LEAD SCORING CASE STUDY

HARSHIT KAMANI & GAURAV MAKHIJA

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.

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 higher lead score have a higher conversion chance and the customers with 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%.

DATA PROVIDED

You have been provided with a leads dataset from the past with around 9000 data points. 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. You can learn more about the dataset from the data dictionary provided in the zip folder at the end of the page. Another thing that you also need to check out for are the levels present in the categorical variables. Many of the categorical variables have a level called 'Select' which needs to be handled because it is as good as a null value (think why?).

GOALS OF THE CASE STUDY

There are quite a few goals for this 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.

There are some more problems presented by the company which your model should be able to adjust to if the company's requirement changes in the future so you will need to handle these as well. These problems are provided in a separate doc file. Please fill it based on the logistic regression model you got in the first step. Also, make sure you include this in your final PPT where you'll make recommendations.

Steps Involved:

- Importing Libraries
- Reading Data Set – leads.csv
- Checking basic data about DS like shape, info, describe, etc. to understand DS in a better way,
- Cleaning data & preparing it for further analysis.
- Checking for Missing Values – Imputing/Dropping as per scenario.
- Performing EDA & Checking for outliers.
- Creating Dummy variables & checking Correlation.
- Splitting the data into 'train' & 'test' sets.
- Building model on 'train' dataset.
- Evaluating the model built using measures like specificity and sensitivity.
- Making predictions on 'test' dataset.

Importing Libraries, Reading Data Set & Checking Basics

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

# model evaluation
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_recall_curve

# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')

plt.style.use("ggplot")
```

```python
leads_dataframe = pd.read_csv("Leads.csv")
pd.set_option('display.max_columns', None)
leads_dataframe.head()
```

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity	Country	Specialization	How did you hear about X Education	What i: you curren occupatio
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	Page Visited on Website	NaN	Select	Select	Unemploye
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	Email Opened	India	Select	Select	Unemploye
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	Email Opened	India	Business Administration	Select	Studen

Data Cleaning / Preparation

Converting some binary variables (Yes/No) to 0/1

```
# List of variables to map
varlist =  ['Do Not Email', 'Do Not Call', 'Search', 'Magazine', 'Newspap
            ,'Digital Advertisement','Through Recommendations','Receive M
            ,'Update me on Supply Chain Content','Get updates on DM Conte
           ,'A free copy of Mastering The Interview']
```

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

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

Converting SELECTs into NaNs:

```
# Listing the categorical variables yet to be encoded
leads_dataframe.select_dtypes(include='object').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 16 columns):
 #   Column                                   Non-Null Count  Dtype
```

Missing Value

```
# Checking the percentage of missing values
round(100*(leads_dataframe.isnull().sum()/len(leads_dataframe.index)), 2)
```

```
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                                   36.58
How did you hear about X Education               78.46
What is your current occupation                  29.11
What matters most to you in choosing a course    29.32
Search                                            0.00
Newspaper Article                                 0.00
X Education Forums                                0.00
Newspaper                                         0.00
Digital Advertisement                             0.00
Through Recommendations                           0.00
```

Data Cleaning / Preparation

Observations

There are five columns that still have high null values: `country` , `specialization` , `occupation` , `course_selection_reason` , and `city` . We will look at them individually to see what can be done

The distribution of data is heavily skewed in towards 'India (95%)' in 'Country' column and towards 'Better Career Prospects & NaN (99.9%)' in 'What matters most to you in choosing a course' column and 'Unemployed (85%)' in 'what is your current occupation' column. So it's safe to drop those columns.

```
# Dropping 'Country' and 'What matters most to you in choosing a course' columns
leads_dataframe = leads_dataframe.drop(['Country', 'What matters most to you in choosing a course', 'What is your current occupat
```

```
# We can impute the MUMBAI into all the NULLs as most of the values belong to MUMBAI
leads_dataframe['City'] = leads_dataframe['City'].replace(np.nan, 'Mumbai')
```

```
# Since there is no significant difference among top 3 specialisation , hence it will be safer to impute NaN with Others
leads_dataframe['Specialization'] = leads_dataframe['Specialization'].replace(np.nan, 'Other_Specialization')
```

```
# For Tags column, more than 30% data is for "Will revert after reading the email" and hence we can impute NULLS with Will revert
leads_dataframe['Tags'] = leads_dataframe['Tags'].replace(np.nan, 'Will revert after reading the email')
```

```
# Checking missing data percentage in the updated dataframe
round(100*(leads_dataframe.isnull().sum()/len(leads_dataframe.index)), 2)
```

Data Cleaning / Preparation

Handling categorical columns having low representation of categories

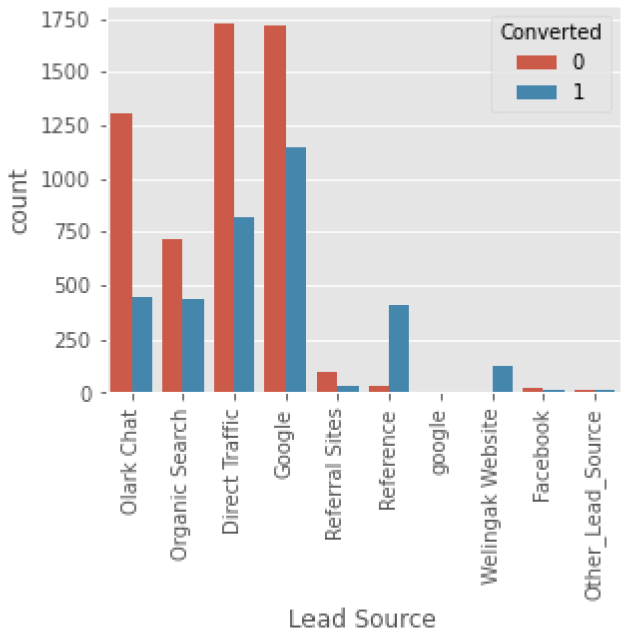
```
# determine unique values for all object datatype columns
for k, v in leads_dataframe.select_dtypes(include='object').nunique().to_dict().items():
    print('{} = {}'.format(k,v))
```

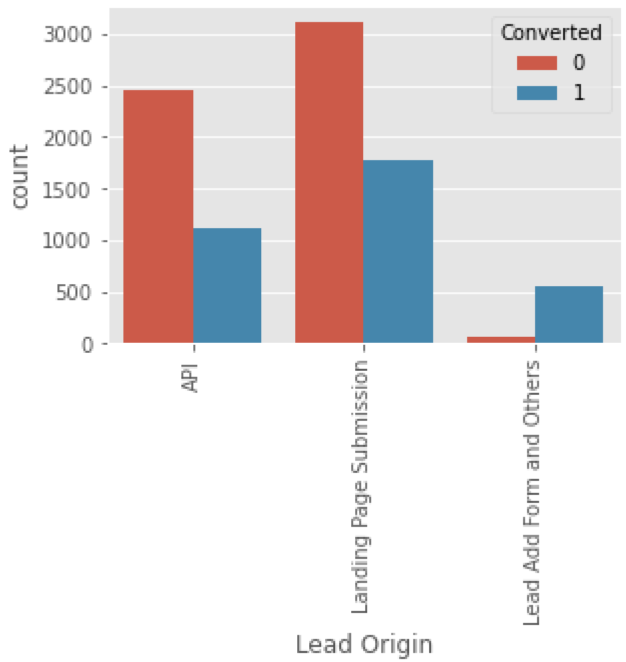
```
Prospect ID = 9074
Lead Origin = 4
Lead Source = 21
Last Activity = 17
Specialization = 19
Tags = 26
City = 6
Last Notable Activity = 16
```

```
# Lead Source Column - We can clearly observe that the count of leads from various sources are close to negligible and hence we c
leads_dataframe['Lead Source'] = leads_dataframe['Lead Source'].replace(['Click2call', 'Live Chat', 'NC_EDM', 'Pay per Click Ads'
  'Social Media', 'WeLearn', 'bing', 'blog', 'testone', 'welearnblog_Home', 'youtubechannel'], 'Other_Lead_Source')
```

```
# Last Activity Column - Converting all the low count categories to the 'Others' category
leads_dataframe['Last Activity'] = leads_dataframe['Last Activity'].replace(['Had a Phone Conversation', 'View in browser link Cl
                                                        'Visited Booth in Tradeshow', 'Approached upfront',
                                                        'Resubscribed to emails','Email Received', 'Email Marked Spam'], 'Other Ac
```

EDA





OBSERVATION:

- The count of leads from the Google and Direct Traffic is maximum
- The conversion rate of the leads from Reference and Welingak Website is maximum

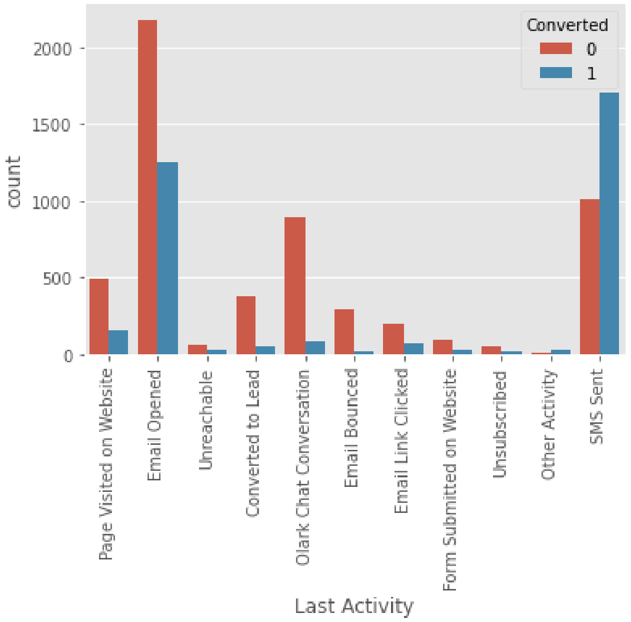
To improve the overall lead conversion rate, we need to focus on increasing the conversion rate of 'Google', 'Olark Chat', 'Organic Search', 'Direct Traffic' and also increasing the number of leads from 'Reference' and 'Welingak Website'

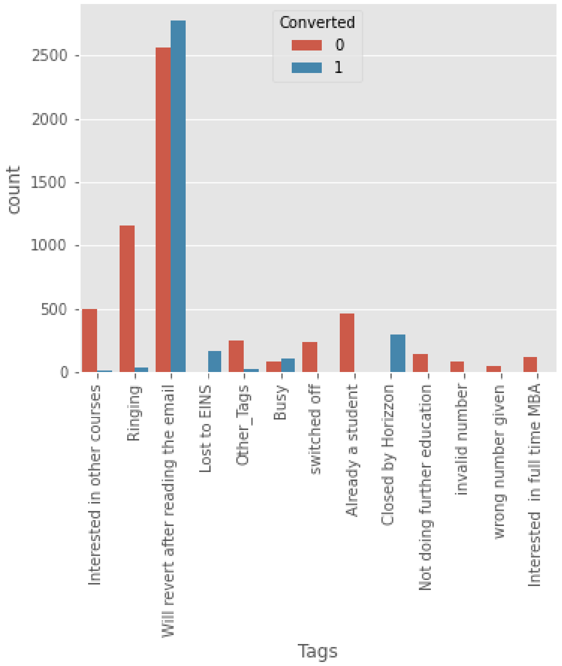
OBSERVATION:

- API and Landing Page Submission has less conversion rate(~30%) but counts of the leads from them are considerable
- The count of leads from the Lead Add Form and Others is pretty low but the conversion rate is very high

To improve the overall lead conversion rate, we need to focus on increasing the conversion rate of 'API' and 'Landing Page Submission' and also increasing the number of leads from 'Lead Add Form'

EDA





OBSERVATION:

- The count of lst activity as "Email Opened" is max
- The conversion rate of SMS sent as last activity is maximum

We should focus on increasing the conversion rate of those having last activity as Email Opened by making a call to those leads and also try to increase the count of the ones having last activity as SMS sent

OBSERVATION:

- Will revert after reading the email' and 'Closed by Horizzon' have high conversion rate

Dummy Variable Creation

```
# dummy encoding for the categorical variables
dummy = pd.get_dummies(leads_dataframe[['Lead Origin', 'Lead Source', 'Last Activity', 'Specialization',
                              'Tags','City','Last Notable Activity']], drop_first=True)
```

```
# getting the cleaned df
leads_dataframe = leads_dataframe.drop(['Lead Origin', 'Lead Source', 'Last Activity', 'Specialization',
                              'Tags','City','Last Notable Activity'], axis=1)
leads_dataframe = pd.concat([leads_dataframe, dummy], axis=1)
```

```
leads_dataframe.head()
```

	Prospect ID	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums	Newspaper	Digital Advertisement	Through Recommendations	A free copy of Mastering The Interview
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	0	0	0	0.0	0	0.0	0	0	0	0	0	0	0
1	2a272436-5132-4136-86fa-dcc88c88f482	0	0	0	5.0	674	2.5	0	0	0	0	0	0	0

Correlation

```
# columns pairs in order of highest absolute correlation
leads_dataframe.corr().abs().unstack().sort_values(ascending=False).drop_duplicates().head(12)
```

```
Do Not Email                                Do Not Email                              1.000000
Last Notable Activity_Unsubscribed          Last Activity_Unsubscribed                0.872656
Last Activity_Email Opened                  Last Notable Activity_Email Opened        0.861636
Last Notable Activity_SMS Sent              Last Activity_SMS Sent                    0.853102
Lead Source_Reference                       Lead Origin_Lead Add Form and Others      0.843166
Last Activity_Email Link Clicked            Last Notable Activity_Email Link Clicked  0.800686
Lead Origin_Landing Page Submission         Specialization_Other_Specialization       0.755381
TotalVisits                                 Page Views Per Visit                      0.737996
Newspaper Article                           X Education Forums                        0.707068
Last Notable Activity_Page Visited on Website  Last Activity_Page Visited on Website  0.691811
Do Not Email                                Last Activity_Email Bounced               0.620041
Last Activity_Unreachable                   Last Notable Activity_Unreachable         0.594369
dtype: float64
```

```
# Dropping variables with high multi-collinearity
leads_dataframe.drop(['Last Notable Activity_Unsubscribed', 'Last Activity_Email Opened', 'Last Notable Activity_SMS Sent', 'Lead
```

```
# Top 5 features correlated with target variable
leads_dataframe.corr()['Converted'].abs().sort_values(ascending=False).head(6)[1:]
```

```
Total Time Spent on Website            0.359261
Tags_Will revert after reading the email  0.348355
Last Activity_SMS Sent                 0.335815
Lead Origin_Lead Add Form and Others   0.291680
```

Train-Test Split

```
# Putting feature variable to X
X = leads_dataframe.drop(['Prospect ID','Converted'], axis=1)
# Putting response variable to y
y = leads_dataframe['Converted']

print(y)

X.head()
```

```
0       0
1       0
2       1
3       0
4       1
       ..
9235    1
9236    0
9237    0
9238    1
9239    1
Name: Converted, Length: 9074, dtype: int64
```

	Do Not Email	Do Not Call	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Search	Newspaper Article	X Education Forums	Newspaper	Digital Advertisement	Through Recommendations	A free copy of Mastering The Interview	Lead Origin_Landing Page Submission	Le Origin_Le Add Fo and Oth
0	0	0	0.0	0	0.0	0	0	0	0	0	0	0	0	

Step 8 - Scaling

```
# Scale the three numeric features present in the dataset
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	Search	Newspaper Article	X Education Forums	Newspaper	Digital Advertisement	Through Recommendations	A free copy of Mastering The Interview	Lead Origin_Landing Page Submission
3009	0	0	-0.432779	-0.160255	-0.179666	0	0	0	0	0	0	1	1
1012	1	0	-0.432779	-0.540048	-0.179666	0	0	0	0	0	0	0	1
9226	0	0	-1.150329	-0.888650	-1.132538	0	0	0	0	0	0	0	0
4750	0	0	-0.432779	1.643304	-0.179666	0	0	0	0	0	0	0	1
7987	0	0	0.643547	2.017593	0.058552	0	0	0	0	0	0	0	1

Model Building & Feature Selection

```
# Logistic regression model
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:**	6272
Model Family:	Binomial	**Df Model:**	78
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1638.7
Date:	Mon, 13 Jun 2022	**Deviance:**	3277.4
Time:	13:20:23	**Pearson chi2:**	1.35e+04
No. Iterations:	24		
Covariance Type:	nonrobust		

	coef	std err
const	-1.6146	0.825

```
# initiate logistic regression
logreg = LogisticRegression()

# initiate rfe
rfe = RFE(logreg, 15)
rfe = rfe.fit(X_train, y_train)
```

```
rfe.support_
```

```
array([ True, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True,  True, False, False,  True, False,
       False,  True,  True,  True,  True, False, False, False, False,
       False, False, False,  True, False,  True, False, False, False,
        True,  True,  True, False, False, False])
```

```
# Let's take a look at which features have been selected by RFE
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('Do Not Email', True, 1),
 ('Do Not Call', False, 49),
 ('TotalVisits', False, 46),
 ('Total Time Spent on Website', False, 7),
```

Model Evaluation

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

```
3009    0.494730
1012    0.572265
9226    0.002035
4750    0.899255
7987    0.978102
1281    0.899255
2880    0.494730
4971    0.818452
7536    0.494730
1248    0.002035
dtype: float64
```

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

```
array([0.49472997, 0.57226519, 0.00203471, 0.89925
       0.89925513, 0.49472997, 0.818452  , 0.49472
```

```
# Creating a dataframe with the true convertion st
y_train_pred_final = pd.DataFrame({'Convert':y_tra
y_train_pred_final['Pros_ID'] = y_train.index
y_train_pred_final.head()
```

```
# Dropping'Tags_invalid number'
X_train.drop('Tags_invalid number', axis = 1, inplace = True)
```

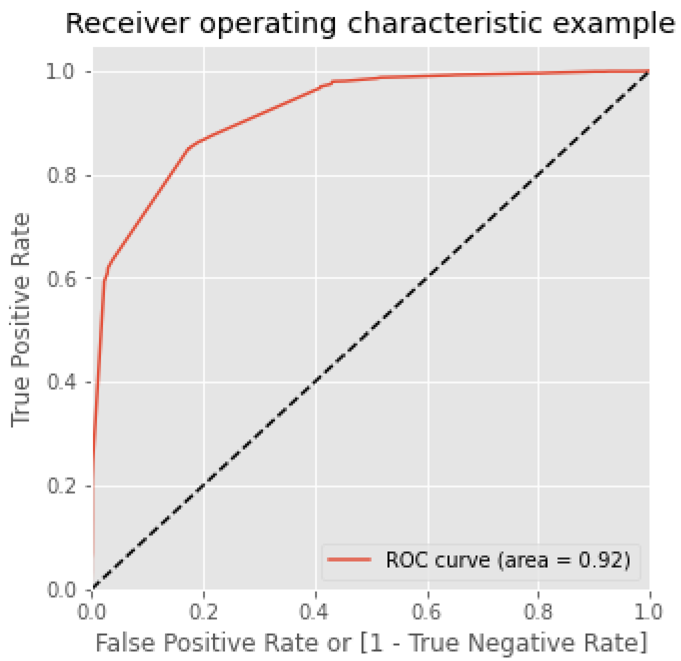
```
# Refit the model with the new set of features
X_train_sm = sm.add_constant(X_train)
logm4 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm4.fit()
res.summary()
```

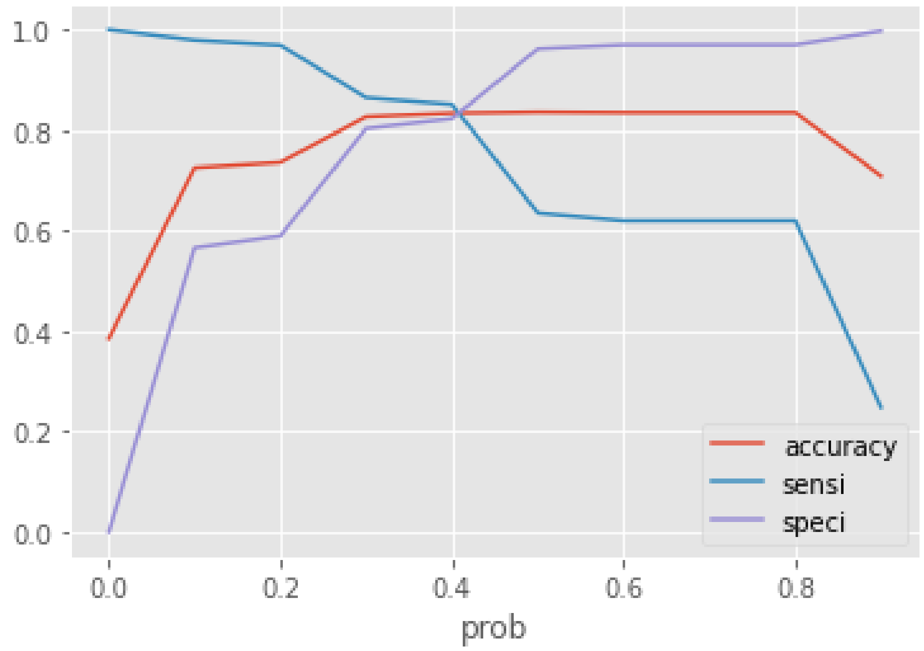
Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6351
Model:	GLM	**Df Residuals:**	6336
Model Family:	Binomial	**Df Model:**	14
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2139.6
Date:	Mon, 13 Jun 2022	**Deviance:**	4279.3
Time:	13:20:46	**Pearson chi2:**	1.17e+04
No. Iterations:	8		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]

ROC Curve





From the curve above, 0.42 is the optimum point to take it as a cut-off probability.

Measuring Accuracy & Other Measures

```
y_train_pred_final['final_predicted'] = y_train_pred_final.Convert_Prob.map( lambda x: 1 if x > 0.42 else 0)
y_train_pred_final.head()
```

	Convert	Convert_Prob	Pros_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	0.495669	3009	0	1	1	1	1	1	0	0	0	0	0	1
1	0	0.573673	1012	1	1	1	1	1	1						
2	0	0.002009	9226	0	1	0	0	0	0						
3	1	0.896470	4750	1	1	1	1	1	1						
4	1	0.978035	7987	1	1	1	1	1	1						

```
# Let's check the accuracy now
metrics.accuracy_score(y_train_pred_final.Convert, y_tra
```

```
0.8337269721303732
```

```
# Let's create the confusion matrix once again
confusion2 = metrics.confusion_matrix(y_train_pred_final
confusion2
```

```
array([[3215,  690],
       [ 366, 2080]], dtype=int64)
```

```
# Let's evaluate the other metrics as well
TP = confusion2[1,1] # true positive
```

```
# Let's evaluate the other metrics as well
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

```
# Calculate Sensitivity
TP/(TP+FN)
```

```
0.8503679476696647
```

```
# Calculate Specificity
TN/(TN+FP)
```

```
0.8233034571062741
```

Observation:

After running the model on the Train Data these are the figures we obtain:

- Accuracy : 83.37%
- Sensitivity : 85.03%
- Specificity :82.33%

Precision & Recall

```
confusion3 = metrics.confusion_matrix(y_train_pred_final.Convert, y_train_pred_final.final_predicted )
confusion3
```

```
array([[3215,  690],
       [ 366, 2080]], dtype=int64)
```

```
precision_score(y_train_pred_final.Convert, y_train_
```

```
0.7509025270758123
```

```
recall_score(y_train_pred_final.Convert, y_train_pre
```

```
0.8503679476696647
```

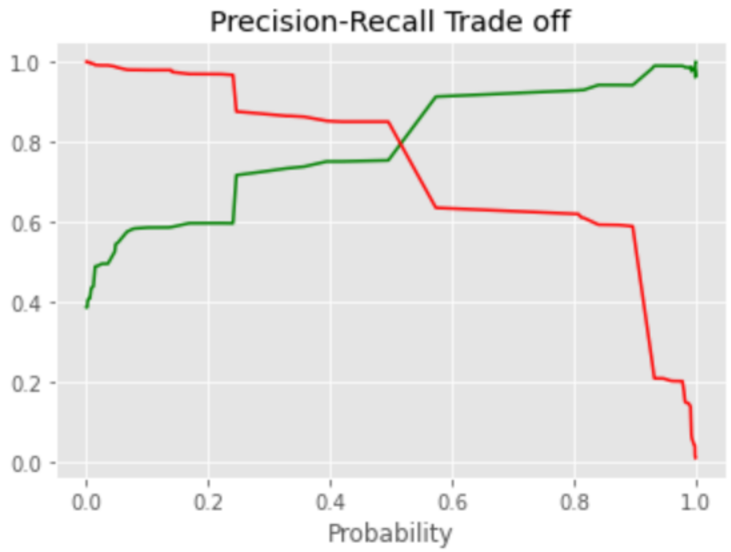
Precision and recall tradeoff

```
y_train_pred_final.Convert, y_train_pred_final.final
```

```
(0        0
 1        0
 2        0
 3        1
```

```
p, r, thresholds = precision_recall_curve(y_train_pred_final.Convert, y_train_pred_final.Convert_Prob)

plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
plt.xlabel("Probability")
plt.title("Precision-Recall Trade off")
plt.show()
```



Prediction on Test Set

```
# Getting the predicted values on the train set
y_test_pred = res.predict(X_test_sm)
```

```
# Coverting it to df
y_pred_1 = pd.DataFrame(y_test_pred)
y_pred_1.head()
```

	0
3271	0.495669
1490	0.495669
7936	0.495669
4216	0.998496
3830	0.495669

```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
```

```
# Putting CustID to index
y_test_df['Prospect ID'] = y_test_df.index
```

```
# Creating confusion matrix
confusion4 = metrics.confusion_matrix(y_pred_final['Converted'], y_pred_final.final_predicted )
confusion4
```

```
array([[1417,  317],
       [ 174,  815]], dtype=int64)
```

```
# Substituting the value of true positive
TP = confusion4[1,1]
# Substituting the value of true negatives
TN = confusion4[0,0]
# Substituting the value of false positives
FP = confusion4[0,1]
# Substituting the value of false negatives
FN = confusion4[1,0]
```

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

```
0.8240647118301314
```

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

```
0.8171856978085352
```

Final Observation

Final Observation: Let us compare the values obtained for Train & Test

Train Data:

- Accuracy : 83.37%
- Sensitivity : 85.03%
- Specificity :82.33%

Test Data:

- Accuracy : 81.96%
- Sensitivity : 82.40%
- Specificity : 81.71%