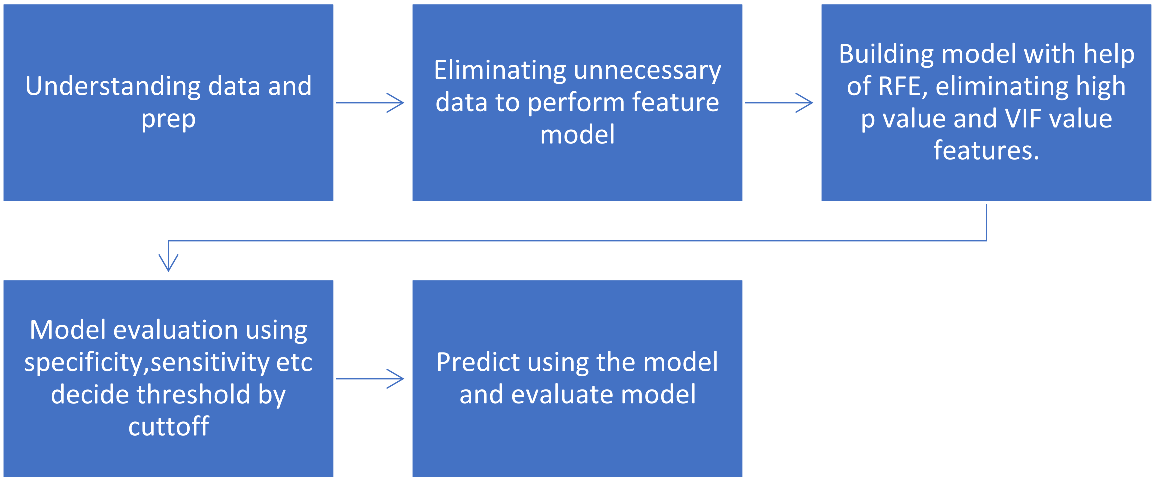
LEAD SCORING CASE STUDY- PPT

TO BUILD A LOGISTIC REGRESSION MODEL TO PREDICT IF A LEAD FOR ONLINE COURSES FOR AN EDUCATION COMPANY AKA X EDUCATION WOULD SUCCESSFULLY BE CONVERTED OR NOT.

BUSINESS OBJECTIVE

- To help X Education to select Hot Leads, i.e. the leads that are most likely to convert into paying customers.
- To build a logistic regression model to assign a lead score value between 0 and 100 to each of the leads which can be used by the company to target potential customers.

PROBLEM SOLVING METHODOLOGY



Understanding data and prep

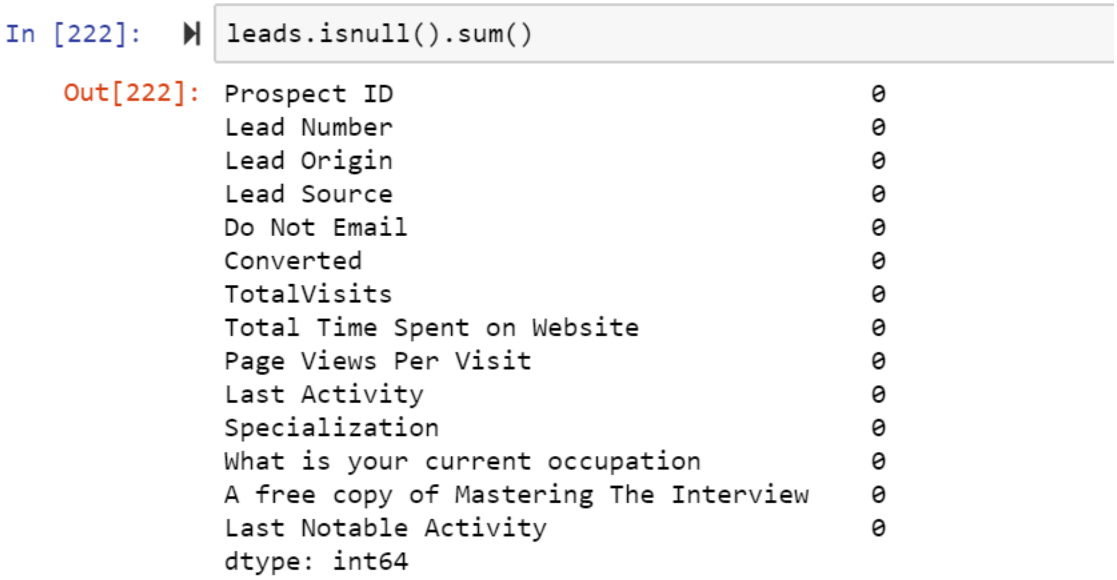
Eliminating unnecessary data to perform feature model

Building model with help of RFE, eliminating high p value and VIF value features.

Model evaluation using specificity,sensitivity etc decide threshold by cuttoff

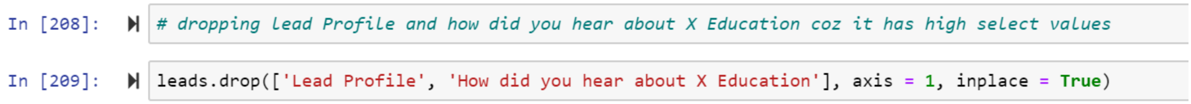
Predict using the model and evaluate model

1. Removing null values



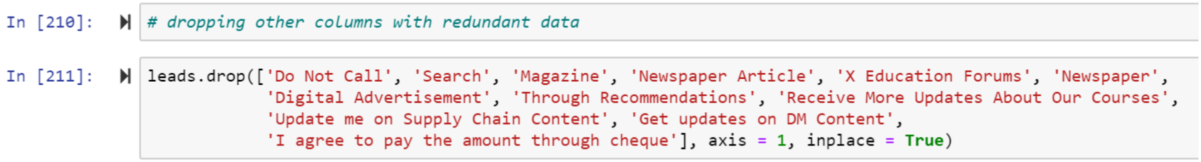
```
In [222]:  leads.isnull().sum()

Out[222]:  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
```

2. Removing columns with high missing values



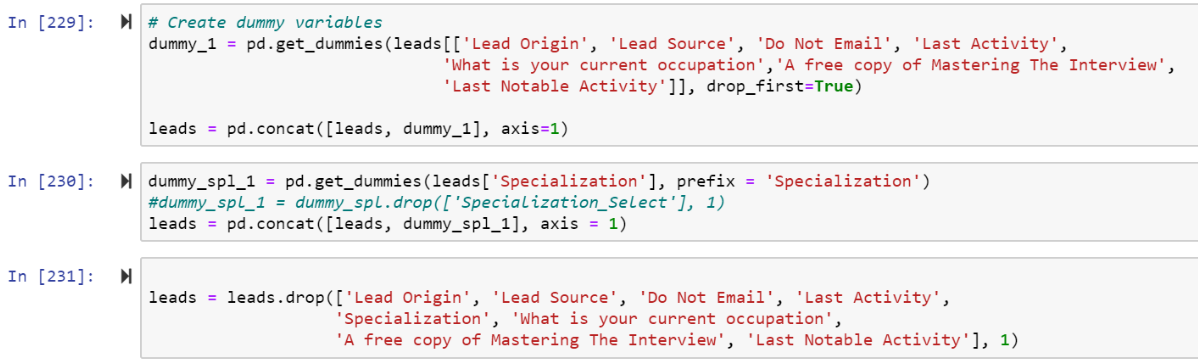
```
In [208]:  # dropping Lead Profile and how did you hear about X Education coz it has high select values

In [209]:  leads.drop(['Lead Profile', 'How did you hear about X Education'], axis = 1, inplace = True)
```

3. Removing redundant data



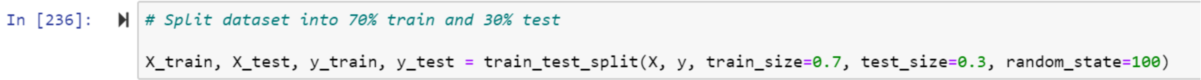
```
In [210]:  # dropping other columns with redundant data

In [211]:  leads.drop(['Do Not Call', 'Search', 'Magazine', 'Newspaper Article', 'X Education Forums', 'Newspaper',
                       'Digital Advertisement', 'Through Recommendations', 'Receive More Updates About Our Courses',
                       'Update me on Supply Chain Content', 'Get updates on DM Content',
                       'I agree to pay the amount through cheque'], axis = 1, inplace = True)
```

4. Dummy Encoding



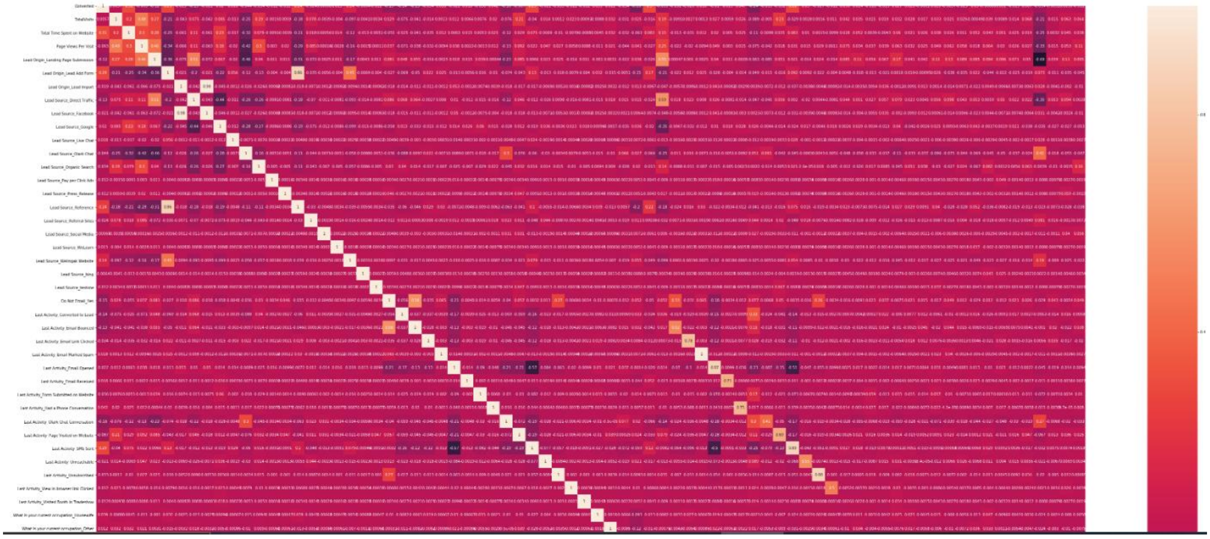
```
In [229]:  # Create dummy variables
           dummy_1 = pd.get_dummies(leads[['Lead Origin', 'Lead Source', 'Do Not Email', 'Last Activity',
                                           'What is your current occupation','A free copy of Mastering The Interview',
                                           'Last Notable Activity']], drop_first=True)

           leads = pd.concat([leads, dummy_1], axis=1)

In [230]:  dummy_spl_1 = pd.get_dummies(leads['Specialization'], prefix = 'Specialization')
           #dummy_spl_1 = dummy_spl.drop(['Specialization_Select'], 1)
           leads = pd.concat([leads, dummy_spl_1], axis = 1)

In [231]:
           leads = leads.drop(['Lead Origin', 'Lead Source', 'Do Not Email', 'Last Activity',
                               'Specialization', 'What is your current occupation',
                               'A free copy of Mastering The Interview', 'Last Notable Activity'], 1)
```

5. Test Train Split

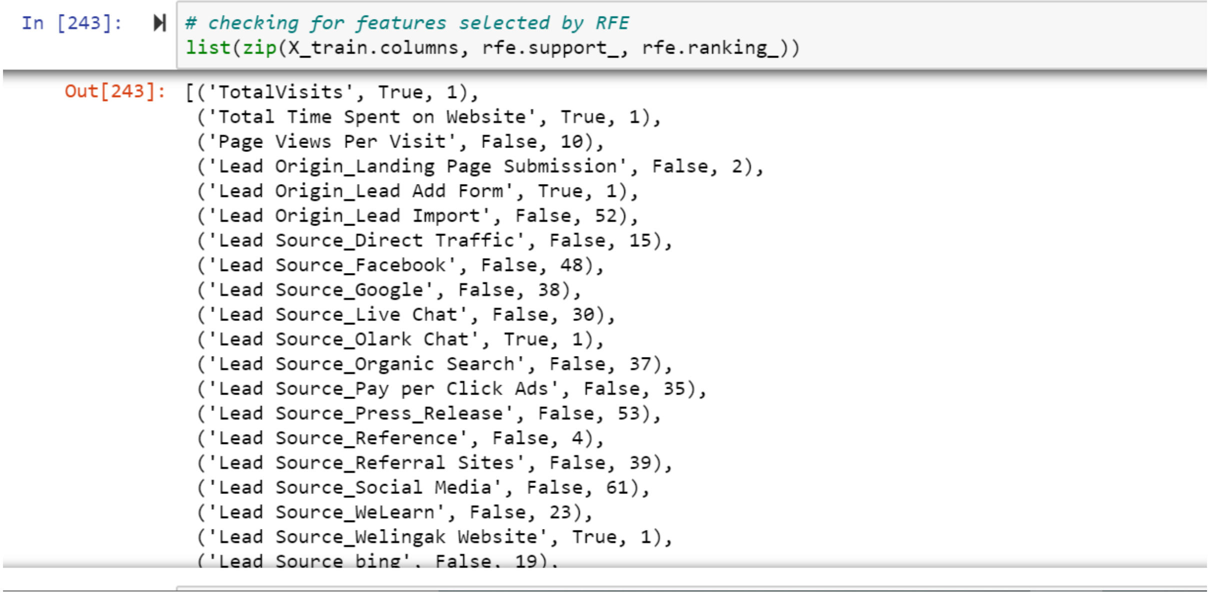


```
In [236]:  # Split dataset into 70% train and 30% test

           X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

6. Checking for correlation



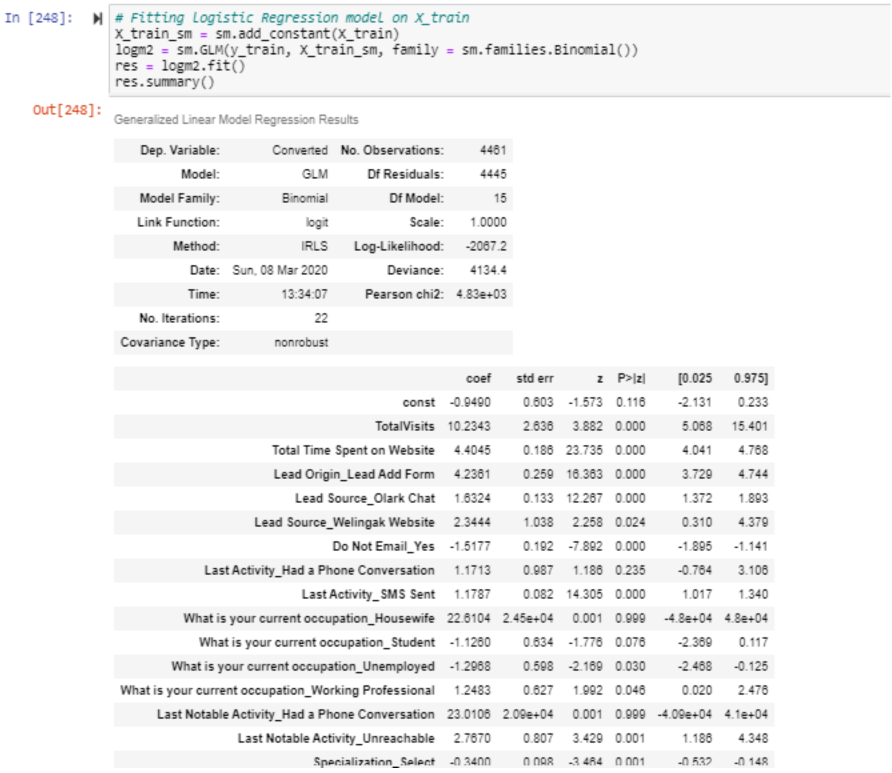
7. Feature Selection



```
In [243]:  # checking for features selected by RFE
           list(zip(X_train.columns, rfe.support_, rfe.ranking_))

Out[243]:  [('TotalVisits', True, 1),
            ('Total Time Spent on Website', True, 1),
            ('Page Views Per Visit', False, 10),
            ('Lead Origin_Landing Page Submission', False, 2),
            ('Lead Origin_Lead Add Form', True, 1),
            ('Lead Origin_Lead Import', False, 52),
            ('Lead Source_Direct Traffic', False, 15),
            ('Lead Source_Facebook', False, 48),
            ('Lead Source_Google', False, 38),
            ('Lead Source_Live Chat', False, 30),
            ('Lead Source_Olark Chat', True, 1),
            ('Lead Source_Organic Search', False, 37),
            ('Lead Source_Pay per Click Ads', False, 35),
            ('Lead Source_Press_Release', False, 53),
            ('Lead Source_Reference', False, 4),
            ('Lead Source_Referral Sites', False, 39),
            ('Lead Source_Social Media', False, 61),
            ('Lead Source_WeLearn', False, 23),
            ('Lead Source_Welingak Website', True, 1),
            ('Lead Source_bing', False, 19),
```

8. Model Building

```
In [245]:  # creating a logistic regression model
```



In [248]:

```
# Fitting Logistic Regression model on X_train
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()
```

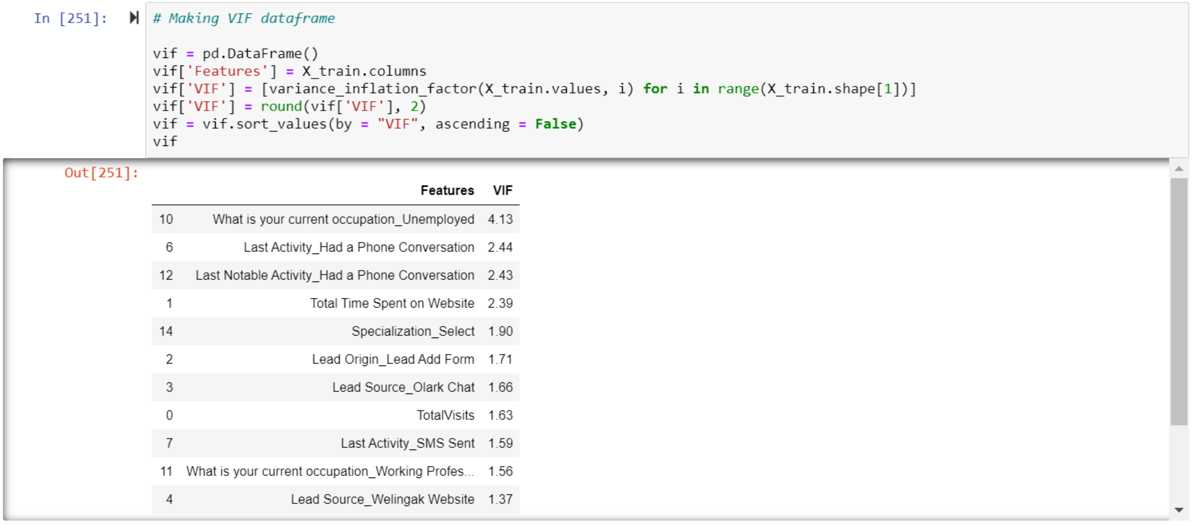
Out[248]:

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:	Sun, 08 Mar 2020	Deviance:	4134.4
Time:	13:34:07	Pearson chi2:	4.83e+03
No. Iterations:	22		
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

10. Checking VIF values



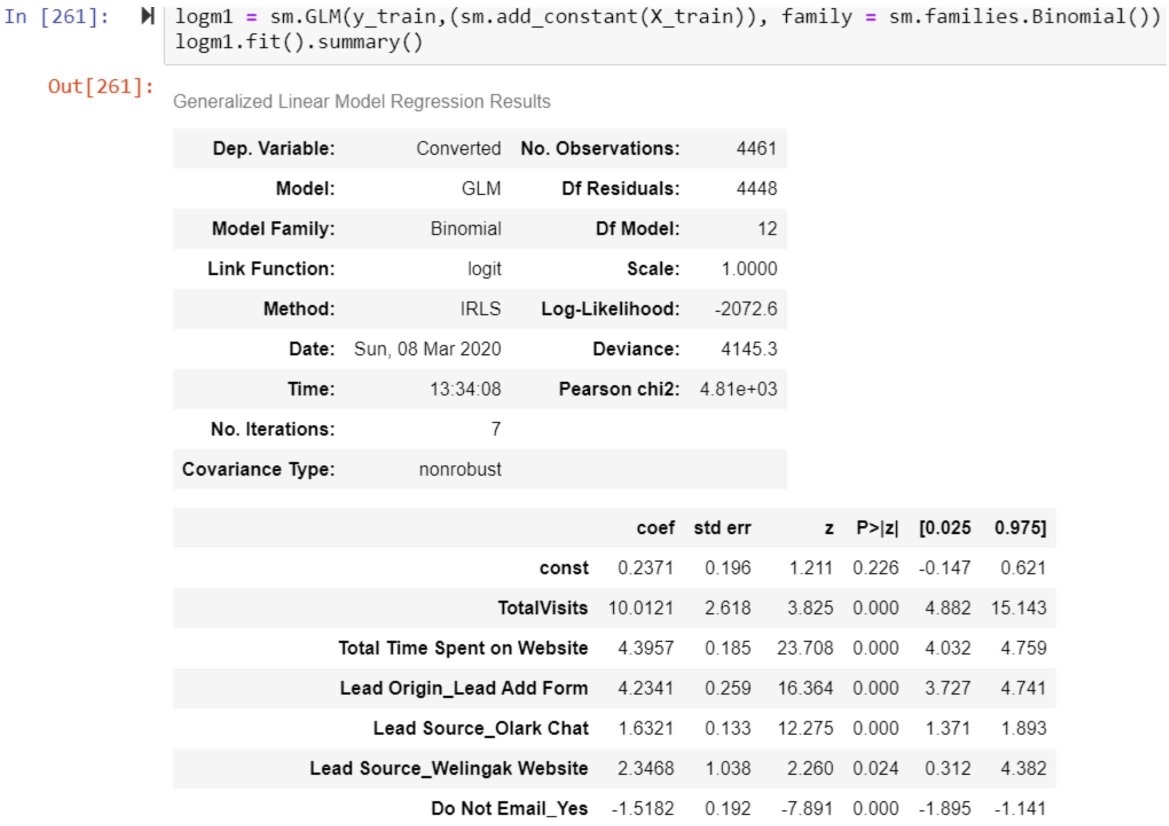
In [251]:

```
# Making VIF dataframe

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
```

Out[251]:

	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

11. Fitting p values and vif values



```
In [261]:  logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Binomial())
           logm1.fit().summary()
```

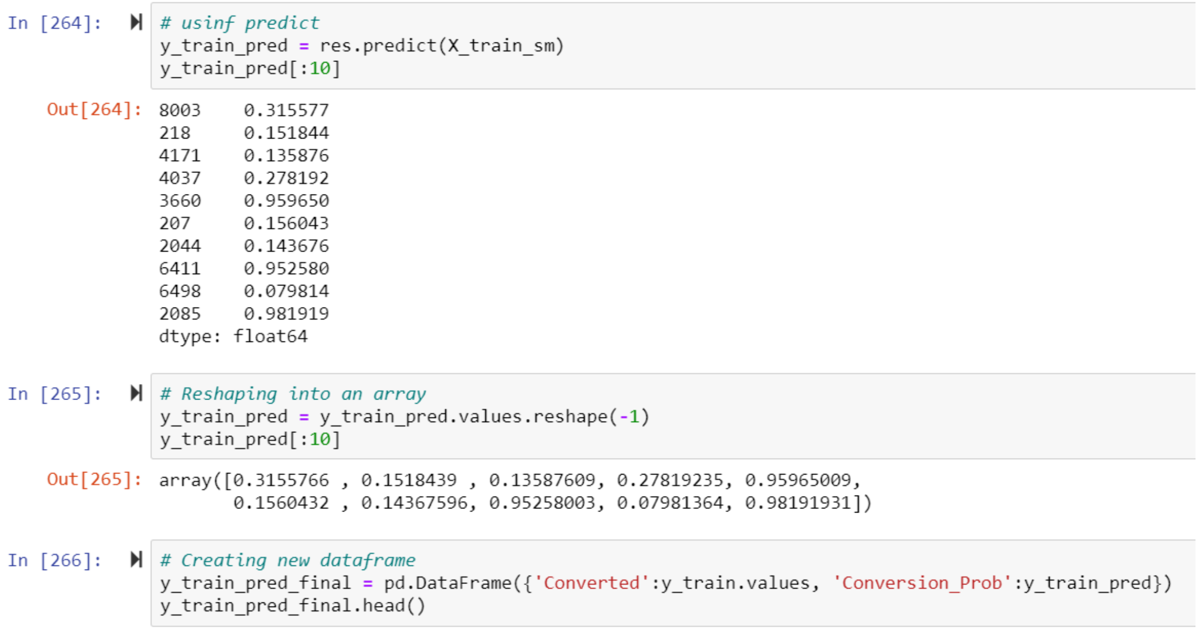
Out[261]:

Generalized Linear Model Regression Results

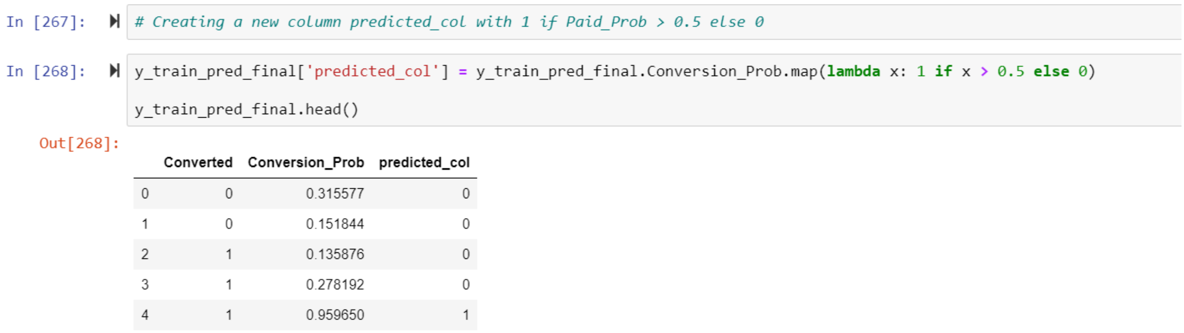
Dep. Variable:	Converted	**No. Observations:**	4461
Model:	GLM	**Df Residuals:**	4448
Model Family:	Binomial	**Df Model:**	12
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2072.6
Date:	Sun, 08 Mar 2020	**Deviance:**	4145.3
Time:	13:34:08	**Pearson chi2:**	4.81e+03
No. Iterations:	7		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2371	0.196	1.211	0.226	-0.147	0.621
TotalVisits	10.0121	2.618	3.825	0.000	4.882	15.143
Total Time Spent on Website	4.3957	0.185	23.708	0.000	4.032	4.759
Lead Origin_Lead Add Form	4.2341	0.259	16.364	0.000	3.727	4.741
Lead Source_Olark Chat	1.6321	0.133	12.275	0.000	1.371	1.893
Lead Source_Welingak Website	2.3468	1.038	2.260	0.024	0.312	4.382
Do Not Email_Yes	-1.5182	0.192	-7.891	0.000	-1.895	-1.141

12. Model Evaluation- predicting the data



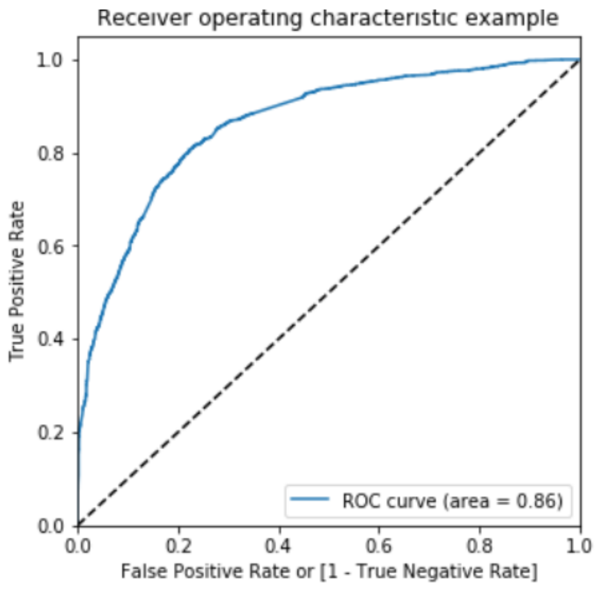
```
In [264]:  # usinf predict
           y_train_pred = res.predict(X_train_sm)
           y_train_pred[:10]

Out[264]:  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

In [265]:  # Reshaping into an array
           y_train_pred = y_train_pred.values.reshape(-1)
           y_train_pred[:10]

Out[265]:  array([0.3155766 , 0.1518439 , 0.13587609, 0.27819235, 0.95965009,
                  0.1560432 , 0.14367596, 0.95258003, 0.07981364, 0.98191931])

In [266]:  # Creating new dataframe
           y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Conversion_Prob':y_train_pred})
           y_train_pred_final.head()
```



```
In [267]:  # Creating a new column predicted_col with 1 if Paid_Prob > 0.5 else 0

In [268]:  y_train_pred_final['predicted_col'] = y_train_pred_final.Conversion_Prob.map(lambda x: 1 if x > 0.5 else 0)

           y_train_pred_final.head()

Out[268]:
```

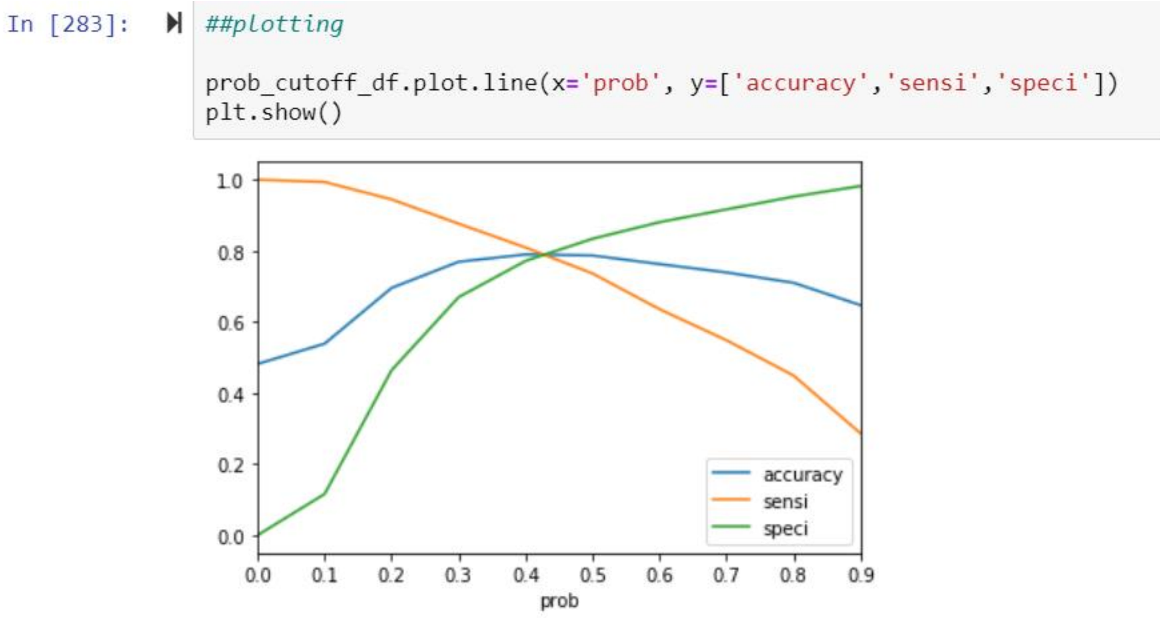
	Converted	Conversion_Prob	predicted_col
0	0	0.315577	0
1	0	0.151844	0
2	1	0.135876	0
3	1	0.278192	0
4	1	0.959650	1

13. Finding optimal cutoff



#here area under the curve of the ROC is 0.88 which shows a good model

14. Optimal probability plot

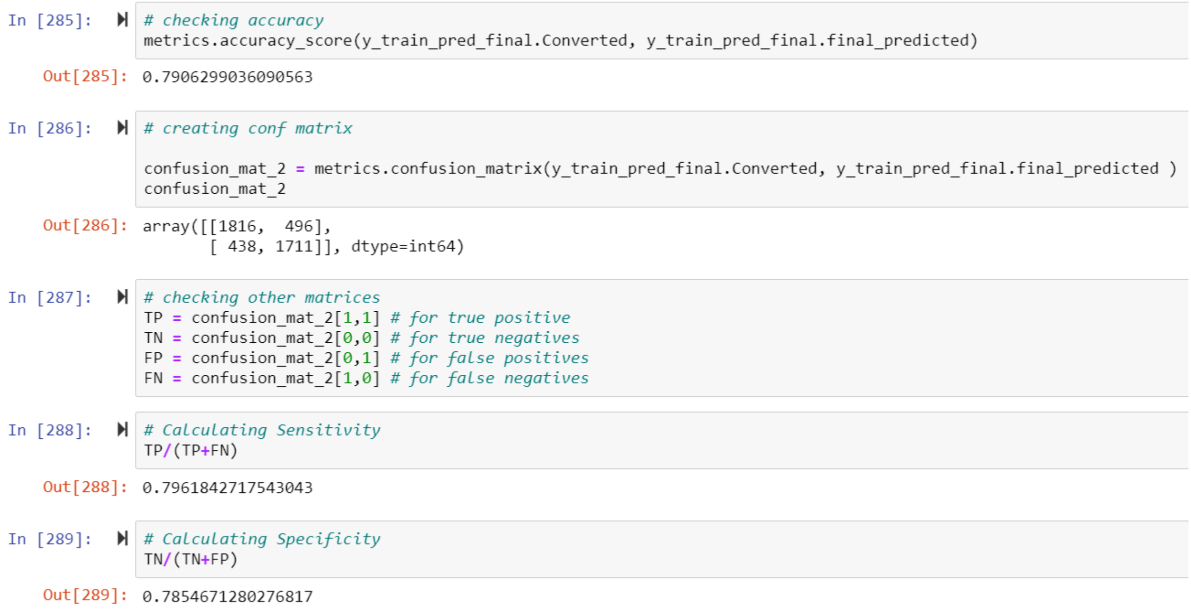


```
In [283]:  ##plotting

           prob_cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
           plt.show()
```

Here when probability thresholds are very low, sensitivity value is very high and specificity is very low. Likewise, for higher probability thresholds, sensitivity values are very low but the specificity values are very high.

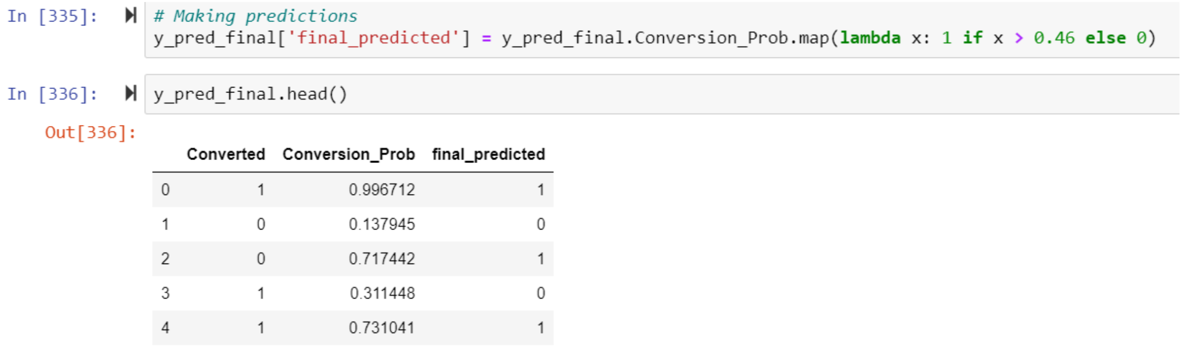
High sensitivity indicates that the model will accurately identify nearly all leads that are likely to convert.

15. Checking accuracy and confusion matrix



```
In [285]:  # checking accuracy
           metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.final_predicted)

Out[285]:  0.7906299036090563

In [286]:  # creating conf matrix

           confusion_mat_2 = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.final_predicted )
           confusion_mat_2

Out[286]:  array([[1816,  496],
                  [ 438, 1711]], dtype=int64)

In [287]:  # checking other matrices
           TP = confusion_mat_2[1,1] # for true positive
           TN = confusion_mat_2[0,0] # for true negatives
           FP = confusion_mat_2[0,1] # for false positives
           FN = confusion_mat_2[1,0] # for false negatives

In [288]:  # Calculating Sensitivity
           TP/(TP+FN)

Out[288]:  0.7961842717543043

In [289]:  # Calculating Specificity
           TN/(TN+FP)

Out[289]:  0.7854671280276817
```

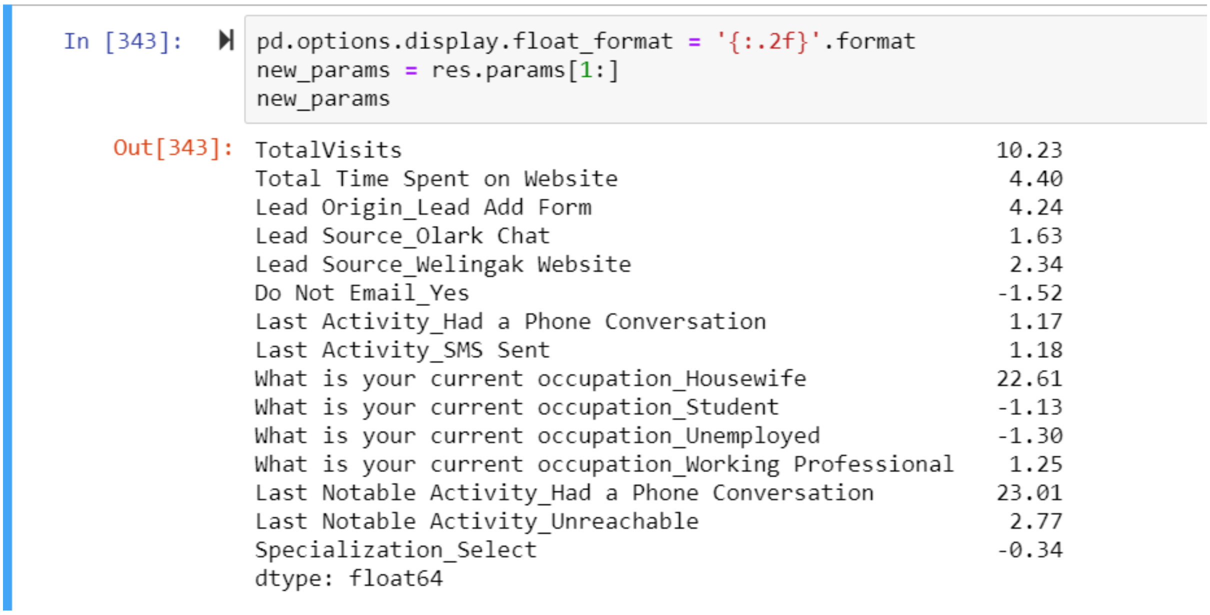
16. Making predictions on test set



```
In [335]:  # Making predictions
           y_pred_final['final_predicted'] = y_pred_final.Conversion_Prob.map(lambda x: 1 if x > 0.46 else 0)

In [336]:  y_pred_final.head()

Out[336]:
```

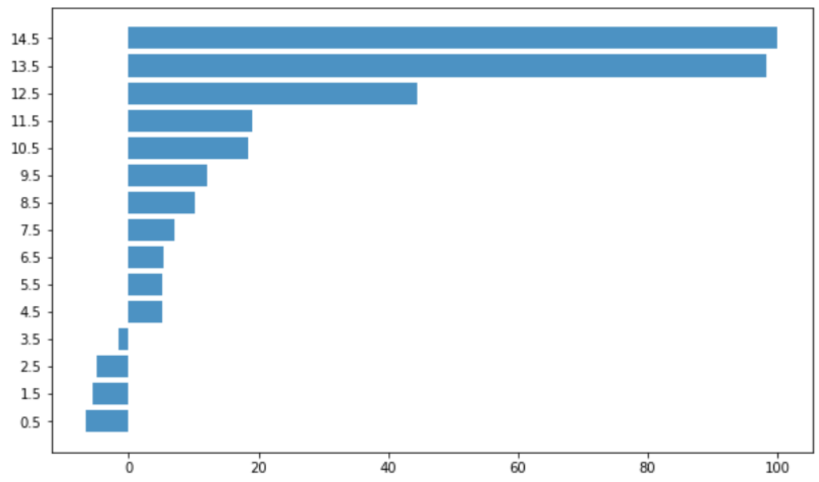
	Converted	Conversion_Prob	final_predicted
0	1	0.996712	1
1	0	0.137945	0
2	0	0.717442	1
3	1	0.311448	0
4	1	0.731041	1

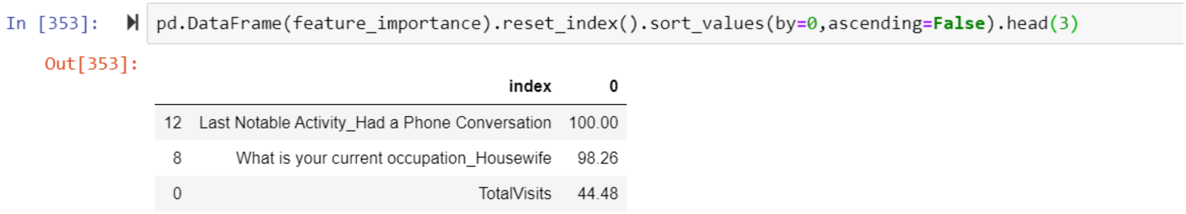
17. Checking for features



```
In [343]:  pd.options.display.float_format = '{:.2f}'.format
           new_params = res.params[1:]
           new_params

Out[343]:  TotalVisits                                           10.23
           Total Time Spent on Website                            4.40
           Lead Origin_Lead Add Form                              4.24
           Lead Source_Olark Chat                                 1.63
           Lead Source_Welingak Website                           2.34
           Do Not Email_Yes                                      -1.52
           Last Activity_Had a Phone Conversation                 1.17
           Last Activity_SMS Sent                                 1.18
           What is your current occupation_Housewife             22.61
           What is your current occupation_Student               -1.13
           What is your current occupation_Unemployed            -1.30
           What is your current occupation_Working Professional   1.25
           Last Notable Activity_Had a Phone Conversation        23.01
           Last Notable Activity_Unreachable                      2.77
           Specialization_Select                                 -0.34
           dtype: float64
```

18. selection of features





```
In [353]:  pd.DataFrame(feature_importance).reset_index().sort_values(by=0,ascending=False).head(3)

Out[353]:
```

	index	0
12	Last Notable Activity_Had a Phone Conversation	100.00
8	What is your current occupation_Housewife	98.26
0	TotalVisits	44.48