Lead Scoring Case Study

ABHIJITH CV
SOMNATH BHONG

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.

Goals of the Case Study

Build a logistic regression model to assign a lead score between 0 and 100 to each of the leads which can be used by the company to target potential leads. A higher score would mean that the lead is hot, i.e. is most likely to convert whereas a lower score would mean that the lead is cold and will mostly not get converted. .

overall approach of the analysis

The files are captured , understood , prepared for analysis (cleaned , processed) ,a predictive model is build and analyses via different methods (statistical summaries and plotting). Then some conclusions are drawn based on the results which might help the company.

INDEX

1. **Importing libraries and Understanding Data**
2. **Data preparation**
3. **Exploratory Data Analysis**
4. **Creating dummy variables for all categorical variables**
5. **Train Test Splitting**
6. **Model Creation**
7. **Prediction and evaluation**
8. **Conclusion**

1. Importing libraries and Understanding Data

```
import numpy as np #  mathematical operations
import pandas as pd # data handling
import matplotlib.pyplot as plt # plots and graphs
%matplotlib inline
import seaborn as sns # APIs for plotting

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

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

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

This dataset consists of various attributes such as Lead Source, Total Time Spent on Website, Total Visits, Last Activity, etc.

which may or may not be useful in ultimately deciding whether a lead will be converted or not.

The target variable, in this case, is the column 'Converted' which tells whether a past lead was converted or not wherein 1 means it was converted and 0 means it wasn't converted.

2. Data preparation

```
# Converting all the values to lower case
df = df.applymap(lambda s:s.lower() if type(s) == str else s)
```

```
# Replacing 'select' is a missing value here
df = df.replace('select',np.nan)
```

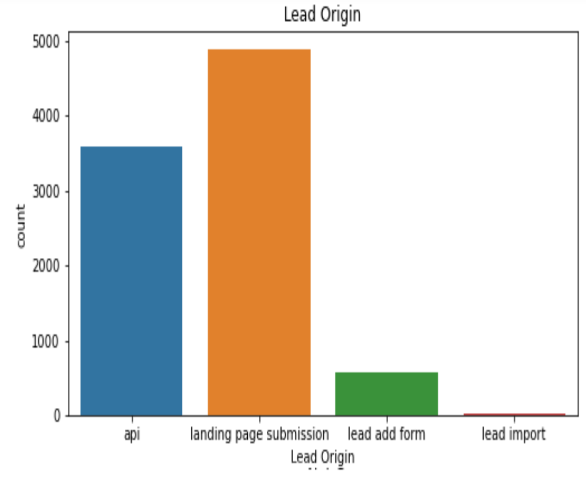
```
# unique value check
df.nunique()
df2.isnull().mean() * 100
```

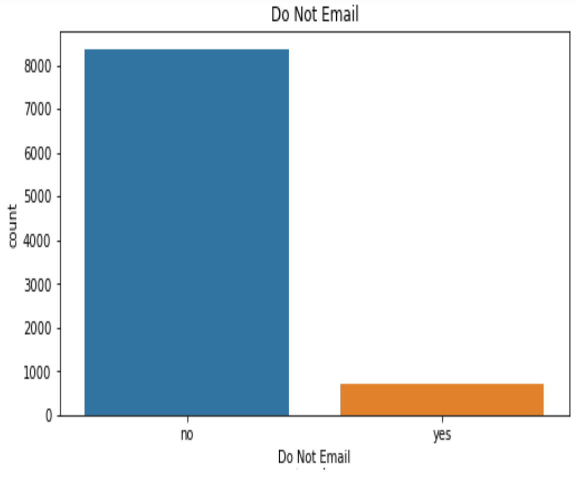
```
Prospect ID                                       0.000000
Lead Origin                                       0.000000
Lead Source                                       0.389610
Do Not Email                                      0.000000
Do Not Call                                       0.000000
Converted                                         0.000000
TotalVisits                                       1.482684
Total Time Spent on Website                       0.000000
Page Views Per Visit                              1.482684
Last Activity                                     1.114719
Country                                          26.634199
Specialization                                   36.580087
What is your current occupation                  29.112554
What matters most to you in choosing a course    29.318182
Search                                            0.000000
Newspaper Article                                 0.000000
X Education Forums                                0.000000
Newspaper                                         0.000000
Digital Advertisement                             0.000000
Through Recommendations                           0.000000
A free copy of Mastering The Interview            0.000000
Last Notable Activity                             0.000000
dtype: float64
```

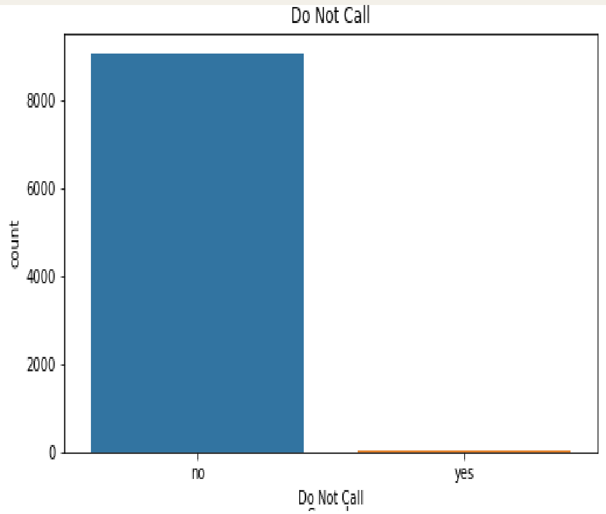
Country,Specialization,What is your current occupation ,What matters most to you in choosing a course are important columns even Tho they have a high number of missing values.

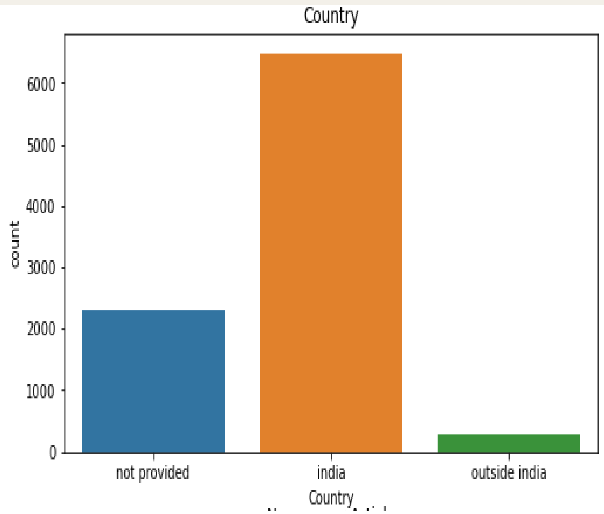
3. Exploratory Data Analysis

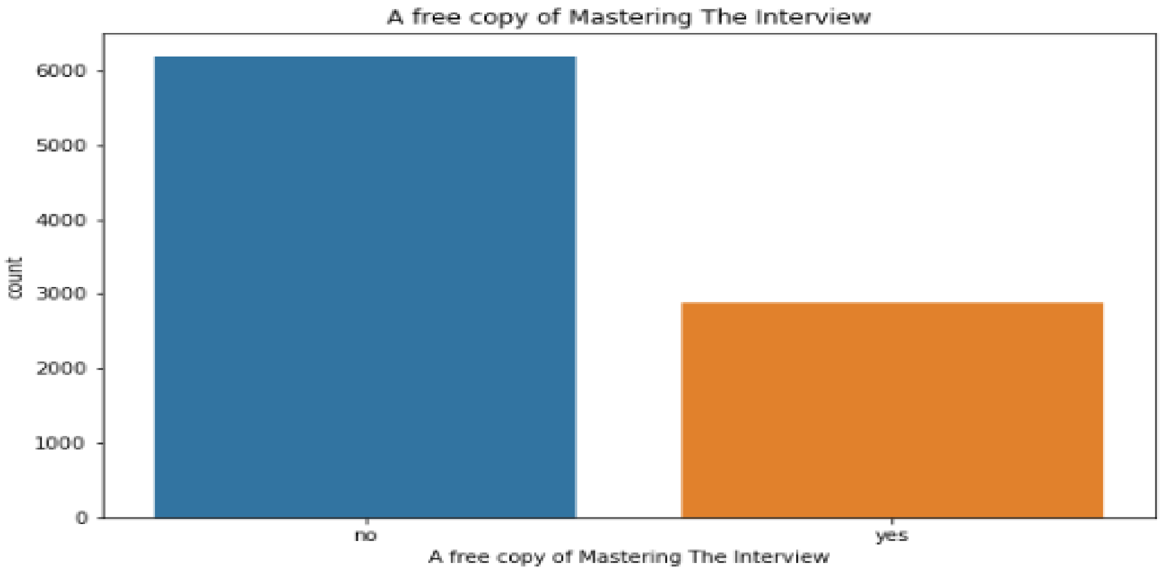
3.1 Categorical Variables Analysis

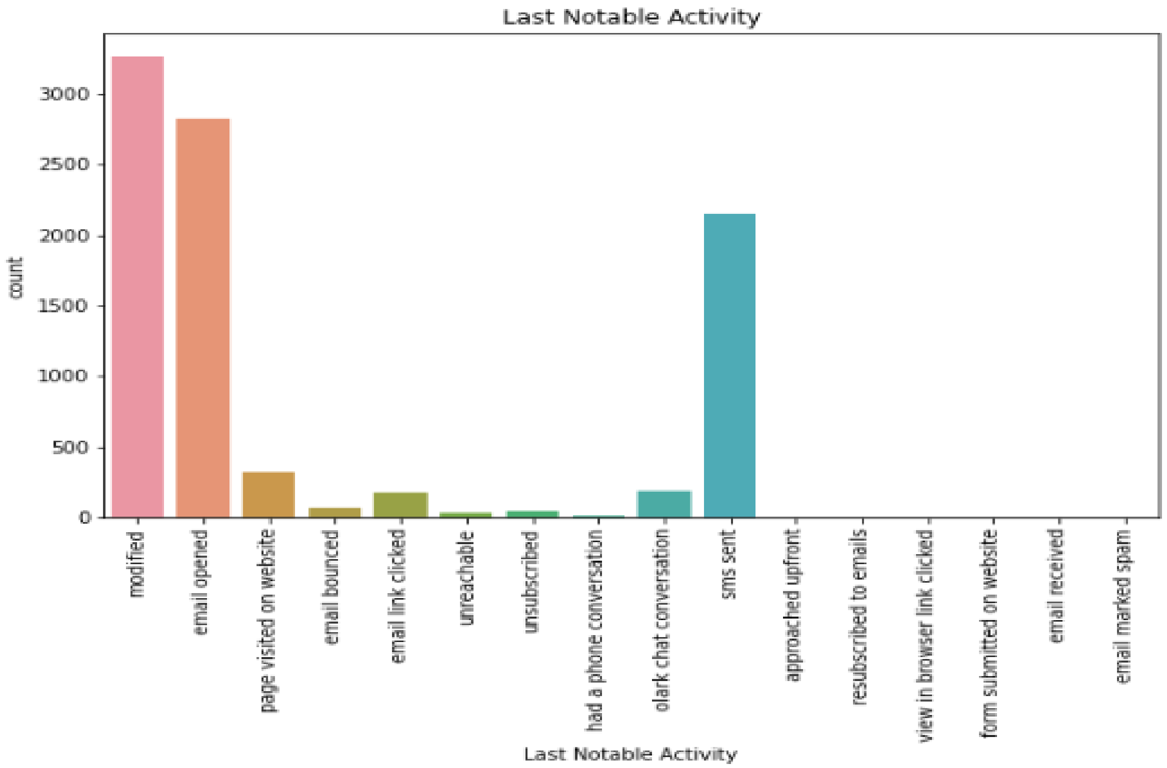




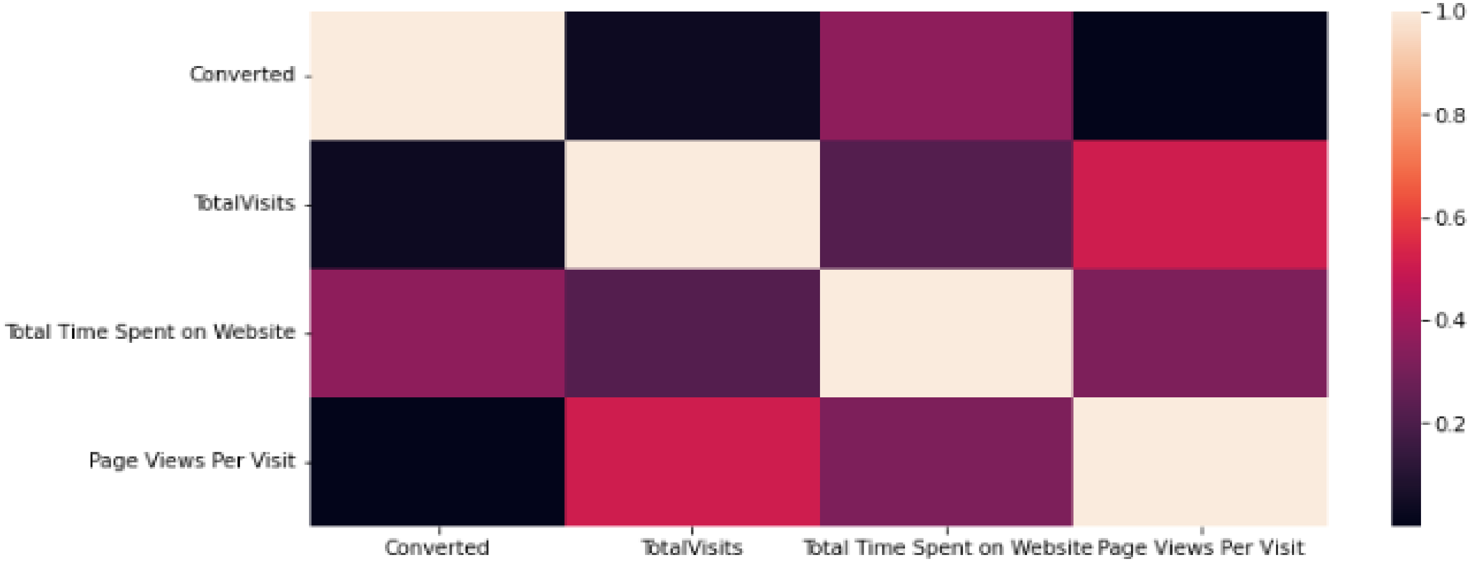


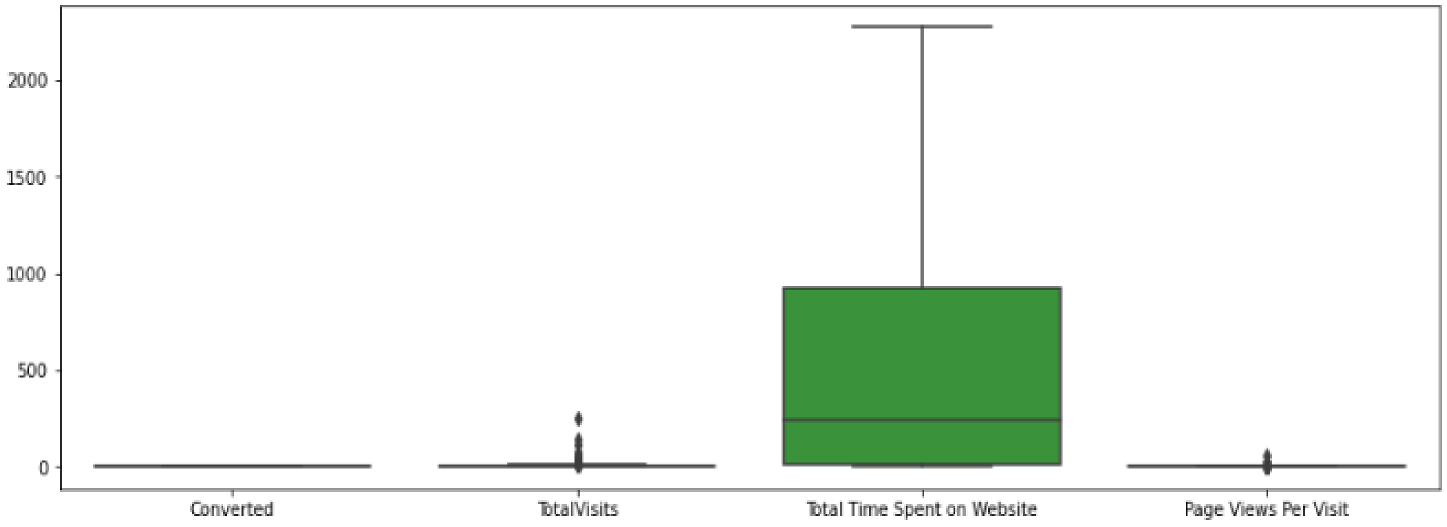






```
plt.figure(figsize=(11,5))
sns.heatmap(df_final.corr())
plt.show()
```





Page Views Per Visit and TotalVisits is moderately correlated other varibles have very low correlation.

outliers for numerical data are not significant

4. Creating dummy variables for all categorical variables

```
# selecting columns that are of object data type.
df_final.loc[:, df_final.dtypes == 'object'].columns
```

```
Index(['Lead Origin', 'Lead Source', 'Do Not Email', 'Do Not Call',
       'Last Activity', 'Country', 'Specialization',
       'What is your current occupation',
       'What matters most to you in choosing a course', 'Search',
       'Newspaper Article', 'X Education Forums', 'Newspaper',
       'Digital Advertisement', 'Through Recommendations',
       'A free copy of Mastering The Interview', 'Last Notable Activity'],
      dtype='object')
```

```
# Creating dummy variables
dummy = pd.get_dummies(df_final[['Lead Origin','Specialization' ,'Lead Source', 'Do Not Email', 'Last Activity', 'What is your cu
# Add the results to the master dataframe
df_final_dum = pd.concat([df_final, dummy], axis=1)
df_final_dum
```

	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Country	...	Last Notable Activity_form submitted on website	Last Notable Activity_had a phone conversation	Last Notable Activity_modified	Last No Activity_ convers
0	api	olark chat	no	no	0	0.0	0	0.00	page visited on website	not provided	...	0	0	1	
1	api	organic search	no	no	0	5.0	674	2.50	email opened	india	...	0	0	0	
2	landing page submission	direct traffic	no	no	1	2.0	1532	2.00	email opened	india	...	0	0	0	
3	landing page submission	direct traffic	no	no	0	1.0	305	1.00	unreachable	india	...	0	0	1	
4	landing page submission	google	no	no	1	2.0	1428	1.00	converted to lead	india	...	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

Dummy variables are useful because they enable us to use a single regression equation to represent multiple groups. This means that we don't need to write out separate equation models for each subgroup. The dummy variables act like 'switches' that turn various parameters on and off in an equation.

5. Train Test Splitting

5.1 Making X as predictor columns and y as targert variable

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

	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Lead Origin_landing page submission	Lead Origin_lead add form	Lead Origin_lead import	Specialization_business administration	Specialization_e-business	Specialization_e-commerce	Specialization_finance management	...
0	0.0	0	0.0	0	0	0	0	0	0	0	...
1	5.0	674	2.5	0	0	0	0	0	0	0	...
2	2.0	1532	2.0	1	0	0	1	0	0	0	...
3	1.0	305	1.0	1	0	0	0	0	0	0	...
4	2.0	1428	1.0	1	0	0	0	0	0	0	...

5 rows × 80 columns

```
# making Converted column as the target variable
y = df_final_dum['Converted']
y.head()
```

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

5.2 Train test split

```
# Split the dataset into 70% and 30% for train and test respectively
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=1)
```

Train test split is a model validation process that allows you to simulate how our model would perform with new data

6. Model Creation

Model 4

```
X_train_sm = sm.add_constant(X_train)
logm4 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm4.fit()
res.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	No. Observations:	6351
Model:	GLM	Df Residuals:	6338
Model Family:	Binomial	Df Model:	12
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-2612.7
Date:	Mon, 12 Sep 2022	Deviance:	5225.3
Time:	23:46:27	Pearson chi2:	6.42e+03
No. Iterations:	7		
Covariance Type:	nonrobust		

	Features	VIF
7	What is your current occupation_unemployed	2.03
0	Total Time Spent on Website	1.82
1	Lead Origin_lead add form	1.54
2	Lead Source_olark chat	1.49
9	Last Notable Activity_sms sent	1.44
5	Last Activity_olark chat conversation	1.37
3	Lead Source_welingak website	1.32
8	What is your current occupation_working profes...	1.30
4	Do Not Email_yes	1.13
11	Last Notable Activity_unsubscribed	1.08
6	What is your current occupation_student	1.04
10	Last Notable Activity_unreachable	1.01

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.2403	0.103	-31.467	0.000	-3.442	-3.038
Total Time Spent on Website	4.5857	0.167	27.383	0.000	4.257	4.914
Lead Origin_lead add form	3.8318	0.232	16.510	0.000	3.377	4.287
Lead Source_olark chat	1.3582	0.105	12.926	0.000	1.152	1.564
Lead Source_welingak website	2.4553	1.037	2.369	0.018	0.424	4.487
Do Not Email_yes	-1.6585	0.182	-9.094	0.000	-2.016	-1.301
Last Activity_olark chat conversation	-1.2007	0.157	-7.624	0.000	-1.509	-0.892
What is your current occupation_student	1.3015	0.224	5.802	0.000	0.862	1.741
What is your current occupation_unemployed	1.1473	0.087	13.138	0.000	0.976	1.318
What is your current occupation_working professional	3.5404	0.196	18.075	0.000	3.157	3.924
Last Notable Activity_sms sent	1.4507	0.080	18.093	0.000	1.294	1.608
Last Notable Activity_unreachable	1.8861	0.556	3.391	0.001	0.796	2.976
Last Notable Activity_unsubscribed	1.5137	0.485	3.124	0.002	0.564	2.463

RFE with 15 variables as output is done and many columns are dropped Arriving at model 4 which has very stable P value and VIF

7. Prediction and evaluation

7.1 Train set prediction

In [60]:
```
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

Out[60]:
```
7656    0.169349
7775    0.144146
5287    0.158272
3315    0.718001
4058    0.996477
363     0.167509
6714    0.726811
4797    0.608436
9109    0.122745
5264    0.043823
dtype: float64
```

In [61]:
```
# Reshaping to an array
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[61]:
```
array([0.16934934, 0.14414593, 0.15827243, 0.71800052, 0.99647661,
       0.16750861, 0.72681067, 0.60843611, 0.12274453, 0.04382329])
```

In [62]:
```
# Data frame with given convertion rate and probablity of predicted ones
y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Conversion_Prob':y_train_pred})
y_train_pred_final.head()
```

Out[62]:

	Converted	Conversion_Prob
0	0	0.169349
1	0	0.144146
2	0	0.158272
3	1	0.718001
4	1	0.996477

7.2 Model Evaluation

```
# Importing metrics from sklearn for evaluation
from sklearn import metrics
```

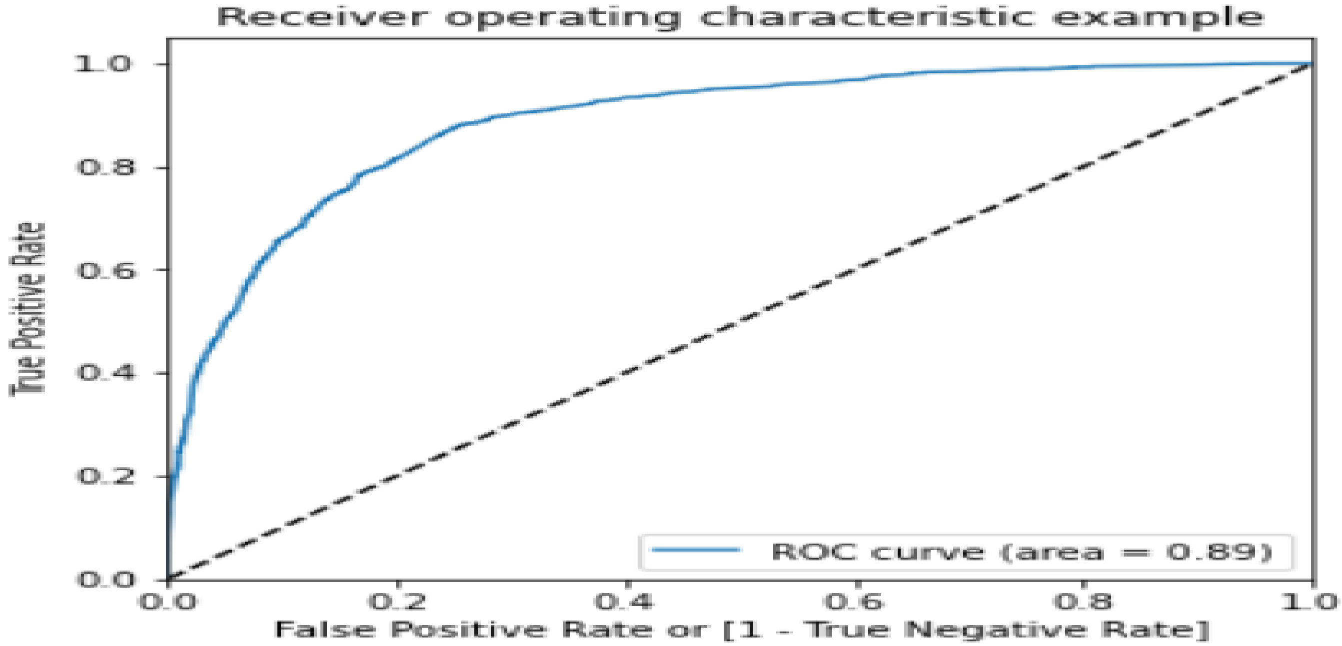
```
# Creating confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Predicted )
confusion
```

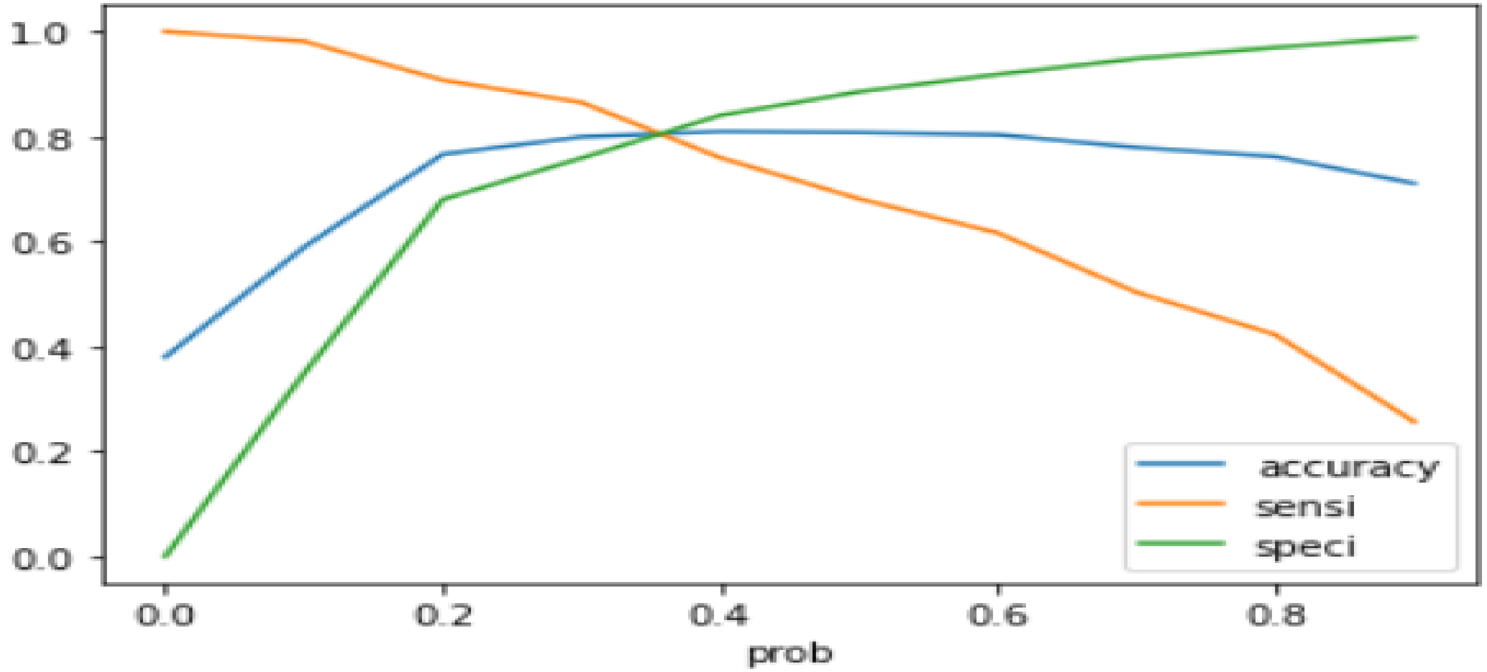
```
array([[3486,  453],
       [ 769, 1643]], dtype=int64)
```

```
# Check the overall accuracy
metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.Predicted)
```

```
0.807589356006928
```

This predictions allows business to make highly accurate guesses as to the likely outcomes of a question based on the collected historical data





ROC curve's area is 88 % which is acceptable

we can see that the best cut off is between .3 and .4

7.5 Prediction on Test set

```
# Making prediction using cut off 0.35
y_pred_final['final_predicted'] = y_pred_final.Conversion_Prob.map(lambda x: 1 if x > 0.35 else 0)
y_pred_final
```

	Converted	Conversion_Prob	final_predicted
0	0	0.426167	1
1	1	0.671714	1
2	0	0.177131	0
3	1	0.696273	1
4	0	0.132149	0
...	...	...	...
2718	1	0.890321	1
2719	0	0.141078	0
2720	0	0.062065	0
2721	0	0.164555	0
2722	0	0.229318	0

2723 rows × 3 columns

```
# Checking the overall accuracy
metrics.accuracy_score(y_pred_final['Converted'], y_pred_final.final_predicted)
```

```
0.8053617333822989
```

```
# Creating confusion matrix
confusion2 = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_final.final_predicted )
#confusion2
```

```
# sensitivity
print('sensitivity-',TP/(TP+FN))
# specificity
print('specificity-',TN/(TN+FP))
```

```
sensitivity- 0.804726368159204
specificity- 0.8075653719218076
```

7.6 Precision-Recall

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

```
# calculating Precision
confusion[1,1]/(confusion[0,1]+confusion[1,1]) # Precision = TP / TP + FP
```

```
0.7838740458015268
```

```
# calculating Recall
confusion[1,1]/(confusion[1,0]+confusion[1,1]) #Recall = TP / TP + FN
```

```
0.6811774461028193
```

The overall accuracy of the model is about 80.53 %

The sensitivity and specificity of the model is also found

The models Precision is about 78 % and its Recall is about 68 %

8. Conclusion

Variables that matters are the most in the potential buyers are,

The total time spend on the Website.
Total number of visits.
When the lead source was:

1. Google
2. Direct traffic
3. Organic search
4. Welingak website

When the last activity was:

1. SMS
2. Olark chat conversation

When the lead origin is Lead add format.
When their current occupation is as a working professionals.