


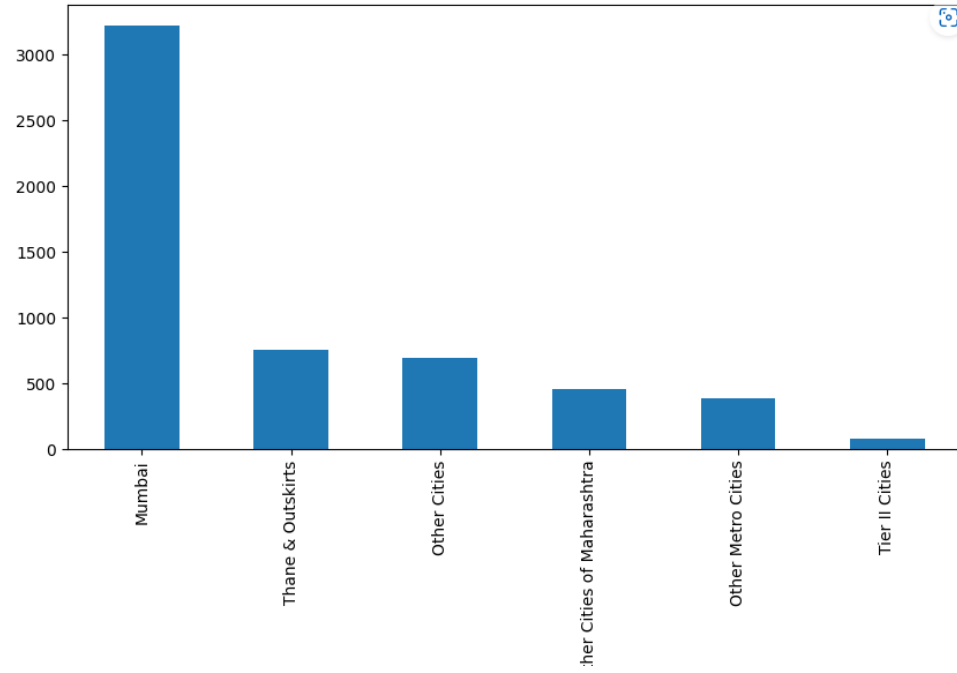
Lead Scoring Case Study



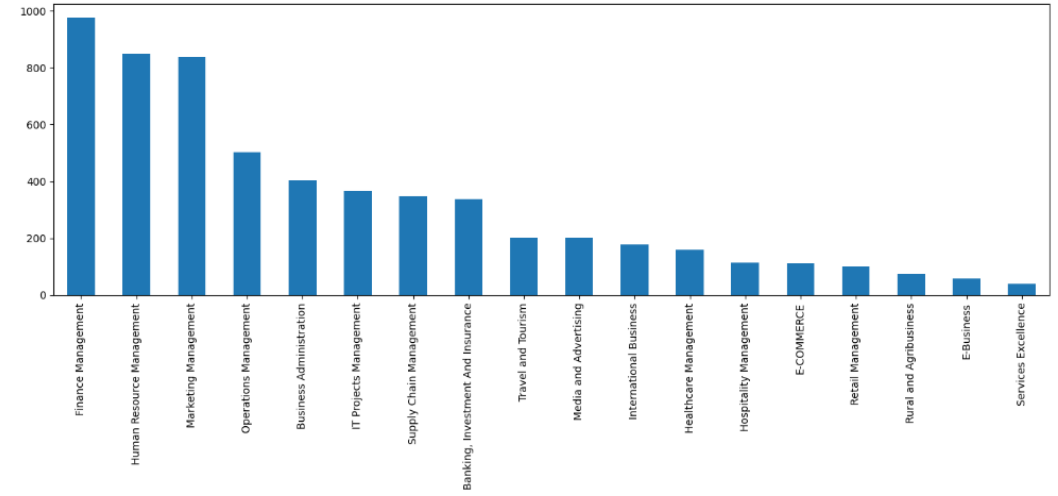
by
Siddhesh Haware
Arathi Unni
Kishlay

Data Cleaning

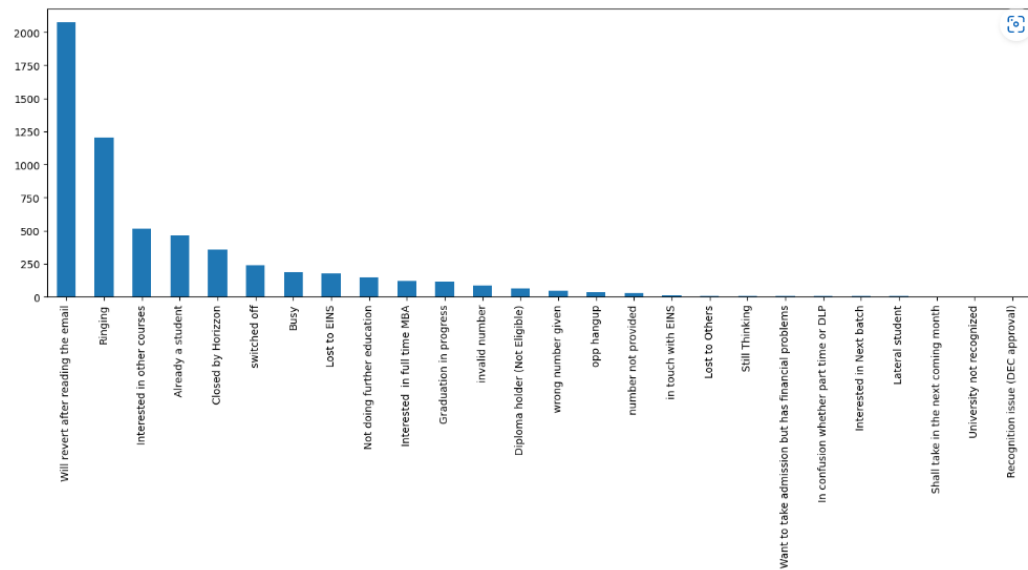
- In data cleaning the first thing we do is we replace all the 'Select' values with NAN values. T
 - Then we check for null value percentage in columns and drop the columns having more than 40% missing values.
 - For the columns having less than 40% missing values we imputed the columns with appropriate values.
 - We also drop rows accordingly.
- 



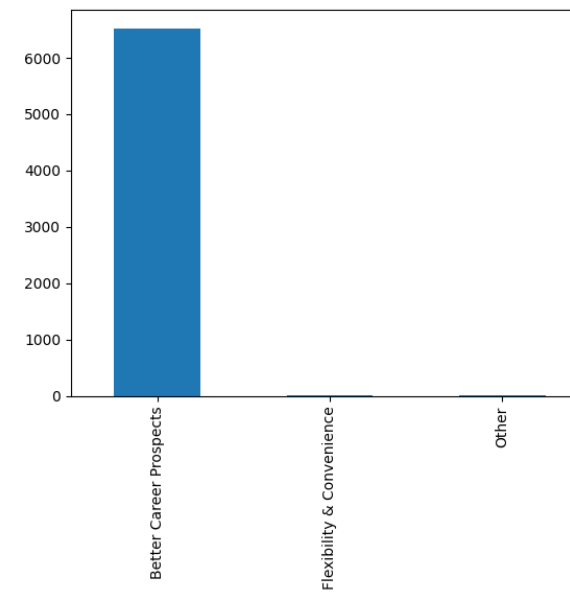
```
In [ ]: mpl.figure(figsize=(17,5))
leads['Specialization'].value_counts().plot.bar()
mpl.xticks(rotation=90)
mpl.show()
```



```
In [ ]: # Visualizing Tags column
mpl.figure(figsize=(17,5))
leads['Tags'].value_counts().plot.bar()
mpl.xticks(rotation=90)
mpl.show()
```

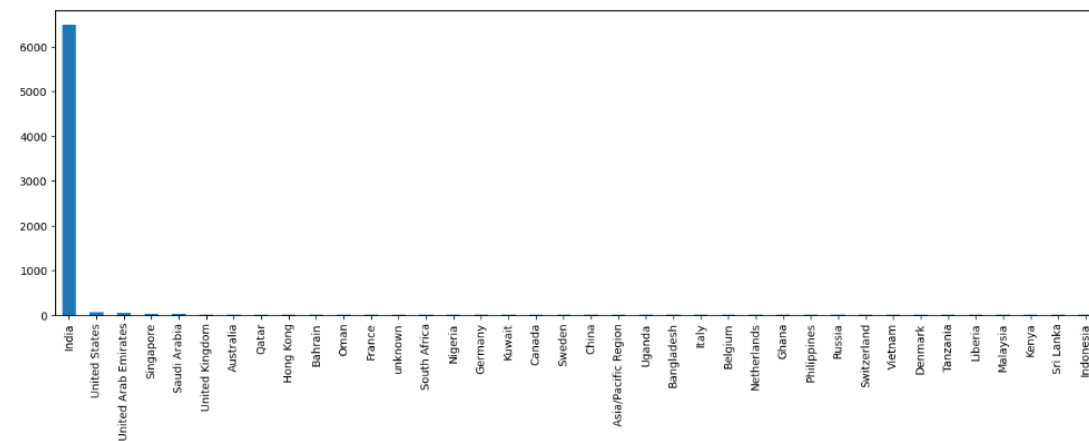
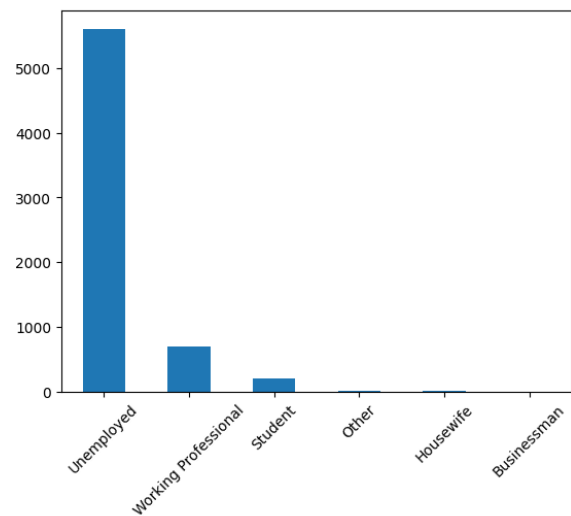


```
In [ ]: # Visualizing this column
leads['What matters most to you in choosing a course'].value_counts().plot.bar()
mpl.xticks(rotation=90)
mpl.show()
```




```
In [ ]: leads['What is your current occupation'].value_counts().plot.bar()
mpl.xticks(rotation=45)
```


```
Out[432]: (array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Unemployed'),
  Text(1, 0, 'Working Professional'),
  Text(2, 0, 'Student'),
  Text(3, 0, 'Other'),
  Text(4, 0, 'Housewife'),
  Text(5, 0, 'Businessman')])
```




Exploratory Data Analysis

- In this section we perform Uni-variate and bi-variate analysis.
 - And by this process we infer important insights about various variables.
 - In EDA we also look for outliers.
- 


Data preparation

- In data preparation we firstly convert binary variables to 1 and 0.
 - After this we perform dummy encoding on categorical variables.
 - After creating Dummy variables we concat dummy variables with the actual data-set and drop the categorical columns on which we performed dummy encoding.
 - Now we split our data-set into X_{train} , X_{test} , y_{train} , y_{test} .
 - After splitting our data-set into train and test, we perform normalization scaling, that is MinMax scaling.
- 

Feature Selection

- 1.We build a model using SK_learn and perform RFE on that. RFE stands for Recursive Feature Elimination.
 - 2.After performing RFE we look for the columns that are selected by the RFE algorithm.
 - 3.We will be using these same columns for further model building purposes.
 - 4.And also we will perform manual feature elimination from this same set of columns.
 - Here we are done with the Feature Selection. Now we will head towards model building.
- 

Model building

- Firstly we create a model using statsmodels.
 - After creating this model we look for it's detailed summary using `.summary()`.
 - After going through summary we will find VIF.
 - VIF stands for Variance Inflation Factor.
 - Now according to the values of P-value, which is present in statsmodels summary, and VIF we will perform manual feature elimination.
 - In manual feature elimination we look for these stats and then accordingly remove the columns, then we again build the model and then again remove the columns, and this process goes on until we get desired result.
- 

Our Final Model

Model 7

```
In [ ]: X_train_sm = sm.add_constant(X_train[col1])
logm5 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm5.fit()
res.summary()
```

Out[542]: 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:	-2656.1
Date:	Sat, 15 Apr 2023	Deviance:	5312.2
Time:	16:10:38	Pearson chi2:	6.63e+03
No. Iterations:	7	Pseudo R-sq. (CS):	0.3883
Covariance Type:	nonrobust		

	coef	std err	z	P> z	[0.025	0.975]
const	-0.3597	0.157	-2.297	0.022	-0.667	-0.053
Do Not Email	-1.1288	0.170	-6.659	0.000	-1.461	-0.797
TotalVisits	0.1850	0.044	4.219	0.000	0.099	0.271
Total Time Spent on Website	1.0345	0.039	26.356	0.000	0.956	1.111
A free copy of Mastering The Interview	-0.2200	0.086	-2.547	0.011	-0.389	-0.051
Lead Origin_Landing Page Submission	-1.0785	0.130	-8.315	0.000	-1.333	-0.824
Lead Origin_Lead Add Form	6.0065	0.731	8.222	0.000	4.575	7.438
Lead Source_Olark Chat	1.1824	0.134	8.856	0.000	0.921	1.444
Lead Source_Reference	-2.6656	0.756	-3.524	0.000	-4.148	-1.183
Last Activity_Email Opened	0.2078	0.105	1.981	0.048	0.002	0.413
Last Activity_Olark Chat Conversation	-1.2647	0.182	-6.935	0.000	-1.622	-0.907
Specialization_Others	-1.1526	0.126	-9.161	0.000	-1.399	-0.906
What is your current occupation_Working Professional	2.7095	0.206	13.182	0.000	2.307	3.112
Last Notable Activity_Modified	-0.2423	0.101	-2.393	0.017	-0.441	-0.044
Last Notable Activity_SMS Sent	1.5949	0.121	13.210	0.000	1.358	1.831

Checking for VIF values:


```
In [ ]: # Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train[col1].columns
vif['VIF'] = [variance_inflation_factor(X_train[col1].values, i) for i in range(X_train[col1].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[543]:

	Features	VIF
4	Lead Origin_Landing Page Submission	4.49
5	Lead Origin_Lead Add Form	4.39
7	Lead Source_Reference	4.13
10	Specialization_Others	2.96
8	Last Activity_Email Opened	2.53
6	Lead Source_Olark Chat	2.52
13	Last Notable Activity_SMS Sent	2.32
12	Last Notable Activity_Modified	2.30
3	A free copy of Mastering The Interview	2.18
9	Last Activity_Olark Chat Conversation	1.70
1	TotalVisits	1.67
2	Total Time Spent on Website	1.32
11	What is your current occupation_Working Profes...	1.21
0	Do Not Email	1.20

Predictions on train data-set

- We compare the actual dependent variable with the predicted dependent variable.
 - We have various metrics on which we judge our model on, such as accuracy, sensitivity-specificity, precision-recall.
 - According to these metrics we check efficiency of our model.
 - Also according to these metrics we find the Optimal cut-off
 - And according to this optimal cutoff, we make the decision whether to consider it as 1 or 0.
- 

Prediction on Train data-set

Metrics beyond simply accuracy

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

```
In [ ]: # Sensitivity of our Logistic regression model  
        print("Sensitivity : ",TP / float(TP+FN))  
  
        Sensitivity : 0.6902985074626866
```


```
In [ ]: # Let us calculate specificity  
        print("Specificity : ",TN / float(TN+FP))  
  
        Specificity : 0.883980705762884
```

```
In [ ]: # Calculate false positive rate - predicting converted Lead when the Lead actually was not converted  
        print("False Positive Rate :",FP/ float(TN+FP))  
  
        False Positive Rate : 0.11601929423711602
```

```
In [ ]: # positive predictive value  
        print("Positive Predictive Value :",TP / float(TP+FP))  
  
        Positive Predictive Value : 0.7846371347785108
```

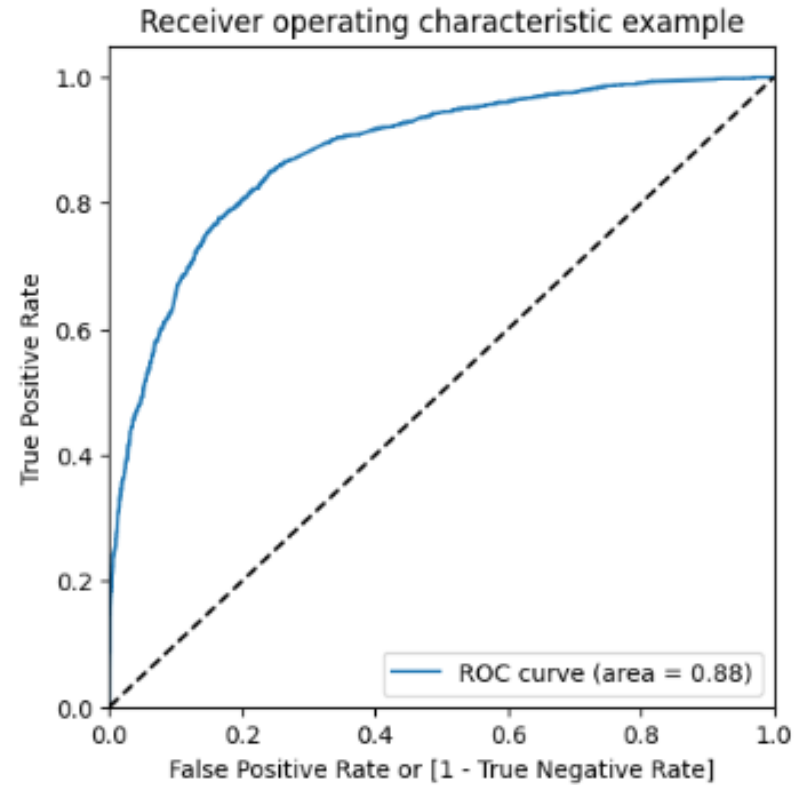
```
In [ ]: # Negative predictive value  
        print ("Negative predictive value :",TN / float(TN+ FN))  
  
        Negative predictive value : 0.8233624970442185
```

Model evaluation on Test data-set

- Firstly we predict the dependent variable values.
 - Then we compare the actual y_{test} values and the predicted y_{test} values.
 - And according to this comparison we judge how good our model is.
 - We use the same metrics that we used to judge our model on train data-set on our test data-set as well.
 - We use accuracy, confusion matrix, sensitivity-specificity, precision-recall, etc to test our model.
- 

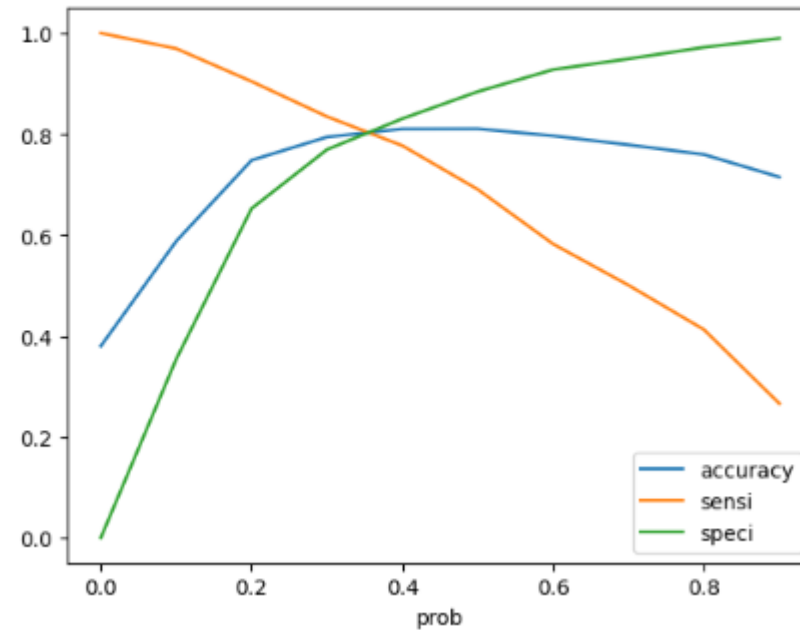
ROC Curve

```
In [ ]: draw_roc(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
```



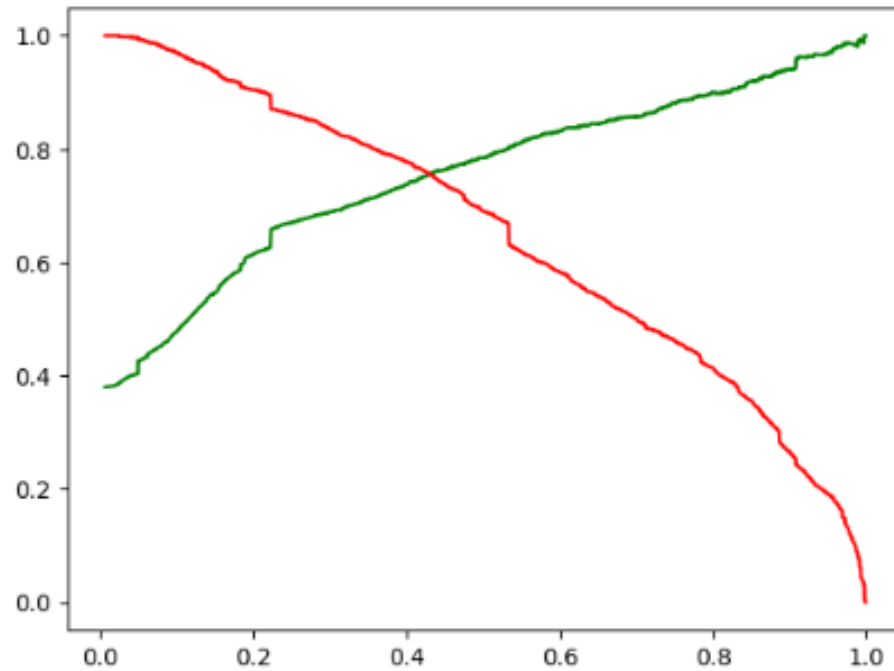
Accuracy-Sensitivity-Specificity

```
In [ ]: # Let's plot accuracy sensitivity and specificity for various probabilities.  
cutoff_df.plot.line(x='prob', y=['accuracy', 'sensi', 'speci'])  
mpl.show()
```



Precision-Recall Tradeoff

```
In [ ]: # plotting a trade-off curve between precision and recall  
mpl.plot(thresholds, p[:-1], "g-")  
mpl.plot(thresholds, r[:-1], "r-")  
mpl.show()
```



Final Observation

Observations:

After running the model on the Test Data , we obtain:

- Accuracy : 81.8 %
- Sensitivity : 81.3 %
- Specificity : 82 %

Results :

1) Comparing the values obtained for Train & Test:

Train Data:

- Accuracy : 80.2 %
- Sensitivity : 80.5 %
- Specificity : 80 %

Test Data:

- Accuracy : 81.8 %
- Sensitivity : 81.3 %
- Specificity : 82 %

Thus we have achieved our goal of getting a ballpark of the target lead conversion rate to be around 80% . The Model seems to predict the Conversion Rate very well and we should be able to give the CEO confidence in making good calls based on this model to get a higher lead conversion rate of 80%.

The End