Analysis to find the probability of Lead Numbers to become student of X Education.

Title and Content Layout with List

1. Problem Statement.
2. Roadmap of analysis.
   i. Treatment of outliers.
   ii. Exploratory Data Analysis.
   iii. Data preparation for modeling.
3. Analysis of the training model.
4. Optimal cut-off
5. Validating Model on Test Data
6. Results

Problem Statement :

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.

Roadmap of analysis

Understand Data, and check for any abnormality in data.

Inspection of missing values treat them accordingly.

Exploratory Data Analysis, for categorical and continuous data, also treat the outliers.

Preparing Data fro model building:

Create dummy variables for categorical columns.

Split the data into Test and Train data set with 70=30 ratio.

Also scaled features such as ['TotalVisits','Total Time Spent on Website', 'Page Views Per Visit'].

Prepare the Logistic regression Model, and select the optimal features using mixed approach of RFE and manual selection.

Train the data and obtain predicted value for Conversion probability of Lead Number.

Check the Accuracy, Sensitivity and Specificity, ROC curve area. Check for the optimal Cutoff for all. And use that probability.

Validate Logistic regression Model on test data-set

Final Result @Optimal cut-off of 0.43:

Accuracy of train is 0.79 & test is 0.79

Sensitivity of train is 0.76 & test is 0.77

Specificity of train is 0.80 & test is 0.80

ROC Curve area of Train Dataset: 0.86 & test : 0.87

Probabilities converted into lead score for Test Data.

Treatment of Missing values.

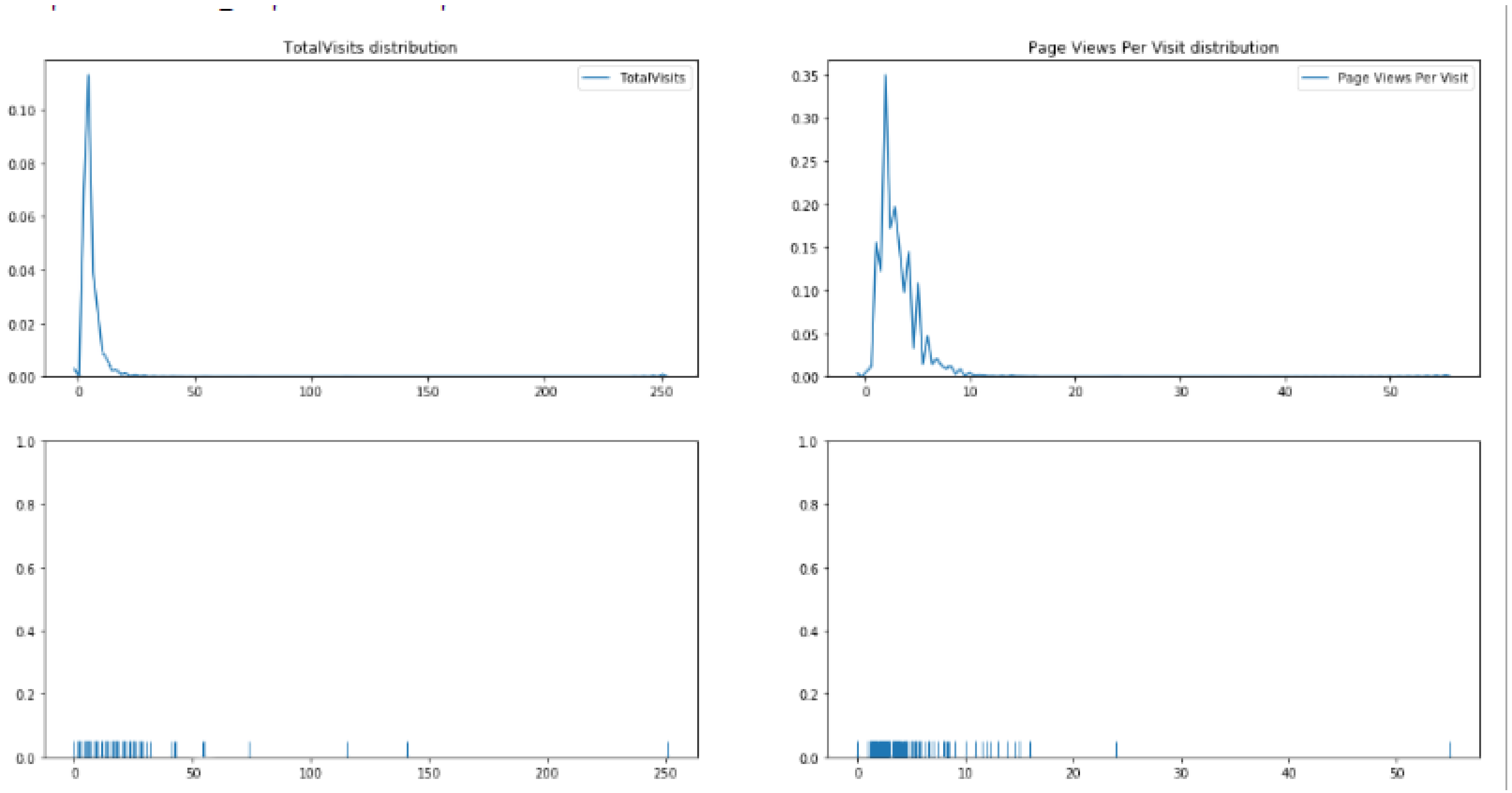
- columns having high missing values are ['How did you hear about X Education','Lead Quality','Lead Profile'] which need to be dropped.
- ***remove below columns as index and score assigned to each customer based on their activity and their profile***.[Asymmetrique Activity Index 45.65, Asymmetrique Profile Index 45.65, Asymmetrique Activity Score 45.65, Asymmetrique Profile Score 45.65]

Analysis of **Categorical columns for imputing missing values**

- 1. Lead Source Columns: 'Google','Direct Traffic','Olark Chat','Organic Search', 'Reference' can be considered as different categories and rest can be considered in Lead_Source_Others category.
- Will impute missing values with mode. Also lowercase google need to be replaced with uppercase Google.
- 2. Last Activity: 'Email Opened','SMS Sent','Olark Chat Conversation','Page Visited on Website' can be considered as different categories and rest other can be considered in Last_Activity_others category. Will impute missing values with mode.
- 3. Country: Looking at country it is clear that 70% is of India and 27% is missing values, which if imputed with mode will comprise of 97%. Hence this column does not give any significant information and can be dropped.
- 4.Specialization: As we can see in Specialization column maximum percentage is of missing value that is 37%, need to check for missing values in rows.
- 5. What is your current occupation: 61% is unemployed and 29% is missing value and after that 8% are working professional and 2% are student. need to check for missing values in rows. Categorised them under Others category
- 6. What matters most to you in choosing a course: Clearly it 71% for better career prospects and 29% missing value, rest less than 1%. Hence this column can be dropped.
- 7.Tags : For Tags 36% is Null values, and rest Will revert after reading the email has 22% and Ringing has 13%, also Tags columns can be dropped as it is given by executives of the company after attempting call to leads. need to check for missing values in rows. Categorised them under Others category
- 8.City: 40% is missing value and 35% is Mumbai, if we impute missing values with mode, then also this column doe not give any significant information. need to check for missing values in rows. Categorised them under Others category

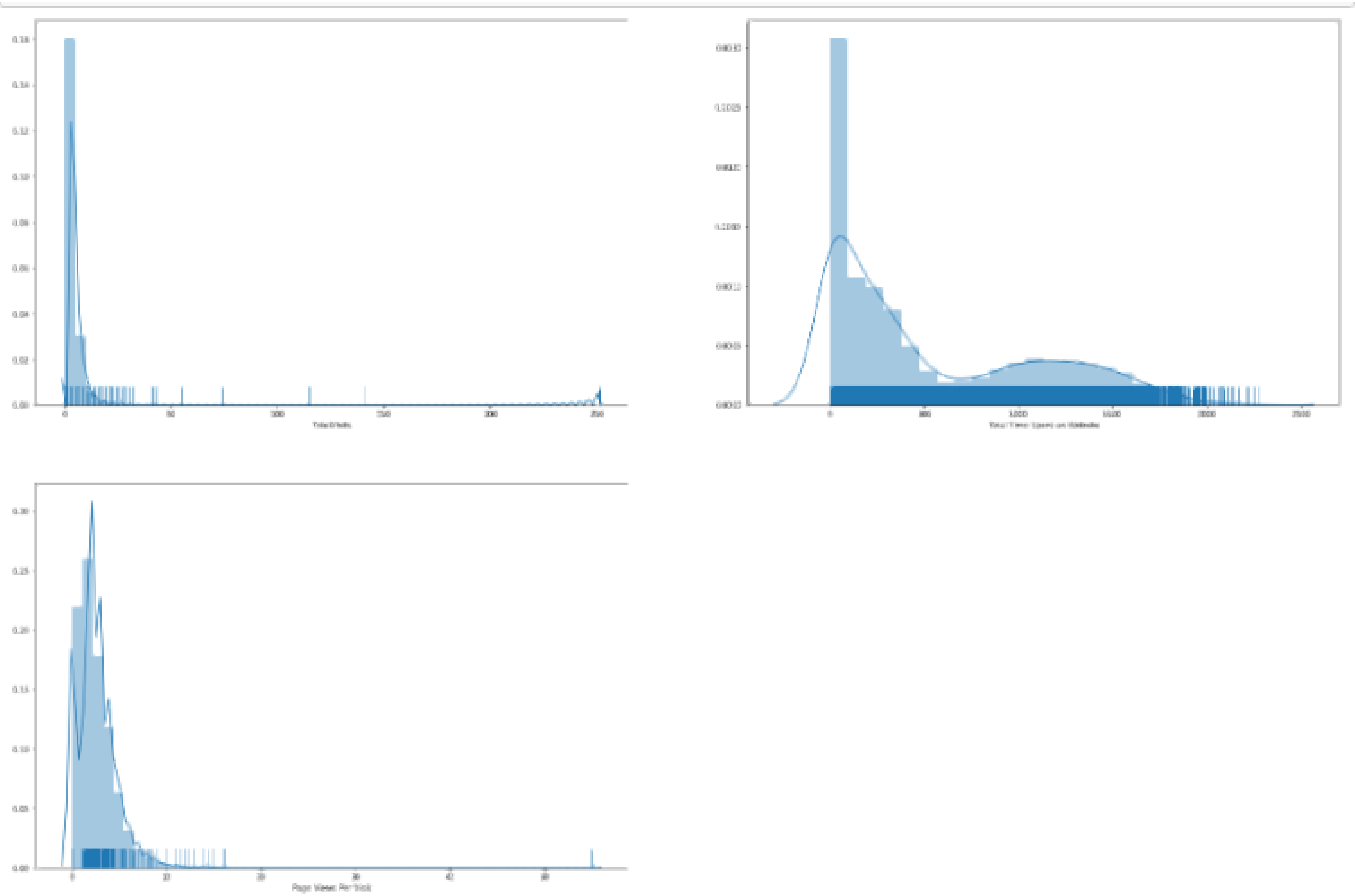
- Analysis of Continuous columns for treating missing values

As both continuous columns seems to be skewed, will impute missing values with median.



Treatment of outliers in TotalVisits and Page View Per Visit

Capped the outliers in soft range for both columns.



Data preparation for modelling

- Created dummy for categorical_col = [['Lead Origin', 'Lead Source', 'Last Activity','Specialization', 'What is your current occupation','City',]]

nce nent	Human Resource Management	IT Projects Management	Marketing Management	Operations Management	Supply Chain Management	Student	Unemployed	Working Professional	Metro Cities	Tier II Cities
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...
0	0	1	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	1	0	0	0

Split data into 70-30 ratio. And analysed the outcome for trained data.

- Split the data into 70-30 and after that analysed the outcome of that data such as Accuracy, Specificity, Sensitivity and ROC curve for the same.

Accuracy

[1670]:

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

executed in 74ms, finished 11:01:35 2020-09-07

```
0.7902635431918009
```

Sensitivity : Number of actual Conversion predicted / Total Number of actual conversion

[1671]:

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

executed in 136ms, finished 11:01:36 2020-09-07

[1672]:

```
Sensitivity = TP/float(TP+FN)
Sensitivity
```

executed in 90ms, finished 11:01:36 2020-09-07

[1672]: 0.7280954339777869

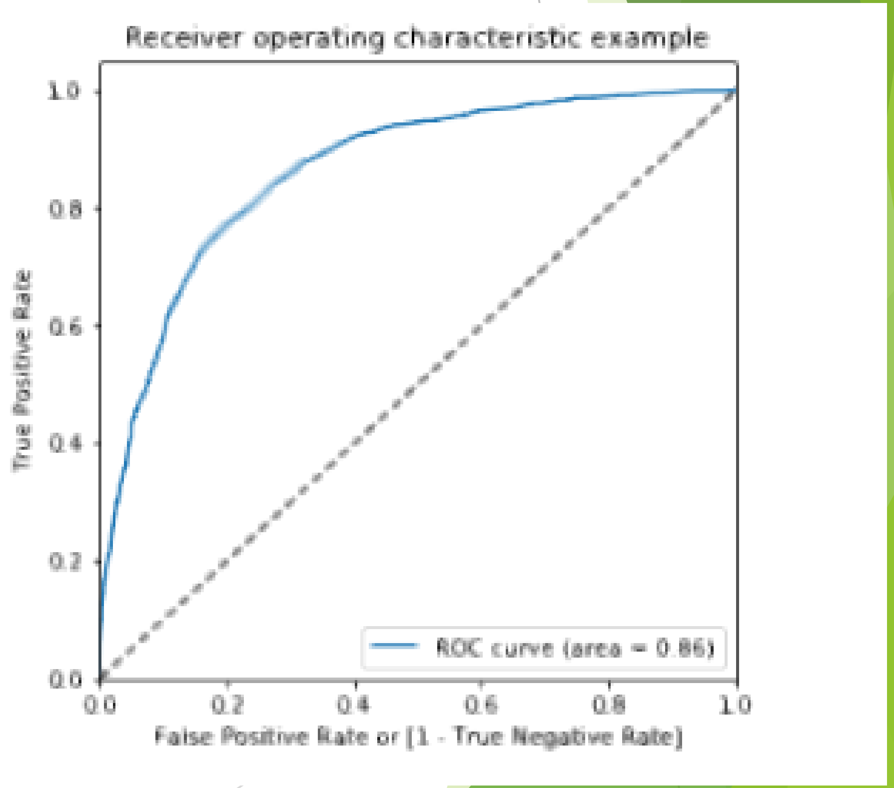
Specificity : Number of actual non-Conversion predicted / Total Number of actual non-conversion

[1673]:

```
Specificity = TN/float(TN+FP)
Specificity
```

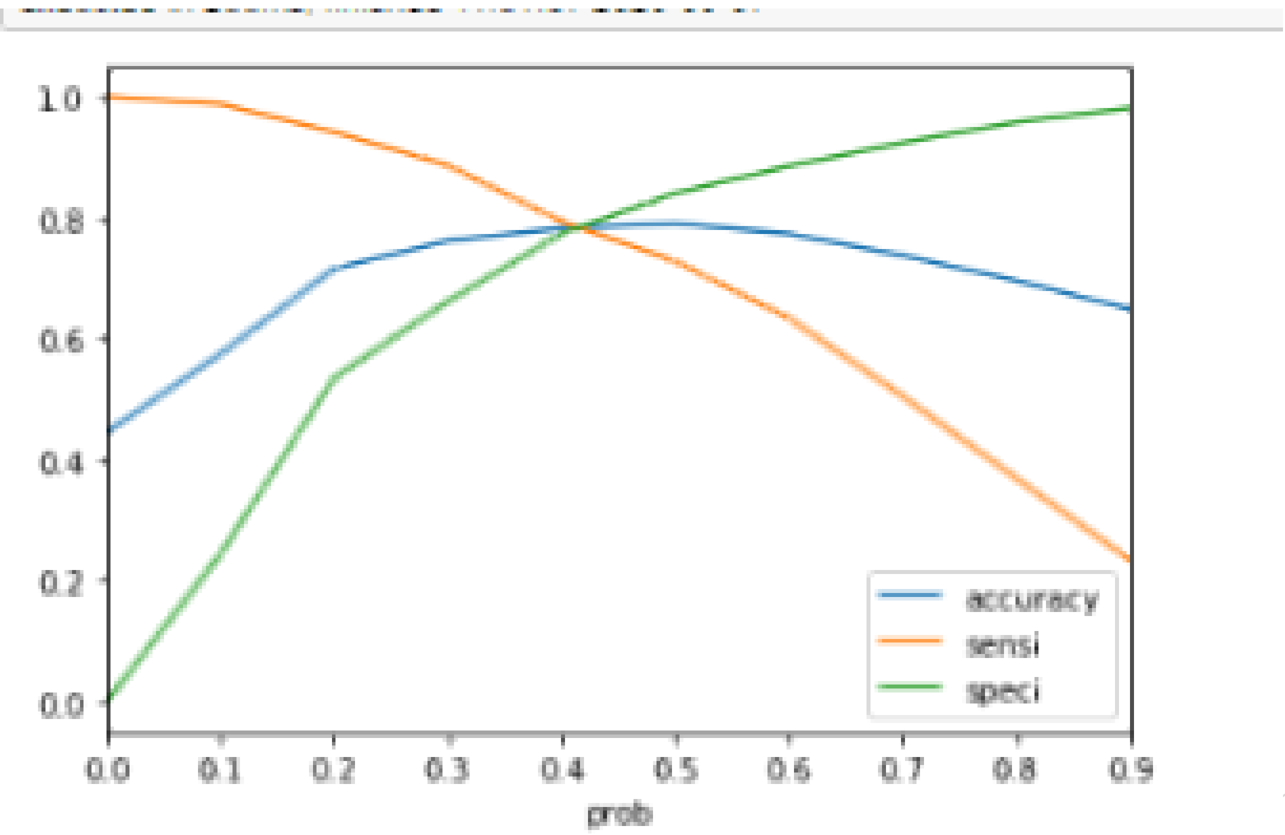
executed in 91ms, finished 11:01:36 2020-09-07

[1673]: 0.840092317837125



Check for the optimal cut-off.

- As it is quite clear that optimum cut-off for would be somewhere around .43. Hence revised the cut-off and again checked for the Accuracy, Sensitivity, Specificity.



Accuracy, Sensitivity, Specificity after revised cut-off.

Accuracy after revision of Probability cut-off at optimal cut-off

n [1721]:

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

executed in 21ms, finished 11:39:45 2020-09-07

ut[1721]: 0.7866032210834554

Sensitivity after revision of Probability cut-off at optimal cut-off ¶

[1684]:

```
Sensitivity = TP/float(TP+FN)
Sensitivity
```

executed in 167ms, finished 11:01:37 2020-09-07

t[1684]: 0.7758124228712464

Specificity after revision of Probability cut-off at optimal cut-off

[1685]:

```
Specificity = TN/float(TN+FP)
Specificity
```

executed in 90ms, finished 11:01:37 2020-09-07

t[1685]: 0.7952522255192879

Validating logistic regression model on Test data-set

Accuracy

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

executed in 22ms, finished 11:11:35 2020-09-07

```
0.7904396073410158
```

Sensitivity : Number of actual Conversion predicted / Total Number of actual conversion ¶

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

executed in 17ms, finished 11:11:40 2020-09-07

```
Sensitivity = TP/float(TP+FN)
Sensitivity
```

executed in 11ms, finished 11:11:43 2020-09-07

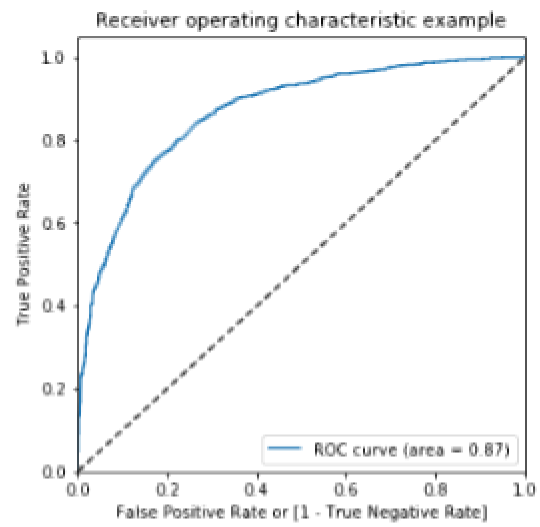
```
0.7730061349693251
```

Specificity : Number of actual non-Conversion predicted / Total Number of actual non-conversion

```
Specificity = TN/float(TN+FP)
Specificity
```

executed in 14ms, finished 11:11:46 2020-09-07

```
0.802930402930403
```



Final conversion of probability into Lead Score for each Lead Number

Conversion of Probability into lead score.

1725]:

```
y_test_pred_final['Lead Score'] = round(y_test_pred_final['Conversion_Probability']*100,2)
y_test_pred_final[['Lead Number','Lead Score']]
```

executed in 59ms, finished 11:42:56 2020-09-07

1725]:

Index	Lead Number	Lead Score
5354	608784	11.09
5162	610375	73.63
9226	579735	63.83
6271	601727	39.34
5386	608539	40.41
...	...	...
7849	589580	69.15
2185	638761	29.55
3393	627679	5.77
3226	629243	21.08
3229	629227	11.86

2343 rows × 2 columns

Final Result:

- Accuracy of train dataset @ optimal cut-off of .43 : 0.79
- Accuracy of test dataset @ optimal cut-off of .43 : 0.79

- Sensitivity of train dataset @ optimal cut-off of .43: 0.76
- Sensitivity of test dataset @ optimal cut-off of .43: 0.77

- Specificity of train @ optimal cut-off of .43: 0.80
- Specificity of test @ optimal cut-off of .43: 0.80

- ROC Curve area of Train Dataset: 0.86
- ROC Curve area of Test Dataset: 0.87

- Probabilities converted into lead score for Test Data.