non-null   int64
 17  Through Recommendations                        9240 non-null   int64
 18  Receive More Updates About Our Courses         9240 non-null   int64
 19  Update me on Supply Chain Content              9240 non-null   int64
 20  Get updates on DM Content                      9240 non-null   int64
 21  Last Notable Activity                          9240 non-null   object
dtypes: int64(14), object(8)
memory usage: 1.6+ MB
```

In [24]:

```
# Creating a dummy variables for 8 categories and dropping the first level.

dummy=pd.get_dummies(Leads_df[['Lead Origin','Lead Source','Last Activity','W
                              'What matters most to you in choosing a course',

# Adding these dummies to our original dataset

Leads_df=pd.concat([Leads_df,dummy],axis=1)

Leads_df.shape
```

Out[24]: (9240, 83)

Now, Removing duplicate columns or repeated columns

In [25]:

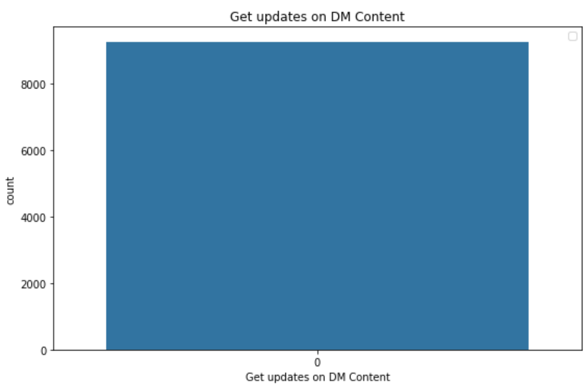
```
# We have created dummies for below categories hence removing the original co

duplicates=['Lead Origin','Lead Source','Last Activity','What is your current
            'What matters most to you in choosing a course','Last Notable Act


Leads_df=Leads_df.drop(duplicates,1)

Leads_df.shape
```

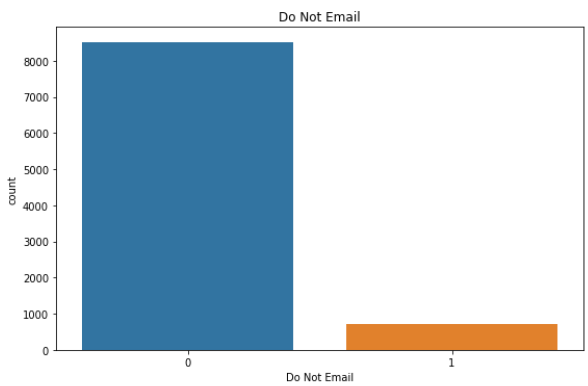
Out[25]: (9240, 77)

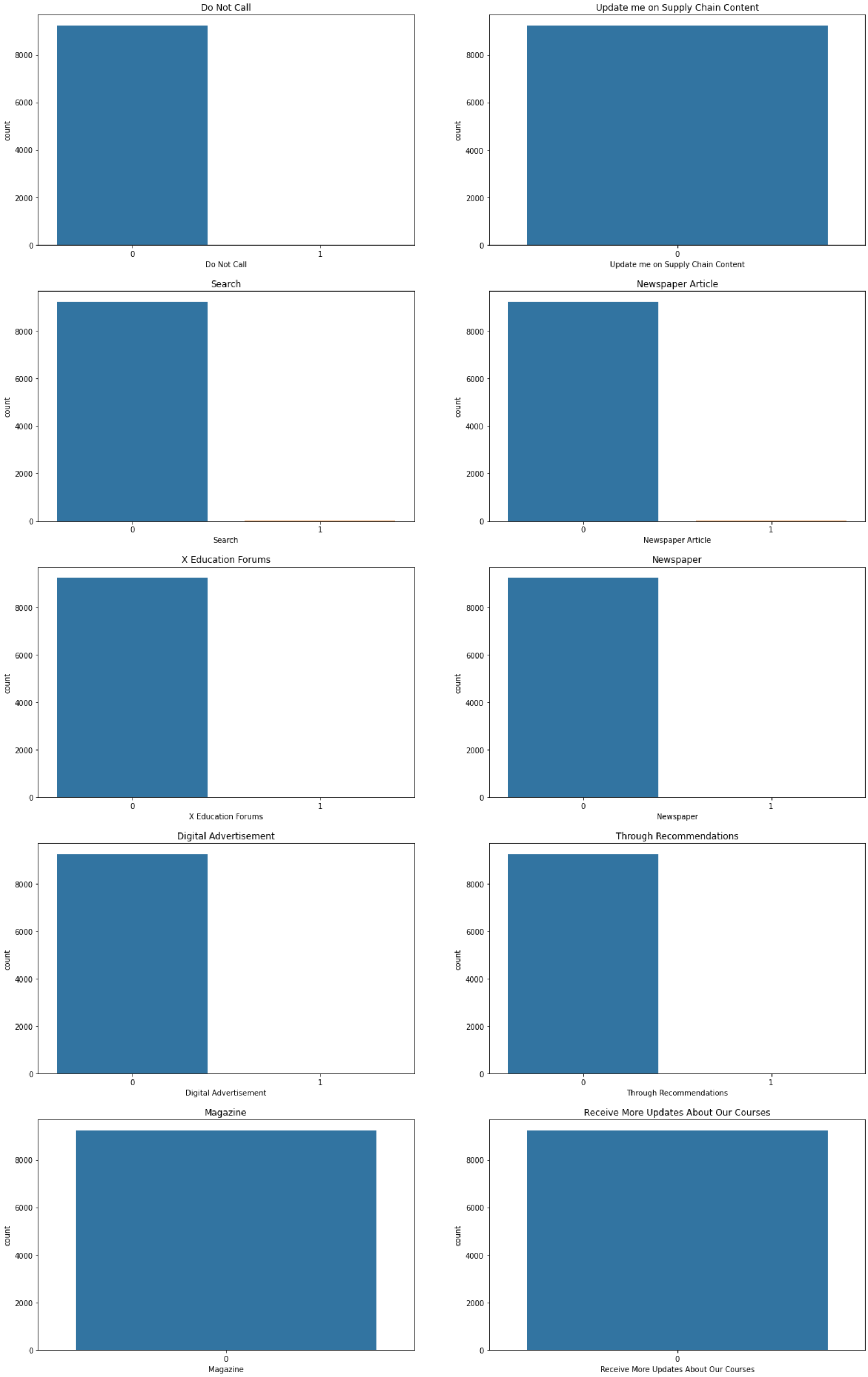
In [26]:

```
plt.figure(figsize = (20,40))

plt.subplot(6,2,1)
sns.countplot(Leads_df['Get updates on DM Content'])
plt.title('Get updates on DM Content')
plt.legend([0, 1],["No", "Yes"])

plt.subplot(6,2,2)
sns.countplot(Leads_df['Do Not Email'])
plt.title('Do Not Email')
```

```
plt.subplot(6,2,3)
sns.countplot(Leads_df['Do Not Call'])
plt.title('Do Not Call')


plt.subplot(6,2,4)
sns.countplot(Leads_df['Update me on Supply Chain Content'])
plt.title('Update me on Supply Chain Content')


plt.subplot(6,2,5)
sns.countplot(Leads_df['Search'])
plt.title('Search')


plt.subplot(6,2,6)
sns.countplot(Leads_df['Newspaper Article'])
plt.title('Newspaper Article')


plt.subplot(6,2,7)
sns.countplot(Leads_df['X Education Forums'])
plt.title('X Education Forums')


plt.subplot(6,2,8)
sns.countplot(Leads_df['Newspaper'])
plt.title('Newspaper')


plt.subplot(6,2,9)
sns.countplot(Leads_df['Digital Advertisement'])
plt.title('Digital Advertisement')


plt.subplot(6,2,10)
sns.countplot(Leads_df['Through Recommendations'])
plt.title('Through Recommendations')


plt.subplot(6,2,11)
sns.countplot(Leads_df['Magazine'])
plt.title('Magazine')


plt.subplot(6,2,12)
sns.countplot(Leads_df['Receive More Updates About Our Courses'])
plt.title('Receive More Updates About Our Courses')


plt.show()
```







In [27]:

```
# Dropping redundant variables(all have NO values)
redundant=['Receive More Updates About Our Courses','Update me on Supply Chai
```

```
Leads_df=Leads_df.drop(redundant,1)
```

In [28]:

```
# Converting some categories to numercial as they are imported as an 'Object'

Leads_df['TotalVisits']=pd.to_numeric(Leads_df['TotalVisits'])
Leads_df['TotalVisits']
Leads_df['Page Views Per Visit']=pd.to_numeric(Leads_df['Page Views Per Visit
Leads_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 73 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Do Not Email
9240 non-null   int64
 1   Do Not Call
9240 non-null   int64
 2   Converted
9240 non-null   int64
 3   TotalVisits
9240 non-null   float64
 4   Total Time Spent on Website
9240 non-null   int64
 5   Page Views Per Visit
9240 non-null   float64
 6   Search
9240 non-null   int64
 7   Newspaper Article
9240 non-null   int64
 8   X Education Forums
9240 non-null   int64
 9   Newspaper
9240 non-null   int64
 10  Digital Advertisement
9240 non-null   int64
 11  Through Recommendations
9240 non-null   int64
 12  Lead Origin_Landing Page Submission
9240 non-null   uint8
 13  Lead Origin_Lead Add Form
9240 non-null   uint8
 14  Lead Origin_Lead Import
9240 non-null   uint8
 15  Lead Origin_Quick Add Form
9240 non-null   uint8
 16  Lead Source_Blog
9240 non-null   uint8
 17  Lead Source_Click2call
9240 non-null   uint8
 18  Lead Source_Direct traffic
9240 non-null   uint8
 19  Lead Source_Facebook
9240 non-null   uint8
 20  Lead Source_Google
9240 non-null   uint8
 21  Lead Source_Live chat
9240 non-null   uint8
 22  Lead Source_Nc_edm
9240 non-null   uint8
```

```
 23  Lead Source_Olark chat
9240 non-null   uint8
 24  Lead Source_Organic search
9240 non-null   uint8
 25  Lead Source_Pay per click ads
9240 non-null   uint8
 26  Lead Source_Press_release
9240 non-null   uint8
 27  Lead Source_Reference
9240 non-null   uint8
 28  Lead Source_Referral sites
9240 non-null   uint8
 29  Lead Source_Social media
9240 non-null   uint8
 30  Lead Source_Testone
9240 non-null   uint8
 31  Lead Source_Welearn
9240 non-null   uint8
 32  Lead Source_Welearnblog_home
9240 non-null   uint8
 33  Lead Source_Welingak website
9240 non-null   uint8
 34  Lead Source_Youtubechannel
9240 non-null   uint8
 35  Last Activity_Converted to Lead
9240 non-null   uint8
 36  Last Activity_Email Bounced
9240 non-null   uint8
 37  Last Activity_Email Link Clicked
9240 non-null   uint8
 38  Last Activity_Email Marked Spam
9240 non-null   uint8
 39  Last Activity_Email Opened
9240 non-null   uint8
 40  Last Activity_Email Received
9240 non-null   uint8
 41  Last Activity_Form Submitted on Website
9240 non-null   uint8
 42  Last Activity_Had a Phone Conversation
9240 non-null   uint8
 43  Last Activity_Olark Chat Conversation
9240 non-null   uint8
 44  Last Activity_Page Visited on Website
9240 non-null   uint8
 45  Last Activity_Resubscribed to emails
9240 non-null   uint8
 46  Last Activity_SMS Sent
9240 non-null   uint8
 47  Last Activity_Unreachable
9240 non-null   uint8
 48  Last Activity_Unsubscribed
9240 non-null   uint8
 49  Last Activity_View in browser link Clicked
9240 non-null   uint8
 50  Last Activity_Visited Booth in Tradeshow
9240 non-null   uint8
 51  What is your current occupation_Housewife
9240 non-null   uint8
 52  What is your current occupation_Other
9240 non-null   uint8
 53  What is your current occupation_Student
9240 non-null   uint8
 54  What is your current occupation_Unemployed
9240 non-null   uint8
 55  What is your current occupation_Working Professional
```

```
9240 non-null   uint8
 56  What matters most to you in choosing a course_Flexibility & Convenienc
e  9240 non-null   uint8
 57  What matters most to you in choosing a course_Other
9240 non-null   uint8
 58  Last Notable Activity_Email Bounced
9240 non-null   uint8
 59  Last Notable Activity_Email Link Clicked
9240 non-null   uint8
 60  Last Notable Activity_Email Marked Spam
9240 non-null   uint8
 61  Last Notable Activity_Email Opened
9240 non-null   uint8
 62  Last Notable Activity_Email Received
9240 non-null   uint8
 63  Last Notable Activity_Form Submitted on Website
9240 non-null   uint8
 64  Last Notable Activity_Had a Phone Conversation
9240 non-null   uint8
 65  Last Notable Activity_Modified
9240 non-null   uint8
 66  Last Notable Activity_Olark Chat Conversation
9240 non-null   uint8
 67  Last Notable Activity_Page Visited on Website
9240 non-null   uint8
 68  Last Notable Activity_Resubscribed to emails
9240 non-null   uint8
 69  Last Notable Activity_SMS Sent
9240 non-null   uint8
 70  Last Notable Activity_Unreachable
9240 non-null   uint8
 71  Last Notable Activity_Unsubscribed
9240 non-null   uint8
 72  Last Notable Activity_View in browser link Clicked
9240 non-null   uint8
dtypes: float64(2), int64(10), uint8(61)
memory usage: 1.4 MB
```

From above it states that all variables are numericals

Checking for Outliers

In [29]:

```
round(Leads_df.describe(percentiles=[0.15,0.35,0.55,0.75,0.95]),2)
```

Out[29]:

	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums
count	9240.00	9240.00	9240.00	9240.00	9240.00	9240.00	9240.00	9240.00	9240.0C
mean	0.08	0.00	0.39	3.39	487.70	2.33	0.00	0.00	0.0C
std	0.27	0.01	0.49	4.84	548.02	2.16	0.04	0.01	0.01
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0C
15%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0C
35%	0.00	0.00	0.00	2.00	98.00	1.50	0.00	0.00	0.0C
50%	0.00	0.00	0.00	3.00	248.00	2.00	0.00	0.00	0.0C
55%	0.00	0.00	0.00	3.00	305.00	2.00	0.00	0.00	0.0C
75%	0.00	0.00	1.00	5.00	936.00	3.00	0.00	0.00	0.0C

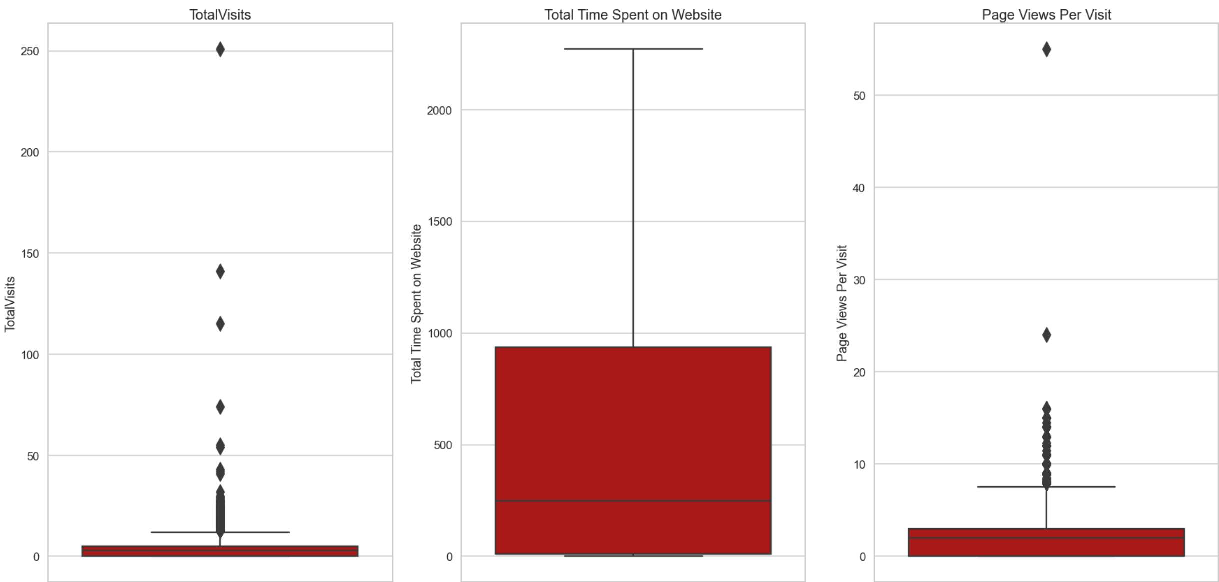
	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums
95%	1.00	0.00	1.00	10.00	1562.00	6.00	0.00	0.00	0.00
max	1.00	1.00	1.00	251.00	2272.00	55.00	1.00	1.00	1.00

11 rows × 73 columns

As we can see there are outliers in 2 variables **'TotalVisits' and 'Page Views Per Visit'**.

Let's visualize the outliers using boxplot to understand the outliers.

In [30]:
```
# Setting size of figure, context and gridlines

plt.figure(figsize=(30,50))
plt.tight_layout()
sns.set_style('whitegrid')
sns.set_context('talk')

# Title names for the columns in the dataset

col={0:'TotalVisits',1:'Total Time Spent on Website',2:'Page Views Per Visit'

# Visualising the outliers with boxplot for all the variables

for i in range(3):
    plt.subplot(3,3,i+1)
    plt.title(col[i],fontsize=20)
    sns.boxplot(y=Leads_df[col[i]],data=Leads_df,palette='gist_heat',fliersiz
```



From the above boxplots we can now confirm that we have two outlier variables in our dataset **('TotalVisits' and 'Page Views Per Visit')**. Now as per business requirement we cannot drop these outliers because it may impact our analysis/model so we will **create bins** for these two outliers.

From above, creating bins surely removed the outliers and hence we are now good to go. Before going to another step let's remove redundant columns/varaibles.

Data Preparation

Train-Test Split

In [31]:
```
# Importing train-test-split method from sklearn - model selection

from sklearn.model_selection import train_test_split
```

In [32]:
```
# Separating target varaible from dependent variable

y=Leads_df['Converted']      # putting target varaible 'Converted' to a new se

y.head()
```

Out[32]:
```
0    0
1    0
2    1
3    0
4    1
Name: Converted, dtype: int64
```

In [33]:
```
# Putting dependent variable in a new dataset called 'X'

X=Leads_df.drop('Converted',1)

X.head()
```

Out[33]:

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums	Newspaper	D Advertise
0	0	0	0.0	0	0.0	0	0	0	0	
1	0	0	5.0	674	2.5	0	0	0	0	
2	0	0	2.0	1532	2.0	0	0	0	0	
3	0	0	1.0	305	1.0	0	0	0	0	
4	0	0	2.0	1428	1.0	0	0	0	0	

5 rows × 72 columns

In [34]:
```
# Splitting the datset into train and test dataset

X_train,X_test,y_train,y_test=train_test_split(X, y, train_size=0.7, test_siz
```

Feature Standardization

In [35]:
```
# Importing Standard Scaler method from sklearn - preprocessing library

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()  # Creating an object
```

In [36]:
```
# Now, Scalling the 'Total Time Spent on Website' variables with standard sca
# and fitting - tranforming the X - train dataset

X_train[['TotalVisits', 'Page Views Per Visit', 'Total Time Spent on Website'

X_train.head()
```

Out[36]:

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums	Newspaper	A
1871	0	0	0.000000	0.000000	0.000000	0	0	0	0	
6795	0	0	0.015936	0.214349	0.024182	0	0	0	0	
3516	0	0	0.019920	0.046655	0.045455	0	0	0	0	
8105	0	0	0.019920	0.541373	0.090909	0	0	0	0	
3934	0	0	0.000000	0.000000	0.000000	0	0	0	0	

5 rows × 72 columns

In [37]:
```
## Checking the conversion rate from 'converted' column as it denotes the tar

(sum(y)/len(y.index))*100
```

Out[37]:
```
38.53896103896104
```

We have conversion rate of almost 39%

Correlation of the dataset

In [38]:
```
# setting the figure size

plt.figure(figsize=(20,15))

# setting the title

plt.title('Correlations')

# Plotting a heatmap

sns.heatmap(Leads_df.corr(method='spearman'))

plt.show()
```

From the above heatmap, we saw that there are two variables having high correlation, so we going to drop them.

Dropping highly correlated dummy variable/categories

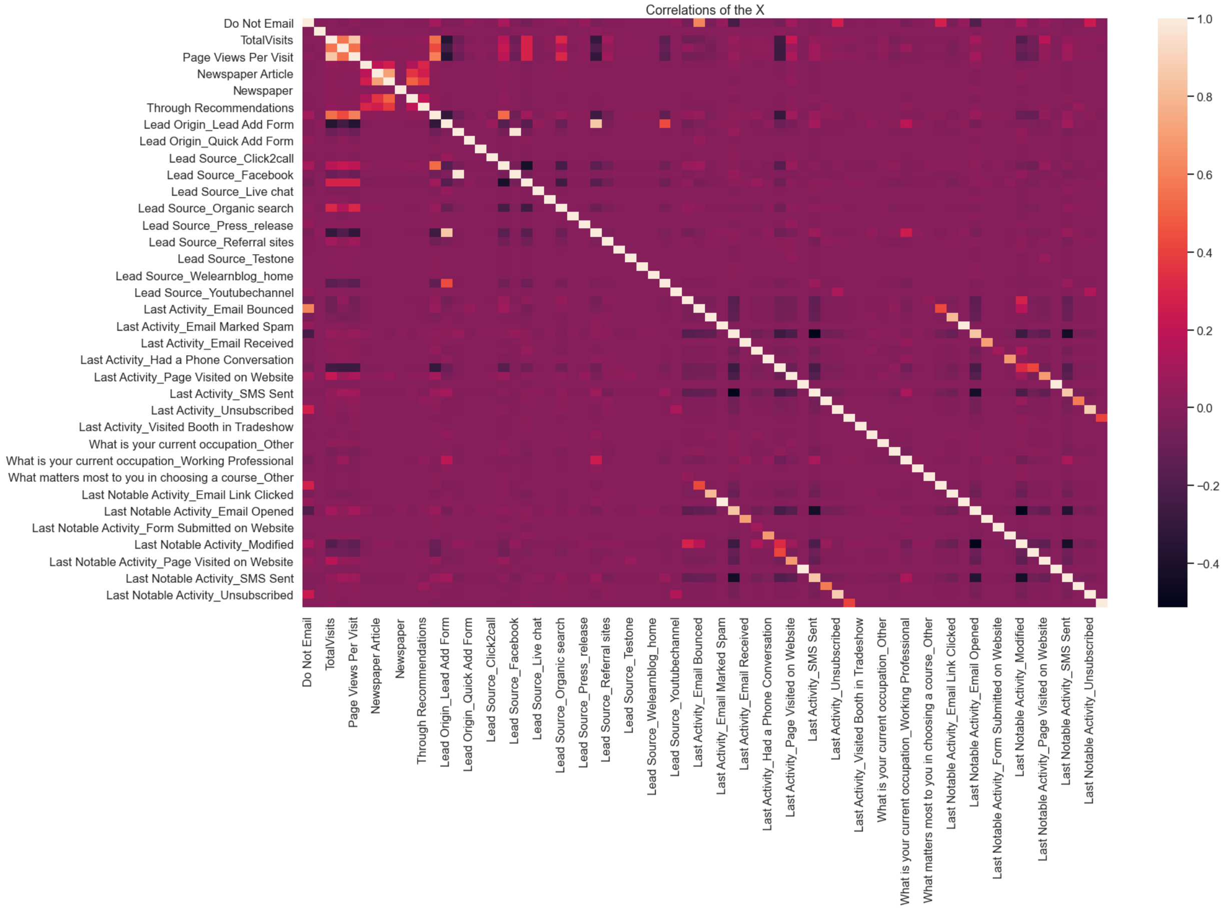
In [39]:

```
corr_dummy=['Lead Source_Olark chat','What is your current occupation_Unemplo
X_train=X_train.drop(corr_dummy,1)       # dropping from X train set
X_test=X_test.drop(corr_dummy,1)        # dropping from X test set
```

Checking again the correlation of the dataset

In [40]:

```
# setting the figure size
plt.figure(figsize=(25,15))
# setting the title
plt.title('Correlations of the X')
# Plotting a heatmap
sns.heatmap(Leads_df[X_train.columns].corr(method='spearman'))
plt.show()
```



Now, both of them are removed and new correlation is shown above by heatmap, We will now proceed with building our model based on the p-values and VIFs, we will again check for correlation as from above heatmap it is difficult to spot the highly correlated variables.

Building a Model

In [41]:

```
# Import 'LogisticRegression'
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
```

Using RFE

In [42]:

```
from sklearn.feature_selection import RFE
```

In [43]:

```
# Running rfe for 15 variables

rfem = RFE(logreg,15)

rfem = rfem.fit(X_train, y_train)    # fitting
```

In [44]:

```
rfem.support_   # checking for ture and false assigned to the variables after
```

Out[44]:

```
array([ True, False,  True,  True,  True, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False,  True, False, False,
```

```
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False,  True, False,
       False, False,  True, False,  True, False, False,  True,  True,
        True,  True, False, False, False, False, False])
```

In [45]:

```python
# Importing statsmodels
import statsmodels.api as sm
```

In [46]:

```python
# selecting columns only which are 'True' in rfem.support_ i.e True columns w

col=X_train.columns[rfem.support_]

X_train_1=sm.add_constant(X_train[col])     # Adding constant
```

In [47]:

```python
# creating 1st model after RFE

logis1=sm.GLM(y_train,X_train_1,family=sm.families.Binomial())

reg1=logis1.fit()

reg1.summary()
```

Out[47]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6468
Model:	GLM	**Df Residuals:**	6452
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2736.8
Date:	Wed, 11 Aug 2021	**Deviance:**	5473.6
Time:	19:31:15	**Pearson chi2:**	7.01e+03
No. Iterations:	21		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.3187	0.084	-3.811	0.000	-0.483	-0.155
Do Not Email	-1.6761	0.168	-9.987	0.000	-2.005	-1.347
TotalVisits	7.9919	2.284	3.500	0.000	3.516	12.468
Total Time Spent on Website	4.1453	0.151	27.512	0.000	3.850	4.441
Page Views Per Visit	-7.6765	1.178	-6.514	0.000	-9.986	-5.367
Lead Origin_Lead Add Form	3.2659	0.189	17.255	0.000	2.895	3.637
Lead Source_Welingak website	2.0179	0.746	2.704	0.007	0.555	3.481
Last Activity_Converted to Lead	-1.0583	0.221	-4.789	0.000	-1.491	-0.625
What is your current occupation_Housewife	22.7910	1.4e+04	0.002	0.999	-2.74e+04	2.74e+04
What is your current occupation_Working Professional	2.7980	0.187	14.976	0.000	2.432	3.164
Last Notable Activity_Email Link Clicked	-1.8064	0.275	-6.561	0.000	-2.346	-1.267
Last Notable Activity_Email Opened	-1.3301	0.086	-15.383	0.000	-1.500	-1.161

Last Notable Activity_Had a Phone Conversation	1.8839	1.105	1.705	0.088	-0.282	4.050
Last Notable Activity_Modified	-1.8961	0.093	-20.421	0.000	-2.078	-1.714
Last Notable Activity_Olark Chat Conversation	-2.3098	0.324	-7.139	0.000	-2.944	-1.676
Last Notable Activity_Page Visited on Website	-1.8445	0.201	-9.186	0.000	-2.238	-1.451

Now, From the above summary presented there are some features having high p -values, we will drop features which is having insignificant values one by one and create new model again and again until all the features attain significant p- value.

Calculating VIF

In [48]:

```
# importing VIFs library
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [49]:

```
# Creating vif dataframe
vif=pd.DataFrame()
# adding same features as the x_train dataset have
vif['Features']=X_train_1[col].columns
# Caculating VIFs
vif['VIF']=[variance_inflation_factor(X_train_1[col].values,i) for i in range
# Rounding the vif values
vif['VIF']=round(vif['VIF'],2)
# Sorting the vif values
vif=vif.sort_values(by='VIF',ascending=False)

vif   # Viewing the dataset
```

Out[49]:

	Features	VIF
3	Page Views Per Visit	2.76
1	TotalVisits	1.98
2	Total Time Spent on Website	1.82
12	Last Notable Activity_Modified	1.56
10	Last Notable Activity_Email Opened	1.41
4	Lead Origin_Lead Add Form	1.40
5	Lead Source_Welingak website	1.24
6	Last Activity_Converted to Lead	1.16
8	What is your current occupation_Working Profes...	1.16
14	Last Notable Activity_Page Visited on Website	1.14

	Features	VIF
0	Do Not Email	1.13
9	Last Notable Activity_Email Link Clicked	1.02
7	What is your current occupation_Housewife	1.01
13	Last Notable Activity_Olark Chat Conversation	1.01
11	Last Notable Activity_Had a Phone Conversation	1.00

As we can see that all features are having vif values less than 5, hence there is no multicollinearity issue in the dataset.

As expained before we will drop the highest in-significant features i.e **'What is your current occupation_Housewife'** having 0.999 p - value.

In [50]:

```
# Dropping the most insignificant values ('What is your current occupation_Ho

X_train_2=X_train_1.drop(['const','What is your current occupation_Housewife'
```

In [51]:

```
# Creating a new model

X_train_2=sm.add_constant(X_train_2)                                    # Adding co
logis2=sm.GLM(y_train,X_train_2,families=sm.families.Binomial())        # Using GLM
reg2=logis2.fit()                                                       # Fitting c
reg2.summary()                                                          # Showing t
```

Out[51]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6468
Model:	GLM	**Df Residuals:**	6453
Model Family:	Gaussian	**Df Model:**	14
Link Function:	identity	**Scale:**	0.14069
Method:	IRLS	**Log-Likelihood:**	-2827.8
Date:	Wed, 11 Aug 2021	**Deviance:**	907.90
Time:	19:31:15	**Pearson chi2:**	908.
No. Iterations:	3		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.4083	0.012	32.752	0.000	0.384	0.433
Do Not Email	-0.1877	0.018	-10.491	0.000	-0.223	-0.153
TotalVisits	1.0230	0.261	3.915	0.000	0.511	1.535
Total Time Spent on Website	0.7259	0.021	35.097	0.000	0.685	0.766
Page Views Per Visit	-0.9736	0.143	-6.790	0.000	-1.255	-0.693
Lead Origin_Lead Add Form	0.4950	0.020	24.308	0.000	0.455	0.535
Lead Source_Welingak website	0.1923	0.044	4.397	0.000	0.107	0.278
Last Activity_Converted to Lead	-0.1247	0.023	-5.317	0.000	-0.171	-0.079
What is your current occupation_Working Professional	0.3459	0.018	18.997	0.000	0.310	0.382

Last Notable Activity_Email Link Clicked	-0.2916	0.036	-8.145	0.000	-0.362	-0.221
Last Notable Activity_Email Opened	-0.2237	0.013	-17.428	0.000	-0.249	-0.199
Last Notable Activity_Had a Phone Conversation	0.2268	0.114	1.997	0.046	0.004	0.449
Last Notable Activity_Modified	-0.3054	0.013	-24.110	0.000	-0.330	-0.281
Last Notable Activity_Olark Chat Conversation	-0.3494	0.036	-9.739	0.000	-0.420	-0.279
Last Notable Activity_Page Visited on Website	-0.3034	0.027	-11.122	0.000	-0.357	-0.250

Now, from the above summary we can say that all the variables present in this model are **significant** as no variables is having p - value greater than 5% hence we can proceed with our next step

Creating VIF

After creating a model with no in significant features lets check the VIF i.e multicollinearity as we have checked earlier there was no such thing were found after creating VIF - all VIF vallues are less than 5 which means our **final model is ready**.

In [52]:

```
# Checking VIF again just to be sure

X_train_2_1=X_train_2.drop('const',1)      # dropping constant and saving in n
vif=pd.DataFrame()                         # Creating new VIF DataFrame
vif['Features']=X_train_2_1.columns        # Adding final train dataset featu

# Now calculating

vif['VIF']=[variance_inflation_factor(X_train_2_1.values,i) for i in range(X_

# Rounding the vif values

vif['VIF']=round(vif['VIF'],2)

# Sorting the vif dataset

vif=vif.sort_values(by='VIF',ascending=False)

vif   # viewing the dataset
```

Out[52]:

	Features	VIF
3	Page Views Per Visit	2.76
1	TotalVisits	1.98
2	Total Time Spent on Website	1.82
11	Last Notable Activity_Modified	1.56
9	Last Notable Activity_Email Opened	1.41
4	Lead Origin_Lead Add Form	1.40
5	Lead Source_Welingak website	1.24
6	Last Activity_Converted to Lead	1.16
7	What is your current occupation_Working Profes...	1.16
13	Last Notable Activity_Page Visited on Website	1.14
0	Do Not Email	1.13
8	Last Notable Activity_Email Link Clicked	1.02

	Features	VIF
12	Last Notable Activity_Olark Chat Conversation	1.01
10	Last Notable Activity_Had a Phone Conversation	1.00

As confirmed earlier, **no sign of multicollinearity** shown from above vif dataframe hence reg2 is our final model and we are going to use it predict the X train dataset.

Predicting a Train model

In [53]:
```
# Predicting the train dataset with our final model

y_train_pred=reg2.predict(X_train_2)

# Creating a new dataset and saving predicted values in it

y_train_pred_final=pd.DataFrame({'Converted':y_train.values,'Converted_probab
                                 'ID':y_train.index})

y_train_pred_final.head()    # viewing first 5 rows
```

Out[53]:

	Converted	Converted_probability	ID
1871	0	0.184626	1871
6795	0	0.332988	6795
3516	0	0.194618	3516
8105	0	0.733167	8105
3934	0	0.102850	3934

ROC Curve Plotting

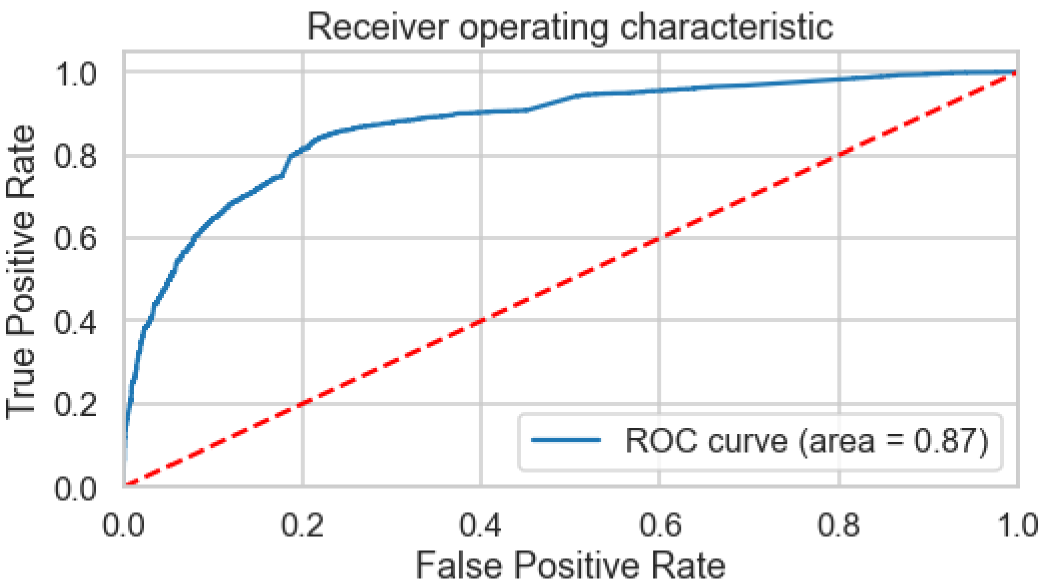
- ROC curve shows the trade off between sensitivity and specificity - means if sensitivity increases specificity will decrease.
- The curve closer to the left side border then right side of the border is more accurate.
- The curve closer to the 45-degree diagonal of the ROC space is less accurate.

In [54]:
```
# Importing necessary libraries for roc curve

from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# Creating a function to plot roc curve with auc score

def edu_roc( real, probability ):

    # Creating roc curve values like false positive rate , true positive rate

    fpr, tpr, thresholds = roc_curve( real, probability,drop_intermediate = F

    # Calculating the auc score(area under the curve)

    auc_score = roc_auc_score( real, probability )

    # Setting the figure size
```

```
    plt.figure(figsize=(8, 4))

    # Plotting the roc curve

    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )

    # Plotting the 45% dotted line
    plt.plot([0, 1], [0, 1], 'r--')

    # Setting the x axis linit

    plt.xlim([0.0, 1.0])

    # Setting the y axis limit

    plt.ylim([0.0, 1.05])

    # Setting the x axis label

    plt.xlabel('False Positive Rate')

    # Setting the y axis label

    plt.ylabel('True Positive Rate')

    # Setting the title

    plt.title('Receiver operating characteristic')

    # Setting the legend on the left below to show the value of auc

    plt.legend(loc="lower right")

    # Showing the plot

    plt.show()

    return None    # no return
```

In [55]:

```
# Calling the roc curve for plotting

edu_roc(y_train_pred_final.Converted, y_train_pred_final.Converted_probabilit
```



Points to be concluded from above roc curve -

- The curve is closer to the left side of the border than to the right side hence our model is having great accuracy.
- The area under the curve is 88% of the total area.

Finding optimal probability cutoff point

In [56]:

```
# creating 10 points out of which one we will choose for our cutoff point

numbers=[float(x)/10 for x in range(10)]  # from 0 to 0.9 with set size 0.1

for i in numbers:
    y_train_pred_final[i]=y_train_pred_final['Converted_probability'].map(lam
y_train_pred_final.head()   # Viewing the first 5 rows
```

Out[56]:

	Converted	Converted_probability	ID	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1871	0	0.184626	1871	1	1	0	0	0	0	0	0	0	0
6795	0	0.332988	6795	1	1	1	1	0	0	0	0	0	0
3516	0	0.194618	3516	1	1	0	0	0	0	0	0	0	0
8105	0	0.733167	8105	1	1	1	1	1	1	1	1	0	0
3934	0	0.102850	3934	1	1	0	0	0	0	0	0	0	0

Now, after creating series of points let's check the possibilities of choosing any one points from 0 to 0.9. We will do this by finding **'Accuracy'**, **'Sensitivity'** and **'Specificity'** for each points. These three methods will tell us how our model is - whether it is having low accuray or high or number of relevance data points is high or low etc.

In [57]:

```
# Caculating accuracy, sensitivity and specificity with probability cutoffs

# importing necessary library

from sklearn.metrics import confusion_matrix

# Creating a dataframe to store all the values to be created

df_cutoffs=pd.DataFrame(columns=['Probability','Accuracy','Sensitvity','Speci

# from 0 to 0.9 with set size 0.1

var=[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

for i in var:
    cm_matrix=confusion_matrix(y_train_pred_final['Converted'],y_train_pred_f
    total=sum(sum(cm_matrix))
    accuracy=(cm_matrix[0,0]+cm_matrix[1,1])/total
    sensitivity=cm_matrix[1,1]/(cm_matrix[1,0]+cm_matrix[1,1])
    specificity=cm_matrix[0,0]/(cm_matrix[0,0]+cm_matrix[0,1])
    df_cutoffs.loc[i]=[i, accuracy, sensitivity, specificity]
print(df_cutoffs)   # Printing the data
```

```
     Probability  Accuracy  Sensitvity  Specificity
0.0          0.0  0.426407    0.997161     0.074713
0.1          0.1  0.490260    0.983779     0.186157
0.2          0.2  0.708256    0.904298     0.587456
0.3          0.3  0.787106    0.865369     0.738881
```

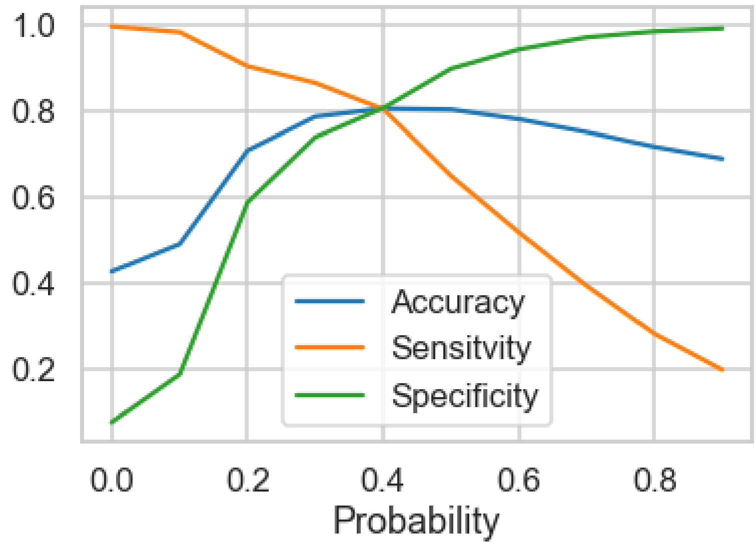
```
0.4          0.4  0.805968   0.804542    0.806847
0.5          0.5  0.803649   0.649635    0.898551
0.6          0.6  0.781231   0.517843    0.943528
0.7          0.7  0.751237   0.393350    0.971764
0.8          0.8  0.716759   0.280616    0.985507
0.9          0.9  0.689085   0.196675    0.992504
```

As we can see from the above data we have created points for accuracy , sensitivity and specificity for all probability points from 0 to 0.9. Out of this we have to choose one as a cutoff point and it is **probability cutoff = 0.4** because all the accuracy , sensitivity and specificity are having nearly same value which is an ideal point to consider for as we can't ignore any one from three.

Let's plot this data and see the convergent point or meeting point for all three point 'accuracy' , 'sensitivity' and 'specificity'

In [58]:

```
# Ploting 'Accuracy' , 'Sensitivity' and 'Specificity' for various probabilit

df_cutoffs.plot.line(x='Probability', y=['Accuracy','Sensitvity','Specificity
plt.show()
```



From the above curve, 0.4 is the optimum point for taking probability cutoff as the meeting point is slightly before from 0.4 hence final cutoff we choose is **0.40**. Also we can see that there is a trade off between sensitivity and specificity.

In [59]:

```
# Predicting the outcomes with probability cutoff as 0.4 by creating new colu

y_train_pred_final['Predicted']=y_train_pred_final['Converted_probability'].m

y_train_pred_final.head()
```

Out[59]:

	Converted	Converted_probability	ID	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Pred
1871	0	0.184626	1871	1	1	0	0	0	0	0	0	0	0	
6795	0	0.332988	6795	1	1	1	1	0	0	0	0	0	0	
3516	0	0.194618	3516	1	1	0	0	0	0	0	0	0	0	
8105	0	0.733167	8105	1	1	1	1	1	1	1	1	0	0	
3934	0	0.102850	3934	1	1	0	0	0	0	0	0	0	0	

Precision and Recall

Let's create precision and recall using confusion matrix for the final dataset ass we know that to attain more stability and predict successfully in our model one needs to check these two important methods which not only will tell us how our model is but also it will show us some insight like what is the score for result relevancy and how many truly relevant results are returned.

In [60]:

```
# Creating confusion matrix to find precision and recall score

confusion_pr=confusion_matrix(y_train_pred_final.Converted,y_train_pred_final
confusion_pr
```

Out[60]:

```
array([[3229,  773],
       [ 482, 1984]], dtype=int64)
```

In [61]:

```
print('Precision',confusion_pr[1,1]/(confusion_pr[0,1]+confusion_pr[1,1]))
print('Recall',confusion_pr[1,1]/(confusion_pr[1,0]+confusion_pr[1,1]))
```

```
Precision 0.7196227783822996
Recall 0.8045417680454177
```

Important point to be noted from the outcomes for precision and recall score -

- Our precison percentage is 72% approximately and recall percentage is 80%
- This means we have very good model which explains relevancy of 72% and true relevant results about 80%.

As per our business objective, the recall percentage I will consider more valuable because it is okay if our precision is little low which means less hot lead customers but we don't want to left out any hot leads which are willing to get converted hence our focus on this will be more on Recall than Precision.

Precision and Recall trade-off

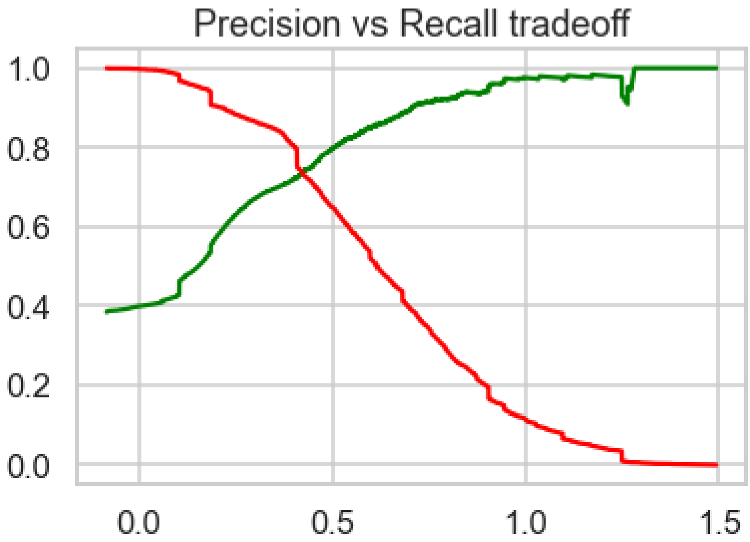
As we all know that Precision and Recall are inversely related means if one increases other will genuinely decrease. Hence we need to see the trade off between these two. Let's check that in below graph.

In [62]:

```
# importing precision recall curve from sklearn library

from sklearn.metrics import precision_recall_curve
```

In [63]:

```
# Creating precision recall curve by crreating three points and plotting

p ,r, thresholds=precision_recall_curve(y_train_pred_final.Converted,y_train_
plt.title('Precision vs Recall tradeoff')
plt.plot(thresholds, p[:-1], "g-")    # Plotting precision
plt.plot(thresholds, r[:-1], "r-")    # Plotting Recall
plt.show()
```



As we can see that there is a trade off between Precision and Recall and the meeting point is nearly at 0.5

Prediction the test dataset

Scaling the test dataset

In [64]:
```
# Scalling the variables 'Total Time Spent on Website' with standard scaler a
X_test[['TotalVisits', 'Page Views Per Visit', 'Total Time Spent on Website']
    'TotalVisits','Page Views Per Visit','Total Time Spent on Website']])
```

Now Predicting

In [65]:
```
# Predicting the test dataset with our final model

test_cols=X_train_2.columns[1:]             # Taking the same column train s
X_test_final=X_test[test_cols]              # Updating it in the final test
X_test_final=sm.add_constant(X_test_final)  # Adding constant to the final s
y_pred_test=reg2.predict(X_test_final)      # Predicting the final test set
```

In [66]:
```
# Creating a new dataset and saving the prediction values in it

y_test_pred_final=pd.DataFrame({'Converted':y_test.values,'Converted_Probabil

y_test_pred_final.head()   # viewing first 5 rows
```

Out[66]:

	Converted	Converted_Probability	ID
4269	1	0.398803	4269
2376	1	0.903256	2376
7766	1	0.563633	7766
9199	0	0.102850	9199
4359	1	0.679584	4359

Model Evaluation

In [67]:

```
# Predicting the outcomes with probability cutoff as 0.4 by creating new colu
y_test_pred_final['Predicted']=y_test_pred_final['Converted_Probability'].map
y_test_pred_final.head()
```

Out[67]:

	Converted	Converted_Probability	ID	Predicted
4269	1	0.398803	4269	0
2376	1	0.903256	2376	1
7766	1	0.563633	7766	1
9199	0	0.102850	9199	0
4359	1	0.679584	4359	1

In [68]:

```
# Checking the accuracy of the test dataset.
from sklearn import metrics    # Importing metrics from sklearn
print('Accuracy score in predicting test dataset :',metrics.accuracy_score(y_
```

```
Accuracy score in predicting test dataset : 0.8051948051948052
```

In [69]:

```
from sklearn.metrics import precision_score, recall_score     # Importing pre
print('Precision score in predicting test dataset:',precision_score(y_test_pr
print('Recall score in predicting test dataset:',recall_score(y_test_pred_fin
```

```
Precision score in predicting test dataset: 0.7766699900299102
Recall score in predicting test dataset: 0.7114155251141553
```

Lead Score assigning

In [70]:

```
# Creating new columns for lead number and lead score
y_test_pred_final['Lead Number']=Leads_df.iloc[y_test_pred_final['ID'],1]
y_test_pred_final['Lead Score']=y_test_pred_final['Converted_Probability'].ap

y_test_pred_final.head()
```

Out[70]:

	Converted	Converted_Probability	ID	Predicted	Lead Number	Lead Score
4269	1	0.398803	4269	0	0	40
2376	1	0.903256	2376	1	0	90
7766	1	0.563633	7766	1	0	56
9199	0	0.102850	9199	0	0	10
4359	1	0.679584	4359	1	0	68

Conclusion

Valuable Insights -

- The Accuracy, Precision and Recall score we got from test set in aceptable range.
- We have high recall score than precision score which we were exactly looking for.
- In business terms, this model has an ability to adjust with the company’s requirements in coming future.
- This concludes that the model is in stable state.
- Important features responsible for good conversion rate or the ones' which contributes more towards the probability of a lead getting converted are :
  - **Total Visits**
  - **Page Views Per Visit**
  - **Total Time Spent on Website** and
  - **Lead Origin_Lead Add Form**