1. Which are the top three variables in your model which contribute most towards the probability of a lead getting converted?

Solution:

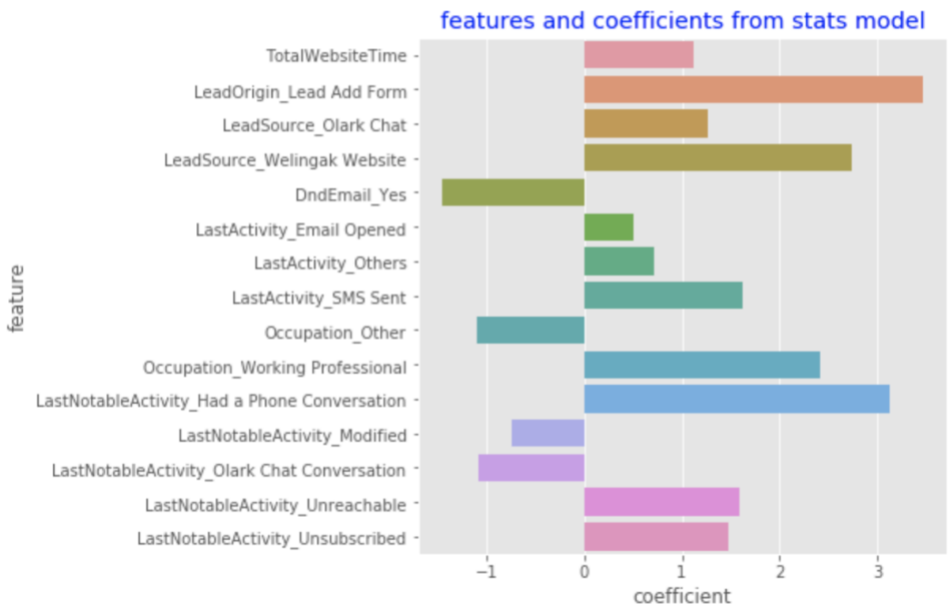
Lets plot features and respective coefficients from stats regression model

In [710]:
```
params_df = pd.DataFrame(columns=['coefficient'],data=lr.params).reset_index()
params_df.rename(columns={'index':'feature'},inplace=True)
params_df.drop(index=0,inplace=True)
params_df.sort_values(by='coefficient',ascending=False)
```

Out[710]:

	feature	coefficient
2	LeadOrigin_Lead Add Form	3.466767
11	LastNotableActivity_Had a Phone Conversation	3.123466
4	LeadSource_Welingak Website	2.733684
10	Occupation_Working Professional	2.406889
8	LastActivity_SMS Sent	1.623893
14	LastNotableActivity_Unreachable	1.593580
15	LastNotableActivity_Unsubscribed	1.470591
3	LeadSource_Olark Chat	1.273367
1	TotalWebsiteTime	1.126039
7	LastActivity_Others	0.721333
6	LastActivity_Email Opened	0.511292
12	LastNotableActivity_Modified	-0.742999
13	LastNotableActivity_Olark Chat Conversation	-1.078162
9	Occupation_Other	-1.092805
5	DndEmail_Yes	-1.452211

```
plt.figure(figsize=(6,6))
plt.title('features and coefficients from stats model',color='blue')
sns.barplot(y='feature',x='coefficient',data=params_df);
```



Top three predictors that contribute in lead conversion are Lead Origin, Last Notable Activity and Lead Source

2. What are the top 3 categorical/dummy variables in the model which should be focused the most on in order to increase the probability of lead conversion?

Solution:

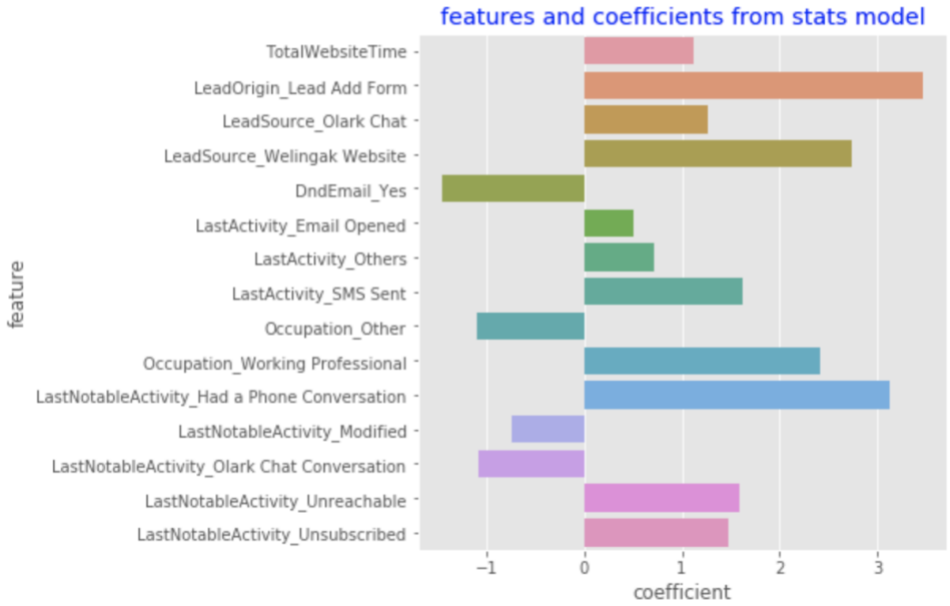
Lets plot features and respective coefficients from stats regression model

In [710]:
```
params_df = pd.DataFrame(columns=['coefficient'],data=lr.params).reset_index()
params_df.rename(columns={'index':'feature'},inplace=True)
params_df.drop(index=0,inplace=True)
params_df.sort_values(by='coefficient',ascending=False)
```

Out[710]:

	feature	coefficient
2	LeadOrigin_Lead Add Form	3.466767
11	LastNotableActivity_Had a Phone Conversation	3.123466
4	LeadSource_Welingak Website	2.733684
10	Occupation_Working Professional	2.406889
8	LastActivity_SMS Sent	1.623893
14	LastNotableActivity_Unreachable	1.593580
15	LastNotableActivity_Unsubscribed	1.470591
3	LeadSource_Olark Chat	1.273367
1	TotalWebsiteTime	1.126039
7	LastActivity_Others	0.721333
6	LastActivity_Email Opened	0.511292
12	LastNotableActivity_Modified	-0.742999
13	LastNotableActivity_Olark Chat Conversation	-1.078162
9	Occupation_Other	-1.092805
5	DndEmail_Yes	-1.452211

```
plt.figure(figsize=(6,6))
plt.title('features and coefficients from stats model',color='blue')
sns.barplot(y='feature',x='coefficient',data=params_df);
```



Top three categorical/dummy variables are LeadOrigin_Lead Add Form, LastNotableActivity_Had a Phone Conversation and LeadSource_Welingak Website

3. X Education has a period of 2 months every year during which they hire some interns. The sales team, in particular, has around 10 interns allotted to them. So during this phase, they wish to make the lead conversion more aggressive. So they want almost all of the potential leads (i.e. the customers who have been predicted as 1 by the model) to be converted and hence, want to make phone calls to as much of such people as possible. Suggest a good strategy they should employ at this stage.

Solution:

Since the company wants to focus on lead conversion aggressively, this means that company is focusing more on True Positive.

For this, we should choose the cutoff such that the sensitivity of the model should increase without compromising much on accuracy of model

From original model we have

```
# Now let's calculate accuracy sensitivity and specificity for various probability cutoffs.
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = confusion_matrix(y_train_pred_final['Converted'], y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

	prob	accuracy	sensi	speci
0.0	0.0	0.382794	1.000000	0.000000
0.1	0.1	0.637011	0.972514	0.428930
0.2	0.2	0.767291	0.921584	0.671597
0.3	0.3	0.803652	0.867017	0.764352
0.4	0.4	0.816958	0.774454	0.843319
0.5	0.5	0.820672	0.707357	0.890950
0.6	0.6	0.814173	0.644301	0.919529
0.7	0.7	0.788179	0.536378	0.944347
0.8	0.8	0.763423	0.436136	0.966408
0.9	0.9	0.715457	0.278092	0.986713

Observations from above plot

- for cutoff 0.3, we have good Sensitivity score without compromising much on accuracy score
- Therefore we will choose cutoff as 0.3

- With cut-off point as 0.3, we have test accuracy of 80%, sensitivity of 86.2% and specificity of 76%

```
result_metrics
```

	cutoff	train_acc	train_sen	train_spec	train_prec	test_acc	test_sen	test_spec	test_prec
0	0.36	0.81	0.81	0.82	0.73	0.81	0.8	0.82	0.74
1	0.35	0.81	0.84	0.79	0.72	0.81	0.83	0.79	0.72
2	0.37	0.81	0.79	0.83	0.74	0.81	0.79	0.83	0.75
3	0.3	0.8	0.87	0.76	0.7	0.8	0.86	0.76	0.7

4. Similarly, at times, the company reaches its target for a quarter before the deadline. During this time, the company wants the sales team to focus on some new work as well. So during this time, the company's aim is to not make phone calls unless it's extremely necessary, i.e. they want to minimize the rate of useless phone calls. Suggest a strategy they should employ at this stage.

Solution:

Since the company wants to to avoid unnecessary phone call, this means that company is focusing on less number of false positive rate.

Since FPR = 1 - Specificity, this means we must set a cutoff such that the Specificity is high from the model thereby resulting in less False Positive Rate

From original model we have

```
# Now let's calculate accuracy sensitivity and specificity for various probability cutoffs.
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = confusion_matrix(y_train_pred_final['Converted'], y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

```
     prob  accuracy     sensi     speci
0.0   0.0  0.382794  1.000000  0.000000
0.1   0.1  0.637011  0.972514  0.428930
0.2   0.2  0.767291  0.921584  0.671597
0.3   0.3  0.803652  0.867017  0.764352
0.4   0.4  0.816958  0.774454  0.843319
0.5   0.5  0.820672  0.707357  0.890950
0.6   0.6  0.814173  0.644301  0.919529
0.7   0.7  0.788179  0.536378  0.944347
0.8   0.8  0.763423  0.436136  0.966408
0.9   0.9  0.715457  0.278092  0.986713
```

Observations from above cells ¶

- for cutoff 0.6, we have good Specificity score without compromising much on accuracy score
- Therefor we will choose cutoff as 0.6

- With cut-off point as 0.6, we have test accuracy of 80.04%, sensitivity of 61% and specificity of 92.3%

```
1 result_metrics
```

	cutoff	train_acc	train_sen	train_spec	train_prec	test_acc	test_sen	test_spec	test_prec
0	0.36	0.81	0.81	0.82	0.73	0.81	0.8	0.82	0.74
1	0.35	0.81	0.84	0.79	0.72	0.81	0.83	0.79	0.72
2	0.37	0.81	0.79	0.83	0.74	0.81	0.79	0.83	0.75
3	0.3	0.8	0.87	0.76	0.7	0.8	0.86	0.76	0.7
4	0.6	0.81	0.64	0.92	0.83	0.8	0.61	0.92	0.84