LEAD SCORING ASSIGNMENT

Group assignment Batch DS67-2024

Dennis, Devika, Dilshad

Introduction to the Problem Statement

- An education company named X Education sells online courses to industry professionals. On any given day, many professionals who are interested in the courses land on their website and browse for courses.
- The company markets its courses on several websites and search engines like Google. Once these people land on the website, they might browse the courses or fill up a form for the course or watch some videos. When these people fill up a form providing their email address or phone number, they are classified to be a lead. Moreover, the company also gets leads through past referrals. Once these leads are acquired, employees from the sales team start making calls, writing emails, etc. Through this process, some of the leads get converted while most do not. The typical lead conversion rate at X education is around 30%.
- Now, although X Education gets a lot of leads, its lead conversion rate is very poor. For example, if, say, they acquire 100 leads in a day, only about 30 of them are converted. To make this process more efficient, the company wishes to identify the most potential leads, also known as ‘Hot Leads’. If they successfully identify this set of leads, the lead conversion rate should go up as the sales team will now be focusing more on communicating with the potential leads rather than making calls to everyone. As you can see, there are a lot of leads generated in the initial stage (top) but only a few of them come out as paying customers from the bottom. In the middle stage, you need to nurture the potential leads well (i.e. educating the leads about the product, constantly communicating etc.) in order to get a higher lead conversion.
- X Education has appointed you to help them select the most promising leads, i.e. the leads that are most likely to convert into paying customers. The company requires you to build a model wherein you need to assign a lead score to each of the leads such that the customers with a higher lead score have a higher conversion chance and the customers with a lower lead score have a lower conversion chance. The CEO, in particular, has given a ballpark of the target lead conversion rate to be around 80%.

Steps Of Our Intended Approach To Building The Required Model

- **Data Collection**
- **Quality Checks**
- **Exploratory Data Analysis**
  - Data Cleaning
  - Categorical Variables Analysis
  - Numerical Variables Analysis
- **Identifying Categorical Variables and Creating Dummy Variables**
- **Model Building Using Logistic Regression**
  - Splitting The Dataset
  - Re-scaling
  - Model Building Using RFE and Statsmodels
  - Deriving Probabilities and Lead Score
  - Confusion Matrix
  - Optimal Cut-OFF
  - Plotting ROC Curve
- **Predictions On Test Dataset**
  - Re-scaling
  - Making Predictions
  - Confusion Matrix
- **Speculation**

A. Data Collection

- **#import important libraries**
- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns

- from sklearn.preprocessing import StandardScaler

- **#import warnings**
- import warnings
- warnings.filterwarnings('ignore')

- sns.set_style('ticks')

- import plotly.express as px
- import plotly.graph_objects as go

Started with loading the dataset and gaining basic insights

```
[2]: #import the dataset and read the first five rows

     leadsdf = pd.read_csv("Leads.csv")
```

```
[3]: leadsdf.head()
```

t[3]:

	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	...	Get updates on DM Content	Lead Profile	City	Asymmetrique Activity Index	Asymmetriqu Profile Inde
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	...	No	Select	Select	02.Medium	02.Mediu
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	...	No	Select	Select	02.Medium	02.Mediu
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	...	No	Potential Lead	Mumbai	02.Medium	01.Hig
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	...	No	Select	Mumbai	02.Medium	01.Hig
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.0	1428	1.0	...	No	Select	Mumbai	02.Medium	01.Hig

5 rows × 37 columns

```
[4]: #checking the shape of data

     leadsdf.shape
```

[4]: (9240, 37)

Getting more info about the data

```
In [5]: #getting some descriptive information about the dataset

        leadsdf.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
```

In [6]:

Slide Type Slide

```
leadsdf.describe()
```

Out[6]:

	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Score	Asymmetrique Profile Score
count	9240.000000	9240.000000	9103.000000	9240.000000	9103.000000	5022.000000	5022.000000
mean	617188.435606	0.385390	3.445238	487.698268	2.362820	14.306252	16.344883
std	23405.995698	0.486714	4.854853	548.021466	2.161418	1.386694	1.811395
min	579533.000000	0.000000	0.000000	0.000000	0.000000	7.000000	11.000000
25%	596484.500000	0.000000	1.000000	12.000000	1.000000	14.000000	15.000000
50%	615479.000000	0.000000	3.000000	248.000000	2.000000	14.000000	16.000000
75%	637387.250000	1.000000	5.000000	936.000000	3.000000	15.000000	18.000000
max	660737.000000	1.000000	251.000000	2272.000000	55.000000	18.000000	20.000000

Slide Type Slide

Conclusion:

- There are total of 9240 rows and 37 columns in the dataset.

B. Quality Checks

The two columns 'Prospect ID' and 'Lead Number' have to be checked for duplicate values.

In [7]:

```
if sum(leadsdf.duplicated(subset = "Prospect ID"))==0:
    print("No duplicate values in 'Prospect ID' column")
else:
    print("Duplicate values in 'Prospect ID' column")
```

```
No duplicate values in 'Prospect ID' column
```

In [8]:

```
if sum(leadsdf.duplicated(subset = "Lead Number"))==0:
    print("No duplicate values in 'Lead Number' column")
else:
    print("Duplicate values in 'Lead Number' column")
```

```
No duplicate values in 'Lead Number' column
```

Slide Type Slide

Conclusion:

- No duplicate values were found in the above two columns meaning they are only used to indicate the ID number of the contacted people and are of no significance hence they can be dropped.

Checking the percentage of missing values in all the columns

In [9]:

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

Out[9]:

```
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
Country                                         26.63
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
Tags                                            36.29
Lead Quality                                    51.59
Update me on Supply Chain Content                0.00
Get updates on DM Content                        0.00
Lead Profile                                    29.32
City                                            15.37
Asymmetrique Activity Index                     45.65
Asymmetrique Profile Index                      45.65
Asymmetrique Activity Score                     45.65
Asymmetrique Profile Score                      45.65
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
```

C. Exploratory Data Analysis

I. Data Cleaning

II. 1. Dropping Non-Significant Columns

[10]: Slid

```
#dropping 'Prospect ID' and 'Lead Number' columns as they only have unique values with no importance in analysis

leadsdf.drop(['Prospect ID','Lead Number'], 1, inplace = True)
```

2. Converting 'Select' values to NA values

In [12]:
```
leadsdf = leadsdf.replace('Select', np.nan)
```

3. Dropping columns having more than 45% missing values in them

In [14]:
```
#dropping more than 45% missing value columns

cols_drop = leadsdf.columns
for i in cols_drop:
    if ((100*(leadsdf[i].isnull().sum()/len(leadsdf.index))) > 45):
        leadsdf.drop(i, 1, inplace = True)
```

Checking missing values percentage in remaining columns after dropping more than 45% missing value columns

In [15]:
```
#checking missing values percentage in remaining columns after

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

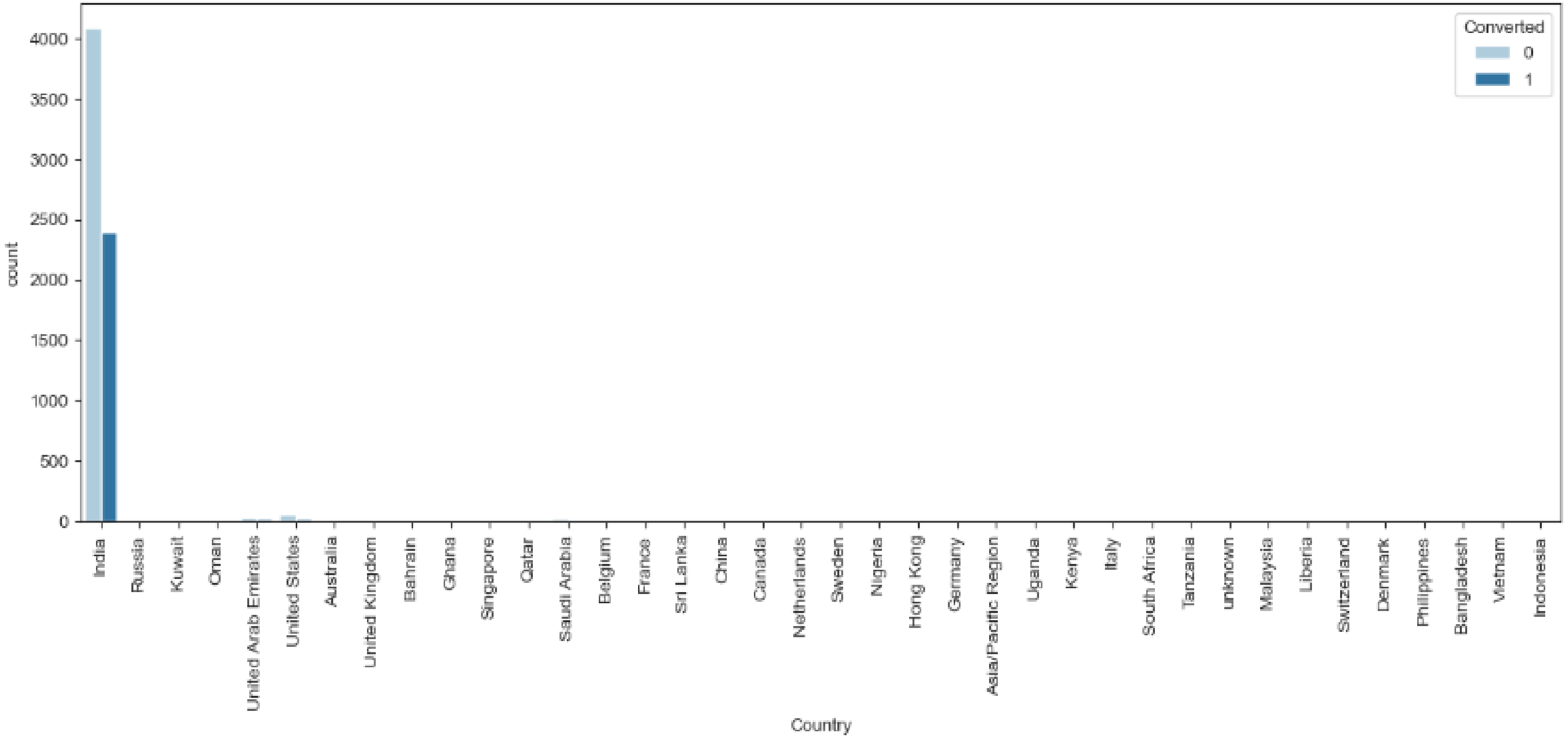
Out[15]:
```
Lead Origin                                      0.00
Lead Source                                      0.39
Do Not Email                                     0.00
Do Not Call                                      0.00
Converted                                        0.00
TotalVisits                                      1.48
Total Time Spent on Website                      0.00
Page Views Per Visit                             1.48
Last Activity                                    1.11
Country                                         26.63
Specialization                                  36.58
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
Update me on Supply Chain Content                0.00
```

Categorical Variables Analysis

1. Analysing 'Country' column

In [16]:
```
#checking all value counts of 'country' column including the NA values

leadsdf['Country'].value_counts(dropna = False)
```

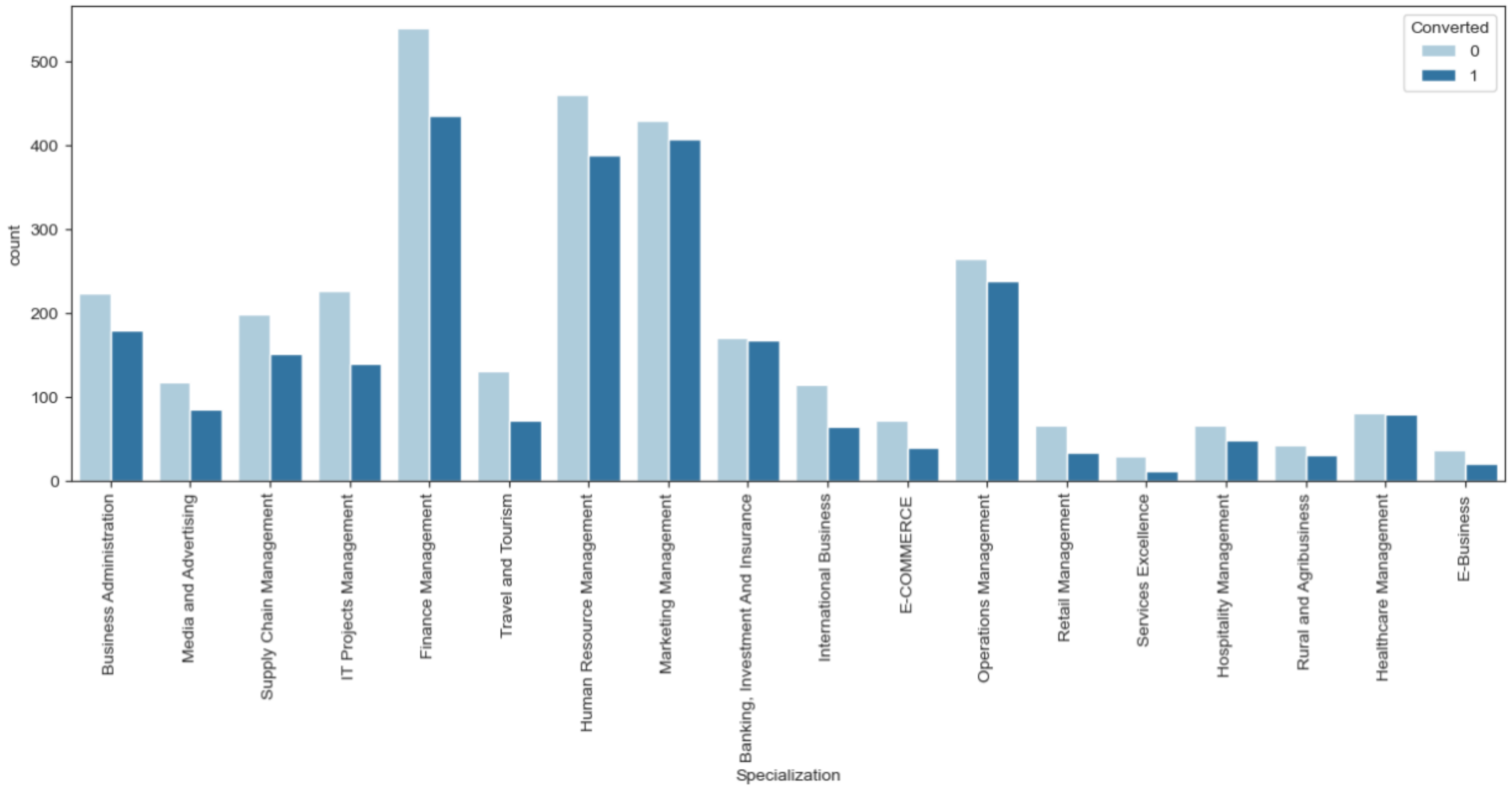
In [17]:
```
#visualising the spread of data in 'country' column

plt.figure(figsize=(15,5))
plt1=sns.countplot(x='Country', hue='Converted', data=leadsdf, palette = 'Paired')
plt1.set_xticklabels(plt1.get_xticklabels(),rotation=90)
plt.show()
```



Conclusion:

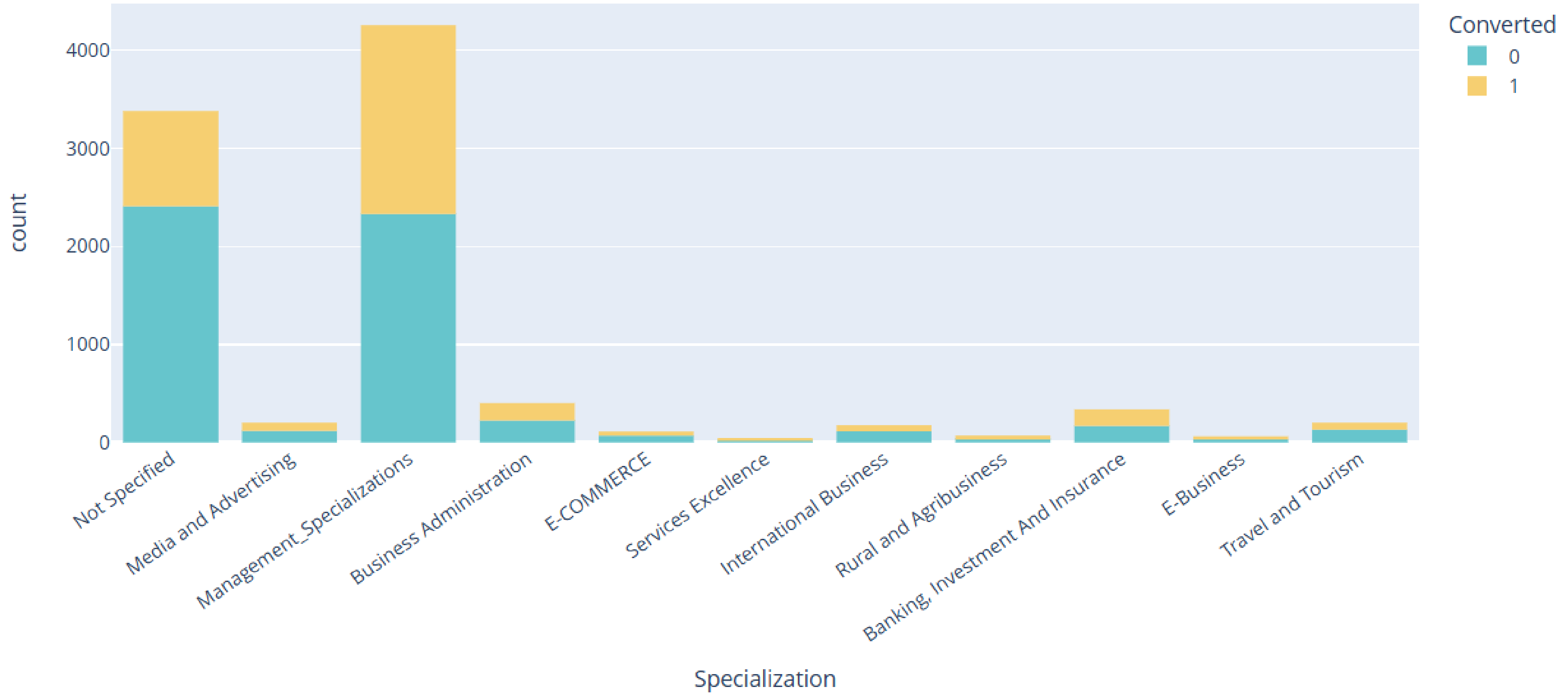
- Since 'India' is most in number in non-missing values, we impute the missing values with 'India'
- We see that number of values of 'India' is quite high (almost 98%) which indicates this data is highly skewed and will influence the model incorrectly hence this column will be dropped towards the end of this analysis.

Similarly, 'City' column NA values also replaced by mode 'Mumbai' ; 'What's your current occupation' NA values replaced by mode 'Unemployed'



For columns like 'Specialisation', NA values replaced with the term 'Not Specified'. As leads may not have mentioned their specializations because they either might still be students or their chosen specialization was not in list.
We also combined similar type of specializations in one term 'Management Specialization' .

After that step, we get better insights from a histogram split on ‘Converted’ column



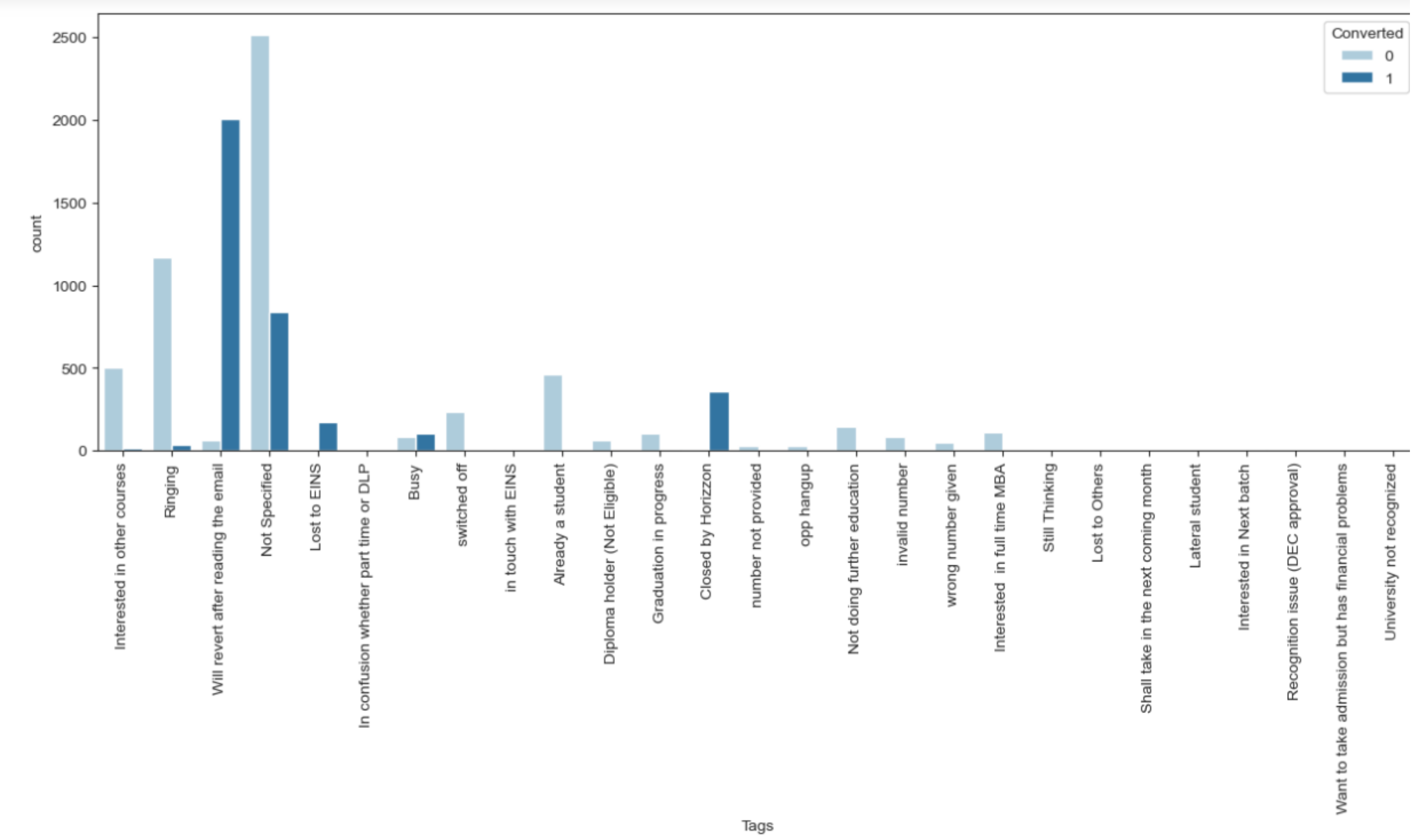
Column : 'What matters most to you in choosing a course'

In [40]:
```
#checking the spread of data in 'What matters most to you in choosing a course' column after data imputation

leadsdf['What matters most to you in choosing a course'].value_counts(dropna = False)
```

Out[40]:
```
Better Career Prospects      9237
Flexibility & Convenience       2
Other                           1
Name: What matters most to you in choosing a course, dtype: int64
```

As data is heavily skewed, we’ll drop this column

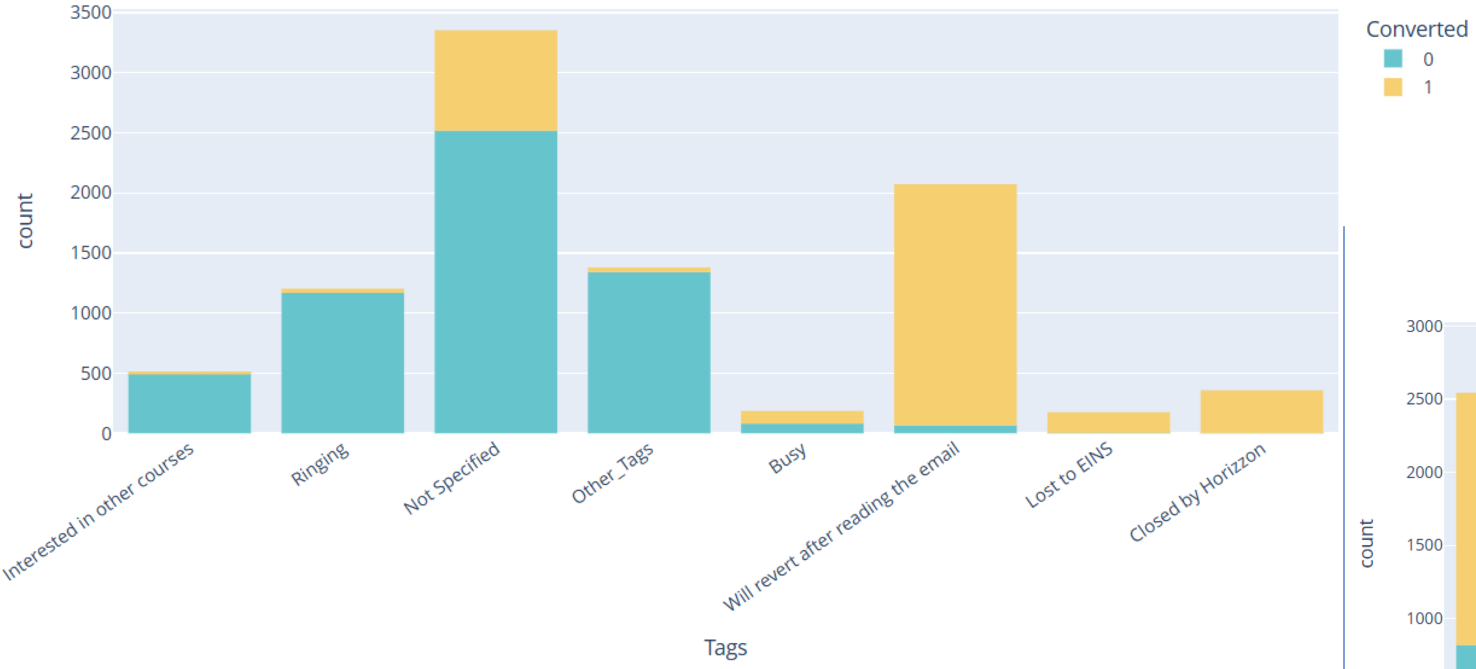
Column :Tags



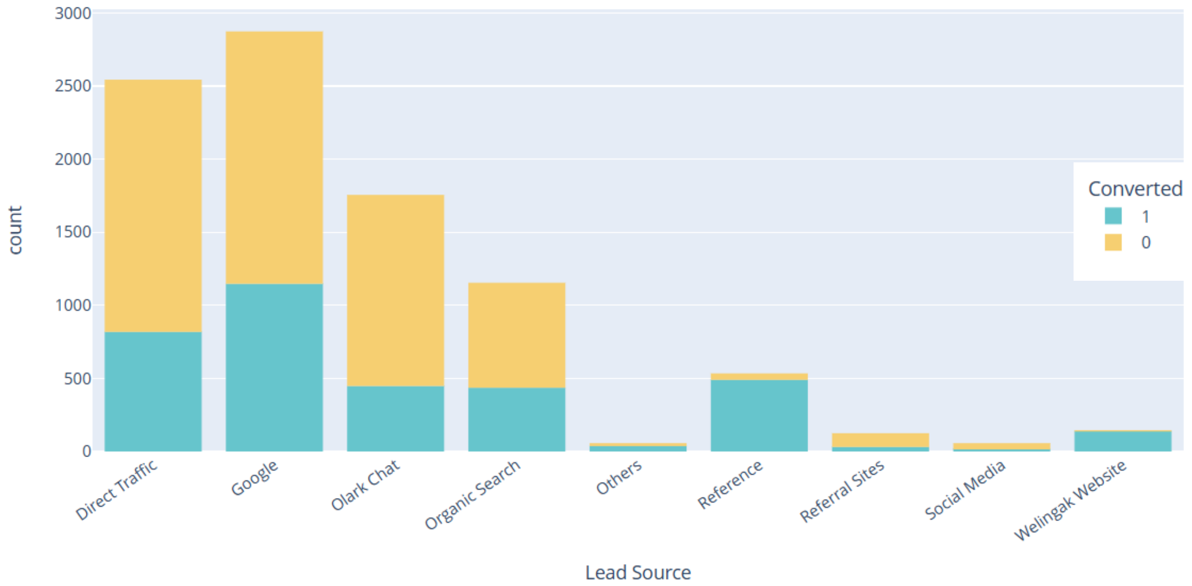
Conclusion:

- Leads tagged 'Will revert after reading mail' have highest chances of being converted followed by 'Lost To EINS', 'Closed By Horizon and 'Busy'
- But there is a lot of data with very low frequence and hence of we club them under one single term like 'Other_Tags' we will be able to read the spread better.

- We will replace 'tags' column low frequency data with term 'Other_Tags'



Similarly for ‘**Lead source**’ column we grouped similar types of low frequency columns like ‘Facebook’ and ‘social Media’



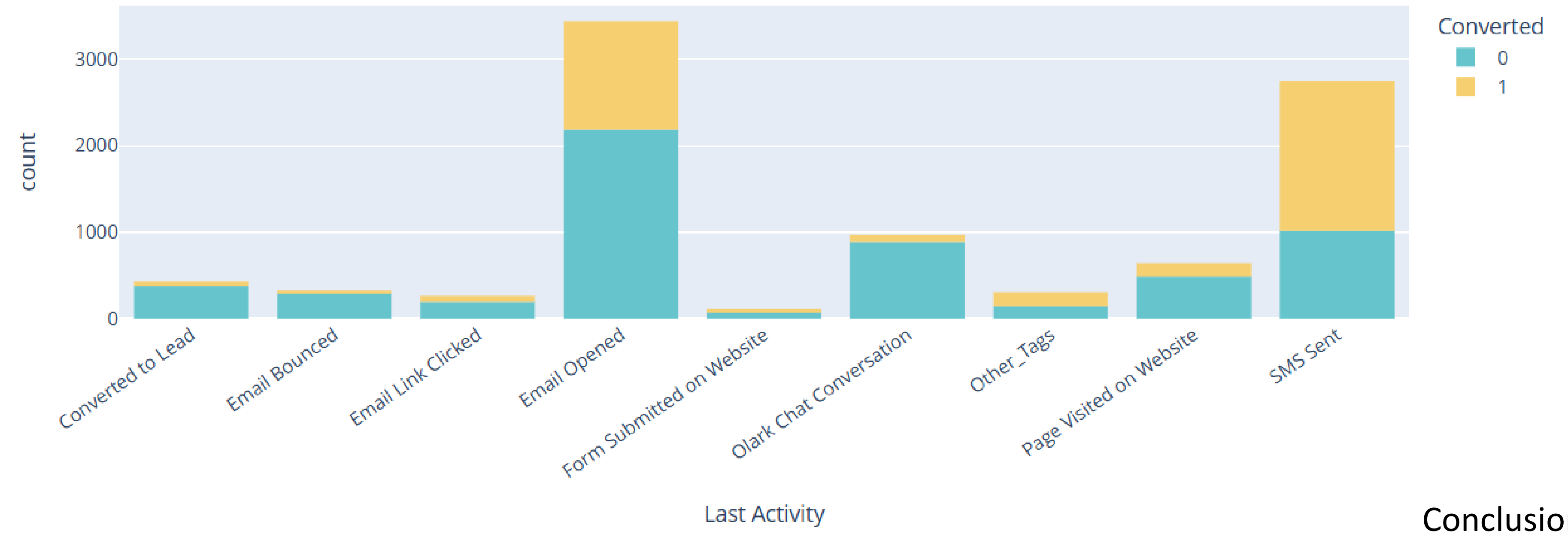
Conclusion:

- Leads tagged 'Will revert after reading mail' have highest chances of being converted followed by 'Lost To EINS', 'Closed By Horizon and 'Busy' and efforts should be made to generate more leads from these tags.
- 'Ringing' and 'Not Specified' tagged leads are more in number and hence efforts should be made to maximise conversion from these tags.

Conclusion:

- 'Google' and 'Direct Traffic' generate most amount of leads and lead conversions while some like 'Welingak Website', 'Reference' and 'Others' are having maximum conversion of leads. This is a very significant column hence we shall retain it.
- To improve overall lead conversion rate, focus should be on improving lead converion of 'direct traffic' and 'google leads' and efforts should be put to generate more leads from 'reference' and 'welingak website' as they are conversion rate is very strong.

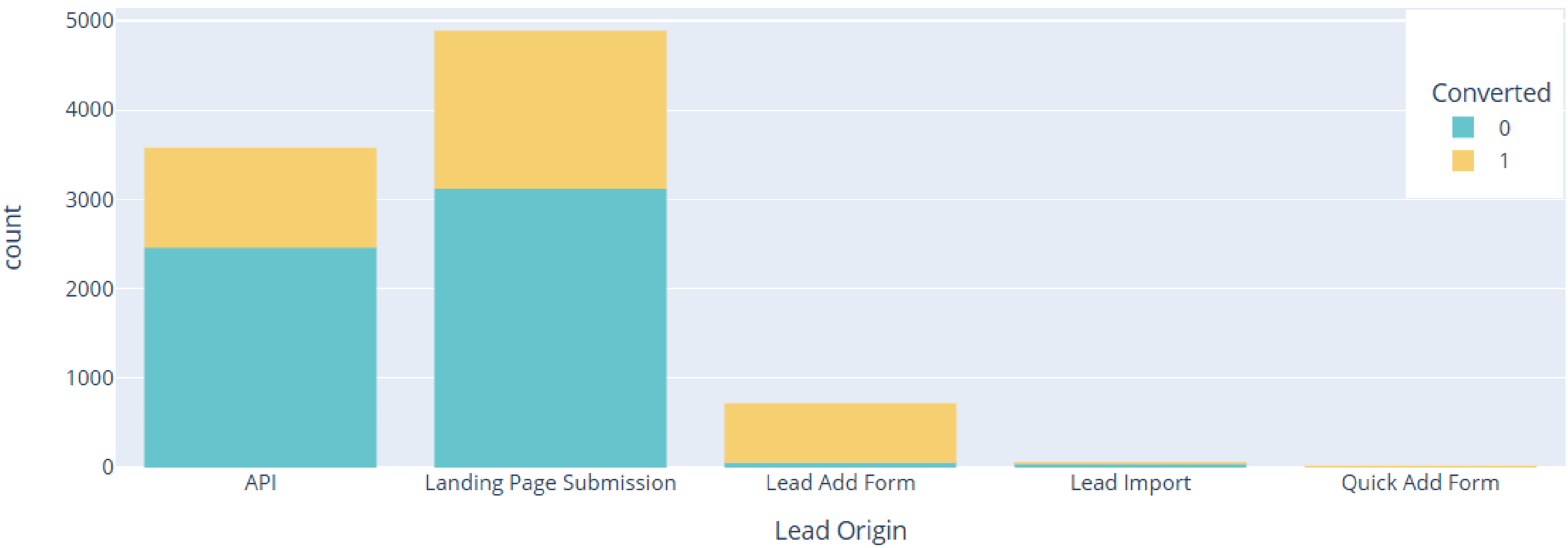
Similarly we clubbed low frequency similar data points in ‘**Last Activity’** column as well.



Conclusion:

- This is another significant column which should be retained as we can clearly see that for leads having last activity as 'SMS Sent' have the most conversion rate.
- 'Email Opened' brings maximum no. of leads and has second most conversion as well.

Column: **Lead Origin**



Conclusion:

- This is another very significant column as we can see that 'Lead Add Form' is a very good origin of leads due to its very strong conversion rate.
- 'Landing Page Submissions' and 'API' bring a higher amount of leads and see more lead conversion as well.
- Lead Import and Quick Add Form get very few leads.
- To improve overall lead conversion rate, we have to improve lead converion of API and Landing Page Submission origin and generate more leads from Lead Add Form.

14. Analysing The Remaining Categorical Columns

```
In [75]: #checking all value counts of 'Search' column

         leadsdf['Search'].value_counts(dropna = False)

Out[75]: No      9226
         Yes       14
         Name: Search, dtype: int64
```

```
In [76]: #checking all value counts of 'Magazine' column

         leadsdf['Magazine'].value_counts(dropna = False)

Out[76]: No    9240
         Name: Magazine, dtype: int64
```

```
In [77]: #checking all value counts of 'Newspaper Article' column

         leadsdf['Newspaper Article'].value_counts(dropna = False)

Out[77]: No      9238
         Yes        2
         Name: Newspaper Article, dtype: int64
```

```
In [78]: #checking all value counts of 'X Education Forums' column

         leadsdf['X Education Forums'].value_counts(dropna = False)

Out[78]: No      9239
         Yes        1
         Name: X Education Forums, dtype: int64
```

```
In [79]: #checking all value counts of 'Newspaper' column

         leadsdf['Newspaper'].value_counts(dropna = False)

Out[79]: No      9239
         Yes        1
         Name: Newspaper, dtype: int64
```

```
In [80]: #checking all value counts of 'Digital Advertisement' column

         leadsdf['Digital Advertisement'].value_counts(dropna = False)

Out[80]: No      9236
         Yes        4
         Name: Digital Advertisement, dtype: int64
```

```
In [81]: #checking all value counts of 'Receive More Updates About Our Courses' column

         leadsdf['Receive More Updates About Our Courses'].value_counts(dropna = False)

Out[81]: No    9240
         Name: Receive More Updates About Our Courses, dtype: int64
```

```
In [82]: #checking all value counts of 'Update me on Supply Chain Content' column

         leadsdf['Update me on Supply Chain Content'].value_counts(dropna = False)

Out[82]: No    9240
         Name: Update me on Supply Chain Content, dtype: int64
```

```
In [83]: #checking all value counts of 'Get updates on DM Content' column

         leadsdf['Get updates on DM Content'].value_counts(dropna = False)
```

```
In [84]: #checking all value counts of 'I agree to pay the amount through cheque' column

         leadsdf['I agree to pay the amount through cheque'].value_counts(dropna = False)

Out[84]: No    9240
         Name: I agree to pay the amount through cheque, dtype: int64
```

Conclusion:

- As we can see the above few columns have highly skewed data and hence it will be better to drop these columns from our analysis.

In [85]:
```python
#checking percentage of missing values in dataset

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

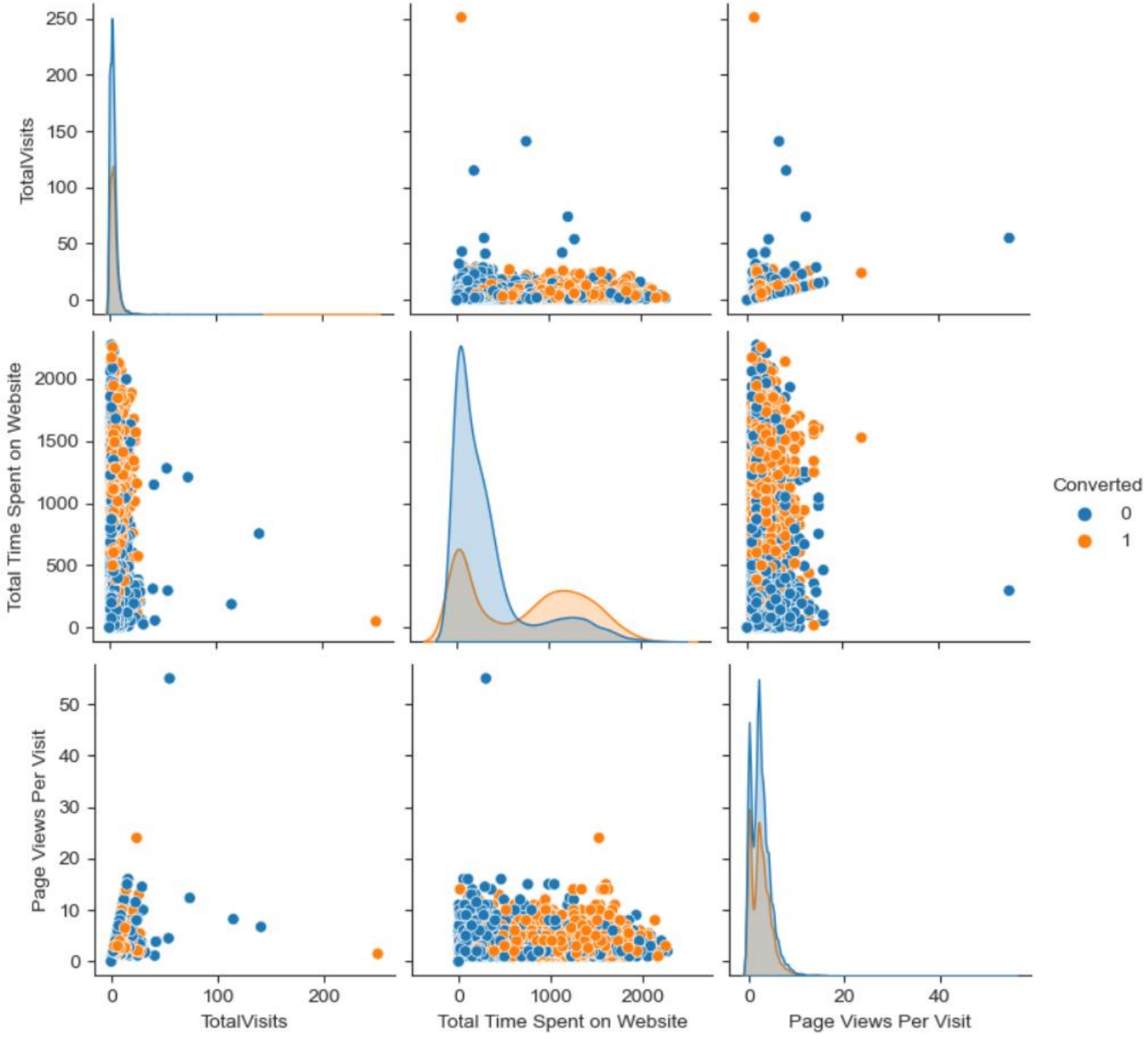
Out[85]:
```
Lead Origin                                      0.00
Lead Source                                      0.00
Do Not Email                                     0.00
Do Not Call                                      0.00
Converted                                        0.00
TotalVisits                                      1.48
Total Time Spent on Website                      0.00
Page Views Per Visit                             1.48
Last Activity                                    0.00
Country                                          0.00
Specialization                                   0.00
What is your current occupation                  0.00
What matters most to you in choosing a course    0.00
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                                             0.00
Update me on Supply Chain Content                0.00
Get updates on DM Content                        0.00
City                                             0.00
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
```

As we can see most of the columns have no missing columns and only 2 columns ahve about 1.5% missing values each which are highly insignificant and hence can be dropped altogether.

Numerical Variables Analysis

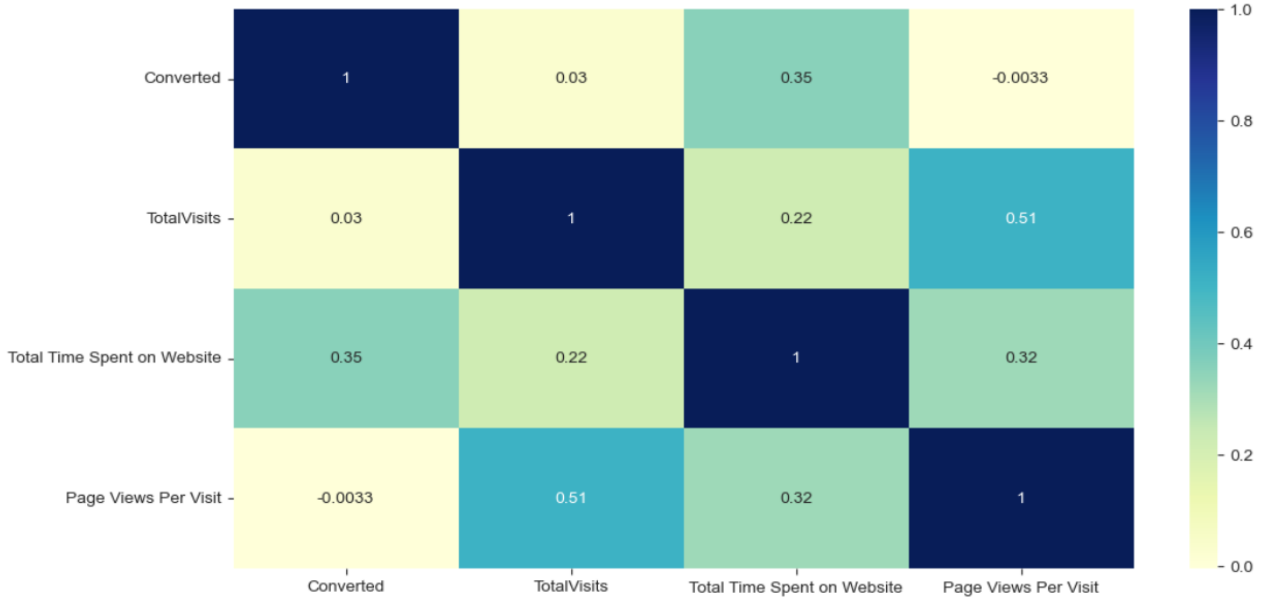
Analysing the correlation of various numerical variable using heatmap.

In [93]:
```
sns.pairplot(leadsdf, hue='Converted')
plt.show()
```



Creating pair plots

Correlation using aheatmap



Conclusion:

- While there is not much to say about the correlation between these numeric variables, 'Total Visits' and 'Page Views Per Visit' have the most correlation with each other. We shall keep this in mind while building our model.

Checking outliers

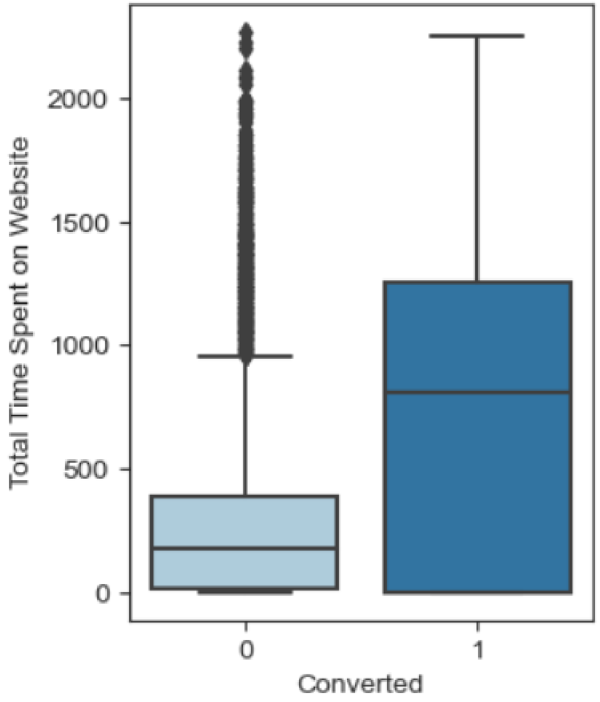
```
In [99]: # checking the spread of percentiles of 'TotalVisits' column

         leadsdf['TotalVisits'].describe(percentiles=[0.05,.25, .5, .75, .90, .95, .99])

Out[99]: count    9103.000000
         mean        3.445238
         std         4.854853
         min         0.000000
         5%          0.000000
         25%         1.000000
         50%         3.000000
         75%         5.000000
         90%         7.000000
         95%        10.000000
         99%        17.000000
         max       251.000000
         Name: TotalVisits, dtype: float64
```

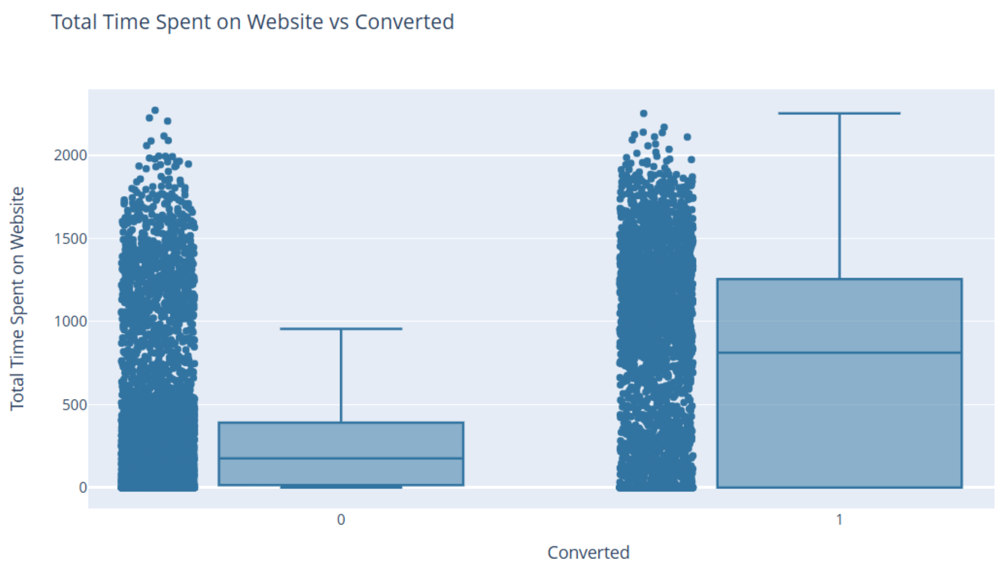
```
In [100]: # checking the spread of percentiles of 'Page Views Per Visit' column

          leadsdf['Page Views Per Visit'].describe(percentiles=[0.05,.25, .5, .75, .90, .95, .99])

Out[100]: count    9103.000000
          mean        2.362820
          std         2.161418
          min         0.000000
          5%          0.000000
          25%         1.000000
          50%         2.000000
          75%         3.000000
          90%         5.000000
          95%         6.000000
          99%         9.000000
          max        55.000000
          Name: Page Views Per Visit, dtype: float64
```

As we can see, the difference between 99th percentile and max value and min value and 25th percentile is very high for both the columns.
Hence we shall drop the top and bottom 1% of datapoints to treat these outliers.

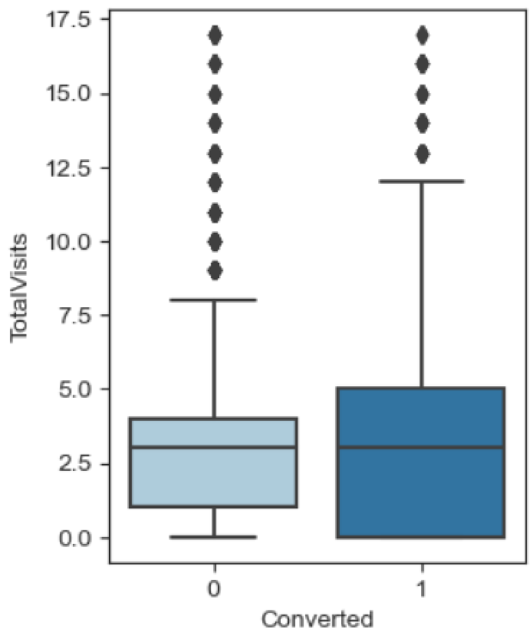
Now we can visualise the box plot better and with lesser outliers.

```
[103]: #visualising the spread of Converted vs Total Time Spent on Website

       plt.figure(figsize=(3,4))
       sns.boxplot(y=leadsdf['Total Time Spent on Website'], x=leadsdf['Converted'], palette = 'Paired')
       plt.show()
```

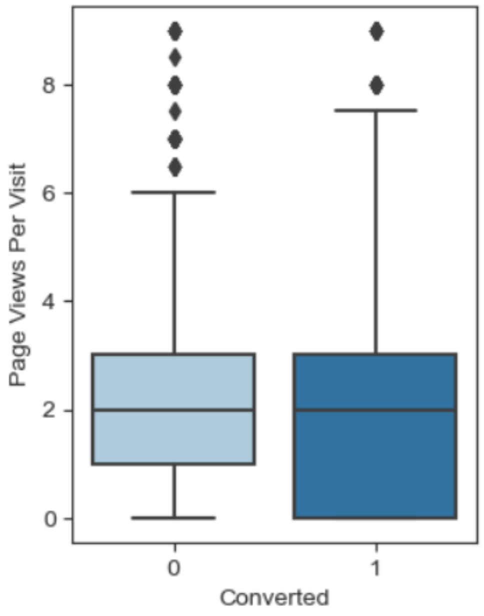




[105]:
```
#visualising the spread of Converted vs Total Visits

plt.figure(figsize=(3,4))
sns.boxplot(y=leadsdf['TotalVisits'], x=leadsdf['Converted'], palette = 'Paired')
plt.show()
```



[106]:
```
#visualising the spread of Converted vs Page Views Per Visit

plt.figure(figsize=(3,4))
sns.boxplot(y=leadsdf['Page Views Per Visit'], x=leadsdf['Converted'], palette = 'Paired')
plt.show()
```



Conclusion:
- From the above visualizations we observe that leads spending more time on website have higher chances of getting converted as we see the median is quite high for the Total Time Spent on Website vs Converted column.
- For the other two there is no conclusive evidence as the medians are same.

1. Identifying Columns Having Categorical Variable

In [107]:
```
leadsdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8953 entries, 0 to 9239
Data columns (total 14 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Lead Origin                             8953 non-null   object
 1   Lead Source                             8953 non-null   object
 2   Do Not Email                            8953 non-null   object
 3   Converted                               8953 non-null   int64
 4   TotalVisits                             8953 non-null   float64
 5   Total Time Spent on Website             8953 non-null   int64
 6   Page Views Per Visit                    8953 non-null   float64
 7   Last Activity                           8953 non-null   object
 8   Specialization                          8953 non-null   object
 9   What is your current occupation         8953 non-null   object
 10  Tags                                    8953 non-null   object
 11  City                                    8953 non-null   object
 12  A free copy of Mastering The Interview  8953 non-null   object
 13  Last Notable Activity                   8953 non-null   object
dtypes: float64(2), int64(2), object(10)
memory usage: 1.0+ MB
```

In [108]:
```
#extracting the columns having object datatype and displaying them.

cat_cols= leadsdf.select_dtypes(include=['object']).columns
cat_cols
```

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

2. Mapping the binary response categorical columns with only 2 responses

Converting ‘Yes’ to 1 and “no’ to 0

3. Creating Dummy Variables For Categorical Variables

- Create Dummy Variables
- Drop The Original Columns For Which Dummy Variables Were Created
- Drop The First Columns As 'p-1' Dummies Can Explain For 'p' Categories

[111]:
```
#create dummy variables and drop first columns

dummy_var_1 = pd.get_dummies(leadsdf[['Lead Origin','What is your current occupation',
                             'City']], drop_first=True)

#adding the results to the main dataframe
leadsdf = pd.concat([leadsdf,dummy_var_1],1)
```

[112]:
```
#create dummy variables and drop first columns

dummy_var_2 = pd.get_dummies(leadsdf['Specialization'], prefix  = 'Specialization')
dummy_var_2 = dummy_var_2.drop(['Specialization_Not Specified'], 1)

#adding the results to the main dataframe
leadsdf = pd.concat([leadsdf, dummy_var_2], axis = 1)
```

[113]:
```
#create dummy variables and drop first columns

dummy_var_3 = pd.get_dummies(leadsdf['Lead Source'], prefix  = 'Lead Source')
dummy_var_3 = dummy_var_3.drop(['Lead Source_Others'], 1)

#adding the results to the main dataframe
leadsdf = pd.concat([leadsdf, dummy_var_3], axis = 1)
```

[114]:
```
#create dummy variables and drop first columns

dummy_var_4 = pd.get_dummies(leadsdf['Last Activity'], prefix  = 'Last Activity')
dummy_var_4 = dummy_var_4.drop(['Last Activity_Other_Tags'], 1)

#adding the results to the main dataframe
leadsdf = pd.concat([leadsdf, dummy_var_4], axis = 1)
```

Created dummy variables and drop first column for all columns having data only as “yes’ and ‘no’

E. Logistic Regression Model Building

1. Splitting The Dataset

- We will now first split the dataset into train and test datasets in ratio of 70-30
- We will use sklearn package and import train_test_split method

[119]:
```
from sklearn.model_selection import train_test_split

#assigning response variable to y
y = leadsdf['Converted']

#assigning remaining variables to X
X = leadsdf.drop('Converted', axis=1)
```

[122]:
```
#splitting the data into 70% train and 30% test dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

[123]:
```
#checking the shape of X_train dataset

X_train.shape
```

t[123]: (6267, 55)

[124]:
```
#checking the shape of y_train dataset

y_train.shape
```

t[124]: (6267,)

2. Rescaling the numerical variables

[126]:
```
#import the required libraries

from sklearn.preprocessing import StandardScaler

#apply scaler to all numeric variables

scaler = StandardScaler()

#extract the numeric columns only

num_cols = X_train.select_dtypes(include=['float64', 'int64']).columns

#fit and train the dataset to rescale the numeric variables

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])

#check the scaled values of numeric columns

X_train.head()
```

3. Model Building using Statsmodels and RFE

- Now we shall start building the model manually, initially starting with the 15 best variables as selected by RFE.
- Further we shall check which model is a better fit by dropping unrequired variables depending on their VIF values/p-values.
- As per convention, we shall drop variables either having VIF > 5 or having p-values > 0.05.
- VIF parameter, which indicates multicollinearity must always be < 5 while p-values, which determine significance of the variables must be < 0.05
- We shall use the statsmodels technique as it allows us to also view a detailed summary of different parameters and make better judgements about the model
- We shall keep dropping/adding variables as required to find the best model and then test it against the test dataset.

[128]:
```
#import the required libraries

from sklearn.linear_model import LogisticRegression

#initialize the logarithmic regression function

logreg = LogisticRegression()

#import RFE

from sklearn.feature_selection import RFE

# running RFE with 15 variables as output

rfe = RFE(estimator=logreg, n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)
```

```
[129]: #display the variables chosen by RFE for builing the initial model al

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

```
[129]: [('TotalVisits', False, 26),
 ('Total Time Spent on Website', False, 2),
 ('Page Views Per Visit', False, 24),
 ('Lead Origin_Landing Page Submission', False, 9),
 ('Lead Origin_Lead Add Form', True, 1),
 ('Lead Origin_Lead Import', False, 19),
 ('What is your current occupation_Housewife', False, 27),
 ('What is your current occupation_Other', False, 29),
 ('What is your current occupation_Student', False, 20),
 ('What is your current occupation_Unemployed', False, 21),
 ('What is your current occupation_Working Professional', False, 7),
 ('City_Other Cities', False, 23),
 ('City_Other Cities of Maharashtra', False, 34),
 ('City_Other Metro Cities', False, 39),
 ('City_Thane & Outskirts', False, 37),
 ('City_Tier II Cities', False, 31),
 ('Specialization_Banking, Investment And Insurance', False, 14),
 ('Specialization_Business Administration', False, 41),
 ('Specialization_E-Business', False, 32),
 ('Specialization_E-COMMERCE', False, 22),
 ('Specialization_International Business', False, 40),
 ('Specialization_Management_Specializations', False, 36),
 ('Specialization_Media and Advertising', False, 35),
 ('Specialization_Rural and Agribusiness', False, 38),
 ('Specialization_Services Excellence', False, 33),
 ('Specialization_Travel and Tourism', False, 8),
 ('Lead Source_Direct Traffic', True, 1),
 ('Lead Source_Google', True, 1),
 ('Lead Source_Olark Chat', False, 30),
 ('Lead Source_Organic Search', True, 1),
 ('Lead Source_Reference', False, 12),
 ('Lead Source_Referral Sites', True, 1),
 ('Lead Source_Social Media', False, 18),
 ('Lead Source_Welingak Website', True, 1),
 ('Last Activity_Converted to Lead', False, 10),
 ('Last Activity_Email Bounced', True, 1),
 ('Last Activity_Email Link Clicked', False, 28),
 ('Last Activity_Email Opened', False, 16),
 ('Last Activity_Form Submitted on Website', False, 15),
 ('Last Activity_Olark Chat Conversation', True, 1),
 ('Last Activity_Page Visited on Website', False, 11),
 ('Last Activity_SMS Sent', False, 3),
 ('Tags_Busy', True, 1),
 ('Tags_Closed by Horizzon', True, 1),
 ('Tags_Interested in other courses', False, 13),
 ('Tags_Lost to EINS', True, 1),
 ('Tags_Not Specified', True, 1),
 ('Tags_Ringing', True, 1),
 ('Tags_Will revert after reading the email', True, 1),
 ('Last Notable Activity_Email Link Clicked', False, 6),
 ('Last Notable Activity_Email Opened', False, 17),
 ('Last Notable Activity_Modified', False, 5),
 ('Last Notable Activity_Olark Chat Conversation', False, 4),
 ('Last Notable Activity_Page Visited on Website', False, 25),
 ('Last Notable Activity_SMS Sent', True, 1)]
```

[131]:
```python
#extract and assign the chosen columns by RFE to a variable.

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

[131]:
```
Index(['Lead Origin_Lead Add Form', 'Lead Source_Direct Traffic',
       'Lead Source_Google', 'Lead Source_Organic Search',
       'Lead Source_Referral Sites', 'Lead Source_Welingak Website',
       'Last Activity_Email Bounced', 'Last Activity_Olark Chat Conversation',
       'Tags_Busy', 'Tags_Closed by Horizzon', 'Tags_Lost to EINS',
       'Tags_Not Specified', 'Tags_Ringing',
       'Tags_Will revert after reading the email',
       'Last Notable Activity_SMS Sent'],
      dtype='object')
```

I. Building the model

Let’s build the model using columns selected for us by RFE

List all the VIF values of predictor variables

[132]:
```python
#import the VIF object

from statsmodels.stats.outliers_influence import variance_inflation_factor

#creating a dataframe consisting of all the predictor variables along with their VIF values

vif = pd.DataFrame()
vif['Features'] = X_train[rfe_col].columns
vif['VIF'] = [variance_inflation_factor(X_train[rfe_col].values, i) for i in range(X_train[rfe_col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[132]:

	Features	VIF
5	Lead Source_Welingak Website	1.35
9	Tags_Closed by Horizzon	1.22
8	Tags_Busy	1.09
6	Last Activity_Email Bounced	1.07
10	Tags_Lost to EINS	1.05
4	Lead Source_Referral Sites	1.02
0	Lead Origin_Lead Add Form	0.75
3	Lead Source_Organic Search	0.40
7	Last Activity_Olark Chat Conversation	0.30
14	Last Notable Activity_SMS Sent	0.24
1	Lead Source_Direct Traffic	0.19
2	Lead Source_Google	0.17
13	Tags_Will revert after reading the email	0.17
12	Tags_Ringing	0.05
11	Tags_Not Specified	0.02

Create the first fitted model using Statsmodels

[133]:
```python
#import the statsmodels library

import statsmodels.api as sm

#add a constant

X_train_sm = sm.add_constant(X_train[rfe_col])

#creating the first fitted model

log_mod_1 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res1 = log_mod_1.fit()

#read the summary of the model

res1.summary()
```

[133]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6267
Model:	GLM	**Df Residuals:**	6251
Model Family:	Binomial	**Df Model:**	15
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1416.1
Date:	Fri, 26 May 2023	**Deviance:**	2832.2
Time:	20:34:24	**Pearson chi2:**	9.42e+03
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.5839
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.7006	0.206	-17.959	0.000	-4.104	-3.297
Lead Origin_Lead Add Form	0.8022	0.441	1.821	0.069	-0.061	1.666
Lead Source_Direct Traffic	-0.6451	0.152	-4.243	0.000	-0.943	-0.347
Lead Source_Google	0.0089	0.135	0.066	0.948	-0.256	0.273
Lead Source_Organic Search	-0.0704	0.175	-0.403	0.687	-0.413	0.272
Lead Source_Referral Sites	-0.3847	0.441	-0.873	0.383	-1.249	0.479
Lead Source_Welingak Website	4.2159	1.105	3.817	0.000	2.051	6.381
Last Activity_Email Bounced	-1.4205	0.418	-3.396	0.001	-2.240	-0.601
Last Activity_Olark Chat Conversation	-1.7121	0.229	-7.483	0.000	-2.161	-1.264
Tags_Busy	2.9959	0.267	11.211	0.000	2.472	3.520
Tags_Closed by Horizzon	9.0151	1.024	8.807	0.000	7.009	11.021
Tags_Lost to EINS	7.8169	0.619	12.623	0.000	6.603	9.031
Tags_Not Specified	2.2780	0.180	12.665	0.000	1.925	2.631
Tags_Ringing	-1.0710	0.274	-3.912	0.000	-1.608	-0.534
Tags_Will revert after reading the email	6.8546	0.237	28.886	0.000	6.390	7.320
Last Notable Activity_SMS Sent	2.4855	0.119	20.947	0.000	2.253	2.718

Conclusion:

- The p-value of the variable 'Lead Source_Google' is highest and > 0.05 and hence it makes sense to drop the variable.

[134]:
```python
#dropping 'Lead Source_Google' column

rfe_col = rfe_col.drop('Lead Source_Google', 1)
```

II. Building the Model-2

List all the VIF values of predictor variables

```
#import the VIF object

from statsmodels.stats.outliers_influence import variance_inflation_factor

#creating a dataframe consisting of all the predictor variables along with their VIF values

vif = pd.DataFrame()
vif['Features'] = X_train[rfe_col].columns
vif['VIF'] = [variance_inflation_factor(X_train[rfe_col].values, i) for i in range(X_train[rfe_col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[135]:

	Features	VIF
4	Lead Source_Welingak Website	1.35
8	Tags_Closed by Horizzon	1.16
5	Last Activity_Email Bounced	1.06
7	Tags_Busy	1.05
9	Tags_Lost to EINS	1.03
3	Lead Source_Referral Sites	1.01
0	Lead Origin_Lead Add Form	0.69
2	Lead Source_Organic Search	0.37
6	Last Activity_Olark Chat Conversation	0.30
13	Last Notable Activity_SMS Sent	0.24
1	Lead Source_Direct Traffic	0.16
12	Tags_Will revert after reading the email	0.14
11	Tags_Ringing	0.05
10	Tags_Not Specified	0.01

Create the second fitted model using Statsmodels

```
#import the statsmodels library

import statsmodels.api as sm

#add a constant

X_train_sm = sm.add_constant(X_train[rfe_col])

#creating the second fitted model

log_mod_2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res2 = log_mod_2.fit()

#read the summary of the model

res2.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6267
Model:	GLM	**Df Residuals:**	6252
Model Family:	Binomial	**Df Model:**	14
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1416.1
Date:	Fri, 26 May 2023	**Deviance:**	2832.2
Time:	20:34:25	**Pearson chi2:**	9.43e+03
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.5839
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.6945	0.184	-20.110	0.000	-4.055	-3.334
Lead Origin_Lead Add Form	0.7961	0.431	1.848	0.065	-0.048	1.640
Lead Source_Direct Traffic	-0.6509	0.124	-5.251	0.000	-0.894	-0.408
Lead Source_Organic Search	-0.0761	0.152	-0.500	0.617	-0.374	0.222
Lead Source_Referral Sites	-0.3902	0.433	-0.902	0.367	-1.238	0.458
Lead Source_Welingak Website	4.2166	1.105	3.817	0.000	2.052	6.382
Last Activity_Email Bounced	-1.4206	0.418	-3.396	0.001	-2.240	-0.601
Last Activity_Olark Chat Conversation	-1.7154	0.223	-7.687	0.000	-2.153	-1.278
Tags_Busy	2.9965	0.267	11.220	0.000	2.473	3.520
Tags_Closed by Horizzon	9.0156	1.024	8.807	0.000	7.009	11.022
Tags_Lost to EINS	7.8168	0.619	12.623	0.000	6.603	9.030
Tags_Not Specified	2.2773	0.179	12.687	0.000	1.925	2.629
Tags_Ringing	-1.0711	0.274	-3.913	0.000	-1.608	-0.535
Tags_Will revert after reading the email	6.8549	0.237	28.892	0.000	6.390	7.320
Last Notable Activity_SMS Sent	2.4857	0.119	20.952	0.000	2.253	2.718

Conclusion:

- The p-value of the variable 'Lead Source_Organic Search' is highest and > 0.05 hence it makes sense to drop the variable.

Similarly, after model 3 the p-value of the variable 'Lead Source_Referral Sites' was highest (0.383) and > 0.05 hence we dropped the variable.

And after model 4, the p-value of the variable 'Lead Origin_Lead Add Form' is highest and > permissible limit (0.05) and hence it makes sense to drop the variable.

V. Building the Model-5

List all the VIF values of predictor variables

[143]:

```
#import the VIF object

from statsmodels.stats.outliers_influence import variance_inflation_factor

#creating a dataframe consisting of all the predictor variables along with their VIF values

vif = pd.DataFrame()
vif['Features'] = X_train[rfe_col].columns
vif['VIF'] = [variance_inflation_factor(X_train[rfe_col].values, i) for i in range(X_train[rfe_col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

:[143]:

	Features	VIF
2	Lead Source_Welingak Website	1.35
6	Tags_Closed by Horizzon	1.15
3	Last Activity_Email Bounced	1.05
5	Tags_Busy	1.04
7	Tags_Lost to EINS	1.02
0	Lead Origin_Lead Add Form	0.68
4	Last Activity_Olark Chat Conversation	0.30
11	Last Notable Activity_SMS Sent	0.24
1	Lead Source_Direct Traffic	0.15
10	Tags_Will revert after reading the email	0.13
9	Tags_Ringing	0.04
8	Tags_Not Specified	0.01

Create the fifth fitted model using Statsmodels

```
#import the statsmodels library

import statsmodels.api as sm

#add a constant

X_train_sm = sm.add_constant(X_train[rfe_col])

#creating the first fitted model

log_mod_5 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res5 = log_mod_5.fit()

#read the summary of the model

res5.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6267
Model:	GLM	**Df Residuals:**	6254
Model Family:	Binomial	**Df Model:**	12
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1416.6
Date:	Fri, 26 May 2023	**Deviance:**	2833.3
Time:	20:34:26	**Pearson chi2:**	9.54e+03
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.5838
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.7263	0.180	-20.713	0.000	-4.079	-3.374
Lead Origin_Lead Add Form	0.8195	0.429	1.908	0.056	-0.022	1.661
Lead Source_Direct Traffic	-0.6284	0.120	-5.238	0.000	-0.864	-0.393
Lead Source_Welingak Website	4.2140	1.105	3.815	0.000	2.049	6.379
Last Activity_Email Bounced	-1.4218	0.417	-3.406	0.001	-2.240	-0.604
Last Activity_Olark Chat Conversation	-1.7035	0.223	-7.644	0.000	-2.140	-1.267
Tags_Busy	3.0036	0.267	11.247	0.000	2.480	3.527
Tags_Closed by Horizzon	9.0213	1.024	8.813	0.000	7.015	11.028
Tags_Lost to EINS	7.8297	0.619	12.647	0.000	6.616	9.043
Tags_Not Specified	2.2881	0.179	12.769	0.000	1.937	2.639
Tags_Ringing	-1.0622	0.274	-3.882	0.000	-1.598	-0.526
Tags_Will revert after reading the email	6.8608	0.237	28.910	0.000	6.396	7.326
Last Notable Activity_SMS Sent	2.4878	0.118	21.013	0.000	2.256	2.720

Conclusion:
- As we can see the VIF < 5 for all variables and p-value < 0.05 for all variables hence we can say that **Model-5 looks to be our best model.**

4. Deriving Probabilities, Predictions and Lead Score on Train Data

[145]:
```
#getting predicted values on train data

y_train_pred = res5.predict(X_train_sm)
y_train_pred[:10]
```

[145]:
```
9196    0.091071
4696    0.050738
3274    0.740706
2164    0.004365
1667    0.958296
7024    0.191826
8018    0.041422
778     0.191826
6942    0.004365
4440    0.112388
dtype: float64
```

```
#creating a dataframe which has y_train dataset values and corresponding y_train predicted values as learnt by X_train

y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Converted_prob':y_train_pred})
y_train_pred_final['Prospect ID'] = y_train.index
y_train_pred_final.head()
```

	Converted	Converted_prob	Prospect ID
9196	1	0.091071	9196
4696	0	0.050738	4696
3274	0	0.740706	3274
2164	0	0.004365	2164
1667	1	0.958296	1667

Lets take 0.5 as cut-off value for deciding whether a lead will be converted or not

```
y_train_pred_final['Predicted'] = y_train_pred_final.Converted_prob.map(lambda x: 1 if x > 0.5 else 0)

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

	Converted	Converted_prob	Prospect ID	Predicted
9196	1	0.091071	9196	0
4696	0	0.050738	4696	0
3274	0	0.740706	3274	1
2164	0	0.004365	2164	0
1667	1	0.958296	1667	1

5. Confusion Matrix

[148]:
```
#import metrics function

from sklearn import metrics

# Confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Predicted )
print(confusion)
```

```
[[3730  152]
 [ 332 2053]]
```

[149]:
```
#checking the overall accuracy

print(metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.Predicted))
```

```
0.922770065422052
```

[150]:
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

[151]:
```
#sensitivity
TP / float(TP+FN)
```

[151]:
```
0.8607966457023061
```

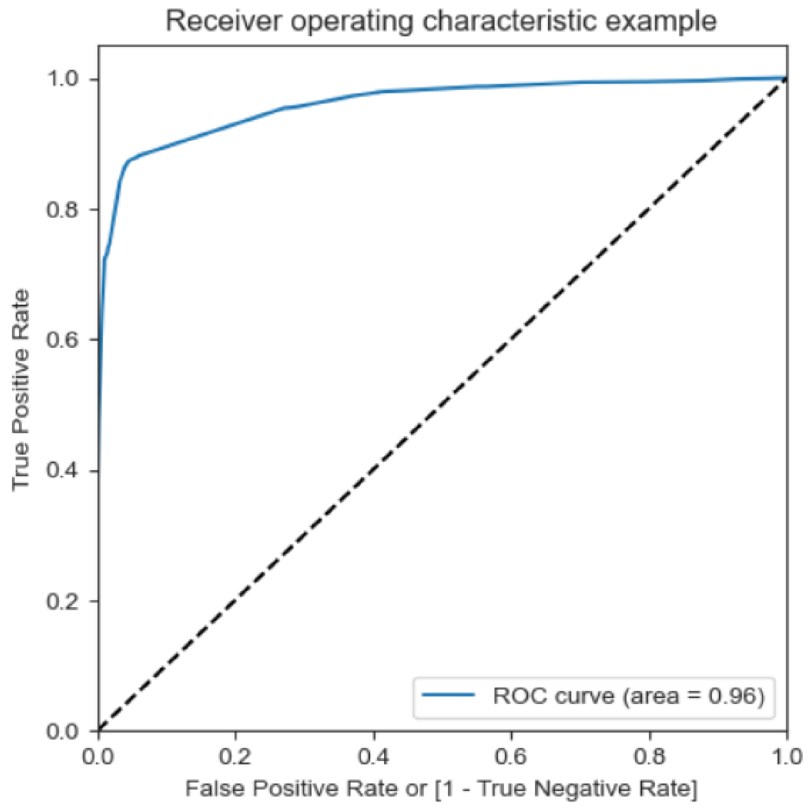
[152]:
```
#specificity
TN / float(TN+FP)
```
[152]: 0.9608449252962391

[153]:
```
#False Postive Rate - predicting conversion when lead does not have convert
print(FP/ float(TN+FP))
```
0.03915507470376095

[154]:
```
#Positive predictive value
print (TP / float(TP+FP))
```
0.9310657596371882

[155]:
```
#Negative predictive value
print (TN / float(TN+ FN))
```
0.9182668636139832

6. Plotting ROC Curve

[159]:
```
draw_roc(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
```


Conclusion:
- ROC curve should be a value closer to 1 for a good model. We have got a value of 0.96 which is extremely good.

7. Optimal Cut-Off

```python
#Creating columns with different probability cutoffs
nums = [float(x)/10 for x in range(10)]
for i in nums:
    y_train_pred_final[i]= y_train_pred_final.Converted_prob.map(lambda x: 1 if x > i else 0)
y_train_pred_final.head()
```

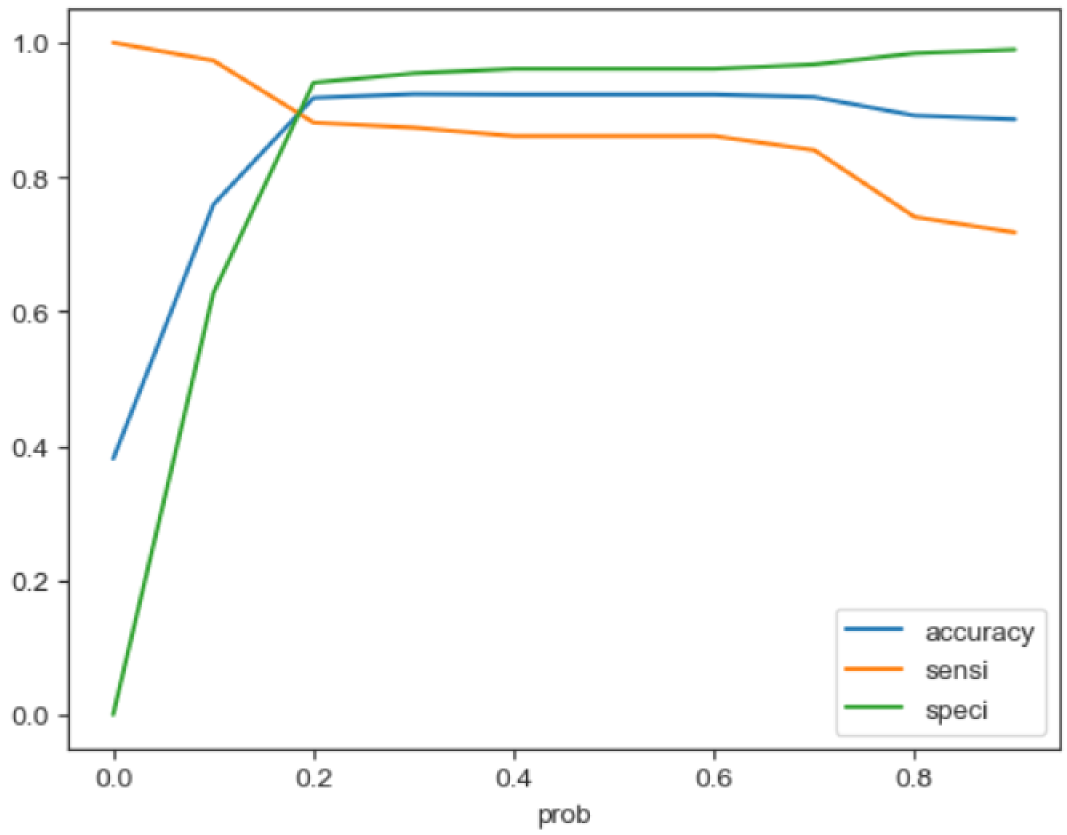
	Converted	Converted_prob	Prospect ID	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
9196	1	0.091071	9196	0	1	0	0	0	0	0	0	0	0	0
4696	0	0.050738	4696	0	1	0	0	0	0	0	0	0	0	0
3274	0	0.740706	3274	1	1	1	1	1	1	1	1	1	0	0
2164	0	0.004365	2164	0	1	0	0	0	0	0	0	0	0	0
1667	1	0.958296	1667	1	1	1	1	1	1	1	1	1	1	1

By making a table giving predicted values for different cut-offs, we can better decide on what the cut-off value should be.

Calculating accuracy sensitivity and specificity for various probability cutoffs.

	prob	accuracy	sensi	speci
0.0	0.0	0.380565	1.000000	0.000000
0.1	0.1	0.758896	0.973166	0.627254
0.2	0.2	0.917664	0.880922	0.940237
0.3	0.3	0.923408	0.873375	0.954147
0.4	0.4	0.922770	0.860797	0.960845
0.5	0.5	0.922770	0.860797	0.960845
0.6	0.6	0.922770	0.860797	0.960845
0.7	0.7	0.918781	0.839832	0.967285
0.8	0.8	0.891336	0.740461	0.984029
0.9	0.9	0.885910	0.717400	0.989438

[163]:
```python
# Let's plot accuracy sensitivity and specificity for various probabilities.
cut_off_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



As we see from the above line plot, 0.2 seems to be the most ideal cut-off point.

In [165]:
```
#using 0.2 as cut-off point for predicting lead conversion

y_train_pred_final['final_Predicted'] = y_train_pred_final.Converted_prob.map( lambda x: 1 if x > 0.2 else 0)

y_train_pred_final.head()
```

Out[165]:

	Converted	Converted_prob	Prospect ID	Predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_Predicted
9196	1	0.091071	9196	0	1	0	0	0	0	0	0	0	0	0	0
4696	0	0.050738	4696	0	1	0	0	0	0	0	0	0	0	0	0
3274	0	0.740706	3274	1	1	1	1	1	1	1	1	1	0	0	1
2164	0	0.004365	2164	0	1	0	0	0	0	0	0	0	0	0	0
1667	1	0.958296	1667	1	1	1	1	1	1	1	1	1	1	1	1

In [166]:
```
#calculating the lead score for each lead and displaying only required columns

y_train_pred_final['Lead_Score'] = y_train_pred_final.Converted_prob.map( lambda x: round(x*100))

y_train_pred_final[['Converted','Converted_prob','Prospect ID','final_Predicted','Lead_Score']].head()
```

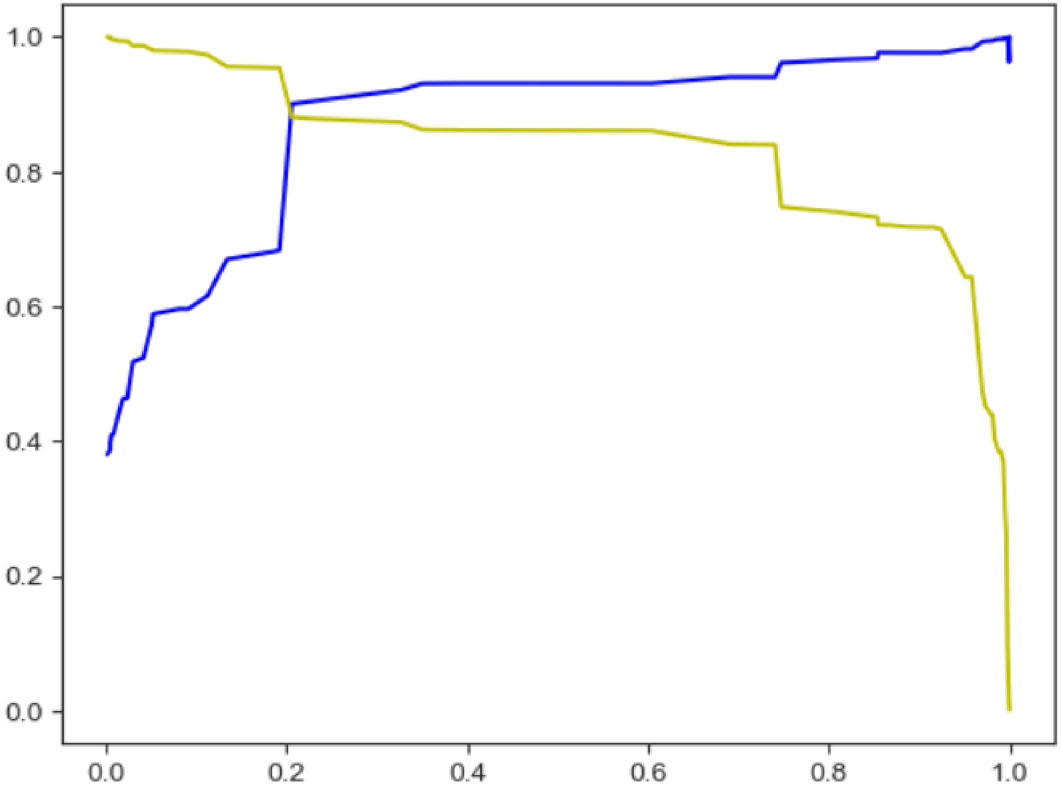
Out[166]:

	Converted	Converted_prob	Prospect ID	final_Predicted	Lead_Score
9196	1	0.091071	9196	0	9
4696	0	0.050738	4696	0	5
3274	0	0.740706	3274	1	74
2164	0	0.004365	2164	0	0
1667	1	0.958296	1667	1	96

```
8. Final Analysis On Training Dataset
******************************
3650   232
284    2101
******************************
Accuracy --> 91.77
Sensitivity --> 88.09
Specificity --> 94.02
Precision --> 90.06
Recall --> 88.09
```

```python
#plotting recall and precision curve

y_train_pred_final.Converted, y_train_pred_final.final_Predicted
p, r, thresholds = precision_recall_curve(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
```

```python
plt.plot(thresholds, p[:-1], "b-")
plt.plot(thresholds, r[:-1], "y-")
plt.show()
```



8. Final Analysis On Training Dataset

- Our model seems to be performing very well as our ROC curve has a value of 0.96 which is extremely good.
- Some important statistics of our model:
  - Accuracy --> 91.77
  - Sensitivity --> 88.09
  - Specificity --> 94.02
  - Precision --> 90.06
  - Recall --> 88.09

F. Making Predictions On Test Dataset With Model Built From Training Dataset

1. Rescaling the Test Dataset Values

[185]:
```
#extracting the numerical columns and assigning them to a variable

num_cols=X_test.select_dtypes(include=['float64', 'int64']).columns

#scaling the test dataset values

X_test[num_cols] = scaler.fit_transform(X_test[num_cols])
```

[187]:
```
#adding a constant

X_test_sm = sm.add_constant(X_test)
```

[188]:
```
#predicting the values in test dataset based on model built on training dataset

y_test_pred = res5.predict(X_test_sm)
```

[189]:
```
y_test_pred[:10]
```

[189]:
```
7681    0.050738
984     0.023517
8135    0.603783
6915    0.008257
2712    0.958296
244     0.008257
4698    0.023517
8287    0.041422
6791    0.958296
8970    0.023517
dtype: float64
```

[190]:
```
# Converting y_pred to a dataframe which is an array

y_pred_1 = pd.DataFrame(y_test_pred)
```

Let's check converted probability of our test dataset

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

```
y_pred_final.head()
```

	Converted	Prospect ID	Converted_prob
0	0	7681	0.050738
1	0	984	0.023517
2	0	8135	0.603783
3	0	6915	0.008257
4	1	2712	0.958296

```
#assigning probabability score based on cutoff of 0.2

y_pred_final['final_Predicted'] = y_pred_final.Converted_prob.map(lambda x: 1 if x > 0.2 else 0)
```

```
y_pred_final.head()
```

	Converted	Prospect ID	Converted_prob	final_Predicted
0	0	7681	0.050738	0
1	0	984	0.023517	0
2	0	8135	0.603783	1
3	0	6915	0.008257	0
4	1	2712	0.958296	1

```
#checking the overall accuracy

accuracy = metrics.accuracy_score(y_pred_final.Converted, y_pred_final.final_Predicted)
accuracy
```

```
0.9225614296351452
```

4. Final Analysis On Test Dataset

- Upon running the trained model on the test dataset we obtain the following figures:
  - Accuracy --> 92.26
    - Sensitivity --> 88.71
    - Specificity --> 94.39
    - Precision --> 90.51
    - Recall --> 88.71

G. Conclusion

Variables to be focussed on for bettering the lead conversion ratio:

- Lead Source_Welingak Website
- Tags_Closed by Horizzon
- Last Activity_Email Bounced
- Tags_Busy
- Tags_Lost to EINS
- Lead Origin_Lead Add Form
- Last Activity_Olark Chat Conversation
- Last Notable Activity_SMS Sent
- Lead Source_Direct Traffic
- Tags_Will revert after reading the email
- Tags_Ringing
- Tags_Not Specified

Comparing the values obtained by our Train and Test dataset:

- Train Dataset
  - Accuracy --> 91.77
  - Sensitivity --> 88.09
  - Specificity --> 94.02
  - Precision --> 90.06
  - Recall --> 88.09
- Test Dataset
  - Accuracy --> 92.26
  - Sensitivity --> 88.71
  - Specificity --> 94.39
  - Precision --> 90.51
  - Recall --> 88.71

Suggestions:

Hence, we can make the following predictions based on the model we have arrived at:

1. Welingak Website is the biggest source for leads which is giving a higher conversion rate, hence we can look into investing into increasing visibility on that website.
2. If the Tag given to a potential lead is 'closed by Horrizon' , 'Busy' , 'Lost to EINS', 'Will revert after reading the email' , or 'Ringing' its better to follow up with them as they have a higher chance of conversion
3. If a lead is having an 'Olark Chat Conversation' or sending an SMS as their last activity, they have a higher chance of conversion, hence it's better to follow up.