Lead Scoring Case Study

Introduction

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%.

Problem Statement

- Lead conversion is low for X Education.
- Company wants to identify “Hot Leads” to increase the conversion.
- Initial lead generation is good but in the funnelling stage there are many leads which is not converted at the bottom.
- Create a model so that customer with high lead score have a higher conversion chance and vice versa.

Analysis

Step 1: Import important libraries and warnings

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

import time, warnings
import datetime as dt

from IPython.display import display
pd.options.display.max_columns = None

import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2: Read file and understand data type

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

```
lead.head()
```

```
lead.shape
```

```
(9240, 37)
```

```
lead.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
```

Analysis

Step 3: Check for Null value and remove null values

```
#checking for null
lead.isnull().sum()
```

```
Prospect ID                                        0
Lead Number                                        0
Lead Origin                                        0
Lead Source                                       36
Do Not Email                                       0
Do Not Call                                        0
Converted                                          0
TotalVisits                                      137
Total Time Spent on Website                        0
Page Views Per Visit                             137
Last Activity                                    103
Country                                         2461
Specialization                                  1438
How did you hear about X Education              2207
What is your current occupation                 2690
What matters most to you in choosing a course   2709
Search                                             0
Magazine                                           0
```

```
#Since there are not much distinct value in city and country hence dropping off
lead.drop(['City'], axis=1, inplace=True)
lead.drop(['Country'], axis=1, inplace=True)
```

```
#checking the % of null value
round(100*(lead.isnull().sum()/len(lead.index)),2)
```

```
Prospect ID                                     0.00
Lead Number                                     0.00
Lead Origin                                     0.00
```

Analysis

Step 4: Variables which were considered for next set of process

```
lead.isnull().sum()
```

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

Analysis

Step 5: Prepare data for modelling

Created dummy variable for the object data type

```
#adding dummies to the object data
lead= pd.get_dummies(data= lead, columns=['Lead Origin', 'Lead Source', 'Do Not Email', 'Last Activity',
                              'What is your current occupation','A free copy of Mastering The Interview',
                              'Last Notable Activity'], drop_first=True)
lead= pd.get_dummies(data= lead, columns=['Specialization'])
lead.head()
#lead = pd.concat([lead, dummy], axis=1)
```

```
lead.replace({False: 0, True: 1}, inplace=True)
```

```
lead.head()
```

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

Analysis

Step 6: Train and Split data

Data was split into 70% train and 30% test

```python
from sklearn.model_selection import train_test_split
```

```python
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```python
#X = Lead.drop(['Converted'], axis=1, inplace =True)
X=lead
X=X.drop(['Converted'],axis=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	Lead Source_Direct Traffic	Lead Source_Facebook	Lead Source_Google	Lead Source_Live Chat	Lead Source_Olark Chat	Source_
0	0.0	0	0.0	0	0	0	0	0	0	0	1	
1	5.0	674	2.5	0	0	0	0	0	0	0	0	
2	2.0	1532	2.0	1	0	0	1	0	0	0	0	
3	1.0	305	1.0	1	0	0	1	0	0	0	0	
4	2.0	1428	1.0	1	0	0	0	0	1	0	0	

```python
y=lead['Converted']
y.head()
```

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

```python
#Split the data into 70% train and 30% test data
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

Analysis

Step 7: Scaling the data

Using scaling to create homogeneous data across all the variables

```
#Using scalling to create homogenious data across all the variables
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()

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

X_train.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	Lead Source_Direct Traffic	Lead Source_Facebook	Lead Source_Google	Lead Source_Live Chat	Lead Source_Olark Chat	So
8003	0.015936	0.029489	0.125	1	0	0	1	0	0	0	0	
218	0.015936	0.082306	0.250	1	0	0	1	0	0	0	0	
4171	0.023904	0.034331	0.375	1	0	0	1	0	0	0	0	
4037	0.000000	0.000000	0.000	0	0	0	0	0	0	0	1	
3660	0.000000	0.000000	0.000	0	1	0	0	0	0	0	0	

Analysis

Step 8: Model Building

Used RFE to identify variable required to create the model

```
# Import 'LogisticRegression' and create a LogisticRegression object

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

```
# Import RFE and select 15 variables

from sklearn.feature_selection import RFE
rfe = RFE(logreg,n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)
```

```
#Features have been selected by RFE

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

```
[('TotalVisits', True, 1),
 ('Total Time Spent on Website', True, 1),
 ('Page Views Per Visit', False, 11),
 ('Lead Origin_Landing Page Submission', False, 2),
 ('Lead Origin_Lead Add Form', True, 1),
 ('Lead Origin_Lead Import', False, 49),
 ('Lead Source_Direct Traffic', False, 16),
 ('Lead Source_Facebook', False, 42),
 ('Lead Source_Google', False, 35),
 ('Lead Source_Live Chat', False, 39),
 ('Lead Source_Olark Chat', True, 1),
 ('Lead Source_Organic Search', False, 34),
 ('Lead Source_Pay per Click Ads', False, 33),
 ('Lead Source_Press_Release', False, 53),
 ('Lead Source_Reference', False, 4),
 ('Lead Source_Referral Sites', False, 36),
 ('Lead Source_Social Media', False, 58),
 ('Lead Source_WeLearn', False, 25),
 ('Lead Source_Welingak Website', True, 1),
 ('Lead Source_bing', False, 21),
 ('Lead Source_testone', False, 32),
```

Analysis

Step 9: Fitting logistic regression on the train data set

Used RFE to identify variable required to create the model

```
# Fitting Logistic Regression model on X_train post adding constant

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

Generalized Linear Model Regression Results

Dep. Variable:	Converted	No. Observations:	4461
Model:	GLM	Df Residuals:	4445
Model Family:	Binomial	Df Model:	15
Link Function:	Logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-2067.2
Date:	Tue, 20 Feb 2024	Deviance:	4134.4
Time:	11:48:44	Pearson chi2:	4.83e+03
No. Iterations:	22	Pseudo R-squ. (CS):	0.3676
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.9490	0.603	-1.573	0.116	-2.131	0.233
TotalVisits	10.2343	2.636	3.882	0.000	5.068	15.401
Total Time Spent on Website	4.4045	0.186	23.735	0.000	4.041	4.768
Lead Origin_Lead Add Form	4.2361	0.259	16.363	0.000	3.729	4.744
Lead Source_Olark Chat	1.6324	0.133	12.267	0.000	1.372	1.893
Lead Source_Welingak Website	2.3444	1.038	2.258	0.024	0.310	4.379
Do Not Email_Yes	-1.5177	0.192	-7.892	0.000	-1.895	-1.141
Last Activity_Had a Phone Conversation	1.1713	0.987	1.186	0.235	-0.764	3.106
Last Activity_SMS Sent	1.1787	0.082	14.305	0.000	1.017	1.340
What is your current occupation_Housewife	22.6104	2.45e+04	0.001	0.999	-4.8e+04	4.8e+04
What is your current occupation_Student	-1.1260	0.634	-1.776	0.076	-2.369	0.117
What is your current occupation_Unemployed	-1.2968	0.598	-2.169	0.030	-2.468	-0.125
What is your current occupation_Working Professional	1.2483	0.627	1.992	0.046	0.020	2.476
Last Notable Activity_Had a Phone Conversation	23.0106	2.09e+04	0.001	0.999	-4.09e+04	4.1e+04
Last Notable Activity_Unreachable	2.7670	0.807	3.429	0.001	1.186	4.348
Specialization_Select	-0.3400	0.098	-3.464	0.001	-0.532	-0.148

Analysis

Step 10: Checking VIF

Checking Variation inflation factor to manually remove any variable where VIF is >5

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

	Features	VIF
10	What is your current occupation_Unemployed	4.13
6	Last Activity_Had a Phone Conversation	2.44
12	Last Notable Activity_Had a Phone Conversation	2.43
1	Total Time Spent on Website	2.39
14	Specialization_Select	1.90
2	Lead Origin_Lead Add Form	1.71
3	Lead Source_Olark Chat	1.66
0	TotalVisits	1.63
7	Last Activity_SMS Sent	1.59
11	What is your current occupation_Working Profes...	1.56
4	Lead Source_Welingak Website	1.37
9	What is your current occupation_Student	1.10
5	Do Not Email_Yes	1.09
8	What is your current occupation_Housewife	1.01
13	Last Notable Activity_Unreachable	1.01

Since VIF for the all the variable was within 5 hence it good to go and now our focus is on p value >0.05

Analysis

Step 11: Checking p-value

For all the variable where p-value is greater than 0.05 remove it

```
X_train.drop('What is your current occupation_Housewife', axis = 1, inplace = True)
```

```
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:	4461
Model:	GLM	Df Residuals:	4447
Model Family:	Binomial	Df Model:	13
Link Function:	Logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-2072.0
Date:	Tue, 20 Feb 2024	Deviance:	4143.9
Time:	11:48:44	Pearson chi2:	4.84e+03
No. Iterations:	7	Pseudo R-squ. (CS):	0.3662
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4009	0.556	-0.721	0.471	-1.491	0.689
TotalVisits	10.1186	2.624	3.856	0.000	4.975	15.262
Total Time Spent on Website	4.3951	0.185	23.707	0.000	4.032	4.758
Lead Origin_Lead Add Form	4.2345	0.259	16.363	0.000	3.727	4.742
Lead Source_Olark Chat	1.6319	0.133	12.268	0.000	1.371	1.893
Lead Source_Welingak Website	2.3476	1.038	2.261	0.024	0.313	4.382
Do Not Email_Yes	-1.5207	0.192	-7.901	0.000	-1.898	-1.143
Last Activity_Had a Phone Conversation	2.7626	0.800	3.454	0.001	1.195	4.330
Last Activity_SMS Sent	1.1783	0.082	14.305	0.000	1.017	1.340
What is your current occupation_Student	-1.6673	0.591	-2.821	0.005	-2.826	-0.509
What is your current occupation_Unemployed	-1.8376	0.552	-3.329	0.001	-2.920	-0.756
What is your current occupation_Working Professional	0.7040	0.583	1.208	0.227	-0.439	1.847
Last Notable Activity_Unreachable	2.7623	0.807	3.422	0.001	1.180	4.345
Specialization_Select	-0.3488	0.098	-3.553	0.000	-0.541	-0.156

Removed variables- 'What is your current occupation_Working Professional, What is your current occupation_Housewife’

Analysis

Step 12: Model Evaluation

```
#  'predict' is used to predict the probabilities on the train set

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

```
8003    0.315577
218     0.151844
4171    0.135876
4037    0.278192
3660    0.959650
207     0.156043
2044    0.143676
6411    0.952580
6498    0.079814
2085    0.981919
dtype: float64
```

```
# Reshaping it into an array

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

```
array([0.3155766 , 0.1518439 , 0.13587609, 0.27819235, 0.95965009,
       0.1560432 , 0.14367596, 0.95258003, 0.07981364, 0.98191931])
```

```
# Create a new dataframe containing the actual conversion flag and the probabilities predicted by the model

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

	Converted	Conversion_Prob
0	0	0.315577
1	0	0.151844
2	1	0.135876
3	1	0.278192
4	1	0.959650

```
#Creating new column 'Predicted' with 1 if Paid_Prob > 0.5 else 0
y_train_pred_final['Predicted'] = y_train_pred_final.Conversion_Prob.map(lambda x: 1 if x > 0.5 else 0)

y_train_pred_final.head()
```

Analysis

Step 12: Model Evaluation

Sensitivity = 79%, Specificity -78% , Accuracy = 79% - hence this is a good model.

```
# Make predictions on the test set using 0.45 as the cutoff

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

```
# Check y_pred_final

y_pred_final.head()
```

	Converted	Conversion_Prob	final_predicted
0	1	0.996712	1
1	0	0.137945	0
2	0	0.717442	1
3	1	0.311448	0
4	1	0.731041	1

```
metrics.accuracy_score(y_pred_final['Converted'], y_pred_final.final_predicted)
```

```
0.7855648535564853
```

```
confusion2 = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_final.final_predicted )
confusion2
```

```
array([[779, 217],
       [193, 723]], dtype=int64)
```

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

```
# Calculate sensitivity
TP / float(TP+FN)
```

```
0.7893013100436681
```

```
# Calculate specificity
TN / float(TN+FP)
```

```
0.7821285140562249
```