LEAD SCORING CASE STUDY

PREPARED BY:

Santhosh Venugopal

Sneha Alphonse Shaji

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%.

GOAL

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.

METHODOLOGY

- Data Loading
  The given dataset as part of the assignment is loaded.

- Data Inspection
  The given data is inspected using the .shape, .info(), .describe() methods in order to get a cursory view of the given data. Also we inspected the given columns and understood the significance of each by referring the data dictionary.
  This step is crucial to move forward to the next step i.e., data cleaning.

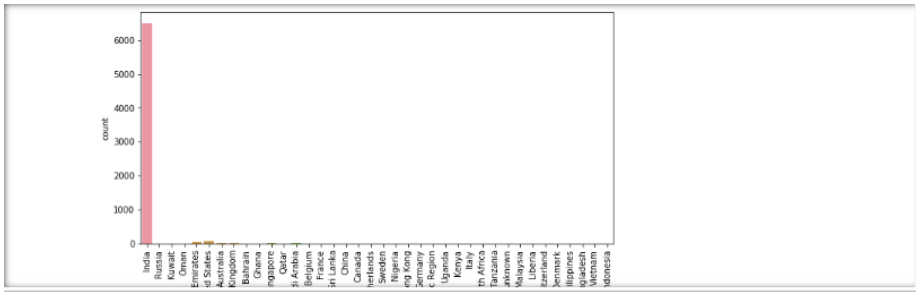
- Data Cleaning
  In order to make the data more readable and easily accessible, we decides to alter the column names and include '_' in place of spaces to avoid clutter.
  Primary or Unique columns, 'Prospect_ID' and 'Lead_Number' were then checked for any duplicate values, and it was found that there were no duplicate values, moreover these columns will not contribute anything to the analysis as such so we decided to drop these columns from the dataframe that we are going to analyze.
  Then we moved on to the missing values present. As mentioned in the problem statement, the value 'Select' must be considered as a 'NAN' value. So, we first replaced all 'Select' values with 'NAN'. Then checked for percentage of null values in each column and dropped columns with >= 45% null values.
  This step enabled creation of a clean dataset that can be now used for further analysis.

- Exploratory Data Analysis
  - Univariate Analysis

    First we picked the categorical variables and checked their significance in our analysis:

    1. Country

    In this column it was observed that the value 'India' was dominant.

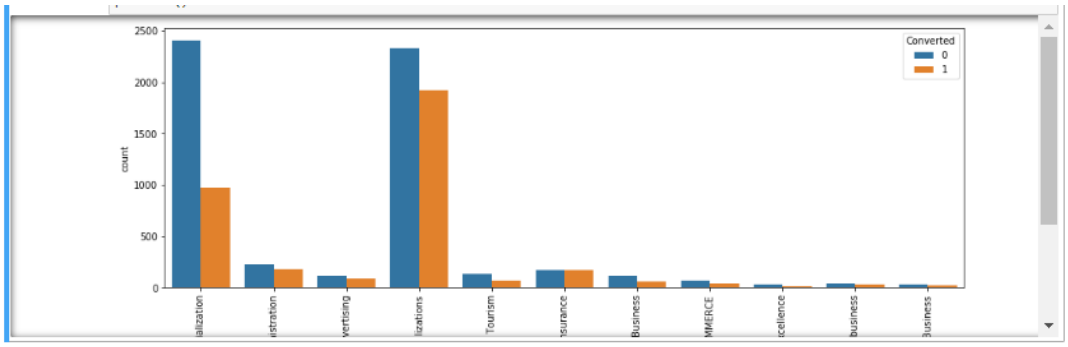


This will lead to class imbalance, so we dropped it.

2. Specialization

The null values were replaced by 'Other Specialization'.

This column was plotted against 'Converted' columne, which I sour target column to get some insights.

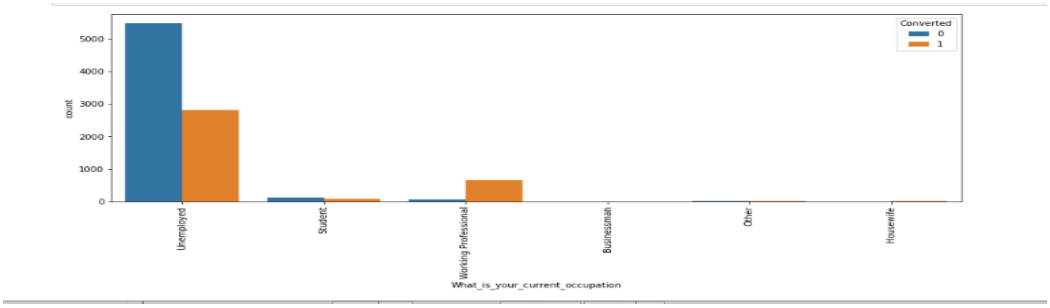


Specification is a significant column. It has good contribution to the overall data

3. Current Occupation

Null values were replaced by 'Unemployed' column.

This column was then plotted against the 'Converted' column.

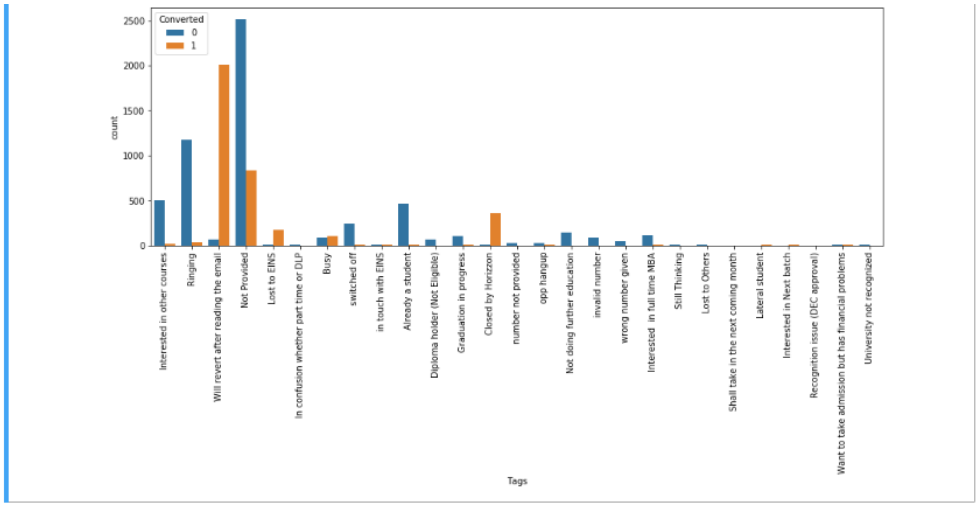


Unemployed has nearly 50% chance for being converted and Working personnel had higher chance to get converted. This column is Significant

4. What matters most to you in choosing a course

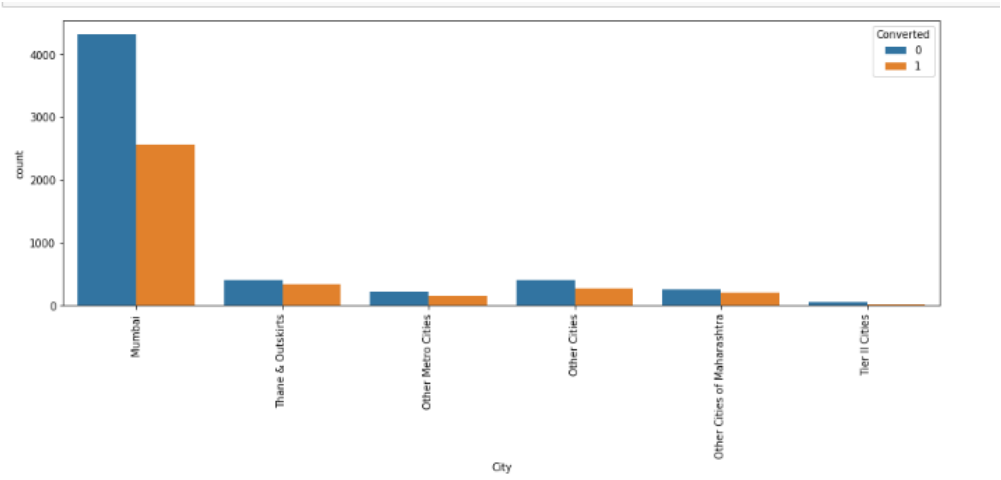
This column would cause class imbalance, hence it was dropped.

5. Tags



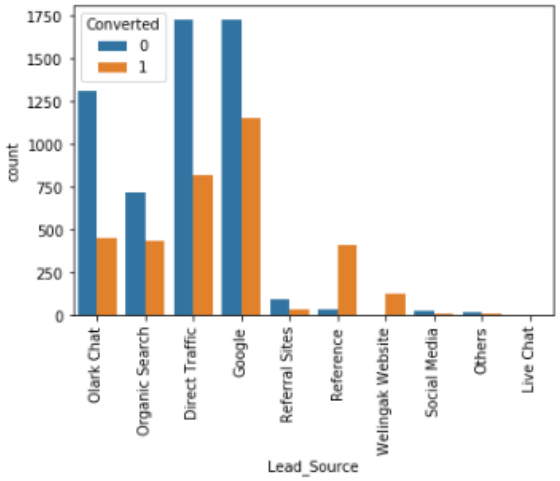
People who are tagged with **"Will Revert after reading the email"** mostly get converted. "Closed by Horizon" had more than 99% chances of being converted.

6. City



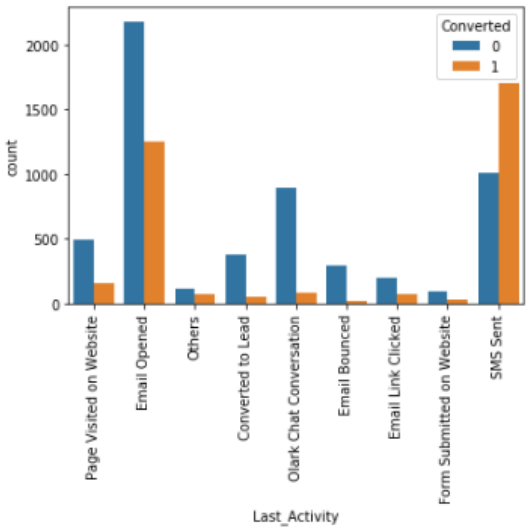
About 60% of people in Mumbai get converted. City column is significant

7. Lead Source



People who spend most of their time on Google got higher chances of being converted. Surprisingly **Reference** has 90% conversion rate though its quantity is low.

8. Last Activity



People who had their last activity as sending SMS showed higher conversion rate. Email opened shows 50% conversion rate.

- Class Imbalance Handling
  Dropped the following columns due to class imbalance:
  1. Do_Not_Call
  2. Search
  3. Magazine
  4. Newspaper_Article
  5. X_Education_Forums
  6. Newspaper
  7. Digital_Advertisement
  8. Through_Recommendations
  9. Receive_More_Updates_About_Our_Courses
  10. Update_me_on_Supply_Chain_Content
  11. Get_updates_on_DM_Content
  12. I_agree_to_pay_the_amount_through_cheque

- Test – Train Split
  Split the dataset into testing and training data for the model.

```
In [110]: X_train,X_test,y_train,y_test = train_test_split(X,y,train_size = 0.7,test_size = 0.3, random_state = 100)
```

- Scaling
  Rescaled numerical columns in order to get uniform results and bring all columns to same scale.
- Model Building
  - Feature Selection using RFE(Recursive Feature Elimination)
    Recursive Feature Elimination (**RFE**) as its title suggests recursively removes features, builds a model using the remaining attributes and calculates model accuracy. **RFE** is able to work out the combination of attributes that contribute to the prediction on the target variable (or class).
    Using RFE we selected top 15 features. Following were those features:

    1. Total_Time_Spent_on_Website
    2. Lead_Origin_Lead Add Form
    3. Lead_Source_Olark Chat

4. Lead_Source_Welingak Website
5. Last_Activity_Email Bounced
6. Last_Activity_SMS Sent
7. Tags_Closed by Horizzon
8. Tags_Interested in other courses
9. Tags_Lost to EINS
10. Tags_Other_Tags
11. Tags_Ringing
12. Tags_Will revert after reading the email
13. Last_Notable_Activity_Modified
14. Last_Notable_Activity_Olark Chat Conversation
15. Last_Notable_Activity_SMS Sent

- Model #1

  For the first model that we created, following is the summary:

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.5859	0.096	-16.496	0.000	-1.774	-1.398
Total_Time_Spent_on_Website	1.0830	0.061	17.798	0.000	0.964	1.202
Lead_Origin_Lead Add Form	1.7763	0.425	4.177	0.000	0.943	2.610
Lead_Source_Olark Chat	1.2092	0.144	8.406	0.000	0.927	1.491
Lead_Source_Welingak Website	3.4826	0.848	4.107	0.000	1.821	5.145
Last_Activity_Email Bounced	-1.4939	0.541	-2.760	0.006	-2.555	-0.433
Last_Activity_SMS Sent	1.3820	0.231	5.994	0.000	0.930	1.834
Tags_Closed by Horizzon	6.4254	0.738	8.705	0.000	4.979	7.872
Tags_Interested in other courses	-2.0275	0.394	-5.143	0.000	-2.800	-1.255
Tags_Lost to EINS	5.3589	0.533	10.063	0.000	4.315	6.403
Tags_Other_Tags	-2.6413	0.228	-11.605	0.000	-3.087	-2.195
Tags_Ringing	-3.6868	0.255	-14.470	0.000	-4.186	-3.187
Tags_Will revert after reading the email	4.4231	0.190	23.339	0.000	4.052	4.794
Last_Notable_Activity_Modified	-1.4460	0.155	-9.352	0.000	-1.749	-1.143
Last_Notable_Activity_Olark Chat Conversation	-1.7724	0.420	-4.216	0.000	-2.597	-0.948
Last_Notable_Activity_SMS Sent	0.7924	0.263	3.009	0.003	0.276	1.309

  From above summary we can see that, there is no high P-value, inferring that no correlation exists.
  Hence, we can further analyze using the Variance Inflation Factor(VIF)

  VIF:

	Features	VIF
14	Last_Notable_Activity_SMS Sent	6.52
5	Last_Activity_SMS Sent	6.36
12	Last_Notable_Activity_Modified	1.92
1	Lead_Origin_Lead Add Form	1.77
11	Tags_Will revert after reading the email	1.60
2	Lead_Source_Olark Chat	1.46
0	Total_Time_Spent_on_Website	1.44
3	Lead_Source_Welingak Website	1.31
6	Tags_Closed by Horizzon	1.20
9	Tags_Other_Tags	1.18
7	Tags_Interested in other courses	1.13
10	Tags_Ringing	1.11
4	Last_Activity_Email Bounced	1.10
8	Tags_Lost to EINS	1.06
13	Last_Notable_Activity_Olark Chat Conversation	1.06

There is a high correlation between **"Last_Notable_Activity_SMS Sent"** and "**Last_Activity_SMS Sent**". Therefore, we will first drop the first column and see the VIF's change.

- Model #2

  Following is the summary of second model:

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.5349	0.094	-16.374	0.000	-1.719	-1.351
Total_Time_Spent_on_Website	1.0773	0.061	17.788	0.000	0.959	1.196
Lead_Origin_Lead Add Form	1.7481	0.430	4.064	0.000	0.905	2.591
Lead_Source_Olark Chat	1.2349	0.143	8.653	0.000	0.955	1.515
Lead_Source_Welingak Website	3.4706	0.853	4.068	0.000	1.799	5.143
Last_Activity_Email Bounced	-1.3974	0.545	-2.566	0.010	-2.465	-0.330
Last_Activity_SMS Sent	1.9819	0.116	17.108	0.000	1.755	2.209
Tags_Closed by Horizzon	6.5893	0.739	8.917	0.000	5.141	8.038
Tags_Interested in other courses	-1.9646	0.392	-5.005	0.000	-2.734	-1.195
Tags_Lost to EINS	5.4909	0.535	10.265	0.000	4.443	6.539
Tags_Other_Tags	-2.5965	0.225	-11.519	0.000	-3.038	-2.155
Tags_Ringing	-3.5930	0.250	-14.349	0.000	-4.084	-3.102
Tags_Will revert after reading the email	4.4970	0.191	23.484	0.000	4.122	4.872
Last_Notable_Activity_Modified	-1.7329	0.128	-13.577	0.000	-1.983	-1.483
Last_Notable_Activity_Olark Chat Conversation	-1.8521	0.421	-4.397	0.000	-2.678	-1.027

Revised VIF:

	Features	VIF
1	Lead_Origin_Lead Add Form	1.77
11	Tags_Will revert after reading the email	1.55
12	Last_Notable_Activity_Modified	1.55
5	Last_Activity_SMS Sent	1.46
0	Total_Time_Spent_on_Website	1.44
2	Lead_Source_Olark Chat	1.43
3	Lead_Source_Welingak Website	1.31
6	Tags_Closed by Horizzon	1.20
9	Tags_Other_Tags	1.16
7	Tags_Interested in other courses	1.11
10	Tags_Ringing	1.10
4	Last_Activity_Email Bounced	1.09

Now, we can see that **multicollinearity** among variables has been removed and thus we can now go ahead with predictions.

- Prediction
  - On training set

    After making prediction on X_train, we got y_train_pred.

    Moved y_train and y_train_pred to a data frame.

33]:

	Converted	Converted_pred	Prospect ID
0	1	0.988634	6676
1	1	0.996319	6138
2	1	0.985684	8650
3	0	0.048376	3423
4	0	0.223347	6552

Then based on the predicated value on training dataset, we created another column where we assigned 1 or 0.

If the value predicted is >0.5 then 1 else 0 was assigned.

|:

	Converted	Converted_pred	Prospect ID	predicted
0	1	0.988634	6676	1
1	1	0.996319	6138	1
2	1	0.985684	8650	1
3	0	0.048376	3423	0
4	0	0.223347	6552	0

  - Confusion Matrix

    Confusion Matrix: A **confusion matrix** is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

i. Positive (P) : Observation is positive (for example: is an apple).
ii. Negative (N) : Observation is not positive (for example: is not an apple).
iii. True Positive (TP) : Observation is positive, and is predicted to be positive.
iv. False Negative (FN) : Observation is positive, but is predicted negative.
v. True Negative (TN) : Observation is negative, and is predicted to be negative.
vi. False Positive (FP) : Observation is negative, but is predicted positive.

```
from sklearn import metrics

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

```
[[3701  170]
 [ 290 2085]]
```

Accuracy:

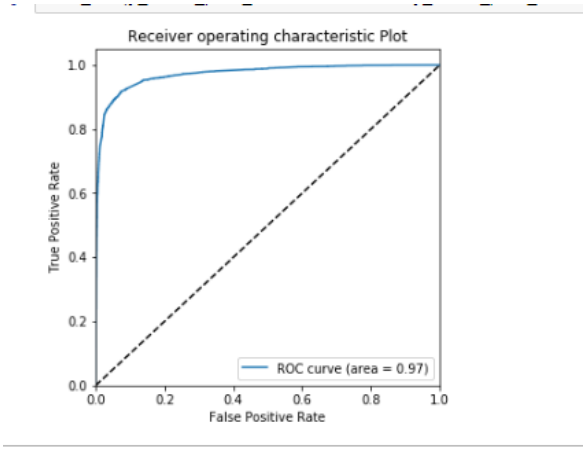
```
In [137]: #Accuracy
          print(metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.predicted))

          0.9263528658341339
```

Specificity and Sensitivity:

```
In [139]: #Sensitivity
          TP / float(TP+FN)

Out[139]: 0.8778947368421053

In [140]: #Specificity
          TN / float(TN+FP)

Out[140]: 0.9560836993025058
```

- ROC Curve

- It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
- The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.



- Optimal Cutoff Probability

  Logistic regression model is a probability, in order to use it as a classifier, we'll have to choose a cutoff value, or you can say it's a threshold value. Where scores above this value will classify as positive, those below as negative.

In [154]:
```
# Let's create columns with different probability cutoffs between 0 to 1
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final.Converted_pred.map(lambda x: 1 if x > i else 0)
y_train_pred_final.head()
```

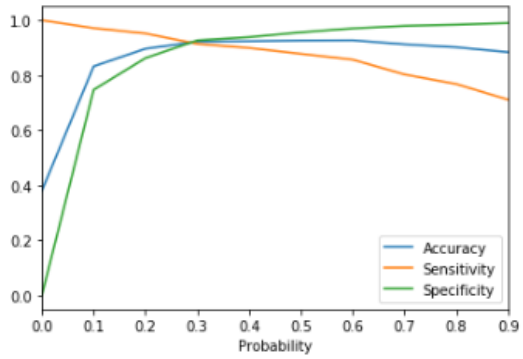
Out[154]:

	Converted	Converted_pred	Prospect ID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	1	0.988634	6676	1	1	1	1	1	1	1	1	1	1	1
1	1	0.996319	6138	1	1	1	1	1	1	1	1	1	1	1
2	1	0.985684	8650	1	1	1	1	1	1	1	1	1	1	1
3	0	0.048376	3423	0	1	0	0	0	0	0	0	0	0	0
4	0	0.223347	6552	0	1	1	1	0	0	0	0	0	0	0

We need to now find the accuracy, sensitivity and specificity of various probability cutoffs:

	Probability	Accuracy	Sensitivity	Specificity
0.0	0.0	0.380243	1.000000	0.000000
0.1	0.1	0.832533	0.971368	0.747352
0.2	0.2	0.896574	0.952000	0.862568
0.3	0.3	0.922030	0.914105	0.926892
0.4	0.4	0.923791	0.899368	0.938776
0.5	0.5	0.926353	0.877895	0.956084
0.6	0.6	0.927153	0.856842	0.970292
0.7	0.7	0.912424	0.802947	0.979592
0.8	0.8	0.901857	0.768000	0.983983
0.9	0.9	0.883926	0.711158	0.989925

Plot these values:



From above plot we can say the 0.3 is an optimum point to be taken as cutoff.

Based on this cutoff value, we then predicted the 'Predicted' column again on the training dataset.
If a value >0.3 we tag it as 1 else 0.

	Converted	Converted_pred	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	1	0.988634	6676	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0.996319	6138	1	1	1	1	1	1	1	1	1	1	1	1
2	1	0.985684	8650	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0.048376	3423	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0.223347	6552	0	1	1	1	0	0	0	0	0	0	0	0

Then we move on to 'Lead Score' assignment:

Assigning Lead Score

```
In [226]: y_train_pred_final['Lead_Score'] = y_train_pred_final.Converted_pred.map( lambda x: round(x*100))

          y_train_pred_final.head()
```

Out[226]:

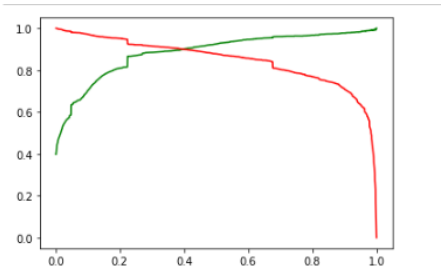
	Converted	Converted_pred	Prospect ID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_predicted	Lead_Score
0	1	0.988634	6676	1	1	1	1	1	1	1	1	1	1	1	1	99
1	1	0.996319	6138	1	1	1	1	1	1	1	1	1	1	1	1	100
2	1	0.985684	8650	1	1	1	1	1	1	1	1	1	1	1	1	99
3	0	0.048376	3423	0	1	0	0	0	0	0	0	0	0	0	0	5
4	0	0.223347	6552	0	1	1	1	0	0	0	0	0	0	0	0	22

After we obtained the value of 'final_predicted' column, we again got the confusion matrix and calculated the Precision and recall values:

```
In [171]: confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.predicted )
          confusion

Out[171]: array([[3701,  170],
                 [ 290, 2085]], dtype=int64)

In [174]: #Precision
          #TP / TP + FP
          confusion[1,1]/(confusion[0,1]+confusion[1,1])

Out[174]: 0.9246119733924612

In [175]: #Recall
          #TP / TP + FN
          confusion[1,1]/(confusion[1,0]+confusion[1,1])

Out[175]: 0.8778947368421053

In [176]: from sklearn.metrics import precision_score, recall_score

In [177]: precision_score(y_train_pred_final.Converted , y_train_pred_final.predicted)

Out[177]: 0.9246119733924612

In [178]: recall_score(y_train_pred_final.Converted, y_train_pred_final.predicted)

Out[178]: 0.8778947368421053
```

- Precision Recall Curve

  A **precision-recall curve** is a plot of the **precision** (y-axis) and the **recall** (x-axis) for different thresholds.



Here, the cutoff is almost equal to 0.3. So, we can keep this model.

- Prediction on test Set

Finally, we move on to predictions on test dataset.

	Prospect ID	Converted	Converted_pred	final_predicted
0	7625	0	0.139377	0
1	5207	1	0.978003	1
2	2390	1	0.996789	1
3	4362	0	0.025761	0
4	1023	0	0.021559	0

Following is the accuracy, sensitivity and specificity obtained:

```
In [218]: #accuracy score
          metrics.accuracy_score(y_pred_final.Converted, y_pred_final.final_predicted)

Out[218]: 0.9241971620612397

In [219]: confusion2 = metrics.confusion_matrix(y_pred_final.Converted, y_pred_final.final_predicted )
          confusion2

Out[219]: array([[1560,  124],
                 [  79,  915]], dtype=int64)

In [220]: TP = confusion2[1,1] # true positive
          TN = confusion2[0,0] # true negatives
          FP = confusion2[0,1] # false positives
          FN = confusion2[1,0] # false negatives

In [221]: #Sensitivity
          TP / float(TP+FN)

Out[221]: 0.920523138832998

In [222]: #Specificity
          TN / float(TN+FP)
```

CONCLUSION

OBSERVATIONS:

Train Dataset

- Accuracy - 92.2 %
- Sensitivity - 91.41 %
- Specificity - 92.69 %

Test Data

- Accuracy = 92.42 %
- Sensitivity = 92.05 %
- Specificity = 92.64 %

Thus, we started with dealing the missing values and did the Univariate Analysis in both the Categorical and Numerical categories and removed the unwanted columns. Later we did the modal building, removed the multi-collinearity. Then we proceeded with finding the optimal cut-off probability. Finally, we tested our Test data and found their respective probability of being converted.