CASE STUDY

Lead scoring

Shashank Verma
&
Dhruv Kartikey

IMPORTED THE “LEADS” DATASET

```
In [4]: xedu_data = pd.read_csv("Leads.csv")
        xedu_data.head()
```

Out[4]:

	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	Asymmetri Profile In
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	...	No	Select	Select	02.Medium	02.Med
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	...	No	Select	Select	02.Medium	02.Med
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.H
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.H
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.H

5 rows × 37 columns

Data Pre-Processing

In [7]:
```
#Data Pre-Processing Step
xedu_data = xedu_data.replace('Select', np.nan)
##Specialisation as 'Select' is of no significance
```

In [8]:
```
xedu_data.shape
```

Out[8]:
```
(9240, 37)
```

In [9]:
```
xedu_data = xedu_data.drop(xedu_data.loc[:,list(round(100*(xedu_data.isnull().sum()/len(xedu_data.index)), 2)>70)].columns, 1)
```

In [10]:
```
xedu_data.shape
```

Out[10]:
```
(9240, 35)
```

In [11]:
```
##Checking if we have prospect id are all primary key or Unique
xedu_data.duplicated(subset='Prospect ID')
##Result : No Duplicate prospect ID is there
```

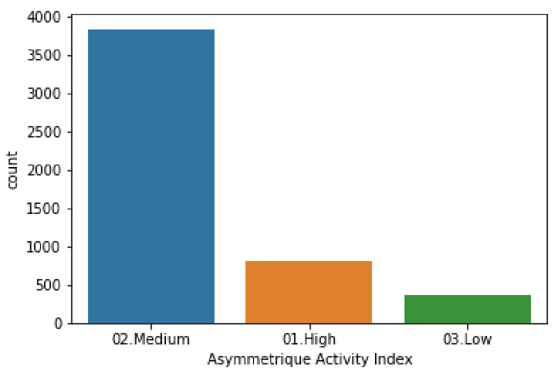
Out[11]:
```
0       False
1       False
2       False
3       False
4       False
5       False
6       False
7       False
8       False
9       False
10      False
```

Data Pre-Processing…

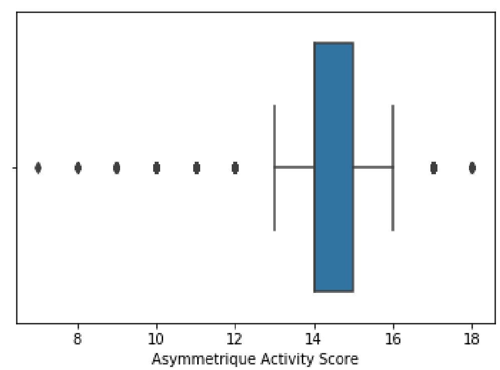
```
In [12]: ## Running a check for nulls
         round(100*(xedu_data.isnull().sum()/len(xedu_data.index)))

Out[12]: Prospect ID                                       0.0
         Lead Number                                       0.0
         Lead Origin                                       0.0
         Lead Source                                       0.0
         Do Not Email                                      0.0
         Do Not Call                                       0.0
         Converted                                         0.0
         TotalVisits                                       1.0
         Total Time Spent on Website                       0.0
         Page Views Per Visit                              1.0
         Last Activity                                     1.0
         Country                                          27.0
         Specialization                                   37.0
         What is your current occupation                  29.0
         What matters most to you in choosing a course    29.0
         Search                                            0.0
         Magazine                                          0.0
         Newspaper Article                                 0.0
         X Education Forums                                0.0
         Newspaper                                         0.0
         Digital Advertisement                             0.0
         Through Recommendations                           0.0
         Receive More Updates About Our Courses            0.0
         Tags                                             36.0
         Lead Quality                                     52.0
         Update me on Supply Chain Content                 0.0
         Get updates on DM Content                         0.0
         City                                             40.0
         Asymmetrique Activity Index                      46.0
```

EXPLORING THE "ASYMMETRIQUE%" ATTRIBUTES

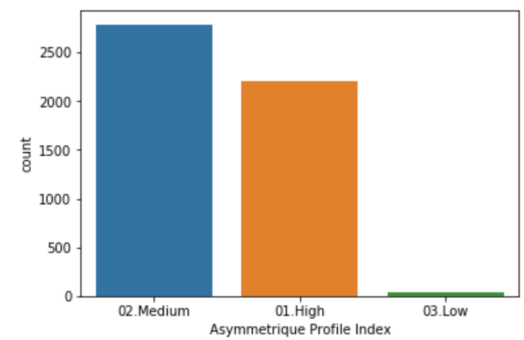
```
sns.countplot(xedu_data['Asymmetrique Activity Index'])
<matplotlib.axes._subplots.AxesSubplot at 0x1be3e3e36d8>
```



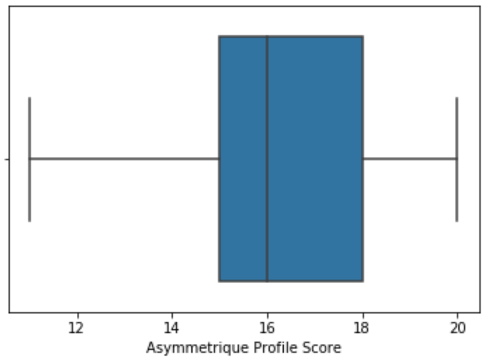

```
sns.boxplot(xedu_data['Asymmetrique Activity Score'])
<matplotlib.axes._subplots.AxesSubplot at 0x1be3e6b8748>
```




```
sns.countplot(xedu_data['Asymmetrique Profile Index'])
<matplotlib.axes._subplots.AxesSubplot at 0x1be3ecd3550>
```




```
sns.boxplot(xedu_data['Asymmetrique Profile Score'])
<matplotlib.axes._subplots.AxesSubplot at 0x1be3ec71128>
```



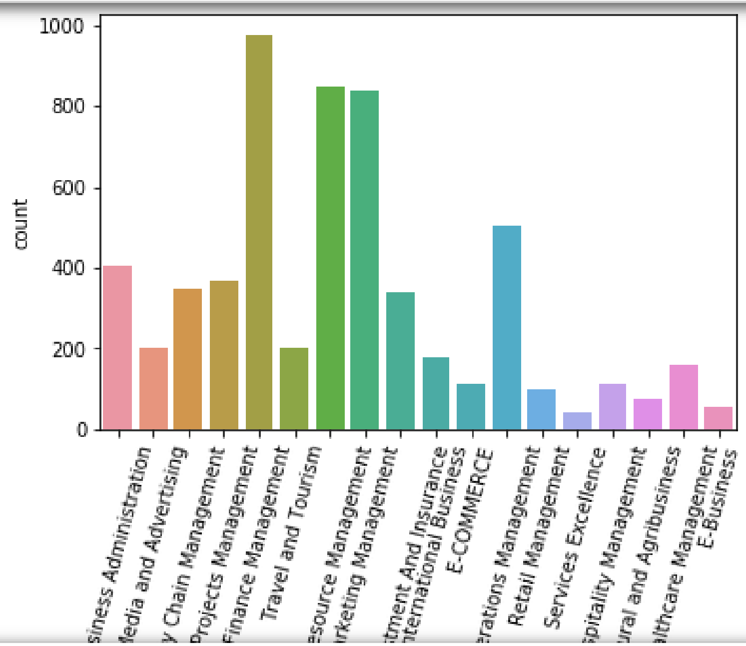
High Variation in the attributes "Asymmmetrique" and also 46% of values are null,
so, it will not contribute well to the model

Lets drop these features:

```
: xedu_data = xedu_data.drop(['Asymmetrique Activity Index','Asymmetrique Activity Score','Asymmetrique Profile Index','Asymmetriq
```

ANALYZING THE ATTRIBUTE 'SPECIALISATION'

```
#Analysing the attribute 'specialisation'
plt.xticks(rotation = 75)
sns.countplot(xedu_data.Specialization)
```

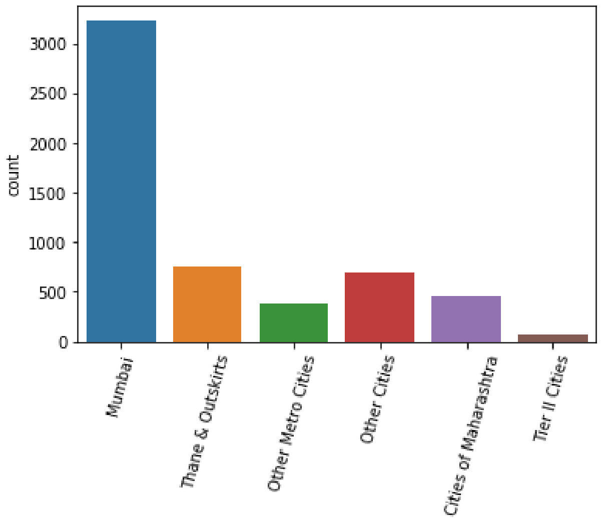


Since 37% of the values in Specialization are null. But our assumption here is that the potential candidate here might be in a general job with no specialization. so will mark it as 'general'.

ANALYZING THE ATTRIBUTE 'CITY'

```
plt.xticks(rotation = 75)
sns.countplot(xedu_data.City)
```

: <matplotlib.axes._subplots.AxesSubplot at 0x1be3e43ef28>



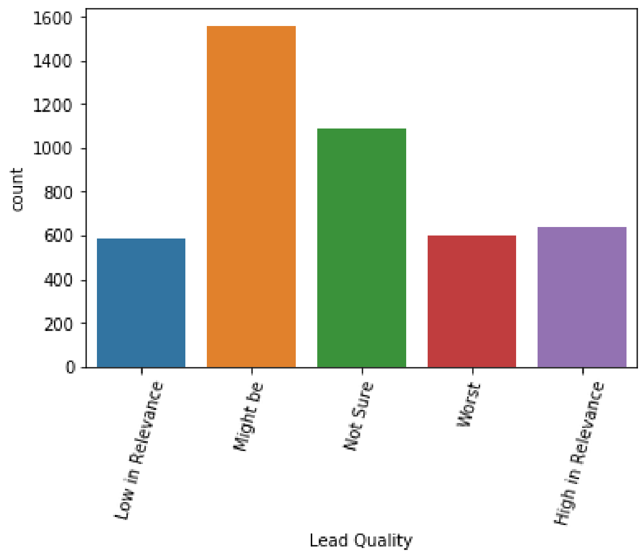
Since most of the bent is toward Mumbai as a city.We can replace blanks with Mumbai

```
xedu_data['City'] = xedu_data['City'].replace(np.nan, 'Mumbai')
```

ANALYZING THE ATTRIBUTE 'LEAD QUALITY'

```
plt.xticks(rotation = 75)
sns.countplot(xedu_data['Lead Quality'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3e4b9be0>
```



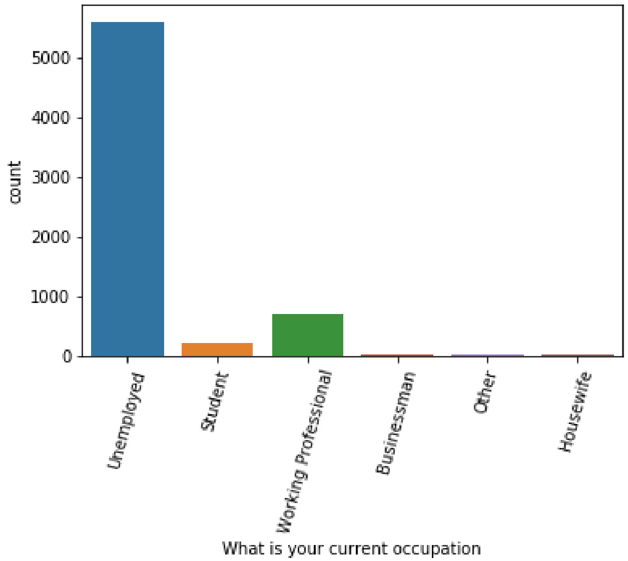
#52% of lead Quality data is blank. But here the spread of classification is proper.
But a 'Not Sure' classification will neither be positive nor negative. So we can consider
the null values in the category of 'Not Sure'

```
xedu_data['Lead Quality'] = xedu_data['Lead Quality'].replace(np.nan, 'Not Sure')
```

ANALYZING THE ATTRIBUTE 'WHAT IS YOUR CURRENT OCCUPATION'

```
plt.xticks(rotation = 75)
sns.countplot(xedu_data['What is your current occupation'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3e4aa400>
```



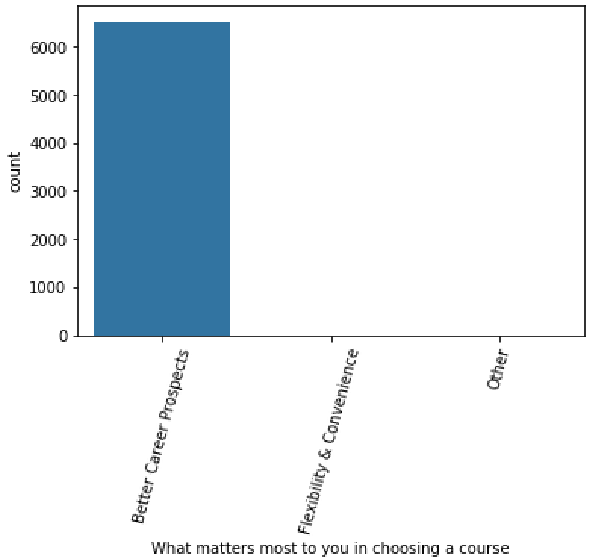
#Since 29% of the 'What is your occupation' is null. And seeing the graph the "Unemployed" #Contributes to majority of the data. Will replace nulls with 'Unemployed'

```
xedu_data['What is your current occupation'] = xedu_data['What is your current occupation'].replace(np.nan, 'Unemployed')
```

ANALYZING THE ATTRIBUTE 'WHAT MATTERS MOST TO YOU IN CHOOSING A COURSE'

```
plt.xticks(rotation = 75)
sns.countplot(xedu_data['What matters most to you in choosing a course'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3e4b9fd0>
```



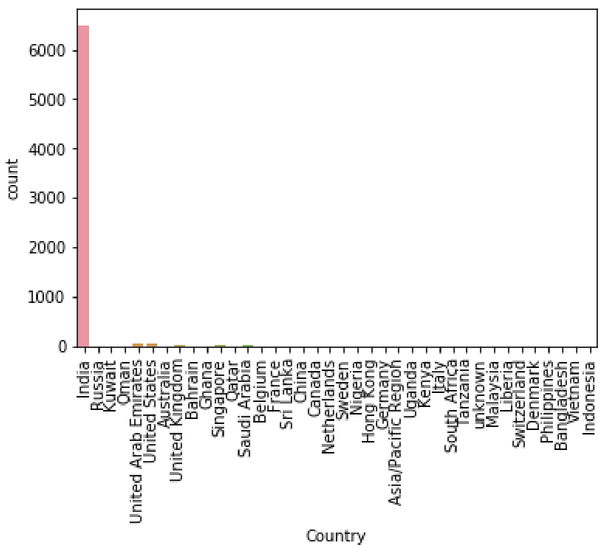
Since 29% of the values are null, we can replace the null with the majority of the occurence
i.e. Better Career Prospects , over here

```
xedu_data['What matters most to you in choosing a course'] = xedu_data['What matters most to you in choosing a course'].replace(np.nan, 'Better Career Prospects')
```

ANALYZING THE ATTRIBUTE 'COUNTRY'

```
plt.xticks(rotation = 90)
sns.countplot(xedu_data['Country'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x1be3e5c4048>



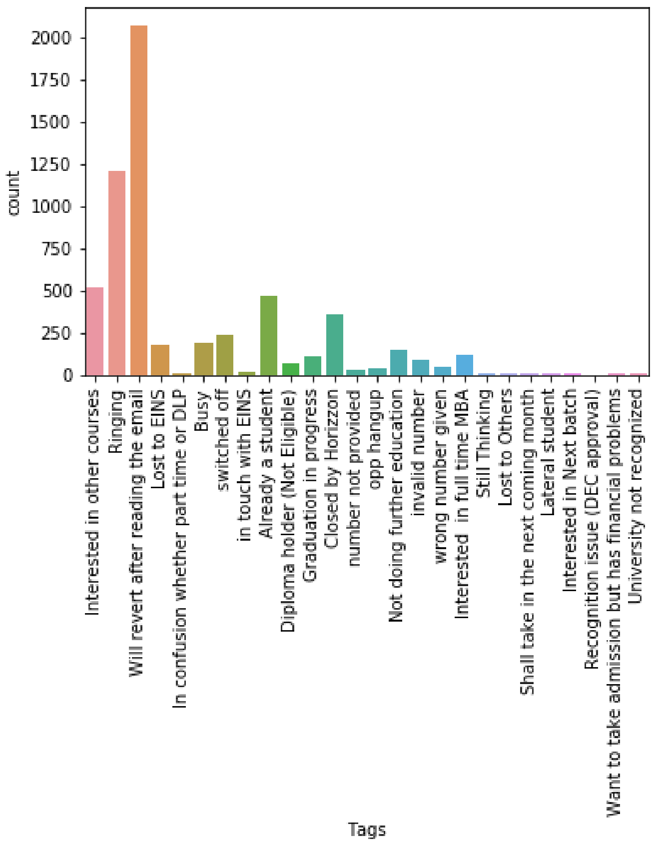
In 'Country', majority of the Data is towards India. And we have around 27% of the data
which is null . So we can easily replace the nulls in the kitty of India

```
xedu_data['Country'] = xedu_data['Country'].replace(np.nan, 'India')
```

ANALYZING THE ATTRIBUTE 'TAGS'

```
plt.xticks(rotation = 90)
sns.countplot(xedu_data['Tags'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3efd6940>
```



We have around 36% of the null data for 'Tags', and seeing the plot, most of the data is
bending to 'Will revert after reading the email'. So we can replace the nulls in the same kitty

xedu_data['Tags'] = xedu_data['Tags'].replace(np.nan, 'Will revert after reading the email')

ANALYZING THE MODIFIED DATA

1. Lets drop others which are having null in the range of 1 or 2

2. Checking for the null values now for the whole dataset

```
# Dropping other which are having null in the range of 1 or 2
xedu_data.dropna(inplace = True)
```

```
# Checking for the null values now for the whole dataset
round(100*(xedu_data.isnull().sum()/len(xedu_data.index)))
```

3. Describe the data to see the statistical details of it.

```
xedu_data.describe()
```

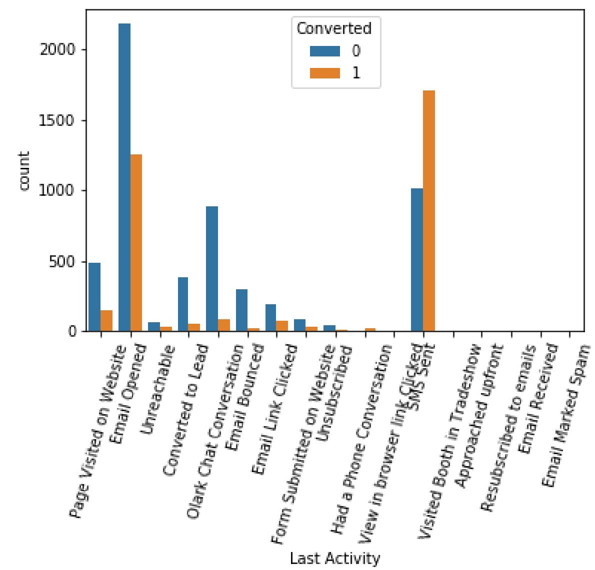
	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit
count	9074.000000	9074.000000	9074.000000	9074.000000	9074.000000
mean	617032.619352	0.378554	3.456028	482.887481	2.370151
std	23348.029512	0.485053	4.858802	545.256560	2.160871
min	579533.000000	0.000000	0.000000	0.000000	0.000000
25%	596406.000000	0.000000	1.000000	11.000000	1.000000
50%	615278.500000	0.000000	3.000000	246.000000	2.000000
75%	637176.500000	1.000000	5.000000	922.750000	3.200000
max	660737.000000	1.000000	251.000000	2272.000000	55.000000

Performing “Exploratory Data Analysis”

1. Univariate Analysis for 'Last Activity'

```
plt.xticks(rotation = 75)
sns.countplot(x = "Last Activity", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3f13e160>
```

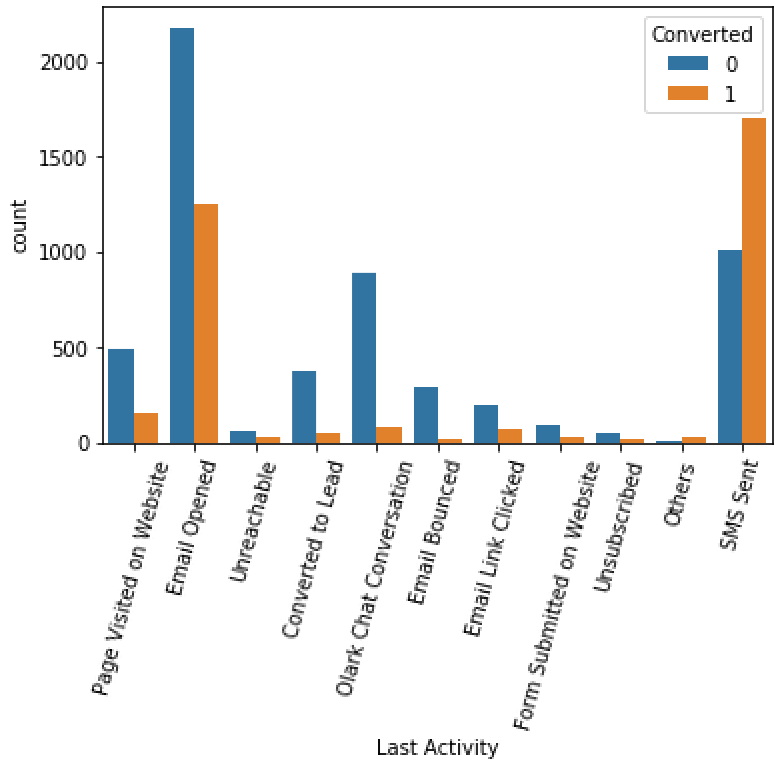



```
# Since categories such as 'Had a Phone Conversation', 'View in browser link Clicked',
#'Visited Booth in Tradeshow', 'Approached upfront','Resubscribed to emails','Email Received',
#'Email Marked Spam' are negligible in justification clubbing them in the category of "Others"
xedu_data['Last Activity'] = xedu_data['Last Activity'].replace(['Had a Phone Conversation', 'View in browser link Clicked',
                                                      'Visited Booth in Tradeshow', 'Approached upfront',
                                                      'Resubscribed to emails','Email Received', 'Email Marked Spam'], 'Others'
```

CONTINUED…

```
# Last Activity new justification post normalising the data
plt.xticks(rotation = 75)
sns.countplot(x = "Last Activity", hue = "Converted", data = xedu_data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1be3ee043c8>

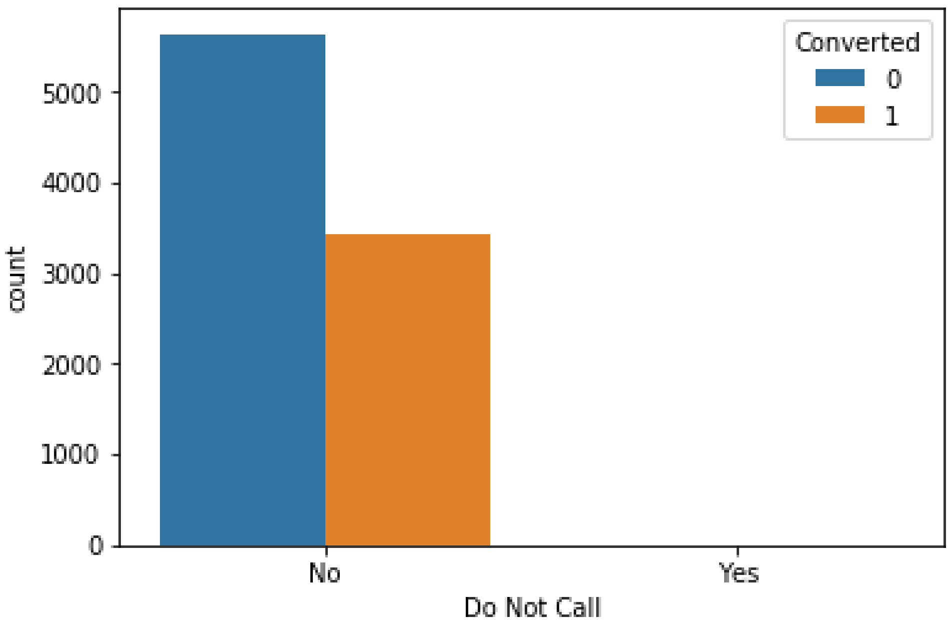


CONTINUED…

2. Univariate Analysis for 'Do Not Call'

```
# Univariate Analysis for 'Do Not Call'
sns.countplot(x = "Do Not Call", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3ee96d30>
```

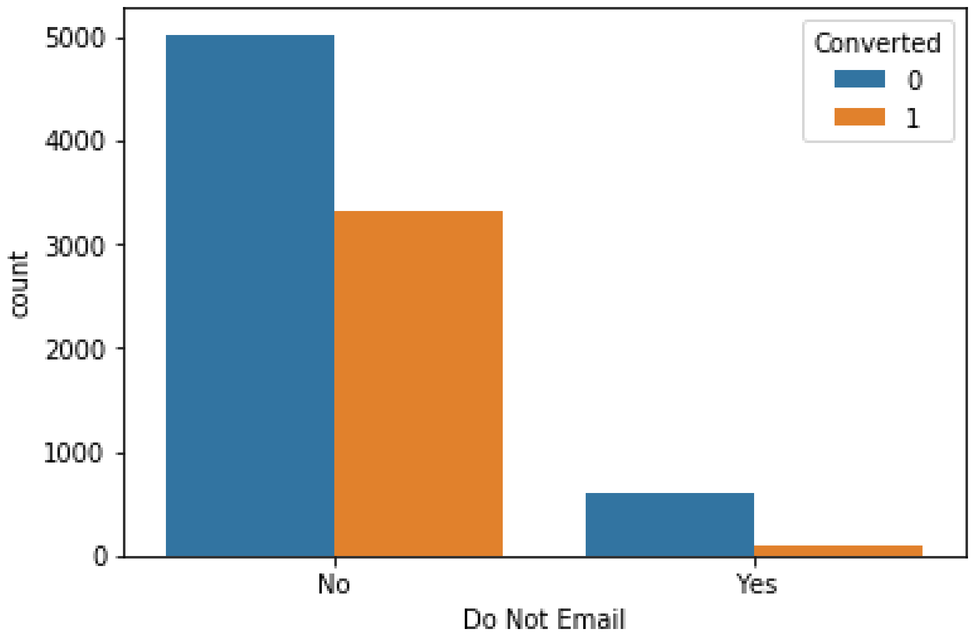


CONTINUED…

3. Univariate Analysis for Do Not Email'

```
# Univariate Analysis for 'Do Not Email'
sns.countplot(x = "Do Not Email", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3eee2fd0>
```

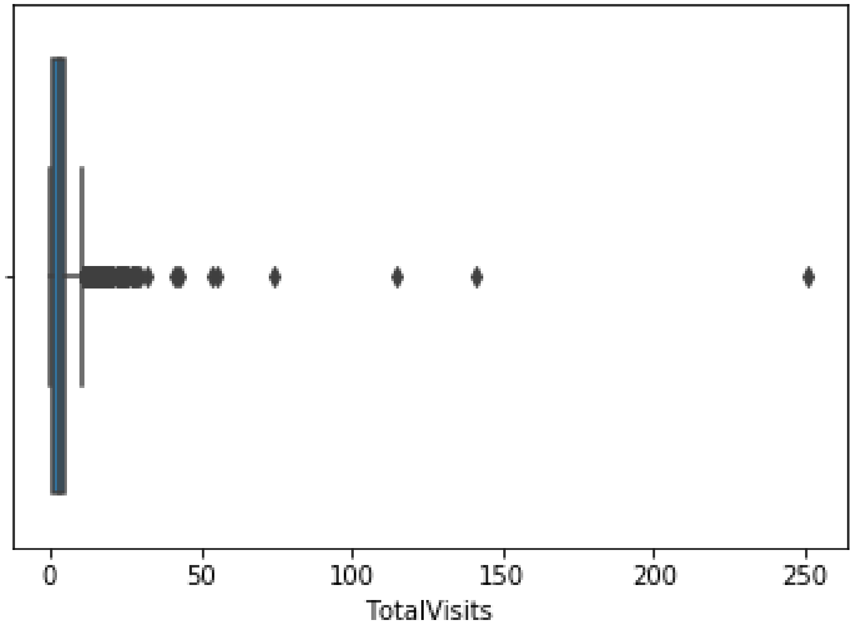


CONTINUED…

4. Univariate Analysis for 'TotalVisits'

```
# Univariate Analysis for 'TotalVisits'
sns.boxplot(xedu_data['TotalVisits'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40329550>
```

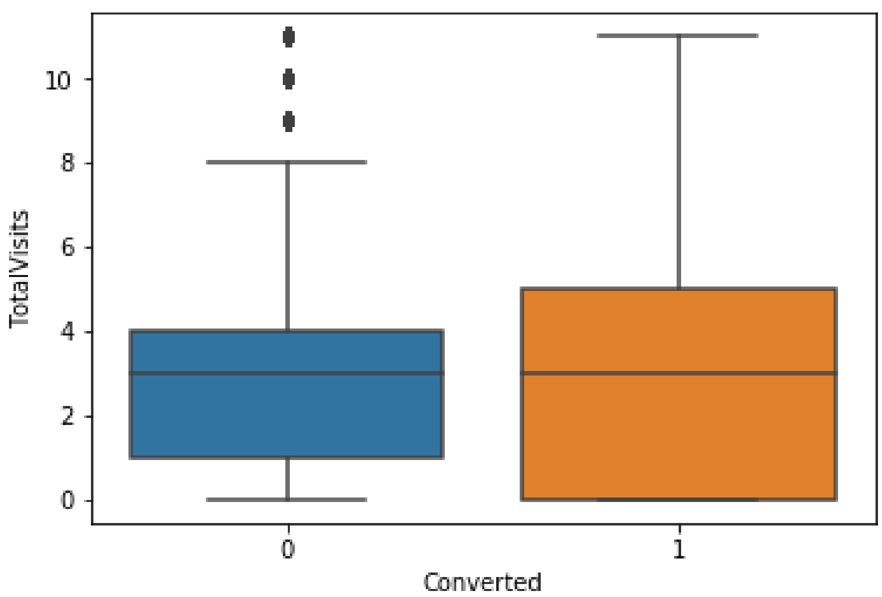


CONTINUED…

Removing Outliers

```
## Removing Outliers at 97% for TotalVisits
per = xedu_data['TotalVisits'].quantile([0.03,0.97]).values
xedu_data['TotalVisits'][xedu_data['TotalVisits'] <= per[0]] = per[0]
xedu_data['TotalVisits'][xedu_data['TotalVisits'] >= per[1]] = per[1]
sns.boxplot(y = 'TotalVisits', x = 'Converted', data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be4037dba8>
```

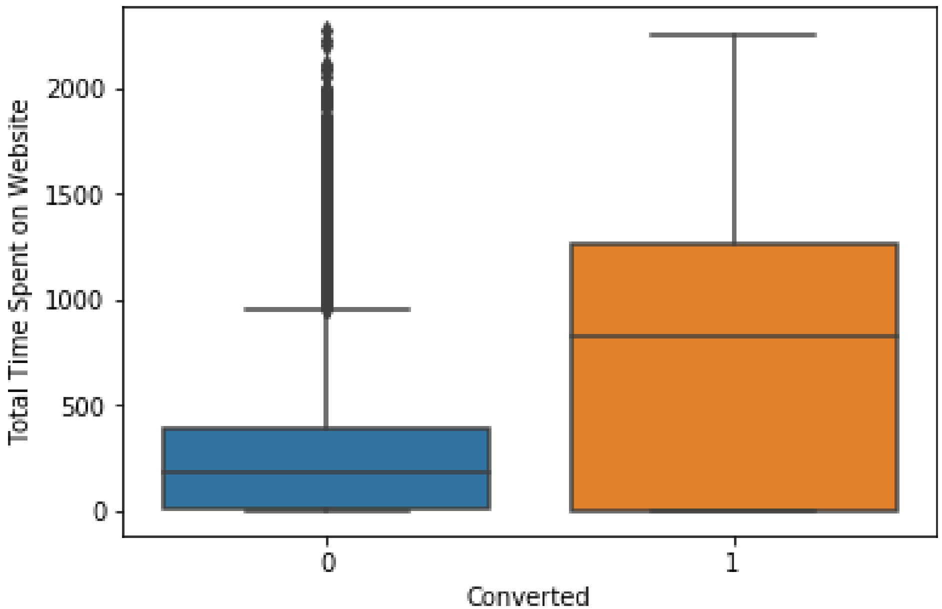


CONTINUED…

5. Univariate Analysis for 'Total Time Spent on Website'

```
# Univariate Analysis for 'Total Time Spent on Website'
sns.boxplot(y = 'Total Time Spent on Website', x = 'Converted', data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be403e9b00>
```

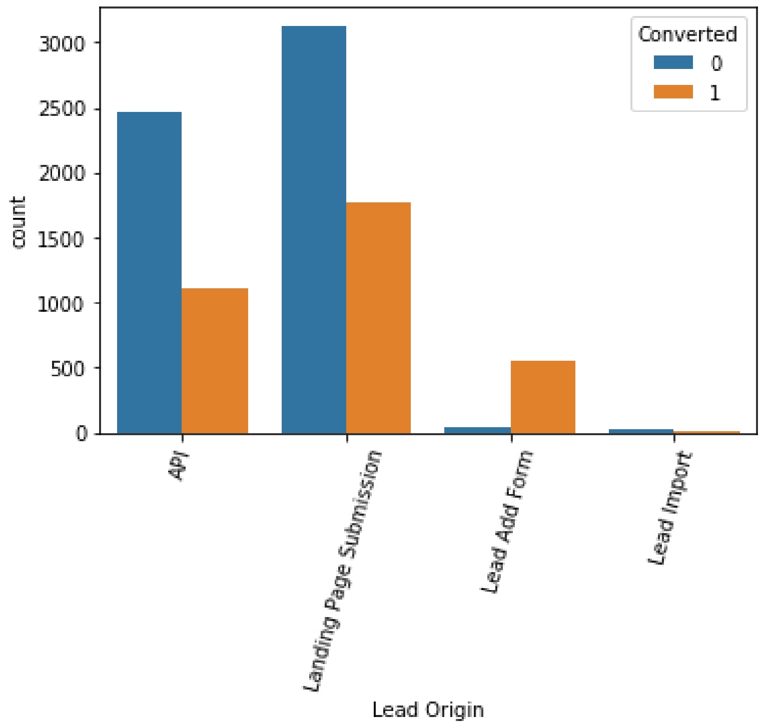


CONTINUED…

6. Univariate Analysis for ‘Lead Origin'

```
# Univariate Analysis for 'Lead Origin'
plt.xticks(rotation = 75)
sns.countplot(x = "Lead Origin", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40457940>
```

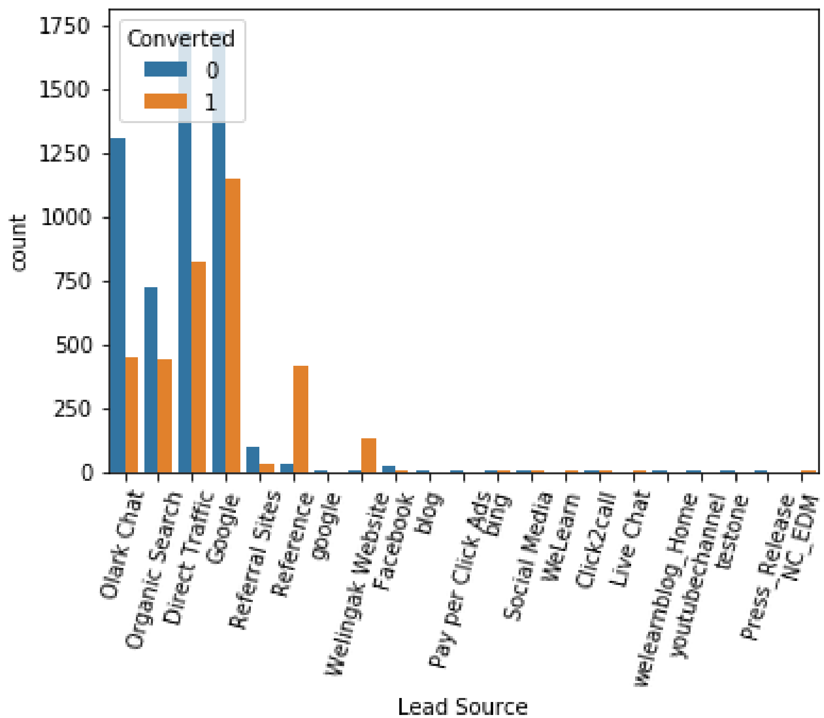


CONTINUED…

7. Univariate Analysis for ‘Lead Source'

```
# Univariate Analysis for 'Lead Source'
plt.xticks(rotation = 75)
sns.countplot(x = "Lead Source", hue = "Converted", data = xedu_data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1be404ba780>

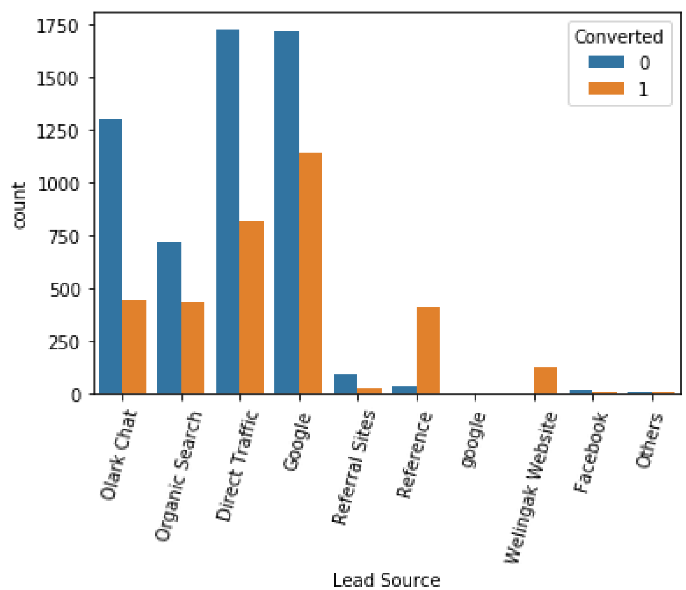


CONTINUED…

Replacing irrelevant values with "Others"

```
## Replacing all the values which holds negligible presence under the tag of "Others"
xedu_data['Lead Source'] = xedu_data['Lead Source'].replace(['Click2call', 'Live Chat', 'NC_EDM', 'Pay per Click Ads', 'Press_Rel
  'Social Media', 'WeLearn', 'bing', 'blog', 'testone', 'welearnblog_Home', 'youtubechannel'], 'Others')
plt.xticks(rotation = 75)
sns.countplot(x = "Lead Source", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40387160>
```

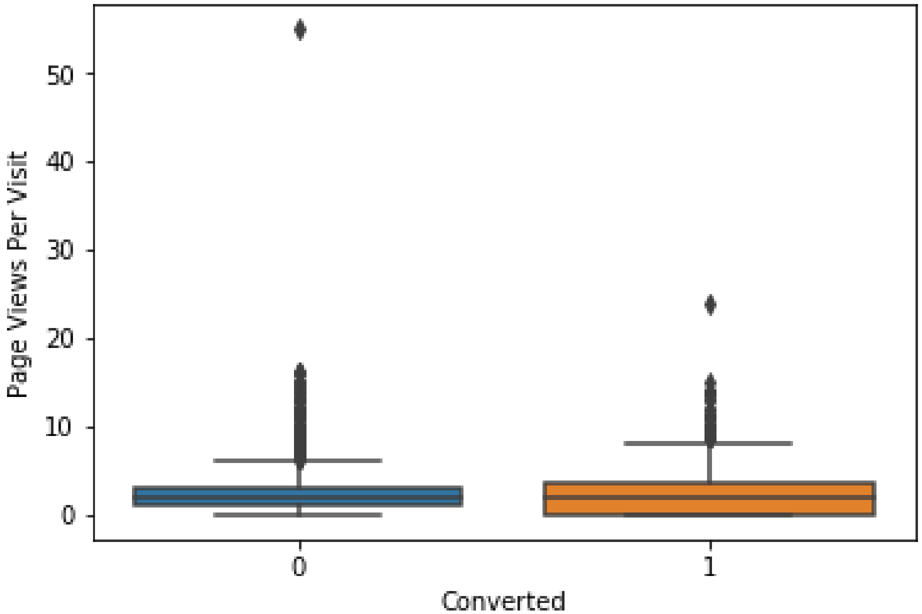


CONTINUED…

8. Univariate Analysis for ‘Page Views Per Visit'

```
# Univariate Analysis for 'Page Views Per Visit'
sns.boxplot(y = 'Page Views Per Visit', x = 'Converted', data = xedu_data)
```

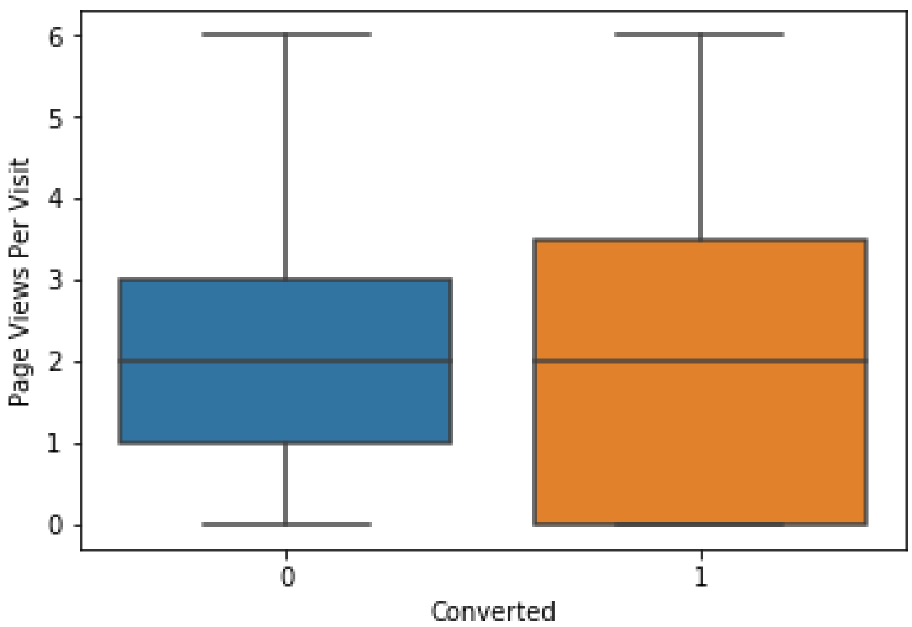
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be4064aa58>
```



CONTINUED…

```
## Removing Outliers at 98% for TotalVisits
per = xedu_data['Page Views Per Visit'].quantile([0.03,0.95]).values
xedu_data['Page Views Per Visit'][xedu_data['Page Views Per Visit'] <= per[0]] = per[0]
xedu_data['Page Views Per Visit'][xedu_data['Page Views Per Visit'] >= per[1]] = per[1]
sns.boxplot(y = 'Page Views Per Visit', x = 'Converted', data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be406c4208>
```

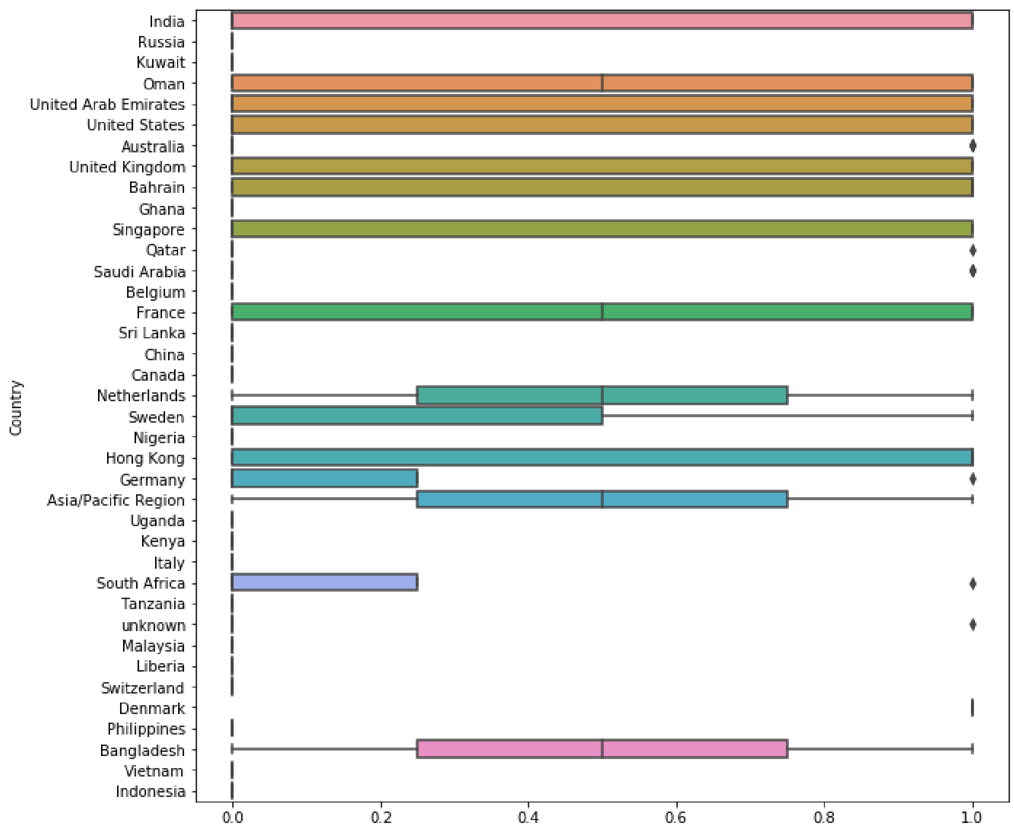


CONTINUED…

9. Univariate Analysis for 'Country'

```
# Univariate Analysis for 'Country'
plt.subplots(figsize = (10,10))
sns.boxplot(y = 'Country', x = 'Converted', data = xedu_data)
```

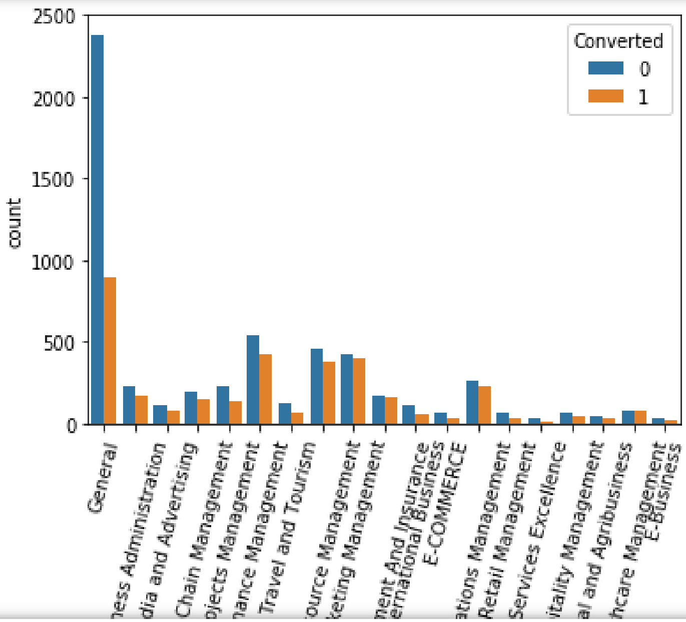
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40729320>
```



CONTINUED…

10. Univariate Analysis for ‘Specialisation'

```
# Univariate Analysis for 'Specialisation'
plt.xticks(rotation = 75)
sns.countplot(x = "Specialization", hue = "Converted", data = xedu_data)
```

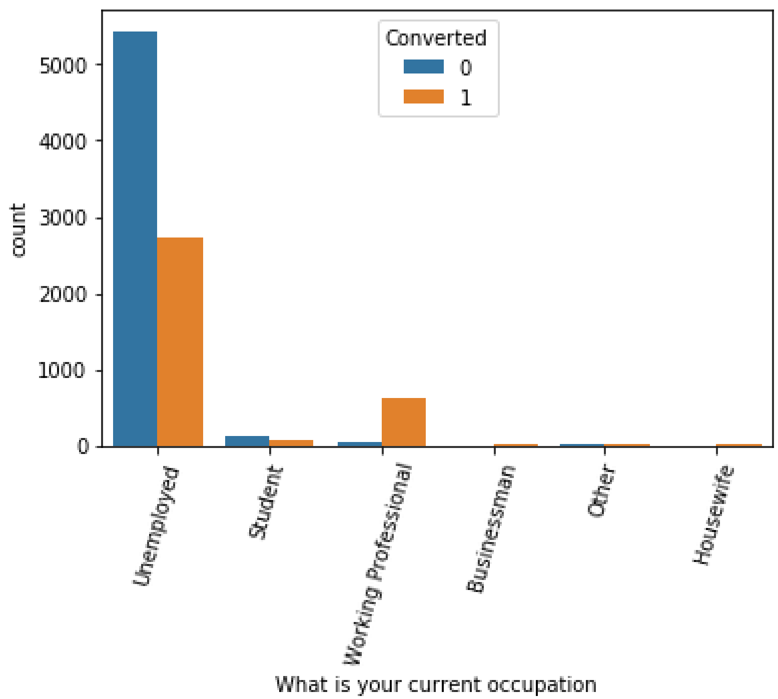


CONTINUED…

11. Univariate Analysis for ‘What is your current occupation'

```
# Univariate Analysis for 'What is your current occupation'
plt.xticks(rotation = 75)
sns.countplot(x = "What is your current occupation", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40a2eb70>
```

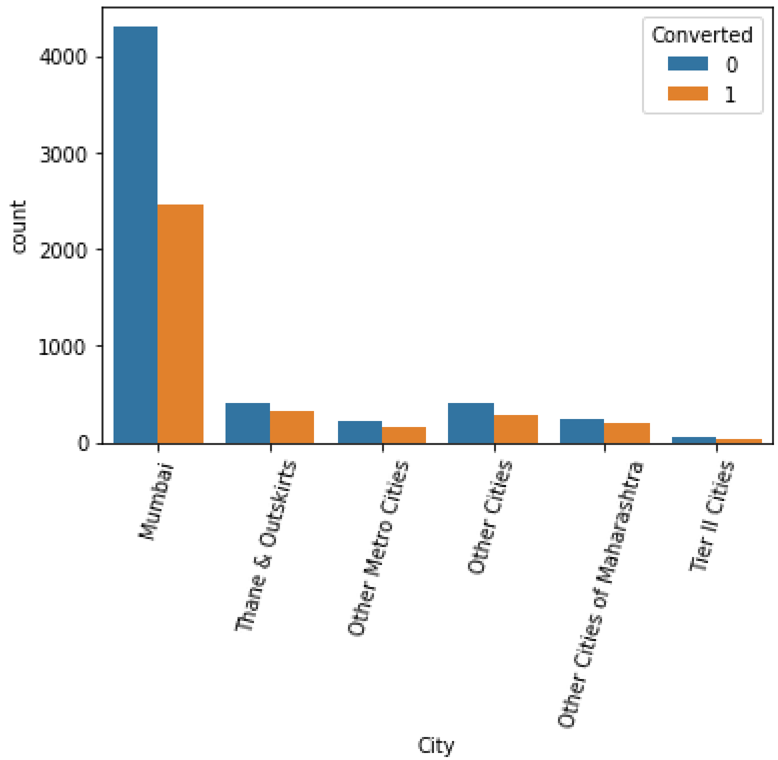


CONTINUED…

12. Univariate Analysis for 'City'

```
# Univariate Analysis for 'City'
sns.countplot(x = "City", hue = "Converted", data = xedu_data)
plt.xticks(rotation = 75)
```

```
(array([0, 1, 2, 3, 4, 5]), <a list of 6 Text xticklabel objects>)
```

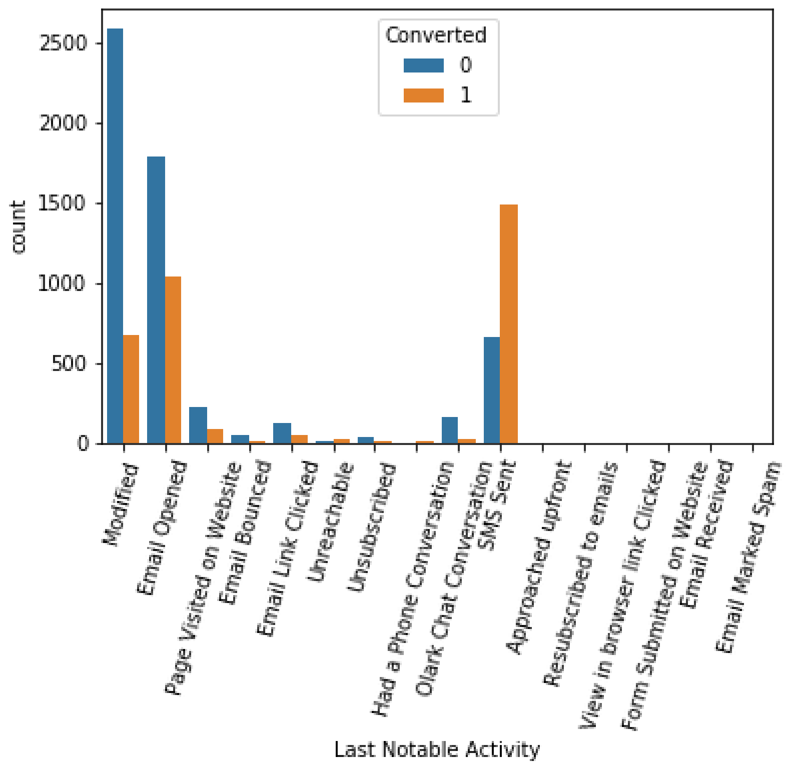


CONTINUED…

13. Univariate Analysis for ‘Last Notable Activity'

```
# Univariate Analysis for 'Last Notable Activity'
plt.xticks(rotation = 75)
sns.countplot(x = "Last Notable Activity", hue = "Converted", data = xedu_data)
```

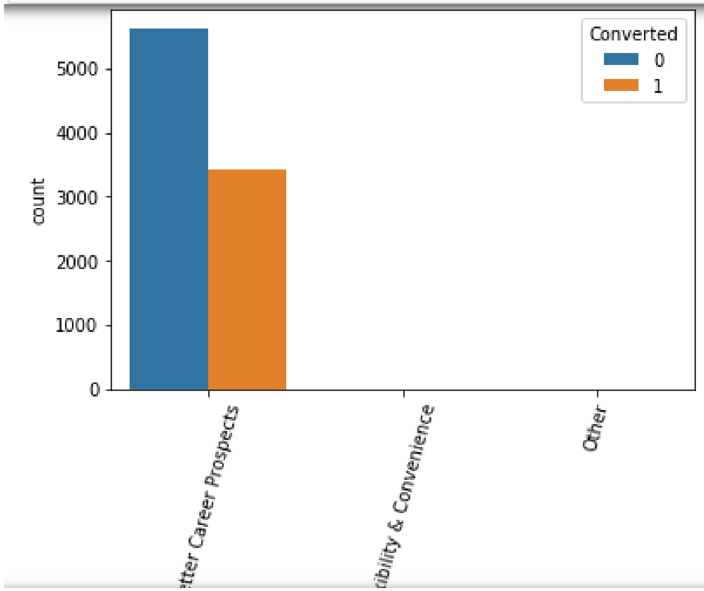
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40b19cc0>
```



CONTINUED…

14. Univariate Analysis for ‘What matters most to you in choosing a course'

```
# Univariate Analysis for 'What matters most to you in choosing a course'
plt.xticks(rotation = 75)
sns.countplot(x = "What matters most to you in choosing a course", hue = "Converted", data = xedu_data)
```



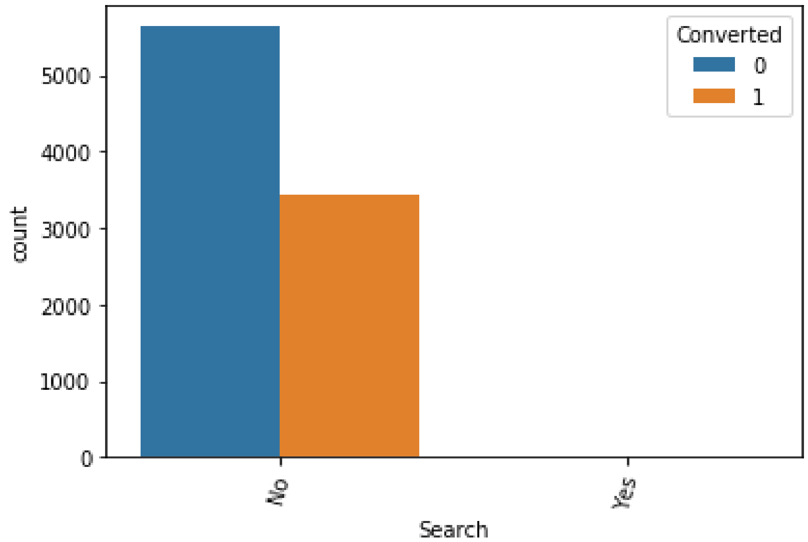
This will be no value addition to the model. Thus should be deleted.

CONTINUED…

15. Univariate Analysis for 'Search'

```
# Univariate Analysis for 'Search'
plt.xticks(rotation = 75)
sns.countplot(x = "Search", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40bb5438>
```



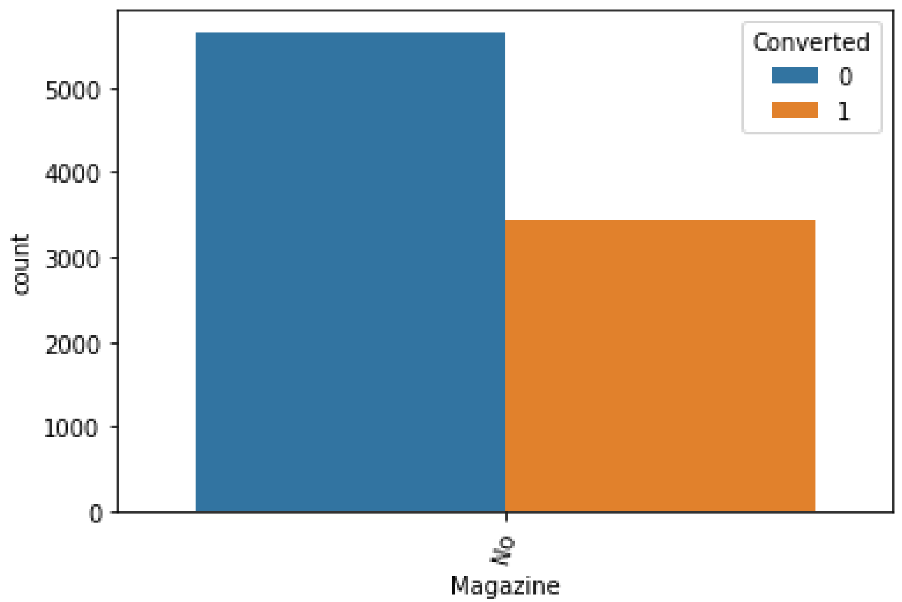
This will be no value addition to the model. Thus should be deleted.

CONTINUED…

16. Univariate Analysis for 'Magazine'

```
# Univariate Analysis for 'Magazine'
plt.xticks(rotation = 75)
sns.countplot(x = "Magazine", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be40eec860>
```



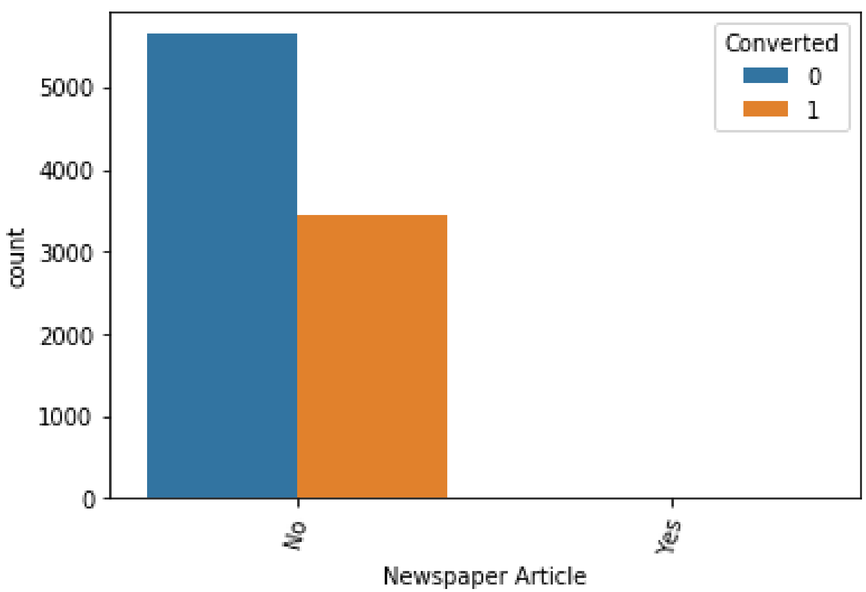
This will be no value addition to the model. Thus should be deleted

CONTINUED…

17. Univariate Analysis for ‘Newspaper Article'

```
# Univariate Analysis for 'Newspaper Article'
plt.xticks(rotation = 75)
sns.countplot(x = "Newspaper Article", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be41f506d8>
```

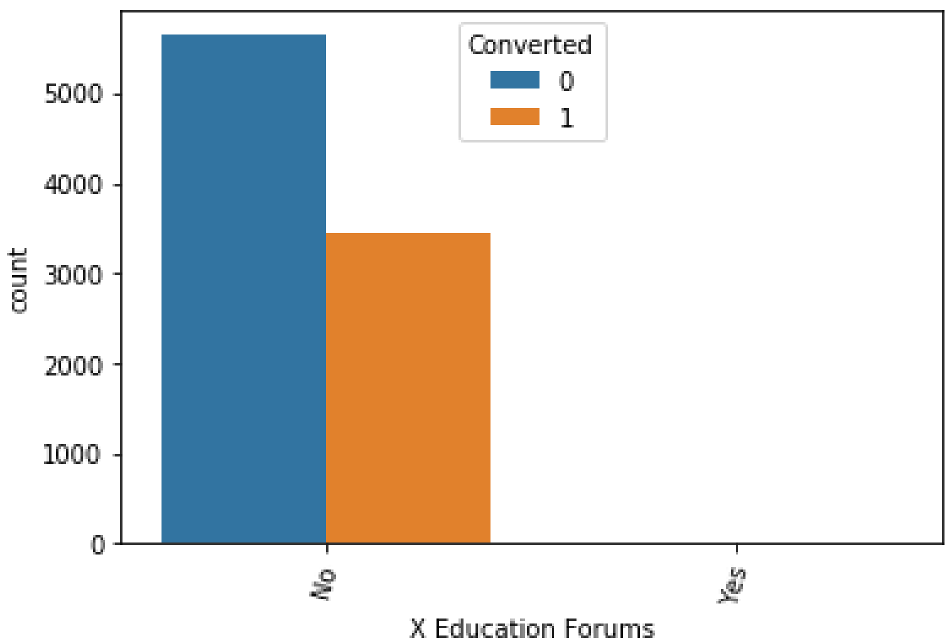


CONTINUED…

18. Univariate Analysis for ‘X Education Forums'

```
# Univariate Analysis for 'X Education Forums'
plt.xticks(rotation = 75)
sns.countplot(x = "X Education Forums", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be41fb3828>
```

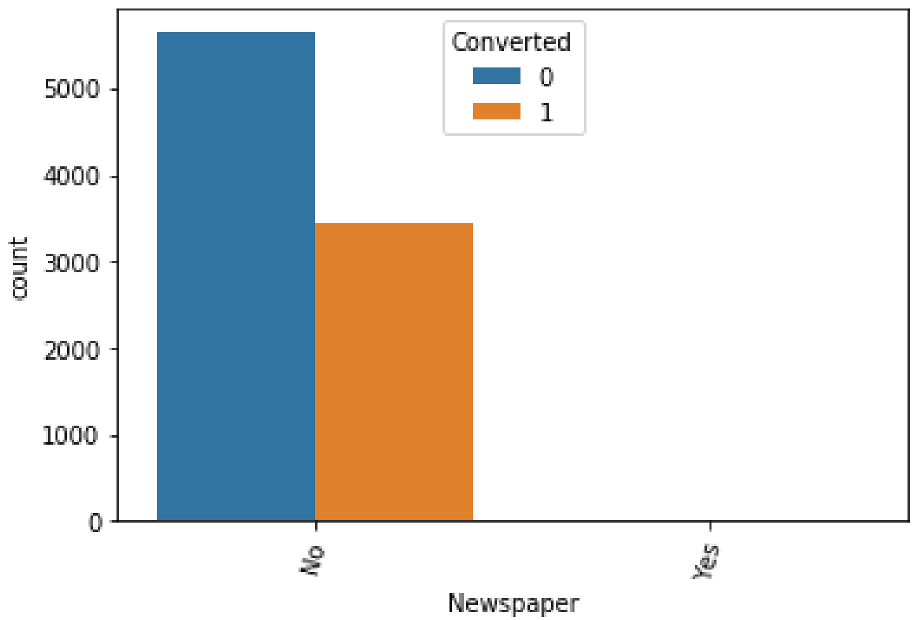


CONTINUED…

19. Univariate Analysis for 'Newspaper'

```
# Univariate Analysis for 'Newspaper'
plt.xticks(rotation = 75)
sns.countplot(x = "Newspaper", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be42002240>
```



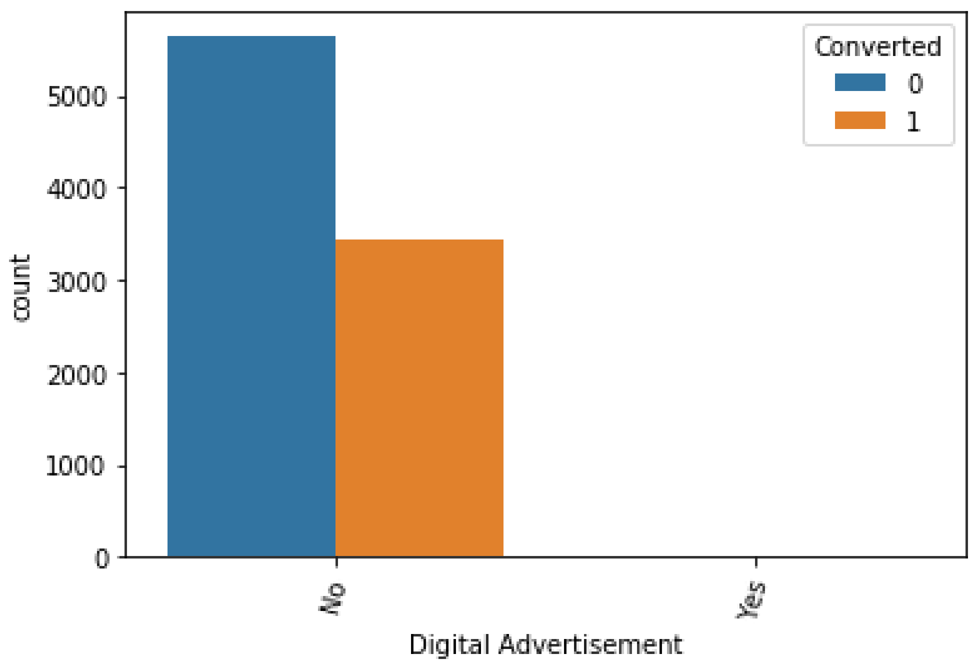
This will be no value addition to the model. Thus should be deleted

CONTINUED…

20. Univariate Analysis for ‘Digital Advertisement'

```
# Univariate Analysis for 'Digital Advertisement'
plt.xticks(rotation = 75)
sns.countplot(x = "Digital Advertisement", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be420537b8>
```

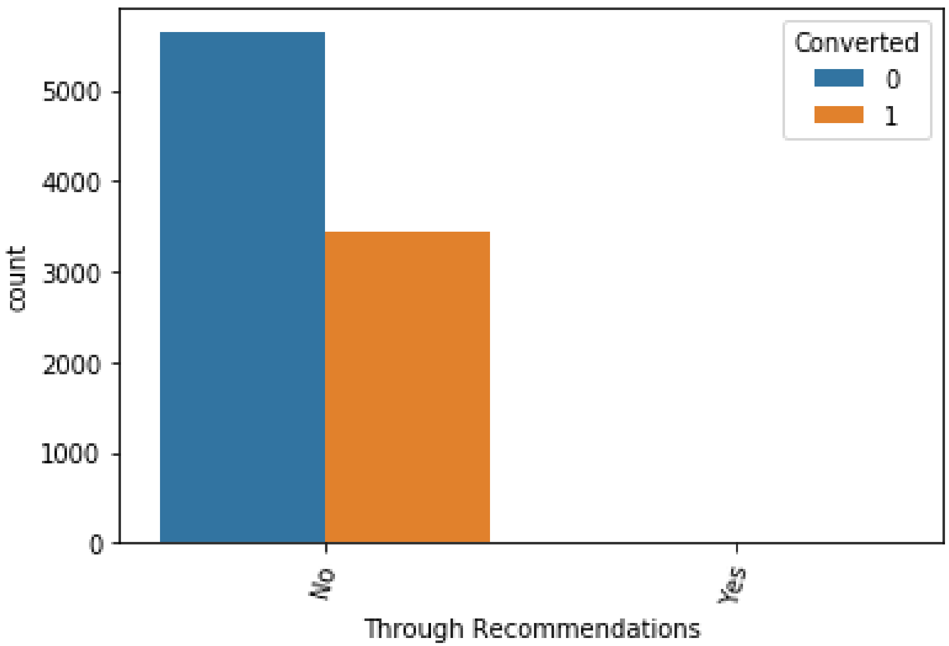


CONTINUED…

21. Univariate Analysis for ‘Through Recommendations'

```
# Univariate Analysis for 'Through Recommendations'
plt.xticks(rotation = 75)
sns.countplot(x = "Through Recommendations", hue = "Converted", data = xedu_data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1be420a5080>

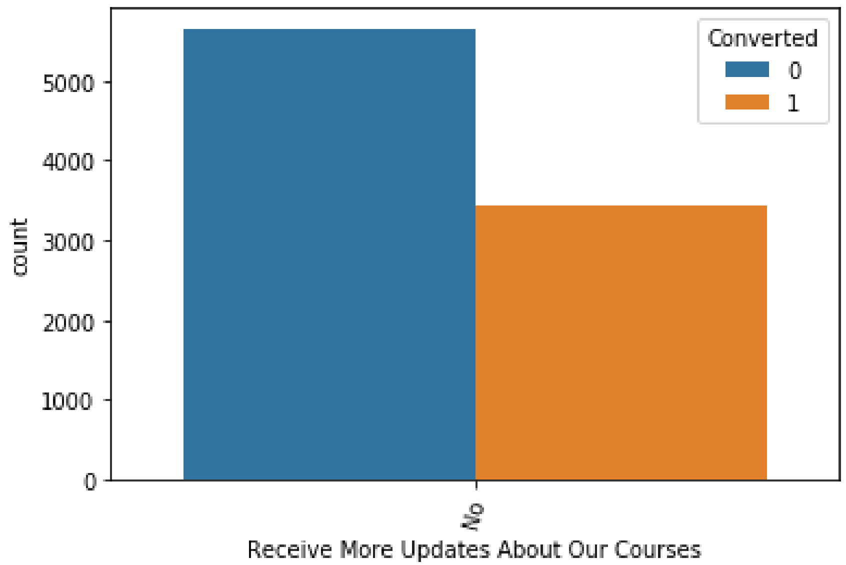


CONTINUED…

22. Univariate Analysis for ‘Receive More Updates About Our Courses'

```
# Univariate Analysis for 'Receive More Updates About Our Courses'
plt.xticks(rotation = 75)
sns.countplot(x = "Receive More Updates About Our Courses", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be4212b8d0>
```



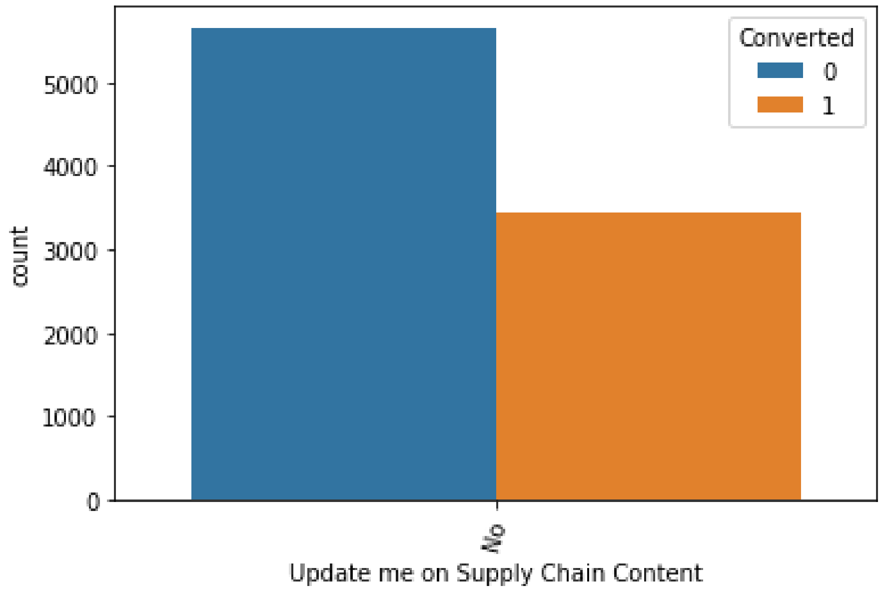
This will be no value addition to the model. Thus should be deleted

CONTINUED…

23. Univariate Analysis for 'Update me on Supply Chain Content'

```
# Univariate Analysis for 'Update me on Supply Chain Content'
plt.xticks(rotation = 75)
sns.countplot(x = "Update me on Supply Chain Content", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be4218aa58>
```



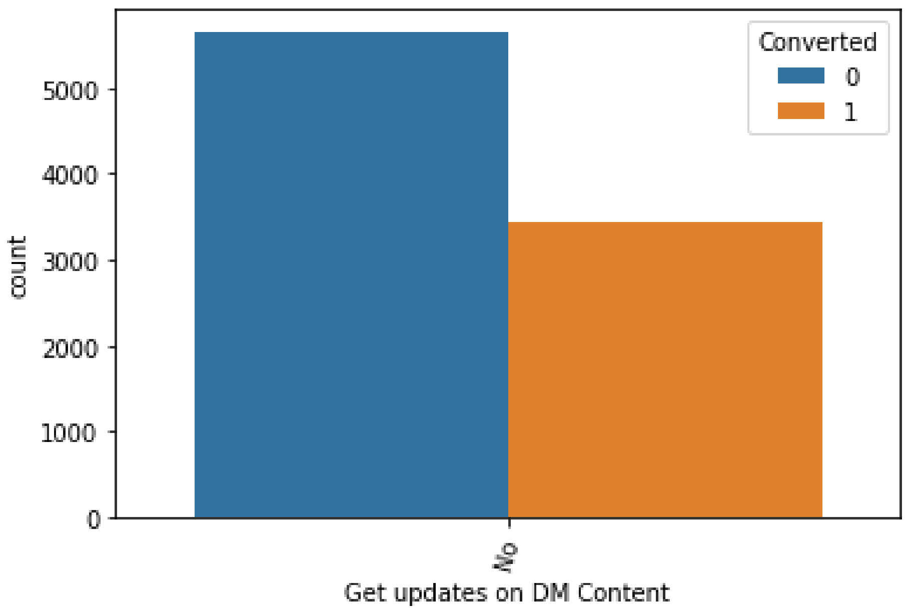
This will be no value addition to the model. Thus should be deleted

CONTINUED…

24. Univariate Analysis for ‘Get updates on DM Content'

```
# Univariate Analysis for 'Get updates on DM Content'
plt.xticks(rotation = 75)
sns.countplot(x = "Get updates on DM Content", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be421be6d8>
```



This will be no value addition to the model. Thus should be deleted

CONTINUED…

25. Univariate Analysis for ‘I agree to pay the amount through cheque'

```
# Univariate Analysis for 'I agree to pay the amount through cheque'
plt.xticks(rotation = 75)
sns.countplot(x = "I agree to pay the amount through cheque", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be42241a90>
```

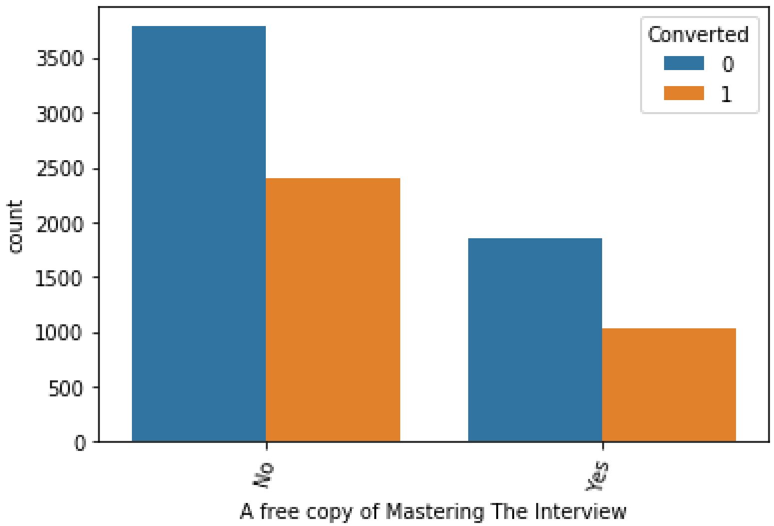
This will be no value addition to the model. Thus should be deleted

CONTINUED…

26. Univariate Analysis for ‘A free copy of Mastering The Interview'

```
# Univariate Analysis for 'A free copy of Mastering The Interview'
plt.xticks(rotation = 75)
sns.countplot(x = "A free copy of Mastering The Interview", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be4229dac8>
```



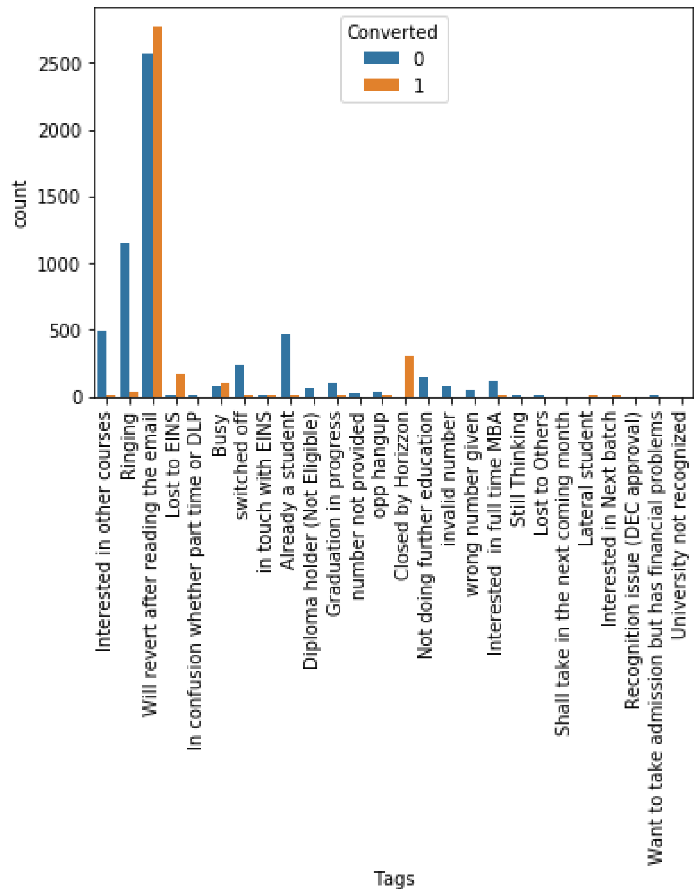
This will be no value addition to the model. Thus should be deleted

CONTINUED…

27. Univariate Analysis for ‘Tags'

```
# Univariate Analysis for 'Tags'
plt.xticks(rotation = 90)
sns.countplot(x = "Tags", hue = "Converted", data = xedu_data)
```

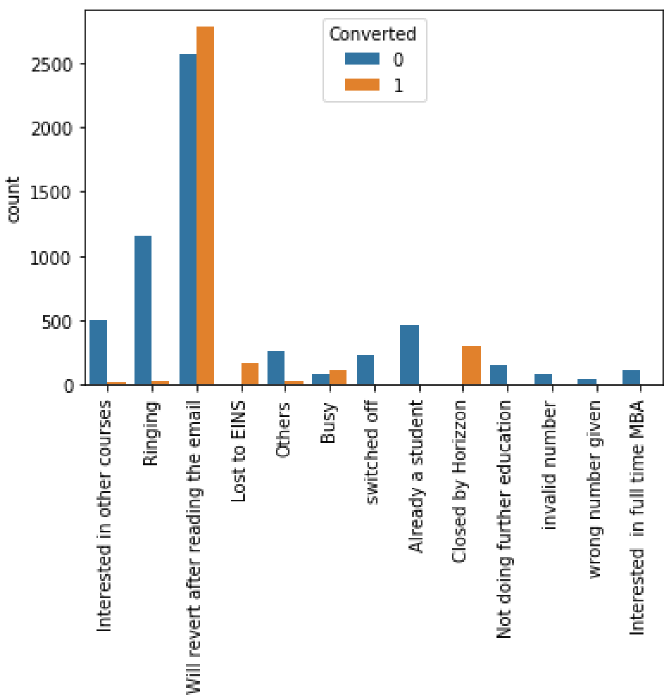
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be42241d30>
```



CONTINUED…

```
# Since most of the categories are not contributing much to the overall model.
# Will put all those tags of minimal contribution to the kitty of "Others"
xedu_data['Tags'] = xedu_data['Tags'].replace(['In confusion whether part time or DLP', 'in touch with EINS','Diploma holder (Not
'Approached upfront','Graduation in progress','number not provided', 'opp hangup','Still Thinking',
'Lost to Others','Shall take in the next coming month','Lateral student','Interested in Next batch',
'Recognition issue (DEC approval)','Want to take admission but has financial problems','University not recognized'], 'Others')
sns.countplot(x = "Tags", hue = "Converted", data = xedu_data)
plt.xticks(rotation = 90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 <a list of 13 Text xticklabel objects>)
```

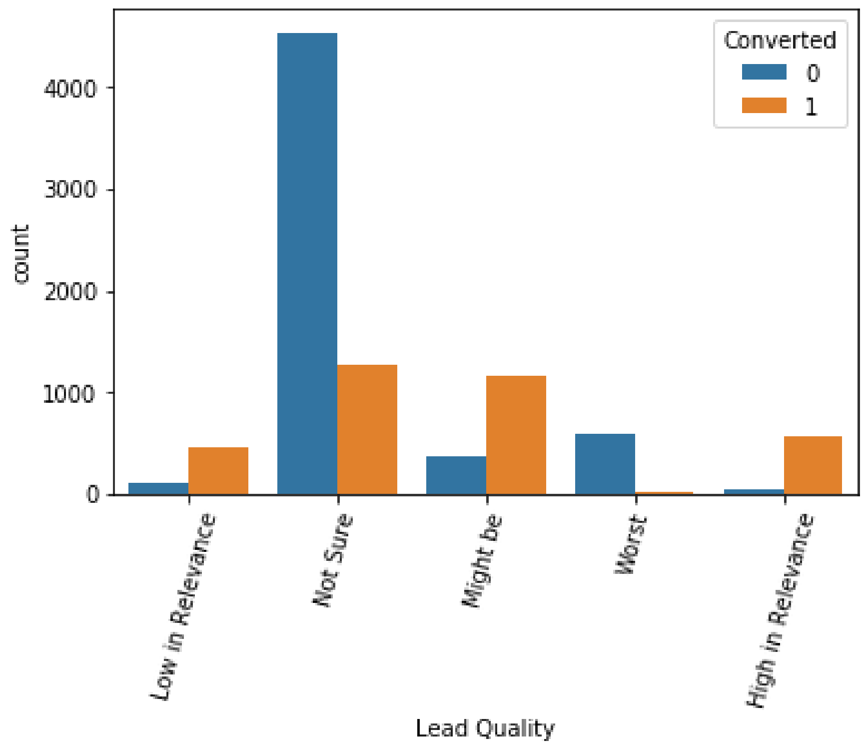


CONTINUED…

28. Univariate Analysis for ‘Lead Quality'

```
# Univariate Analysis for 'Lead Quality'
plt.xticks(rotation = 75)
sns.countplot(x = "Lead Quality", hue = "Converted", data = xedu_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1be424a8f60>
```



END OF EXPLORATORY DATA ANALYSIS

```
## Deleting all the columns that do not contribute much to the over all model building
xedu_data = xedu_data.drop(['Lead Number','What matters most to you in choosing a course','Search','Magazine','Newspaper Article
'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','A free copy of Mastering The Interview','Country'],1)
xedu_data.head()
```

	Prospect ID	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Specialization	What is your current occupation	Tags	Lead Quality	C
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	API	Olark Chat	No	No	0	0.0	0	0.0	Page Visited on Website	General	Unemployed	Interested in other courses	Low in Relevance	Mum
1	2a272436-5132-4136-86fa-dcc88c88f482	API	Organic Search	No	No	0	5.0	674	2.5	Email Opened	General	Unemployed	Ringing	Not Sure	Mum
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	Email Opened	Business Administration	Student	Will revert after reading the email	Might be	Mum
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	Unreachable	Media and Advertising	Unemployed	Ringing	Not Sure	Mum
4	3256f628-e534-4826-9d63-4a8b88782852	Landing Page Submission	Google	No	No	1	2.0	1428	1.0	Converted to Lead	General	Unemployed	Will revert after reading the email	Might be	Mum

CONVERTING BINARY VARIABLES

Let's map 'Do Not Email', 'Do Not Call' features for binary values.

```
################ CONVERTING BINARY VARIABLES ##############3

# List of variables to map

varlist =  ['Do Not Email', 'Do Not Call']

# Defining the map function
def binary_map(x):
    return x.map({'Yes': 1, "No": 0})

# Applying the function to the housing list
xedu_data[varlist] = xedu_data[varlist].apply(binary_map)
```

```
# Creating a dummy variable for some of the categorical variables and dropping the first one.
dum1 = pd.get_dummies(xedu_data[['Tags','Lead Quality','City','Last Notable Activity','Lead Origin',
'Lead Source', 'Last Activity', 'Specialization','What is your current occupation',]], drop_first=True)
dum1.head()
```

	Tags_Busy	Tags_Closed by Horizzon	Tags_Interested in full time MBA	Tags_Interested in other courses	Tags_Lost to EINS	Tags_Not doing further education	Tags_Others	Tags_Ringing	Tags_Will revert after reading the email	Tags_invalid number	...	Specialization_Re Managem
0	0	0	0	1	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	1	0	0	...	
2	0	0	0	0	0	0	0	0	1	0	...	
3	0	0	0	0	0	0	0	1	0	0	...	
4	0	0	0	0	0	0	0	0	1	0	...	

5 rows × 81 columns

CONTINUED…

Concat the new dummies values data set with the ‘xedu_data’ dataset.

```
xedu_data = pd.concat([xedu_data, dum1], axis=1)
xedu_data.head()
```

	Prospect ID	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	...	Specialization_Retail Management	Specialization_Rural and Agribusiness	Speciali:
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	API	Olark Chat	0	0	0	0.0	0	0.0	Page Visited on Website	...	0	0	
1	2a272436-5132-4136-86fa-dcc88c88f482	API	Organic Search	0	0	0	5.0	674	2.5	Email Opened	...	0	0	

Drop duplicate values

```
#Dropping duplicate values
xedu_data = xedu_data.drop(['Tags','Lead Quality','City','Last Notable Activity','Lead Origin', 'Lead Source', 'Last Activity',
```

CONTINUED…

Let's now define the X and y variables for model selection.

```
#Defining X Variable
X = xedu_data.drop(['Prospect ID','Converted'], axis=1)
X.head()
```

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Tags_Busy	Tags_Closed by Horizzon	Tags_Interested in full time MBA	Tags_Interested in other courses	Tags_Lost to EINS	...	Specialization_Retail Management	Specialization_Rura and Agribusines
0	0	0	0.0	0	0.0	0	0	0	1	0	...	0	
1	0	0	5.0	674	2.5	0	0	0	0	0	...	0	
2	0	0	2.0	1532	2.0	0	0	0	0	0	...	0	
3	0	0	1.0	305	1.0	0	0	0	0	0	...	0	
4	0	0	2.0	1428	1.0	0	0	0	0	0	...	0	

5 rows × 86 columns

```
# Defining Y variable
y=xedu_data['Converted']
y.head()
```

```
0    0
1    0
2    1
3    0
4    1
Name: Converted, dtype: int64
```

CONTINUED…

Perform test-train split using sklearn.model_selection

```
from sklearn.model_selection import train_test_split
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

```
X_train.shape
```

```
(6351, 86)
```

```
X_test.shape
```

```
(2723, 86)
```

```
y_train.shape
```

```
(6351,)
```

```
y_test.shape
```

```
(2723,)
```

PERFORMING FEATURE SCALING

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train[['TotalVisits','Total Time Spent on Website','Page Views Per Visit']] = scaler.fit_transform(X_train[['TotalVisits','Tota

X_train.head()
```

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Tags_Busy	Tags_Closed by Horizzon	Tags_Interested in full time MBA	Tags_Interested in other courses	Tags_Lost to EINS	...	Specialization_Retail Management	Specializat and Agr
3009	0	0	-0.431325	-0.160255	-0.155018	0	0	0	0	0	...	0	
1012	1	0	-0.431325	-0.540048	-0.155018	0	0	0	0	0	...	0	
9226	0	0	-1.124566	-0.888650	-1.265540	0	0	0	0	0	...	0	
4750	0	0	-0.431325	1.643304	-0.155018	0	0	0	0	0	...	0	
7987	0	0	0.608537	2.017593	0.122613	0	0	0	0	1	...	0	

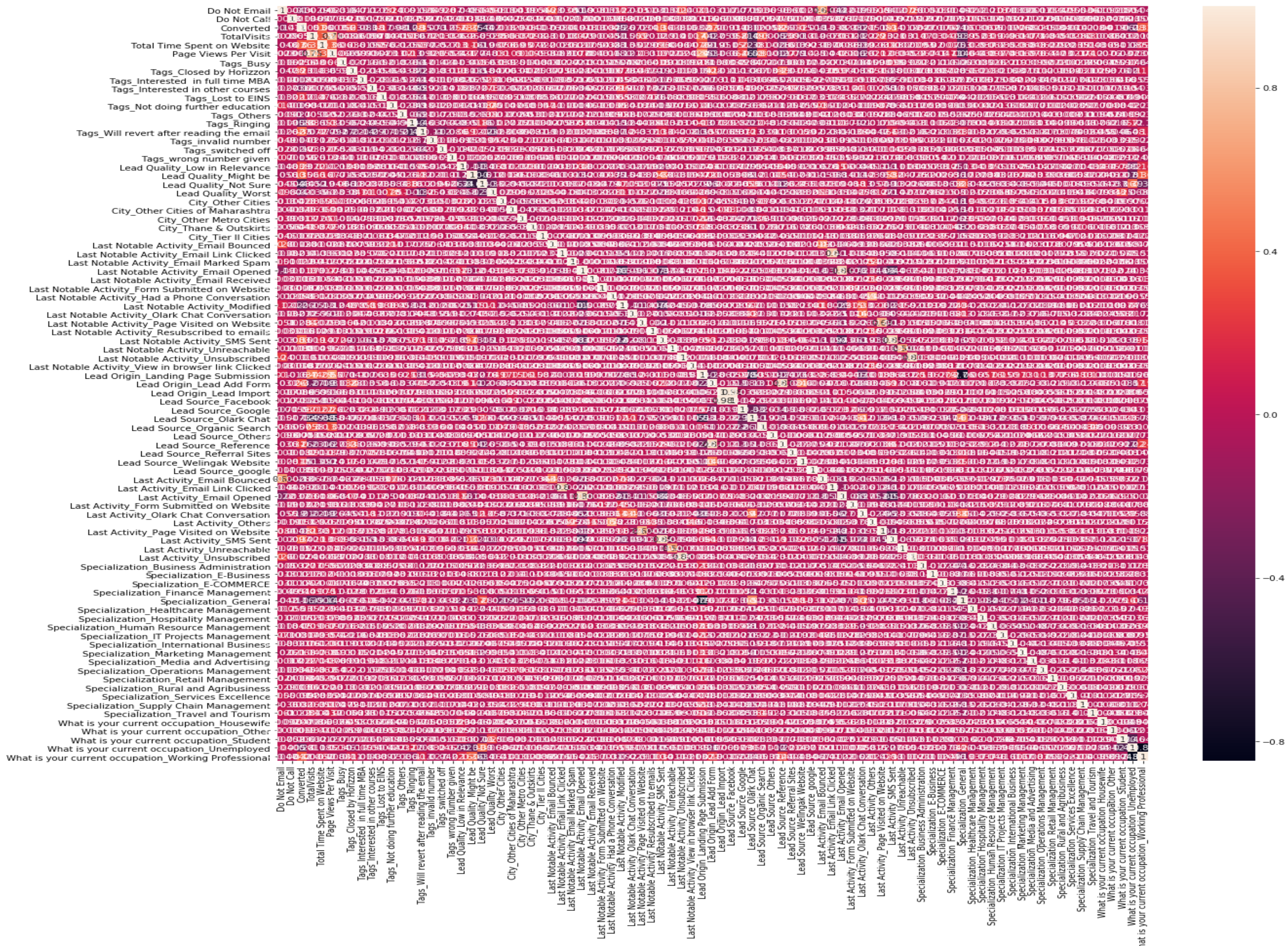
5 rows × 86 columns

Checking the Conversion Rate of the Leads

```
# Checking the Conversion Rate of the Leads
Conv = (sum(xedu_data['Converted'])/len(xedu_data['Converted'].index))*100
Conv
```

```
37.85541106458012
```

CORRELATIONS MATRIX



REGRESSION LOGISTIC MODEL BUILDING

Use the statsmodels.api and fit the model.

```
import statsmodels.api as sm
```

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

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6351
Model:	GLM	**Df Residuals:**	6264
Model Family:	Binomial	**Df Model:**	86
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1249.5
Date:	Sun, 17 Nov 2019	**Deviance:**	2499.0
Time:	10:01:45	**Pearson chi2:**	3.87e+04
No. Iterations:	24	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	23.1365	2.16e+05	0.000	1.000	-4.23e+05	4.23e+05
Do Not Email	-1.3892	0.327	-4.246	0.000	-2.030	-0.748

CONTINUED…

USE RFE Selection

```
    ## USE RFE Selection
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 13 variables as output
rfe = rfe.fit(X_train, y_train)
```

Extract the columns, ranking and selection status from RFE

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

```
[('Do Not Email', True, 1),
 ('Do Not Call', False, 34),
 ('TotalVisits', False, 44),
 ('Total Time Spent on Website', False, 3),
 ('Page Views Per Visit', False, 41),
 ('Tags_Busy', True, 1),
 ('Tags_Closed by Horizzon', True, 1),
 ('Tags_Interested  in full time MBA', False, 18),
 ('Tags_Interested in other courses', False, 10),
 ('Tags_Lost to EINS', True, 1),
 ('Tags_Not doing further education', False, 11),
 ('Tags_Others', False, 30),
 ('Tags_Ringing', True, 1),
 ('Tags_Will revert after reading the email', True, 1),
 ('Tags_invalid number', True, 1),
 ('Tags_switched off', True, 1),
```

CONTINUED…

Model assessment using Statsmodel

In [98]:

```
## Model assessment using Statsmodel
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[98]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6351
Model:	GLM	**Df Residuals:**	6335
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1580.6
Date:	Sun, 17 Nov 2019	**Deviance:**	3161.3
Time:	10:01:56	**Pearson chi2:**	3.11e+04
No. Iterations:	24	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.8547	0.215	-8.636	0.000	-2.276	-1.434
Do Not Email	-1.3106	0.213	-6.154	0.000	-1.728	-0.893
Tags_Busy	3.5477	0.332	10.680	0.000	2.897	4.199
Tags_Closed by Horizzon	7.7377	0.762	10.152	0.000	6.244	9.231
Tags_Lost to EINS	8.9540	0.753	11.887	0.000	7.478	10.430
Tags_Ringing	-1.9696	0.340	-5.800	0.000	-2.635	-1.304
Tags_Will revert after reading the email	3.7332	0.228	16.340	0.000	3.285	4.181
Tags_invalid number	-23.4649	2.21e+04	-0.001	0.999	-4.34e+04	4.33e+04
Tags_switched off	-2.5711	0.589	-4.367	0.000	-3.725	-1.417
Tags_wrong number given	-23.0779	3.17e+04	-0.001	0.999	-6.21e+04	6.2e+04
Lead Quality_Not Sure	-3.3496	0.129	-26.033	0.000	-3.602	-3.097

CONTINUED…

Drop 'Tags_invalid number' because of its high P-Value

```
col1 = col.drop('Tags_invalid number',1)
```

```
col1
```

```
Index(['Do Not Email', 'Tags_Busy', 'Tags_Closed by Horizzon',
       'Tags_Lost to EINS', 'Tags_Ringing',
       'Tags_Will revert after reading the email', 'Tags_switched off',
       'Tags_wrong number given', 'Lead Quality_Not Sure',
       'Lead Quality_Worst', 'Last Notable Activity_SMS Sent',
       'Lead Origin_Lead Add Form', 'Lead Source_Welingak Website',
       'What is your current occupation_Working Professional'],
      dtype='object')
```

CONTINUED…

Re-Run the model using statsapi

```
X_train_sm = sm.add_constant(X_train[col1])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6351
Model:	GLM	**Df Residuals:**	6336
Model Family:	Binomial	**Df Model:**	14
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1586.7
Date:	Sun, 17 Nov 2019	**Deviance:**	3173.3
Time:	10:01:57	**Pearson chi2:**	3.07e+04
No. Iterations:	22	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.0195	0.217	-9.308	0.000	-2.445	-1.594
Do Not Email	-1.3018	0.212	-6.130	0.000	-1.718	-0.886
Tags_Busy	3.7300	0.331	11.270	0.000	3.081	4.379
Tags_Closed by Horizzon	7.8904	0.763	10.345	0.000	6.396	9.385
Tags_Lost to EINS	9.1124	0.754	12.086	0.000	7.635	10.590
Tags_Ringing	-1.7713	0.338	-5.244	0.000	-2.433	-1.109
Tags_Will revert after reading the email	3.8970	0.230	16.954	0.000	3.446	4.348
Tags_switched off	-2.3666	0.588	-4.028	0.000	-3.518	-1.215
Tags_wrong number given	-20.8825	1.17e+04	-0.002	0.999	-2.29e+04	2.28e+04
Lead Quality_Not Sure	-3.3417	0.128	-26.020	0.000	-3.593	-3.090
Lead Quality_Worst	-3.7822	0.848	-4.462	0.000	-5.444	-2.121
Last Notable Activity_SMS Sent	2.7503	0.120	22.841	0.000	2.514	2.986
Lead Origin_Lead Add Form	1.0769	0.362	2.974	0.003	0.367	1.787
Lead Source_Welingak Website	3.4268	0.818	4.190	0.000	1.824	5.030
What is your current occupation_Working Professional	1.3240	0.290	4.567	0.000	0.756	1.892

CONTINUED…

Getting the predicted values on the train set

```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

```
3009    0.187836
1012    0.191249
9226    0.000798
4750    0.783501
7987    0.977050
1281    0.990319
2880    0.187836
4971    0.753835
7536    0.867329
1248    0.000798
dtype: float64
```

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

```
array([1.87835911e-01, 1.91249095e-01, 7.98103307e-04, 7.83501245e-01,
       9.77049675e-01, 9.90319037e-01, 1.87835911e-01, 7.53834532e-01,
       8.67328801e-01, 7.98103307e-04])
```

Drop the feature 'Tags wrong number given' due to high P-Value

```
col2 = col1.drop('Tags_wrong number given',1)
```

CONTINUED…

Re-Run the model

```
X_train_sm = sm.add_constant(X_train[col2])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	No. Observations:	6351
Model:	GLM	Df Residuals:	6337
Model Family:	Binomial	Df Model:	13
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-1588.8
Date:	Sun, 17 Nov 2019	Deviance:	3177.6
Time:	10:01:57	Pearson chi2:	3.08e+04
No. Iterations:	8	Covariance Type:	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.0888	0.216	-9.654	0.000	-2.513	-1.665
Do Not Email	-1.3012	0.212	-6.134	0.000	-1.717	-0.885
Tags_Busy	3.8040	0.330	11.532	0.000	3.157	4.450
Tags_Closed by Horizzon	7.9562	0.763	10.433	0.000	6.461	9.451
Tags_Lost to EINS	9.1785	0.754	12.177	0.000	7.701	10.656
Tags_Ringing	-1.6947	0.337	-5.036	0.000	-2.354	-1.035
Tags_Will revert after reading the email	3.9665	0.229	17.311	0.000	3.517	4.416
Tags_switched off	-2.2882	0.587	-3.900	0.000	-3.438	-1.138
Lead Quality_Not Sure	-3.3406	0.128	-26.026	0.000	-3.592	-3.089
Lead Quality_Worst	-3.7624	0.850	-4.426	0.000	-5.428	-2.096
Last Notable Activity_SMS Sent	2.7406	0.120	22.847	0.000	2.506	2.976
Lead Origin_Lead Add Form	1.0894	0.363	3.001	0.003	0.378	1.801
Lead Source_Welingak Website	3.4138	0.818	4.173	0.000	1.810	5.017

CONTINUED…

```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

```
3009    0.188037
1012    0.194070
9226    0.000805
4750    0.782077
7987    0.977003
1281    0.990228
2880    0.188037
4971    0.753104
7536    0.867357
1248    0.000805
dtype: float64
```

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

```
array([1.88037158e-01, 1.94070077e-01, 8.04879357e-04, 7.82076694e-01,
       9.77003470e-01, 9.90227993e-01, 1.88037158e-01, 7.53103755e-01,
       8.67356930e-01, 8.04879357e-04])
```

```
###  Data set with converted flag and probabilities
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(10)
```

	Converted	Converted_prob	Prospect ID
0	0	0.188037	3009
1	0	0.194070	1012
2	0	0.000805	9226
3	1	0.782077	4750
4	1	0.977003	7987
5	1	0.990228	1281
6	0	0.188037	2880
7	1	0.753104	4971
8	1	0.867357	7536
9	0	0.000805	1248

Calculate the Converted probability.

CONTINUED…

Let’s replace 1 if Churn_Prob > 0.5 else 0 in the new column pred

```
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
0	0	0.188037	3009	0
1	0	0.194070	1012	0
2	0	0.000805	9226	0
3	1	0.782077	4750	1
4	1	0.977003	7987	1

CONTINUED…

Let's now check the confusion matrix , accuracy and VIF

```
from sklearn import metrics
```

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

```
[[3756  149]
 [ 363 2083]]
```

```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.predicted))
```

```
0.9193827743662415
```

```
### Analysing Variance Inflation Factor (VIF)
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train[col2].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_train[col2].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

	Features	VIF
5	Tags_Will revert after reading the email	2.89
9	Last Notable Activity_SMS Sent	2.85
12	What is your current occupation_Working Profes...	1.62
4	Tags_Ringing	1.56
11	Lead Source_Welingak Website	1.54
2	Tags_Closed by Horizzon	1.15
0	Do Not Email	1.11
1	Tags_Busy	1.11
7	Lead Quality_Not Sure	1.11
3	Tags_Lost to EINS	1.05
6	Tags_switched off	1.04

As all the VIFs are less than 5, so no need to drop any features further

CONTINUED…

Let's now check the Metrics Beyond Accuracy

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

```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

```
0.8515944399018807
```

```
# Let us calculate specificity
TN / float(TN+FP)
```

```
0.9618437900128041
```

```
# Calculate false postive rate - predicting churn when customer does not have churned
FP/ float(TN+FP)
```

```
0.038156209987195905
```

```
# positive predictive value
print (TP / float(TP+FP))
```

```
0.9332437275985663
```

```
# Negative predictive value
print (TN / float(TN+ FN))
```

```
0.9118718135469774
```

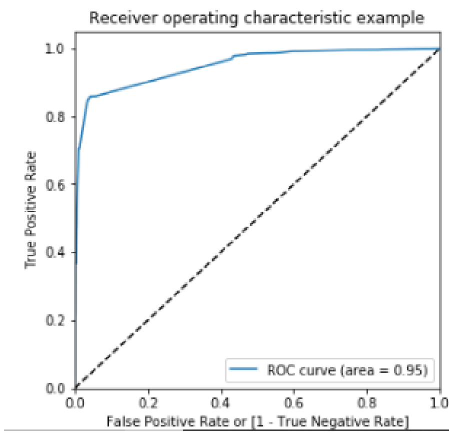
CONTINUED…

Plotting the ROC Curve

```python
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
```

```python
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Converted, y_train_pred_final.Converted_prob, drop_intermediate
```

```python
draw_roc(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
```



CONTINUED…

Optimal Cutoff Point Determination

```python
# 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.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
0	0	0.188037	3009	0	1	1	0	0	0	0	0	0	0	0
1	0	0.194070	1012	0	1	1	0	0	0	0	0	0	0	0
2	0	0.000805	9226	0	1	0	0	0	0	0	0	0	0	0
3	1	0.782077	4750	1	1	1	1	1	1	1	1	1	0	0
4	1	0.977003	7987	1	1	1	1	1	1	1	1	1	1	1

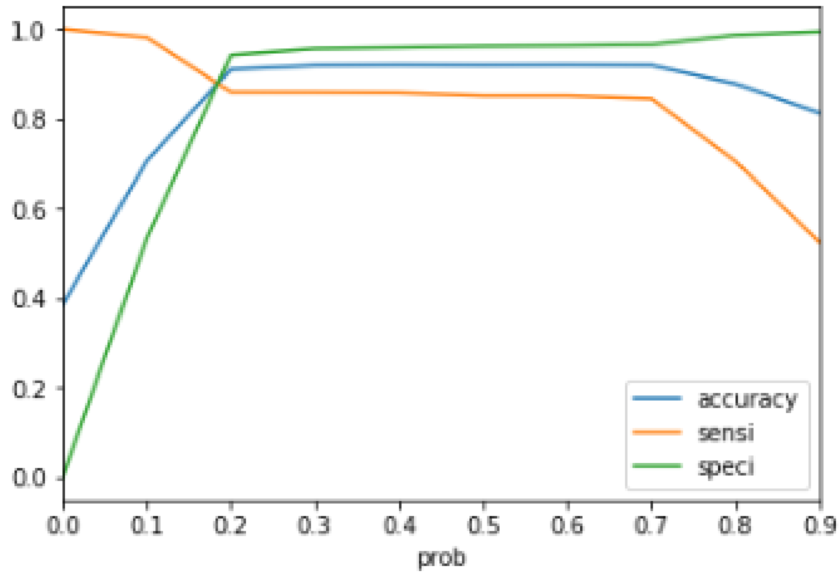
```python
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix

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_pred_final[i] )
    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  0.385136  1.000000  0.000000
0.1   0.1  0.705873  0.981603  0.533163
0.2   0.2  0.910408  0.859771  0.942125
0.3   0.3  0.918910  0.859362  0.956210
0.4   0.4  0.920013  0.858136  0.958771
0.5   0.5  0.919383  0.851594  0.961844
0.6   0.6  0.920170  0.851594  0.963124
0.7   0.7  0.919225  0.845053  0.965685
0.8   0.8  0.878287  0.705233  0.986684
0.9   0.9  0.813258  0.524530  0.994110
```

CONTINUED…

Plotting accuracy,sensitivity and specificity for various probabilities.

```
#plotting accuracy sensitivity and specificity for various probabilities.
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



0.2 is the optimum point to take it as a cutoff probability

CONTINUED…

```
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()
```

	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
0	0	0.188037	3009	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0.194070	1012	0	1	1	0	0	0	0	0	0	0	0	0
2	0	0.000805	9226	0	1	0	0	0	0	0	0	0	0	0	0
3	1	0.782077	4750	1	1	1	1	1	1	1	1	1	0	0	1
4	1	0.977003	7987	1	1	1	1	1	1	1	1	1	1	1	1

```
#### Assignment of Score
y_train_pred_final['Score'] = y_train_pred_final.Converted_prob.map( lambda x: round(x*100))

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	final_predicted	Score
0	0	0.188037	3009	0	1	1	0	0	0	0	0	0	0	0	0	19
1	0	0.194070	1012	0	1	1	0	0	0	0	0	0	0	0	0	19
2	0	0.000805	9226	0	1	0	0	0	0	0	0	0	0	0	0	0
3	1	0.782077	4750	1	1	1	1	1	1	1	1	1	0	0	1	78
4	1	0.977003	7987	1	1	1	1	1	1	1	1	1	1	1	1	98

CONTINUED…

Assignment of Score

```python
#### Assignment of Score
y_train_pred_final['Score'] = y_train_pred_final.Converted_prob.map( lambda x: round(x*100))

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	final_predicted	Score
0	0	0.188037	3009	0	1	1	0	0	0	0	0	0	0	0	0	19
1	0	0.194070	1012	0	1	1	0	0	0	0	0	0	0	0	0	19
2	0	0.000805	9226	0	1	0	0	0	0	0	0	0	0	0	0	0
3	1	0.782077	4750	1	1	1	1	1	1	1	1	1	0	0	1	78
4	1	0.977003	7987	1	1	1	1	1	1	1	1	1	1	1	1	98

```python
metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.final_predicted)

confusion2 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.final_predicted )
confusion2
```

```
array([[3679,  226],
       [ 343, 2103]], dtype=int64)
```

CONTINUED…

Check the Sensitivity and Specificity.

```
TP = confusion2[1,1]
TN = confusion2[0,0]
FP = confusion2[0,1]
FN = confusion2[1,0]
```

```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

```
0.8597710547833197
```

```
# Let us calculate specificity
TN / float(TN+FP)
```

```
0.9421254801536492
```

```
print(FP/ float(TN+FP))
```

```
0.05787451984635083
```

```
print (TP / float(TP+FP))
```

```
0.9029626449119794
```

```
print (TN / float(TN+ FN))
```

```
0.9147190452511188
```

CONTINUED…

Calculate and see the PRECISION AND RECALL values

```
############# PRECISION AND RECALL  #############

confusion3 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.predicted )
confusion3
```

```
array([[3756,  149],
       [ 363, 2083]], dtype=int64)
```

```
# Precision
confusion[1,1]/(confusion[0,1]+confusion[1,1])
```

```
0.9332437275985663
```

```
# Recall
confusion[1,1]/(confusion[1,0]+confusion[1,1])
```

```
0.8515944399018807
```

```
# Using sklearn utilities for the same
from sklearn.metrics import precision_score, recall_score
precision_score(y_train_pred_final.Converted, y_train_pred_final.predicted)
```

```
0.9332437275985663
```

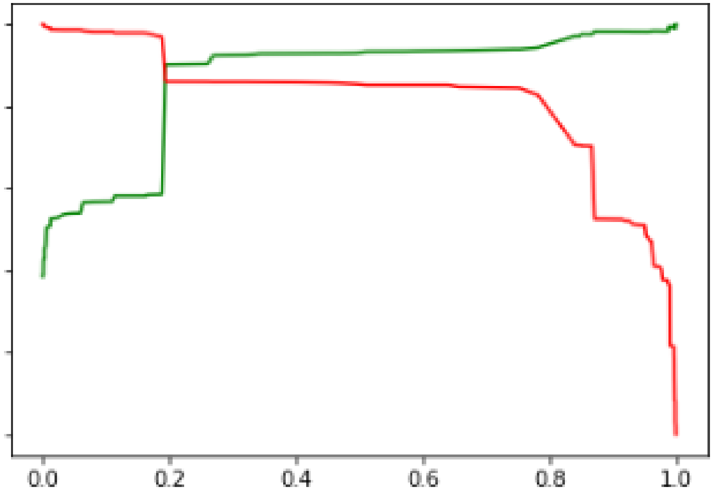
```
recall_score(y_train_pred_final.Converted, y_train_pred_final.predicted)
```

```
0.8515944399018807
```

CONTINUED…

Precision and Recall Tradeoff

```
## Precision and Recall Tradeoff

from sklearn.metrics import precision_recall_curve
p, r, thresholds = precision_recall_curve(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
plt.show()
```



DOING PREDICTION ON THE TEST SET

```
X_test[['TotalVisits','Total Time Spent on Website','Page Views Per Visit']] = scaler.fit_transform(X_test[['TotalVisits','T
X_train.head()
```

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Tags_Busy	Tags_Closed by Horizzon	Tags_Interested in full time MBA	Tags_Interested in other courses	Tags_Lost to EINS	...	Specialization_Retail Management	Specializati and Agri
3009	0	0	-0.431325	-0.160255	-0.155018	0	0	0	0	0	...	0	
1012	1	0	-0.431325	-0.540048	-0.155018	0	0	0	0	0	...	0	
9226	0	0	-1.124566	-0.888650	-1.265540	0	0	0	0	0	...	0	
4750	0	0	-0.431325	1.643304	-0.155018	0	0	0	0	0	...	0	
7987	0	0	0.608537	2.017593	0.122613	0	0	0	0	1	...	0	

5 rows × 86 columns

```
X_test = X_test[col2]
X_test.head()
```

	Do Not Email	Tags_Busy	Tags_Closed by Horizzon	Tags_Lost to EINS	Tags_Ringing	Tags_Will revert after reading the email	Tags_switched off	Lead Quality_Not Sure	Lead Quality_Worst	Last Notable Activity_SMS Sent	Lead Origin_Lead Add Form	Source_We W
3271	0	0	0	0	0	1	0	1	0	0	0	
1490	0	0	0	0	0	1	0	0	0	0	0	
7936	0	0	0	0	0	1	0	1	0	0	0	
4216	0	0	1	0	0	0	0	0	0	0	1	
3830	0	0	0	0	0	1	0	1	0	0	0	

CONTINUED…

Add constant and predict

```
X_test_sm = sm.add_constant(X_test)
```

```
#Predicting on the test set
y_test_pred = res.predict(X_test_sm)
```

```
y_test_pred[:10]
```

```
3271    0.188037
1490    0.961508
7936    0.188037
4216    0.999049
3830    0.188037
1800    0.961508
6507    0.012329
4821    0.000445
4223    0.996691
4714    0.188037
dtype: float64
```

```
# Converting y_pred to a dataframe which is an array
y_pred_1 = pd.DataFrame(y_test_pred)
```

```
y_pred_1.head(10)
```

	0
3271	0.188037
1490	0.961508
7936	0.188037
4216	0.999049
3830	0.188037
1800	0.961508
6507	0.012329
4821	0.000445
4223	0.996691
4714	0.188037

CONTINUED…

```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
y_test_df['Prospect ID'] = y_test_df.index
```

```
# Removing 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)
```

```
# Appending y_test_df and y_pred_1
y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
y_pred_final.head(10)
```

	Converted	Prospect ID	0
0	0	3271	0.188037
1	1	1490	0.961508
2	0	7936	0.188037
3	1	4216	0.999049
4	0	3830	0.188037
5	1	1800	0.961508
6	0	6507	0.012329
7	0	4821	0.000445
8	1	4223	0.996691
9	0	4714	0.188037

```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 0 : 'Converted_Prob'})
```

```
# Rearranging the columns
y_pred_final = y_pred_final.reindex_axis(['Prospect ID','Converted','Converted_Prob'], axis=1)
y_pred_final.head(10)
```

	Prospect ID	Converted	Converted_Prob
0	3271	0	0.188037
1	1490	1	0.961508
2	7936	0	0.188037
3	4216	1	0.999049

CONTINUED…

Let's do the final prediction and check the overall accuracy

```
y_pred_final['final_predicted'] = y_pred_final.Converted_Prob.map(lambda x: 1 if x > 0.42 else 0)
```

```
y_pred_final.head()
```

	Prospect ID	Converted	Converted_Prob	final_predicted
0	3271	0	0.188037	0
1	1490	1	0.961508	1
2	7936	0	0.188037	0
3	4216	1	0.999049	1
4	3830	0	0.188037	0

```
# Let's check the overall accuracy.
metrics.accuracy_score(y_pred_final.Converted, y_pred_final.final_predicted)
```

```
0.9140653690782226
```

CONTINUED…

Let's do the final check on the Sensitivity and Specificity.

```
confusion3 = metrics.confusion_matrix(y_pred_final.Converted, y_pred_final.final_predicted )
confusion3
```

```
array([[1659,   75],
       [ 159,  830]], dtype=int64)
```

```
TP = confusion3[1,1] # true positive
TN = confusion3[0,0] # true negatives
FP = confusion3[0,1] # false positives
FN = confusion3[1,0] # false negatives
```

```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

```
0.839231547017189
```

```
# Let us calculate specificity
TN / float(TN+FP)
```

```
0.9567474048442907
```

THE END