In [72]:
```
# Supress unnecessary warnings

import warnings
warnings.filterwarnings('ignore')
```

In [73]:
```
# Import the NumPy and Pandas packages

import numpy as np
import pandas as pd
```

In [74]:
```
# Read the dataset

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

In [75]:
```
# Look at the first few entries

leads.head()
```

Out[75]:

	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

5 rows × 37 columns

In [76]:
```
# Inspect the shape of the dataset

leads.shape
```

Out[76]: (9240, 37)

In [77]:
```
# Inspect the different columsn in the dataset

leads.columns
```

Out[77]:
```
Index(['Prospect ID', 'Lead Number', 'Lead Origin', 'Lead Source',
       'Do Not Email', 'Do Not Call', 'Converted', 'TotalVisits',
       'Total Time Spent on Website', 'Page Views Per Visit', 'Last Activi
ty',
       'Country', 'Specialization', 'How did you hear about X Education',
       'What is your current occupation',
       'What matters most to you in choosing a course', 'Search', 'Magazin
e',
       'Newspaper Article', 'X Education Forums', 'Newspaper',
       'Digital Advertisement', 'Through Recommendations',
       'Receive More Updates About Our Courses', 'Tags', 'Lead Quality',
       'Update me on Supply Chain Content', 'Get updates on DM Content',
       'Lead Profile', 'City', 'Asymmetrique Activity Index',
       'Asymmetrique Profile Index', 'Asymmetrique Activity Score',
       'Asymmetrique Profile Score',
       'I agree to pay the amount through cheque',
       'A free copy of Mastering The Interview', 'Last Notable Activity'],
      dtype='object')
```

As you can see, the feature variables are quite intuitive. If you don't understand them completely, please refer to the data dictionary.

In [78]:
```
# Check the summary of the dataset

leads.describe()
```

Out[78]:

	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Score	A
count	9240.000000	9240.000000	9103.000000	9240.000000	9103.000000	5022.000000	
mean	617188.435606	0.385390	3.445238	487.698268	2.362820	14.306252	
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	
25%	596484.500000	0.000000	1.000000	12.000000	1.000000	14.000000	
50%	615479.000000	0.000000	3.000000	248.000000	2.000000	14.000000	
75%	637387.250000	1.000000	5.000000	936.000000	3.000000	15.000000	
max	660737.000000	1.000000	251.000000	2272.000000	55.000000	18.000000	

In [79]:
```
# Check the info to see the types of the feature variables and the null val

leads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 37 columns):
Prospect ID                                      9240 non-null object
Lead Number                                      9240 non-null int64
Lead Origin                                      9240 non-null object
Lead Source                                      9204 non-null object
Do Not Email                                     9240 non-null object
Do Not Call                                      9240 non-null object
Converted                                        9240 non-null int64
TotalVisits                                      9103 non-null float64
Total Time Spent on Website                      9240 non-null int64
Page Views Per Visit                             9103 non-null float64
Last Activity                                    9137 non-null object
Country                                          6779 non-null object
Specialization                                   7802 non-null object
How did you hear about X Education               7033 non-null object
What is your current occupation                  6550 non-null object
What matters most to you in choosing a course    6531 non-null object
```

Looks like there are quite a few categorical variables present in this dataset for which we will need to create dummy variables. Also, there are a lot of null values present as well, so we will need to treat them accordingly.

Step 1: Data Cleaning and Preparation

In [80]:
```
# Check the number of missing values in each column

leads.isnull().sum()
```

Out[80]:
```
Prospect ID                                         0
Lead Number                                         0
Lead Origin                                         0
Lead Source                                        36
Do Not Email                                        0
Do Not Call                                         0
Converted                                           0
TotalVisits                                       137
Total Time Spent on Website                         0
Page Views Per Visit                              137
Last Activity                                     103
Country                                          2461
Specialization                                   1438
How did you hear about X Education               2207
What is your current occupation                  2690
What matters most to you in choosing a course    2709
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
Tags                                             3353
Lead Quality                                     4767
Update me on Supply Chain Content                   0
Get updates on DM Content                           0
Lead Profile                                     2709
City                                             1420
Asymmetrique Activity Index                      4218
Asymmetrique Profile Index                       4218
Asymmetrique Activity Score                      4218
Asymmetrique Profile Score                       4218
I agree to pay the amount through cheque            0
A free copy of Mastering The Interview              0
Last Notable Activity                               0
dtype: int64
```

As you can see there are a lot of columnw which have high number of missing values. Clearly, these columns are not useful. Since, there are 9000 datapoints in our dataframe, let's eliminate the columns having greater than 3000 missing values as they are of no use to us.

In [81]:
```
# Drop all the columns in which greater than 3000 missing values are presen

for col in leads.columns:
    if leads[col].isnull().sum() > 3000:
        leads.drop(col, 1, inplace=True)
```

In [82]:
```
# Check the number of null values again

leads.isnull().sum()
```

Out[82]:
```
Prospect ID                                         0
Lead Number                                         0
Lead Origin                                         0
Lead Source                                        36
Do Not Email                                        0
Do Not Call                                         0
Converted                                           0
TotalVisits                                       137
Total Time Spent on Website                         0
Page Views Per Visit                              137
Last Activity                                     103
Country                                          2461
Specialization                                   1438
How did you hear about X Education               2207
What is your current occupation                  2690
What matters most to you in choosing a course    2709
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
Lead Profile                                     2709
City                                             1420
I agree to pay the amount through cheque            0
A free copy of Mastering The Interview              0
Last Notable Activity                               0
dtype: int64
```

As you might be able to interpret, the variable `City` won't be of any use in our analysis. So it's best that we drop it.

In [83]:
```
leads.drop(['City'], axis = 1, inplace = True)
```

In [84]:
```
# Same goes for the variable 'Country'

leads.drop(['Country'], axis = 1, inplace = True)
```

In [85]:
```
# Let's now check the percentage of missing values in each column

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

Out[85]:
```
Prospect ID                                       0.00
Lead Number                                       0.00
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                                   15.56
How did you hear about X Education               23.89
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
Lead Profile                                     29.32
I agree to pay the amount through cheque          0.00
A free copy of Mastering The Interview            0.00
Last Notable Activity                             0.00
dtype: float64
```

In [86]:
```
# Check the number of null values again

leads.isnull().sum()
```

Out[86]:
```
Prospect ID                                         0
Lead Number                                         0
Lead Origin                                         0
Lead Source                                        36
Do Not Email                                        0
Do Not Call                                         0
Converted                                           0
TotalVisits                                       137
Total Time Spent on Website                         0
Page Views Per Visit                              137
Last Activity                                     103
Specialization                                   1438
How did you hear about X Education               2207
What is your current occupation                  2690
What matters most to you in choosing a course    2709
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
Lead Profile                                     2709
I agree to pay the amount through cheque            0
A free copy of Mastering The Interview              0
Last Notable Activity                               0
dtype: int64
```

Now recall that there are a few columns in which there is a level called 'Select' which basically means that the student had not selected the option for that particular column which is why it shows 'Select'. These values are as good as missing values and hence we need to identify the value counts of the level 'Select' in all the columns that it is present.

In [87]:
```
# Get the value counts of all the columns

for column in leads:
    print(leads[column].astype('category').value_counts())
    print('___________________________________________________')
```

```
fffb0e5e-9f92-4017-9f42-781a69da4154    1
56453aec-3f7b-4f30-870c-8f966d393100    1
53ac14bd-2bb2-4315-a21c-94562d1b6b2d    1
53aabd84-5dcc-4299-bbe3-62f3764b07b1    1
539ffa32-1be7-4fe1-b04c-faf1bab763cf    1
539eb309-df36-4a89-ac58-6d3651393910    1
5398e7ff-74db-4074-89fb-4fd9a603f521    1
53953744-234a-4cb9-9af4-bcc47eb472f4    1
539366d9-f633-455a-99e4-dbc5907db28e    1
5390c5fe-b12c-4f6e-ae92-908672abb0a1    1
5379ee79-64b7-44f8-8c56-0e1ca2d5b887    1
537963c8-22d9-459d-8aae-ddac40580ffb    1
53744d5a-0483-42c0-80b0-8990a4d2356d    1
53715ab1-2106-4c4e-8493-81cc465eb9ce    1
536cdc6b-f4c1-449d-bfd8-9ef0ac912dbb    1
53690d88-52f0-4ce5-b6b8-a13570a6db35    1
5363bd79-576c-48ed-83e4-024c81ea00c5    1
53c4e210-3344-4737-813f-74ef9a747ab6    1
53dbb914-71e7-458a-9749-cfb4d655eac2    1
```

The following three columns now have the level 'Select'. Let's check them once again.

In [88]:
```
leads['Lead Profile'].astype('category').value_counts()
```

Out[88]:
```
Select                         4146
Potential Lead                 1613
Other Leads                     487
Student of SomeSchool           241
Lateral Student                  24
Dual Specialization Student      20
Name: Lead Profile, dtype: int64
```

In [89]:
```
leads['How did you hear about X Education'].value_counts()
```

Out[89]:
```
Select                   5043
Online Search             808
Word Of Mouth             348
Student of SomeSchool     310
Other                     186
Multiple Sources          152
Advertisements             70
Social Media               67
Email                      26
SMS                        23
Name: How did you hear about X Education, dtype: int64
```

In [90]:
```
leads['Specialization'].value_counts()
```

Out[90]:
```
Select                               1942
Finance Management                    976
Human Resource Management             848
Marketing Management                  838
Operations Management                 503
Business Administration               403
IT Projects Management                366
Supply Chain Management               349
Banking, Investment And Insurance     338
Media and Advertising                 203
Travel and Tourism                    203
International Business                178
Healthcare Management                 159
Hospitality Management                114
E-COMMERCE                            112
Retail Management                     100
Rural and Agribusiness                 73
E-Business                             57
Services Excellence                    40
Name: Specialization, dtype: int64
```

Clearly the levels `Lead Profile` and `How did you hear about X Education` have a lot of rows which have the value `Select` which is of no use to the analysis so it's best that we drop them.

In [91]:
```
leads.drop(['Lead Profile', 'How did you hear about X Education'], axis = 1
```

Also notice that when you got the value counts of all the columns, there were a few columns in which only one value was majorly present for all the data points. These include `Do Not Call` , `Search` , `Magazine` , `Newspaper Article` , `X Education Forums` , `Newspaper` , `Digital Advertisement` , `Through Recommendations` , `Receive More Updates About Our Courses` , `Update me on Supply Chain Content` , `Get updates on DM Content` , `I agree to pay the amount through cheque` . Since practically all of the values for these variables are `No` , it's best that we drop these columns as they won't help with our analysis.

In [92]:
```
leads.drop(['Do Not Call', 'Search', 'Magazine', 'Newspaper Article', 'X Ed
            'Digital Advertisement', 'Through Recommendations', 'Receive Mo
            'Update me on Supply Chain Content', 'Get updates on DM Content
            'I agree to pay the amount through cheque'], axis = 1, inplace
```

Also, the variable `What matters most to you in choosing a course` has the level `Better Career Prospects` `6528` times while the other two levels appear once twice and once respectively. So we should drop this column as well.

In [93]:
```
leads['What matters most to you in choosing a course'].value_counts()
```

Out[93]:
```
Better Career Prospects      6528
Flexibility & Convenience       2
Other                           1
Name: What matters most to you in choosing a course, dtype: int64
```

In [94]:
```
# Drop the null value rows present in the variable 'What matters most to yo

leads.drop(['What matters most to you in choosing a course'], axis = 1, inp
```

In [95]:
```
# Check the number of null values again

leads.isnull().sum()
```

Out[95]:
```
Prospect ID                                  0
Lead Number                                  0
Lead Origin                                  0
Lead Source                                 36
Do Not Email                                 0
Converted                                    0
TotalVisits                                137
Total Time Spent on Website                  0
Page Views Per Visit                       137
Last Activity                              103
Specialization                            1438
What is your current occupation           2690
A free copy of Mastering The Interview       0
Last Notable Activity                        0
dtype: int64
```

Now, there's the column `What is your current occupation` which has a lot of null values. Now you can drop the entire row but since we have already lost so many feature variables, we choose not to drop it as it might turn out to be significant in the analysis. So let's just drop the null rows for the column `What is you current occupation` .

In [96]:
```
leads = leads[~pd.isnull(leads['What is your current occupation'])]
```

In [97]:
```
# Check the number of null values again

leads.isnull().sum()
```

Out[97]:
```
Prospect ID                                 0
Lead Number                                 0
Lead Origin                                 0
Lead Source                                36
Do Not Email                                0
Converted                                   0
TotalVisits                               130
Total Time Spent on Website                 0
Page Views Per Visit                      130
Last Activity                             103
Specialization                             18
What is your current occupation             0
A free copy of Mastering The Interview      0
Last Notable Activity                       0
dtype: int64
```

Since now the number of null values present in the columns are quite small we can simply drop the rows in which these null values are present.

In [98]:
```
# Drop the null value rows in the column 'TotalVisits'

leads = leads[~pd.isnull(leads['TotalVisits'])]
```

In [99]:
```
# Check the null values again

leads.isnull().sum()
```

Out[99]:
```
Prospect ID                                 0
Lead Number                                 0
Lead Origin                                 0
Lead Source                                29
Do Not Email                                0
Converted                                   0
TotalVisits                                 0
Total Time Spent on Website                 0
Page Views Per Visit                        0
Last Activity                               0
Specialization                             18
What is your current occupation             0
A free copy of Mastering The Interview      0
Last Notable Activity                       0
dtype: int64
```

In [100]:
```
# Drop the null values rows in the column 'Lead Source'

leads = leads[~pd.isnull(leads['Lead Source'])]
```

In [101]:
```
# Check the number of null values again

leads.isnull().sum()
```

Out[101]:
```
Prospect ID                                 0
Lead Number                                 0
Lead Origin                                 0
Lead Source                                 0
Do Not Email                                0
Converted                                   0
TotalVisits                                 0
Total Time Spent on Website                 0
Page Views Per Visit                        0
Last Activity                               0
Specialization                             18
What is your current occupation             0
A free copy of Mastering The Interview      0
Last Notable Activity                       0
dtype: int64
```

In [102]:
```
# Drop the null values rows in the column 'Specialization'

leads = leads[~pd.isnull(leads['Specialization'])]
```

In [103]:
```
# Check the the number of null values again

leads.isnull().sum()
```

Out[103]:
```
Prospect ID                               0
Lead Number                               0
Lead Origin                               0
Lead Source                               0
Do Not Email                              0
Converted                                 0
TotalVisits                               0
Total Time Spent on Website               0
Page Views Per Visit                      0
Last Activity                             0
Specialization                            0
What is your current occupation           0
A free copy of Mastering The Interview    0
Last Notable Activity                     0
dtype: int64
```

Now your data doesn't have any null values. Let's now check the percentage of rows that we have retained.

In [104]:
```
print(len(leads.index))
print(len(leads.index)/9240)
```

```
6373
0.6897186147186147
```

We still have around 69% of the rows which seems good enough.

In [105]:
```
# Let's look at the dataset again

leads.head()
```

Out[105]:

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Converted	TotalVisits	Total Time Spent on Website	Page Views Pe Visi
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	0	0.0	0	0.(
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	0	5.0	674	2.!
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	1	2.0	1532	2.(
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	0	1.0	305	1.(
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	1	2.0	1428	1.(

Now, clearly the variables `Prospect ID` and `Lead Number` won't be of any use in the analysis, so it's best that we drop these two variables.

In [106]:
```
leads.drop(['Prospect ID', 'Lead Number'], 1, inplace = True)
```

In [107]:
```
leads.head()
```

Out[107]:

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

Dummy variable creation

The next step is to deal with the categorical variables present in the dataset. So first take a look at which variables are actually categorical variables.

In [108]:
```
# Check the columns which are of type 'object'

temp = leads.loc[:, leads.dtypes == 'object']
temp.columns
```

Out[108]:
```
Index(['Lead Origin', 'Lead Source', 'Do Not Email', 'Last Activity',
       'Specialization', 'What is your current occupation',
       'A free copy of Mastering The Interview', 'Last Notable Activity'],
      dtype='object')
```

In [109]:
```
# Create dummy variables using the 'get_dummies' command
dummy = pd.get_dummies(leads[['Lead Origin', 'Lead Source', 'Do Not Email',
                              'What is your current occupation','A free cop
                              'Last Notable Activity']], drop_first=True)

# Add the results to the master dataframe
leads = pd.concat([leads, dummy], axis=1)
```

In [110]:
```
# Creating dummy variable separately for the variable 'Specialization' sinc
# drop that level by specifying it explicitly

dummy_spl = pd.get_dummies(leads['Specialization'], prefix = 'Specializatio
dummy_spl = dummy_spl.drop(['Specialization_Select'], 1)
leads = pd.concat([leads, dummy_spl], axis = 1)
```

In [111]:
```
# Drop the variables for which the dummy variables have been created

leads = leads.drop(['Lead Origin', 'Lead Source', 'Do Not Email', 'Last Act
                    'Specialization', 'What is your current occupation',
                    'A free copy of Mastering The Interview', 'Last Notable
```

In [112]:
```
# Let's take a look at the dataset again

leads.head()
```

Out[112]:

	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Source
0	0	0.0	0	0.0	0	0	0	
1	0	5.0	674	2.5	0	0	0	
2	1	2.0	1532	2.0	1	0	0	
3	0	1.0	305	1.0	1	0	0	
4	1	2.0	1428	1.0	1	0	0	

5 rows × 75 columns

Test-Train Split

The next step is to split the dataset into training an testing sets.

In [113]:
```
# Import the required library

from sklearn.model_selection import train_test_split
```

In [114]:
```
# Put all the feature variables in X

X = leads.drop(['Converted'], 1)
X.head()
```

Out[114]:

	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Lead Source_Direct Traffic	Sou
0	0.0	0	0.0	0	0	0	0	
1	5.0	674	2.5	0	0	0	0	
2	2.0	1532	2.0	1	0	0	1	
3	1.0	305	1.0	1	0	0	1	
4	2.0	1428	1.0	1	0	0	0	

5 rows × 74 columns

In [115]:
```
# Put the target variable in y

y = leads['Converted']

y.head()
```

Out[115]:
```
0    0
1    0
2    1
3    0
4    1
Name: Converted, dtype: int64
```

In [116]:
```
# Split the dataset into 70% train and 30% test

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

Scaling

Now there are a few numeric variables present in the dataset which have different scales. So let's go ahead and scale these variables.

In [117]:
```
# Import MinMax scaler

from sklearn.preprocessing import MinMaxScaler
```

In [118]:
```
# Scale the three numeric features present in the dataset

scaler = MinMaxScaler()

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

X_train.head()
```

Out[118]:

	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Lead Source_Direct Traffic
8003	0.015936	0.029489	0.125	1	0	0	1
218	0.015936	0.082306	0.250	1	0	0	1
4171	0.023904	0.034331	0.375	1	0	0	1
4037	0.000000	0.000000	0.000	0	0	0	0
3660	0.000000	0.000000	0.000	0	1	0	0

5 rows × 74 columns

Looking at the correlations

Let's now look at the correlations. Since the number of variables are pretty high, it's better that we look at the table instead of plotting a heatmap

In [119]:
```python
# Looking at the correlation table

leads.corr()
```

Specialization_Marketing Management	0.049520	-0.000493	0.052437	0.017799	0.084975
Specialization_Media and Advertising	-0.000862	0.038725	0.043356	0.063772	0.093730
Specialization_Operations Management	0.031349	0.008929	0.050860	0.030364	0.095849
Specialization_Retail Management	-0.018603	0.014223	0.024919	0.026099	0.070983
Specialization_Rural and Agribusiness	0.006964	0.068015	0.018767	0.027465	0.050077
Specialization_Services Excellence	-0.005142	0.015114	0.003203	0.015230	0.039433
Specialization_Supply Chain Management	0.005785	0.063383	0.045386	0.052972	0.111610
Specialization_Travel and Tourism	-0.011762	0.064384	0.037867	0.111284	0.094875

Step 2: Model Building

Let's now move to model building. As you can see that there are a lot of variables present in the dataset which we cannot deal with. So the best way to approach this is to select a small set of features from this pool of variables using RFE.

In [120]:
```python
# Import 'LogisticRegression' and create a LogisticRegression object

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [121]:
```python
# Import RFE and select 15 variables

from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 15 variables as output
rfe = rfe.fit(X_train, y_train)
```

In [122]:
```
# Let's take a look at which features have been selected by RFE

list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

Out[122]:
```
[('TotalVisits', True, 1),
 ('Total Time Spent on Website', True, 1),
 ('Page Views Per Visit', False, 7),
 ('Lead Origin_Landing Page Submission', False, 11),
 ('Lead Origin_Lead Add Form', True, 1),
 ('Lead Origin_Lead Import', False, 56),
 ('Lead Source_Direct Traffic', False, 23),
 ('Lead Source_Facebook', False, 51),
 ('Lead Source_Google', False, 36),
 ('Lead Source_Live Chat', False, 42),
 ('Lead Source_Olark Chat', True, 1),
 ('Lead Source_Organic Search', False, 35),
 ('Lead Source_Pay per Click Ads', False, 41),
 ('Lead Source_Press_Release', False, 52),
 ('Lead Source_Reference', True, 1),
 ('Lead Source_Referral Sites', False, 37),
 ('Lead Source_Social Media', False, 57),
 ('Lead Source_WeLearn', False, 38),
 ('Lead Source_Welingak Website', True, 1),
```

In [123]:
```
# Put all the columns selected by RFE in the variable 'col'

col = X_train.columns[rfe.support_]
```

Now you have all the variables selected by RFE and since we care about the statistics part, i.e. the p-values and the VIFs, let's use these variables to create a logistic regression model using statsmodels.

In [124]:
```
# Select only the columns selected by RFE

X_train = X_train[col]
```

In [125]:
```
# Import statsmodels

import statsmodels.api as sm
```

In [126]:
```
# Fit a logistic Regression model on X_train after adding a constant and ou

X_train_sm = sm.add_constant(X_train)
logm2 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[126]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4445
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2072.8
Date:	Fri, 08 Nov 2019	**Deviance:**	4145.5
Time:	12:24:30	**Pearson chi2:**	4.84e+03
No. Iterations:	22	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.0061	0.600	-1.677	0.094	-2.182	0.170
TotalVisits	11.3439	2.682	4.230	0.000	6.088	16.600
Total Time Spent on Website	4.4312	0.185	23.924	0.000	4.068	4.794
Lead Origin_Lead Add Form	2.9483	1.191	2.475	0.013	0.614	5.283
Lead Source_Olark Chat	1.4584	0.122	11.962	0.000	1.219	1.697
Lead Source_Reference	1.2994	1.214	1.070	0.285	-1.080	3.679
Lead Source_Welingak Website	3.4159	1.558	2.192	0.028	0.362	6.470
Do Not Email_Yes	-1.5053	0.193	-7.781	0.000	-1.884	-1.126
Last Activity_Had a Phone Conversation	1.0397	0.983	1.058	0.290	-0.887	2.966
Last Activity_SMS Sent	1.1827	0.082	14.362	0.000	1.021	1.344
What is your current occupation_Housewife	22.6492	2.45e+04	0.001	0.999	-4.8e+04	4.8e+04
What is your current occupation_Student	-1.1544	0.630	-1.831	0.067	-2.390	0.081
What is your current occupation_Unemployed	-1.3395	0.594	-2.254	0.024	-2.505	-0.175
What is your current occupation_Working Professional	1.2743	0.623	2.045	0.041	0.053	2.496
Last Notable Activity_Had a Phone Conversation	23.1932	2.08e+04	0.001	0.999	-4.08e+04	4.08e+04
Last Notable Activity_Unreachable	2.7868	0.807	3.453	0.001	1.205	4.369

There are quite a few variable which have a p-value greater than `0.05` . We will need to take care of them. But first, let's also look at the VIFs.

In [127]:
```
# Import 'variance_inflation_factor'

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [128]:
```
# Make a VIF dataframe for all the variables present

vif = pd.DataFrame()
vif['Features'] = X_train.columns
vif['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[128]:

	Features	VIF
2	Lead Origin_Lead Add Form	84.19
4	Lead Source_Reference	65.18
5	Lead Source_Welingak Website	20.03
11	What is your current occupation_Unemployed	3.65
7	Last Activity_Had a Phone Conversation	2.44
13	Last Notable Activity_Had a Phone Conversation	2.43
1	Total Time Spent on Website	2.38
0	TotalVisits	1.62
8	Last Activity_SMS Sent	1.59
12	What is your current occupation_Working Profes...	1.56
3	Lead Source_Olark Chat	1.44
6	Do Not Email_Yes	1.09

VIFs seem to be in a decent range except for three variables.

Let's first drop the variable `Lead Source_Reference` since it has a high p-value as well as a high VIF.

In [129]:
```
X_train.drop('Lead Source_Reference', axis = 1, inplace = True)
```

In [130]:
```
# Refit the model with the new set of features

logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Bin
logm1.fit().summary()
```

Out[130]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4446
Model Family:	Binomial	**Df Model:**	14
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2073.2
Date:	Fri, 08 Nov 2019	**Deviance:**	4146.5
Time:	12:24:31	**Pearson chi2:**	4.82e+03
No. Iterations:	22	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.0057	0.600	-1.677	0.094	-2.181	0.170
TotalVisits	11.3428	2.682	4.229	0.000	6.086	16.599
Total Time Spent on Website	4.4312	0.185	23.924	0.000	4.068	4.794
Lead Origin_Lead Add Form	4.2084	0.259	16.277	0.000	3.702	4.715
Lead Source_Olark Chat	1.4583	0.122	11.960	0.000	1.219	1.697
Lead Source_Welingak Website	2.1557	1.037	2.079	0.038	0.124	4.188
Do Not Email_Yes	-1.5036	0.193	-7.779	0.000	-1.882	-1.125
Last Activity_Had a Phone Conversation	1.0398	0.983	1.058	0.290	-0.887	2.966
Last Activity_SMS Sent	1.1827	0.082	14.362	0.000	1.021	1.344
What is your current occupation_Housewife	22.6511	2.45e+04	0.001	0.999	-4.8e+04	4.8e+04
What is your current occupation_Student	-1.1537	0.630	-1.830	0.067	-2.389	0.082
What is your current occupation_Unemployed	-1.3401	0.594	-2.255	0.024	-2.505	-0.175
What is your current occupation_Working Professional	1.2748	0.623	2.046	0.041	0.053	2.496
Last Notable Activity_Had a Phone Conversation	23.1934	2.08e+04	0.001	0.999	-4.08e+04	4.08e+04
Last Notable Activity_Unreachable	2.7872	0.807	3.454	0.001	1.205	4.369

The variable `Lead Profile_Dual Specialization Student` also needs to be dropped.

In [131]:
```
# Make a VIF dataframe for all the variables present

vif = pd.DataFrame()
vif['Features'] = X_train.columns
vif['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[131]:

	Features	VIF
10	What is your current occupation_Unemployed	3.65
6	Last Activity_Had a Phone Conversation	2.44
12	Last Notable Activity_Had a Phone Conversation	2.43
1	Total Time Spent on Website	2.38
2	Lead Origin_Lead Add Form	1.71
0	TotalVisits	1.62
7	Last Activity_SMS Sent	1.59
11	What is your current occupation_Working Profes...	1.56
3	Lead Source_Olark Chat	1.44
4	Lead Source_Welingak Website	1.33
5	Do Not Email_Yes	1.09
9	What is your current occupation_Student	1.09

The VIFs are now all less than 5. So let's drop the ones with the high p-values beginning with `Last Notable Activity_Had a Phone Conversation` .

In [132]:
```
X_train.drop('Last Notable Activity_Had a Phone Conversation', axis = 1, in
```

In [133]:
```
# Refit the model with the new set of features

logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Bin
logm1.fit().summary()
```

Out[133]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4447
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2076.1
Date:	Fri, 08 Nov 2019	**Deviance:**	4152.2
Time:	12:24:31	**Pearson chi2:**	4.82e+03
No. Iterations:	21	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.0069	0.600	-1.679	0.093	-2.182	0.168
TotalVisits	11.4551	2.686	4.265	0.000	6.191	16.720
Total Time Spent on Website	4.4237	0.185	23.900	0.000	4.061	4.787
Lead Origin_Lead Add Form	4.2082	0.259	16.276	0.000	3.701	4.715
Lead Source_Olark Chat	1.4581	0.122	11.958	0.000	1.219	1.697
Lead Source_Welingak Website	2.1557	1.037	2.079	0.038	0.124	4.188
Do Not Email_Yes	-1.5037	0.193	-7.780	0.000	-1.882	-1.125
Last Activity_Had a Phone Conversation	2.7502	0.802	3.430	0.001	1.179	4.322
Last Activity_SMS Sent	1.1826	0.082	14.364	0.000	1.021	1.344
What is your current occupation_Housewife	21.6525	1.49e+04	0.001	0.999	-2.91e+04	2.91e+04
What is your current occupation_Student	-1.1520	0.630	-1.828	0.068	-2.387	0.083
What is your current occupation_Unemployed	-1.3385	0.594	-2.253	0.024	-2.503	-0.174
What is your current occupation_Working Professional	1.2743	0.623	2.045	0.041	0.053	2.495
Last Notable Activity_Unreachable	2.7862	0.807	3.453	0.001	1.205	4.368

Drop `What is your current occupation_Housewife` .

In [134]:
```
X_train.drop('What is your current occupation_Housewife', axis = 1, inplace
```

In [135]:
```
# Refit the model with the new set of features

logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Bin
logm1.fit().summary()
```

Out[135]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4448
Model Family:	Binomial	**Df Model:**	12
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2078.3
Date:	Fri, 08 Nov 2019	**Deviance:**	4156.7
Time:	12:24:31	**Pearson chi2:**	4.83e+03
No. Iterations:	7	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4528	0.554	-0.818	0.413	-1.538	0.632
TotalVisits	11.2586	2.672	4.214	0.000	6.023	16.495
Total Time Spent on Website	4.4217	0.185	23.898	0.000	4.059	4.784
Lead Origin_Lead Add Form	4.2057	0.258	16.274	0.000	3.699	4.712
Lead Source_Olark Chat	1.4530	0.122	11.930	0.000	1.214	1.692
Lead Source_Welingak Website	2.1541	1.037	2.078	0.038	0.122	4.186
Do Not Email_Yes	-1.5063	0.193	-7.785	0.000	-1.886	-1.127
Last Activity_Had a Phone Conversation	2.7515	0.802	3.432	0.001	1.180	4.323
Last Activity_SMS Sent	1.1823	0.082	14.362	0.000	1.021	1.344
What is your current occupation_Student	-1.7017	0.588	-2.893	0.004	-2.855	-0.549
What is your current occupation_Unemployed	-1.8879	0.550	-3.435	0.001	-2.965	-0.811
What is your current occupation_Working Professional	0.7246	0.581	1.248	0.212	-0.413	1.862
Last Notable Activity_Unreachable	2.7834	0.807	3.448	0.001	1.201	4.365

Drop `What is your current occupation_Working Professional` .

In [136]:
```
X_train.drop('What is your current occupation_Working Professional', axis =
```

In [142]:
```
# Refit the model with the new set of features

logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Bin
res = logm1.fit()
res.summary()
```

Out[142]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4449
Model Family:	Binomial	**Df Model:**	11
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2079.1
Date:	Fri, 08 Nov 2019	**Deviance:**	4158.1
Time:	12:26:47	**Pearson chi2:**	4.80e+03
No. Iterations:	7	**Covariance Type:**	nonrobust

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	0.2040	0.196	1.043	0.297	-0.179	0.587
TotalVisits	11.1489	2.665	4.184	0.000	5.926	16.371
Total Time Spent on Website	4.4223	0.185	23.899	0.000	4.060	4.785
Lead Origin_Lead Add Form	4.2051	0.258	16.275	0.000	3.699	4.712
Lead Source_Olark Chat	1.4526	0.122	11.934	0.000	1.214	1.691
Lead Source_Welingak Website	2.1526	1.037	2.076	0.038	0.121	4.185
Do Not Email_Yes	-1.5037	0.193	-7.774	0.000	-1.883	-1.125
Last Activity_Had a Phone Conversation	2.7552	0.802	3.438	0.001	1.184	4.326
Last Activity_SMS Sent	1.1856	0.082	14.421	0.000	1.024	1.347
What is your current occupation_Student	-2.3578	0.281	-8.392	0.000	-2.908	-1.807
What is your current occupation_Unemployed	-2.5445	0.186	-13.699	0.000	-2.908	-2.180
Last Notable Activity_Unreachable	2.7846	0.807	3.449	0.001	1.202	4.367

All the p-values are now in the appropriate range. Let's also check the VIFs again in case we had missed something.

In [138]:
```
# Make a VIF dataframe for all the variables present

vif = pd.DataFrame()
vif['Features'] = X_train.columns
vif['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[138]:

	Features	VIF
9	What is your current occupation_Unemployed	2.82
1	Total Time Spent on Website	2.00
0	TotalVisits	1.54
7	Last Activity_SMS Sent	1.51
2	Lead Origin_Lead Add Form	1.45
3	Lead Source_Olark Chat	1.33
4	Lead Source_Welingak Website	1.30
5	Do Not Email_Yes	1.08
8	What is your current occupation_Student	1.06
6	Last Activity_Had a Phone Conversation	1.01
10	Last Notable Activity_Unreachable	1.01

We are good to go!

Step 3: Model Evaluation

Now, both the p-values and VIFs seem decent enough for all the variables. So let's go ahead and make predictions using this final set of features.

In [144]:
```
# Use 'predict' to predict the probabilities on the train set

y_train_pred = res.predict(sm.add_constant(X_train))
y_train_pred[:10]
```

Out[144]:
```
8003    0.300117
218     0.142002
4171    0.127629
4037    0.291558
3660    0.954795
207     0.194426
2044    0.178073
6411    0.949460
6498    0.075995
2085    0.982316
dtype: float64
```

In [145]:
```
# Reshaping it into an array

y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[145]:
```
array([0.30011695, 0.14200165, 0.12762885, 0.29155814, 0.95479546,
       0.19442563, 0.17807328, 0.94946006, 0.07599465, 0.98231619])
```

Creating a dataframe with the actual conversion flag and the predicted probabilities

In [146]:
```
# Create a new dataframe containing the actual conversion flag and the prob

y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Conversion_
y_train_pred_final.head()
```

Out[146]:

	Converted	Conversion_Prob
0	0	0.300117
1	0	0.142002
2	1	0.127629
3	1	0.291558
4	1	0.954795

Creating new column 'Predicted' with 1 if Paid_Prob > 0.5 else 0

In [147]:
```
y_train_pred_final['Predicted'] = y_train_pred_final.Conversion_Prob.map(la

# Let's see the head
y_train_pred_final.head()
```

Out[147]:

	Converted	Conversion_Prob	Predicted
0	0	0.300117	0
1	0	0.142002	0
2	1	0.127629	0
3	1	0.291558	0
4	1	0.954795	1

Now that you have the probabilities and have also made conversion predictions using them, it's time to evaluate the model.

In [148]:
```
# Import metrics from sklearn for evaluation

from sklearn import metrics
```

In [149]:
```
# Create confusion matrix

confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_
print(confusion)
```

```
[[1929  383]
 [ 560 1589]]
```

In [150]:
```
# Predicted     not_churn    churn
# Actual
# not_churn        2543      463
# churn            692       1652
```

In [151]:
```
# Let's check the overall accuracy

print(metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_fin
```

```
0.7886124187401928
```

In [152]:
```
# Let's evaluate the other metrics as well

TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [153]:
```
# Calculate the sensitivity

TP/(TP+FN)
```

Out[153]:
```
0.739413680781759
```

In [154]:
```
# Calculate the specificity

TN/(TN+FP)
```

Out[154]:
```
0.8343425605536332
```

Finding the Optimal Cutoff

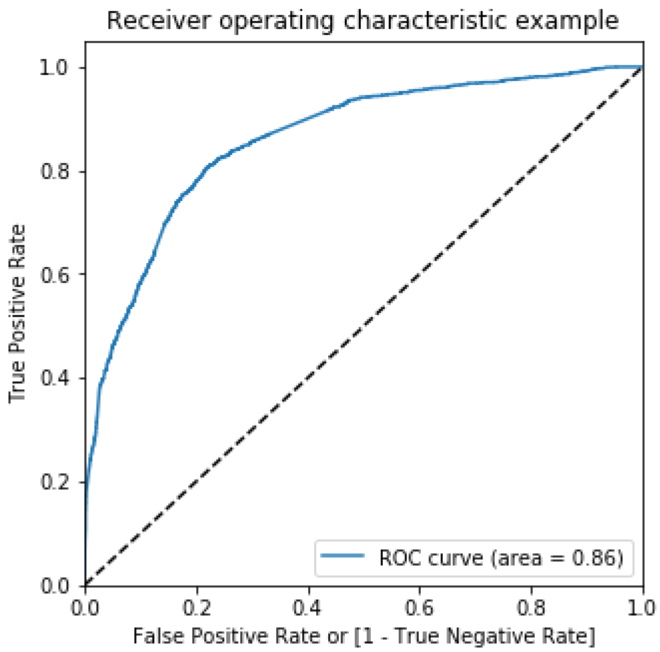
Now 0.5 was just arbitrary to loosely check the model performace. But in order to get good results, you need to optimise the threshold. So first let's plot an ROC curve to see what AUC we get.

In [155]:
```
# ROC function

def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

In [156]:
```
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Converted, y_t
```

In [157]:
```
# Import matplotlib to plot the ROC curve

import matplotlib.pyplot as plt
```

In [158]:
```
# Call the ROC function

draw_roc(y_train_pred_final.Converted, y_train_pred_final.Conversion_Prob)
```



The area under the curve of the ROC is 0.86 which is quite good. So we seem to have a good model. Let's also check the sensitivity and specificity tradeoff to find the optimal cutoff point.

In [159]:
```
# Let's create columns with different probability cutoffs

numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final.Conversion_Prob.map(lambda x:
y_train_pred_final.head()
```

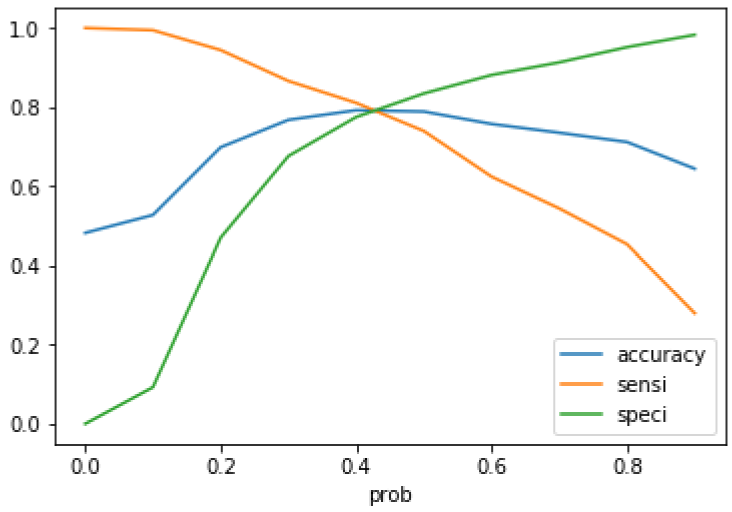
Out[159]:

	Converted	Conversion_Prob	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	0.300117	0	1	1	1	1	0	0	0	0	0	0
1	0	0.142002	0	1	1	0	0	0	0	0	0	0	0
2	1	0.127629	0	1	1	0	0	0	0	0	0	0	0
3	1	0.291558	0	1	1	1	0	0	0	0	0	0	0
4	1	0.954795	1	1	1	1	1	1	1	1	1	1	1

In [160]:
```
# Let's create a dataframe to see the values of accuracy, sensitivity, and

cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix

# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pr
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

```
     prob  accuracy     sensi     speci
0.0   0.0  0.481731  1.000000  0.000000
0.1   0.1  0.527012  0.994416  0.092561
0.2   0.2  0.698274  0.944160  0.469723
0.3   0.3  0.767541  0.865984  0.676038
0.4   0.4  0.791975  0.810610  0.774654
0.5   0.5  0.788612  0.739414  0.834343
0.6   0.6  0.757229  0.624011  0.881055
0.7   0.7  0.735037  0.543509  0.913062
0.8   0.8  0.711500  0.453234  0.951557
0.9   0.9  0.644026  0.279665  0.982699
```

In [161]:
```
# Let's plot it as well

cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



As you can see that around 0.42, you get the optimal values of the three metrics. So let's choose 0.42 as our cutoff now.

In [162]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.Conversion_Prob.

y_train_pred_final.head()
```

Out[162]:

	Converted	Conversion_Prob	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	fin
0	0	0.300117	0	1	1	1	1	0	0	0	0	0	0	
1	0	0.142002	0	1	1	0	0	0	0	0	0	0	0	
2	1	0.127629	0	1	1	0	0	0	0	0	0	0	0	
3	1	0.291558	0	1	1	1	0	0	0	0	0	0	0	
4	1	0.954795	1	1	1	1	1	1	1	1	1	1	1	

In [163]:
```
# Let's check the accuracy now

metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.fin
```

Out[163]: 0.7908540685944856

In [164]:
```
# Let's create the confusion matrix once again

confusion2 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train
confusion2
```

Out[164]:
```
array([[1823,  489],
       [ 444, 1705]], dtype=int64)
```

In [165]:
```
# Let's evaluate the other metrics as well

TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [166]:
```
# Calculate Sensitivity

TP/(TP+FN)
```

Out[166]:
```
0.793392275476966
```

In [167]:
```
# Calculate Specificity

TN/(TN+FP)
```

Out[167]:
```
0.7884948096885813
```

This cutoff point seems good to go!

Step 4: Making Predictions on the Test Set

Let's now make predicitons on the test set.

In [168]:
```
# Scale the test set as well using just 'transform'

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

In [169]:
```
# Select the columns in X_train for X_test as well

X_test = X_test[col]
X_test.head()
```

Out[169]:

	TotalVisits	Total Time Spent on Website	Lead Origin_Lead Add Form	Lead Source_Olark Chat	Lead Source_Reference	Lead Source_Welingak Website
4771	0.000000	0.000000	1	0	1	0
6122	0.027888	0.029049	0	0	0	0
9202	0.015936	0.416813	0	0	0	0
6570	0.011952	0.378961	0	0	0	0
2668	0.031873	0.395246	0	0	0	0

In [170]:
```
# Add a constant to X_test

X_test_sm = sm.add_constant(X_test[col])
```

In [171]:
```
# Check X_test_sm

X_test_sm
```

Out[171]:

	const	TotalVisits	Total Time Spent on Website	Lead Origin_Lead Add Form	Lead Source_Olark Chat	Lead Source_Reference	Source_
4771	1.0	0.000000	0.000000	1	0	1	
6122	1.0	0.027888	0.029049	0	0	0	
9202	1.0	0.015936	0.416813	0	0	0	
6570	1.0	0.011952	0.378961	0	0	0	
2668	1.0	0.031873	0.395246	0	0	0	
4233	1.0	0.000000	0.000000	0	1	0	
3368	1.0	0.007968	0.705106	0	0	0	
9091	1.0	0.035857	0.406690	0	0	0	
5972	1.0	0.007968	0.030810	0	0	0	

In [176]:
```
# Drop the required columns from X_test as well

X_test.drop(['Lead Source_Reference', 'What is your current occupation_Hous
             'What is your current occupation_Working Professional', 'Last
```

In [177]:
```
# Make predictions on the test set and store it in the variable 'y_test_pre

y_test_pred = res.predict(sm.add_constant(X_test))
```

In [178]:
```
y_test_pred[:10]
```

Out[178]:
```
4771    0.996296
6122    0.129992
9202    0.703937
6570    0.299564
2668    0.720796
4233    0.792250
3368    0.704038
9091    0.464521
5972    0.282978
3631    0.786460
dtype: float64
```

In [179]:
```
# Converting y_pred to a dataframe

y_pred_1 = pd.DataFrame(y_test_pred)
```

In [180]:
```
# Let's see the head

y_pred_1.head()
```

Out[180]:

	0
4771	0.996296
6122	0.129992
9202	0.703937
6570	0.299564
2668	0.720796

In [181]:
```
# Converting y_test to dataframe

y_test_df = pd.DataFrame(y_test)
```

In [182]:
```
# Remove index for both dataframes to append them side by side

y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

In [183]:
```
# Append y_test_df and y_pred_1

y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
```

In [184]:
```
# Check 'y_pred_final'

y_pred_final.head()
```

Out[184]:

	Converted	0
0	1	0.996296
1	0	0.129992
2	0	0.703937
3	1	0.299564
4	1	0.720796

In [185]:
```
# Rename the column

y_pred_final= y_pred_final.rename(columns = {0 : 'Conversion_Prob'})
```

In [186]:
```
# Let's see the head of y_pred_final

y_pred_final.head()
```

Out[186]:

	Converted	Conversion_Prob
0	1	0.996296
1	0	0.129992
2	0	0.703937
3	1	0.299564
4	1	0.720796

In [188]:
```
# Make predictions on the test set using 0.45 as the cutoff

y_pred_final['final_predicted'] = y_pred_final.Conversion_Prob.map(lambda x
```

In [189]:
```
# Check y_pred_final

y_pred_final.head()
```

Out[189]:

	Converted	Conversion_Prob	final_predicted
0	1	0.996296	1
1	0	0.129992	0
2	0	0.703937	1
3	1	0.299564	0
4	1	0.720796	1

In [190]:
```
# Let's check the overall accuracy

metrics.accuracy_score(y_pred_final['Converted'], y_pred_final.final_predic
```

Out[190]:
```
0.7845188284518828
```

In [191]:
```
confusion2 = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_fin
confusion2
```

Out[191]:
```
array([[786, 210],
       [202, 714]], dtype=int64)
```

In [192]:
```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [193]:
```
# Calculate sensitivity
TP / float(TP+FN)
```

Out[193]:
```
0.7794759825327511
```

In [194]:
```
# Calculate specificity
TN / float(TN+FP)
```

Out[194]:
```
0.7891566265060241
```

Precision-Recall View

Let's now also build the training model using the precision-recall view

In [114]:
```
#Looking at the confusion matrix again
```

In [195]:
```
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_
confusion
```

Out[195]:
```
array([[1929,  383],
       [ 560, 1589]], dtype=int64)
```

Precision

TP / TP + FP

In [196]:
```
confusion[1,1]/(confusion[0,1]+confusion[1,1])
```

Out[196]:
```
0.8057809330628803
```

Recall

TP / TP + FN

In [197]:
```
confusion[1,1]/(confusion[1,0]+confusion[1,1])
```

Out[197]:
```
0.739413680781759
```

Precision and recall tradeoff

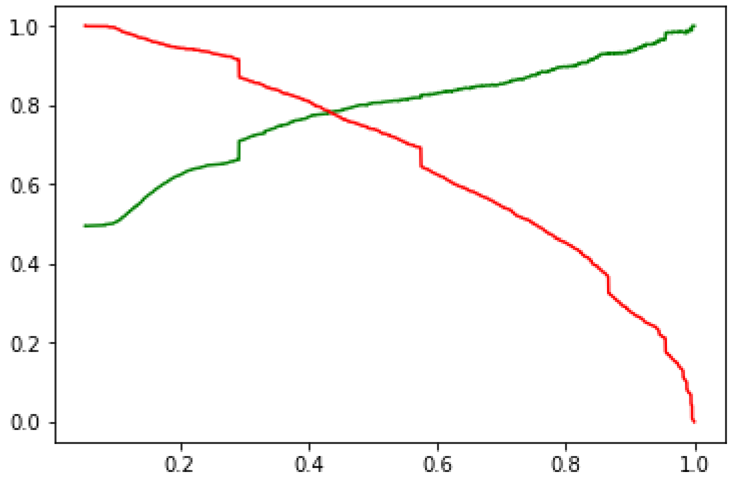
In [198]:
```
from sklearn.metrics import precision_recall_curve
```

In [199]:
```
y_train_pred_final.Converted, y_train_pred_final.Predicted
```

Out[199]:
```
(0       0
 1       0
 2       1
 3       1
 4       1
 5       0
 6       0
 7       1
 8       0
 9       1
 10      0
 11      0
 12      1
 13      1
 14      0
 15      1
 16      1
 17      1
 18      1
```

In [200]:
```
p, r, thresholds = precision_recall_curve(y_train_pred_final.Converted, y_t
```

In [201]:
```
plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
plt.show()
```



In [202]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.Conversion_Prob.

y_train_pred_final.head()
```

Out[202]:

	Converted	Conversion_Prob	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	fin
0	0	0.300117	0	1	1	1	1	0	0	0	0	0	0	
1	0	0.142002	0	1	1	0	0	0	0	0	0	0	0	
2	1	0.127629	0	1	1	0	0	0	0	0	0	0	0	
3	1	0.291558	0	1	1	1	0	0	0	0	0	0	0	
4	1	0.954795	1	1	1	1	1	1	1	1	1	1	1	

In [203]:
```
# Let's check the accuracy now

metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.fin
```

Out[203]: 0.7895090786819099

In [204]:
```
# Let's create the confusion matrix once again

confusion2 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train
confusion2
```

Out[204]:
```
array([[1852,  460],
       [ 479, 1670]], dtype=int64)
```

In [205]:
```
# Let's evaluate the other metrics as well

TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [206]:
```
# Calculate Precision

TP/(TP+FP)
```

Out[206]: 0.784037558685446

In [207]:
```
# Calculate Recall

TP/(TP+FN)
```

Out[207]: 0.7771056305258259

This cutoff point seems good to go!

Step 4: Making Predictions on the Test Set

Let's now make predicitons on the test set.

In [210]:
```
# Make predictions on the test set and store it in the variable 'y_test_pre

y_test_pred = res.predict(sm.add_constant(X_test))
```

In [211]:
```
y_test_pred[:10]
```

Out[211]:
```
4771    0.996296
6122    0.129992
9202    0.703937
6570    0.299564
2668    0.720796
4233    0.792250
3368    0.704038
9091    0.464521
5972    0.282978
3631    0.786460
dtype: float64
```

In [212]:
```
# Converting y_pred to a dataframe

y_pred_1 = pd.DataFrame(y_test_pred)
```

In [213]:
```
# Let's see the head

y_pred_1.head()
```

Out[213]:

	0
4771	0.996296
6122	0.129992
9202	0.703937
6570	0.299564
2668	0.720796

In [214]:
```
# Converting y_test to dataframe

y_test_df = pd.DataFrame(y_test)
```

In [215]:
```
# Remove index for both dataframes to append them side by side

y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

In [216]:
```
# Append y_test_df and y_pred_1

y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
```

In [217]:
```
# Check 'y_pred_final'

y_pred_final.head()
```

Out[217]:

	Converted	0
0	1	0.996296
1	0	0.129992
2	0	0.703937
3	1	0.299564
4	1	0.720796

In [218]:
```
# Rename the column

y_pred_final= y_pred_final.rename(columns = {0 : 'Conversion_Prob'})
```

In [219]:
```
# Let's see the head of y_pred_final

y_pred_final.head()
```

Out[219]:

	Converted	Conversion_Prob
0	1	0.996296
1	0	0.129992
2	0	0.703937
3	1	0.299564
4	1	0.720796

In [220]:
```
# Make predictions on the test set using 0.44 as the cutoff

y_pred_final['final_predicted'] = y_pred_final.Conversion_Prob.map(lambda x
```

In [221]:
```
# Check y_pred_final

y_pred_final.head()
```

Out[221]:

	Converted	Conversion_Prob	final_predicted
0	1	0.996296	1
1	0	0.129992	0
2	0	0.703937	1
3	1	0.299564	0
4	1	0.720796	1

In [222]:
```
# Let's check the overall accuracy

metrics.accuracy_score(y_pred_final['Converted'], y_pred_final.final_predic
```

Out[222]: 0.7866108786610879

In [223]:
```
confusion2 = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_fin
confusion2
```

Out[223]:
```
array([[801, 195],
       [213, 703]], dtype=int64)
```

In [224]:
```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [225]:
```
# Calculate Precision

TP/(TP+FP)
```

Out[225]:
```
0.7828507795100222
```

In [226]:
```
# Calculate Recall

TP/(TP+FN)
```

Out[226]:
```
0.767467248908297
```