The **Most Important Results** of this Logistic Regression Case Study are:

- Target Variable vs Features Heat Map.
- P_values of features selected to be less than 0.05
- VIFs of selected features to be less than 5.
- Train data metrics of accuracy , sensitivity and specificity
- ROC AUC Curve details
- Optimal Cut off graph
- Lead Scores generated on Train and test data sets.
- Test data metrics of accuracy , sensitivity and specificity.

All the above metrics are portrayed by given screenshots sequentially below:

1. **Target Variable vs Features Heat Map.**

In [108]:
```
# Let's Visualize the correlation matrix with respected to TARGET variable 'Converted' to get business sense

# data = np.asarray(leads.corrwith(leads["Converted"])).reshape(len(leads.corrwith(leads["Converted"])),1)
data=leads.corr().loc[:,['Converted']]
plt.figure(figsize = (4,20))        # Size of the figure
sns.heatmap(data,cmap='RdYlGn',annot = True)
plt.show()
```

2. **P_values of features selected to be less than 0.05**

jupyter Achal_Surbhi_LOR_Lead_Scoring_v2 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [185]:
```
# Let's re-run the model using the selected variables
X_train_sm = sm.add_constant(X_train[col])
logm4 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
lor_model = logm4.fit()
lor_model.summary()
```

Out[185]:

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	2890
Model:	GLM	**Df Residuals:**	2882
Model Family:	Binomial	**Df Model:**	7
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1401.8
Date:	Mon, 17 May 2021	**Deviance:**	2803.6
Time:	20:24:44	**Pearson chi2:**	3.65e+03
No. Iterations:	6		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.5127	0.209	7.244	0.000	1.103	1.922
Do Not Email	-1.7219	0.243	-7.078	0.000	-2.199	-1.245
Total Time Spent on Website	1.0928	0.052	21.019	0.000	0.991	1.195
Dmy Lead Origin_Lead Add Form	2.2188	0.400	5.543	0.000	1.434	3.003
Dmy Lead Source_Direct Traffic	-2.1043	0.220	-9.547	0.000	-2.536	-1.672
Dmy Lead Source_Google	-1.7215	0.223	-7.726	0.000	-2.158	-1.285
Dmy Lead Source_Organic Search	-1.8426	0.245	-7.525	0.000	-2.323	-1.363
Dmy What is your current occupation_Working Professional	2.8259	0.246	11.474	0.000	2.343	3.309

3. VIFs of selected features to be less than 5.

jupyter Achal_Surbhi_LOR_Lead_Scoring_v2 Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [189]:
```
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_train[col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[189]:

	Features	VIF
6	Dmy What is your current occupation_Working Pr...	1.14
0	Do Not Email	1.11
3	Dmy Lead Source_Direct Traffic	1.11
4	Dmy Lead Source_Google	1.07
1	Total Time Spent on Website	1.06
2	Dmy Lead Origin_Lead Add Form	1.04
5	Dmy Lead Source_Organic Search	1.04

FEATURE SELECTION INFERENCE:

- We will work with this model obtained as all constraints on vif and p_value are satisfied
- The top three variables which have positive coefficient and make the lead a hot lead are:
  - Dmy What is your current occupation_Working Professional
  - Dmy Lead Origin_Lead Add Form
  - Total Time Spent on Website
- The top three variables which have negative coefficient whose increase in value reduces the probability of making the lead hot are:
  - Dmy Lead Source_Direct Traffic
  - Dmy Lead Source_Organic Search
  - Do Not Email

4. Train data metrics of accuracy , sensitivity and specificity

```
TP: 1075
TN: 1196
FP: 288
FN: 331
```

```
In [193]: # Let's see the sensitivity of our logistic regression model
          TP / float(TP+FN)

Out[193]: 0.7645803698435277
```

```
In [194]: # Let us calculate specificity
          TN / float(TN+FP)

Out[194]: 0.8059299191374663
```

```
In [195]: # Calculate false postive rate - predicting converted when customer has not converted
          print(FP/ float(TN+FP))

          0.1940700808625337
```

```
In [196]: # positive predictive value : This is also called as 'Precision' as you will see this in future
          print (TP / float(TP+FP))

          0.7887013939838592
```

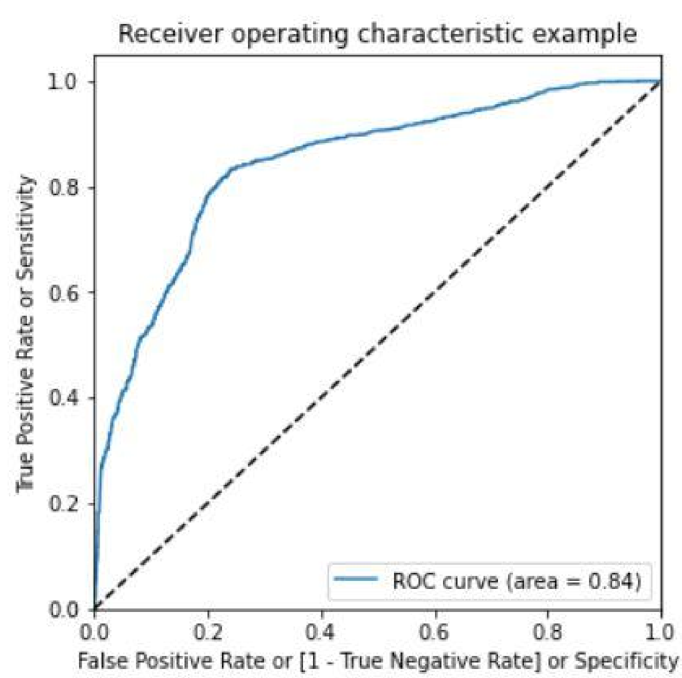
```
In [197]: # Negative predictive value
          print (TN / float(TN+ FN))

          0.7832351015062213
```

5. ROC AUC Curve details

```
In [199]: fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.L

In [200]: draw_roc(y_train_pred_final.Lead, y_train_pred_final.Lead_Prob
```



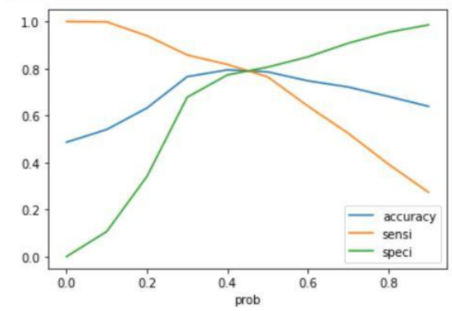
6. Optimal Cut off graph

The **key takeaway** from this code is the accuracy, sensitivity, and specificity values which have been calculated using the appropriate elements in the confusion matrix. The code outputted the following above dataframe:

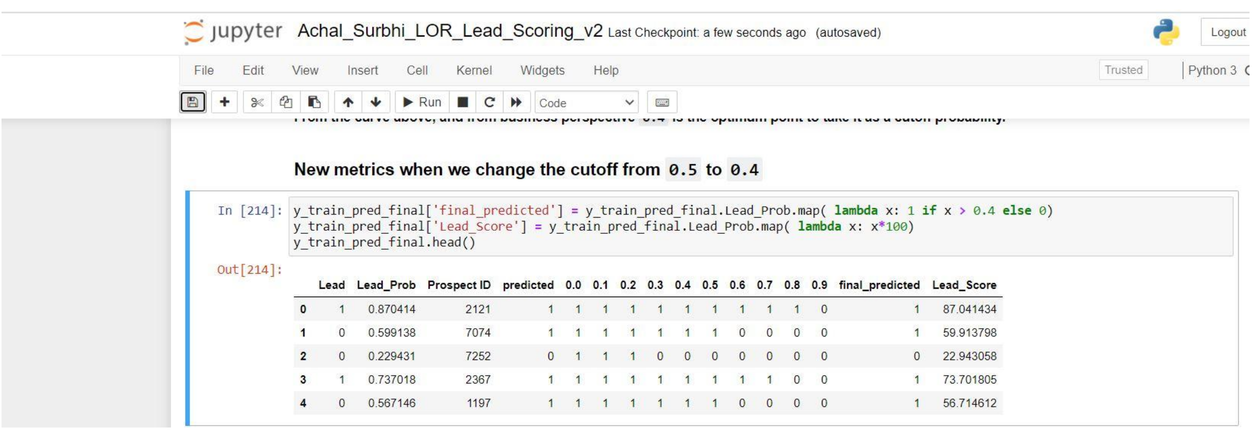
As you can see, when the probability thresholds are very low, the sensitivity is very high and specificity is very low. Similarly, for larger probability thresholds, the sensitivity values are very low but the specificity values are very high. And at about `0.4` , the three metrics seem to be almost equal with decent values.

Also since from business perspective working towards increasing sensitivity and decreasing `False Negatives` (That is Hot Leads who are wrongly predicted as cold leads is our objective and hence, we choose `0.4` as the optimal cut-off point. The following graph also showcases that at about `0.4` , the three metrics intersect.

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



7. Lead Scores generated on Train and test data sets.


jupyter Achal_Surbhi_LOR_Lead_Scoring_v2 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

New metrics when we change the cutoff from `0.5` to `0.4`

```
In [214]: y_train_pred_final['final_predicted'] = y_train_pred_final.Lead_Prob.map( lambda x: 1 if x > 0.4 else 0)
y_train_pred_final['Lead_Score'] = y_train_pred_final.Lead_Prob.map( lambda x: x*100)
y_train_pred_final.head()
```

Out[214]:

	Lead	Lead_Prob	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	0.870414	2121	1	1	1	1	1	1	1	1	1	1	0	1	87.041434
1	0	0.599138	7074	1	1	1	1	1	1	1	0	0	0	0	1	59.913798
2	0	0.229431	7252	0	1	1	1	0	0	0	0	0	0	0	0	22.943058
3	1	0.737018	2367	1	1	1	1	1	1	1	1	1	0	0	1	73.701805
4	0	0.567146	1197	1	1	1	1	1	1	1	0	0	0	0	1	56.714612

8. Test data metrics of accuracy , sensitivity and specificity.

Let's check the overall accuracy of Test Data Set: 0.7917675544794189

In [227]:
```
metrics.accuracy_score(y_pred_final.Converted, y_pred_final.final_predicted)
```

Out[227]:
```
0.7917675544794189
```

In [228]:
```
confusion3 = metrics.confusion_matrix(y_pred_final.Converted, y_pred_final.final_predicted )
confusion3
```

Out[228]:
```
array([[474, 150],
       [108, 507]], dtype=int64)
```

In [229]:
```
TP = confusion3[1,1] # true positive
TN = confusion3[0,0] # true negatives
FP = confusion3[0,1] # false positives
FN = confusion3[1,0] # false negatives
```

Model evaluation on 'Test' Data with `0.4` as the cut off with "senstivity and specificty view":

Sensitivty of Test Set: 0.824390243902439

In [230]:
```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

Out[230]:
```
0.824390243902439
```

Specificity of Test Set: 0.7596153846153846

In [231]:
```
# Let us calculate specificity
TN / float(TN+FP)
```

Out[231]:
```
0.7596153846153846
```

This PPT is submitted by Achal Kagwad and Surbhi Chaplot