Lead Score Case Study

Presented By –

Ritvik Yash

Content

- Problem Statement
- Data Preparation
- EDA
- Training Data
- Building Model
- Model Evaluation
- Observations

Problem Statement

X Education, an online course provider for professionals, aims to boost its current 30% lead conversion rate. The company gathers leads via website forms and video interactions. To enhance efficiency, they seek to identify 'Hot Leads' using a machine learning model, targeting an 80% conversion rate. Provided with 9000 data points, the goal is to build a logistic regression model to predict the likelihood of lead conversion, allowing the sales team to focus on the most promising leads.

Data Preparation

- Importing the Libraries, loading the data in data frame. Checking on the shape and datatypes
- Checking on the Null & Select values and replacing it with Not known.
- Processing null values accordingly & dropping irrelevant columns.

1

```
In [1]:  #Import the required Libraries.

         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
         import statsmodels.api as sm
         from statsmodels.stats.outliers_influence import variance_inflation_factor
         from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
In [55]:  #read the application data and display top 5 rows
          leads_score_data=pd.read_csv("Leads.csv",sep=",",encoding='utf-8')
          leads_score_data.head()
```

Out[55]:

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Las Activit
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	Page Visite on Websit
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	Ema Opene
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	Ema Opene
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	Unreachabl

2

```
In [56]:  #number of rows and columns in the data

          leads_score_data.shape

Out[56]: (9240, 37)
```

```
In [57]:  #displaying the detailed information like column names and non-null counts

          leads_score_data.info()

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 9240 entries, 0 to 9239
          Data columns (total 37 columns):
           #   Column                                    Non-Null Count  Dtype
          ---  ------                                    --------------  -----
           0   Prospect ID                               9240 non-null   object
           1   Lead Number                               9240 non-null   int64
           2   Lead Origin                               9240 non-null   object
           3   Lead Source                               9204 non-null   object
           4   Do Not Email                              9240 non-null   object
           5   Do Not Call                               9240 non-null   object
           6   Converted                                 9240 non-null   int64
           7   TotalVisits                               9103 non-null   float64
           8   Total Time Spent on Website               9240 non-null   int64
           9   Page Views Per Visit                      9103 non-null   float64
           10  Last Activity                             9137 non-null   object
           11  Country                                   6779 non-null   object
           12  Specialization                            7802 non-null   object
           13  How did you hear about X Education        7033 non-null   object
```

```
In [58]:  #null values in leads score data
          def null_values_percentage(dataframe):
              return round(dataframe.isnull().sum()/len(dataframe.index)*100,3)
```

```
In [60]:  missing=null_values_percentage(leads_score_data)
          print(missing)

          Prospect ID                               0.000
          Lead Number                               0.000
```

3

```
In [61]:  # Converting 'Select' values to Not Known.
          leads_score_data = leads_score_data.replace('Select','Not Known')
```

```
In [62]:  leads_score_data['Lead Quality'].value_counts()

Out[62]:  Might be             1560
          Not Sure             1092
          High in Relevance     637
          Worst                 601
          Low in Relevance      583
          Name: Lead Quality, dtype: int64
```

```
In [63]:  # Creating a new category called 'Not Known' for Lead Quality
          leads_score_data['Lead Quality'] = leads_score_data['Lead Quality'].replace(np.nan, 'Not Known')
```

4

```
In [65]:  #drop irrelevant columns
          irrelavant_columns = ['Asymmetrique Profile Index','Asymmetrique Activity Score','Asymmetrique Profile Score']
          leads_score_data.drop(columns=irrelavant_columns, axis=1, inplace=True)
```

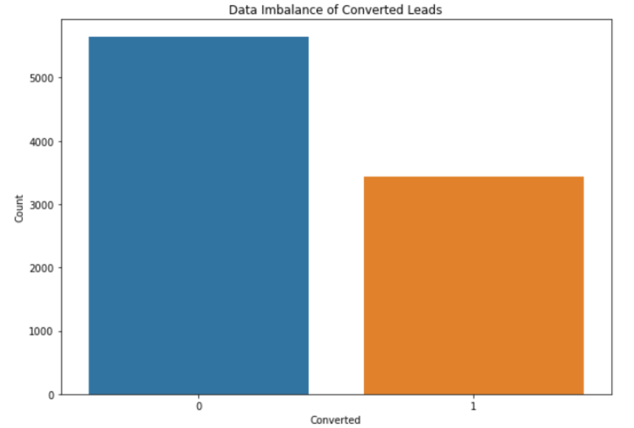
```
In [66]:  #convert the values in the column and replace null values
          leads_score_data['Asymmetrique Activity Index'] = leads_score_data['Asymmetrique Activity Index'].apply(lambda x:
```

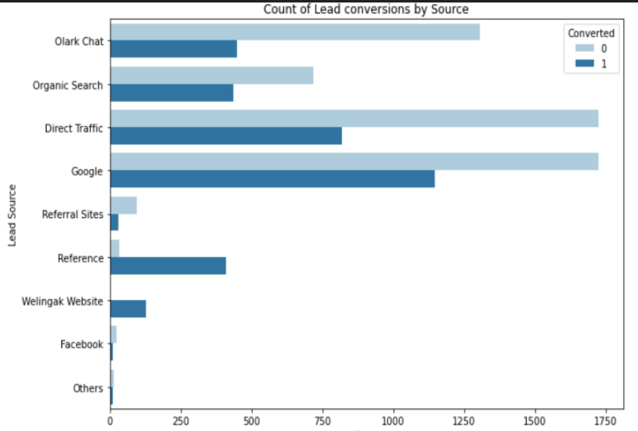
```
In [67]:  # Value counts for 'Asymmetrique Activity Index'
          leads_score_data['Asymmetrique Activity Index'].value_counts()

Out[67]:  3    4580
          2    3839
          1     821
          Name: Asymmetrique Activity Index, dtype: int64
```

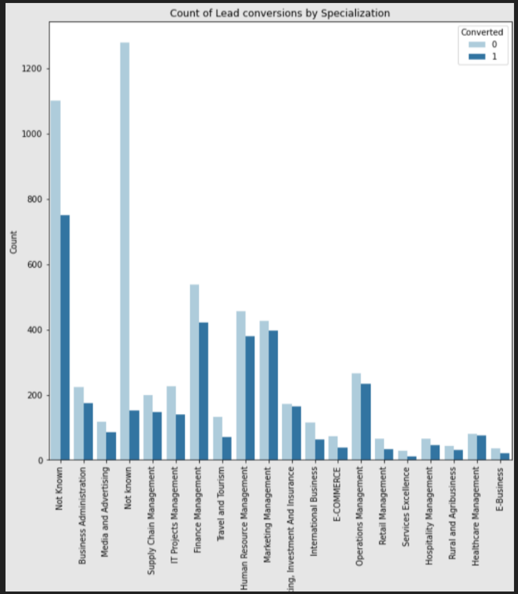
```
In [68]:  # Null values in 'leads_score_data'
          print(null_values_percentage(leads_score_data))

          Prospect ID                               0.000
          Lead Number                               0.000
          Lead Origin                               0.000
```

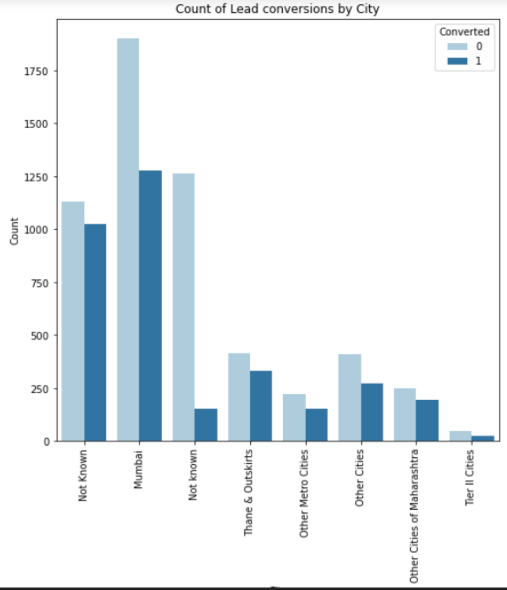
Exploratory Data Analysis(EDA)

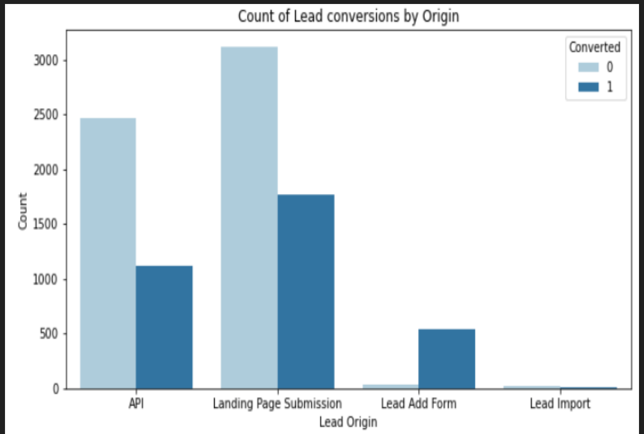
- Data Imbalance of Conversion with respect to different attributes

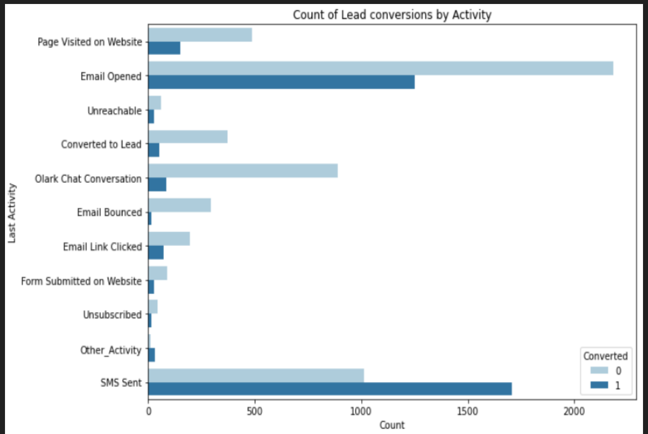


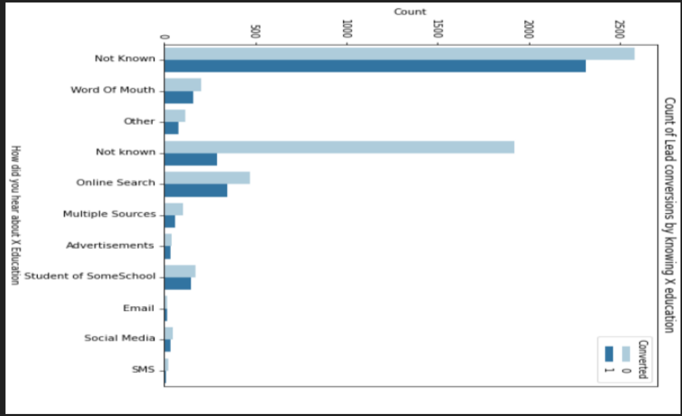




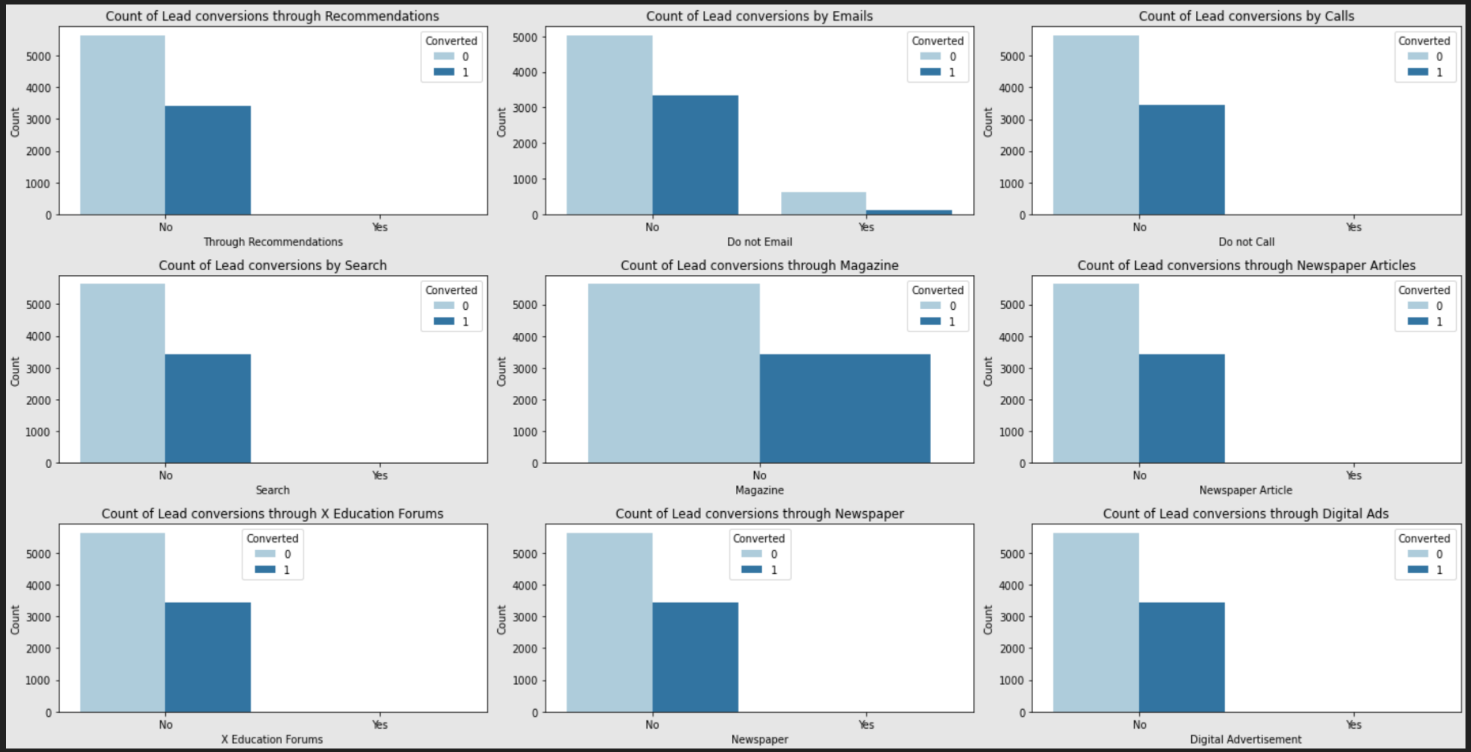


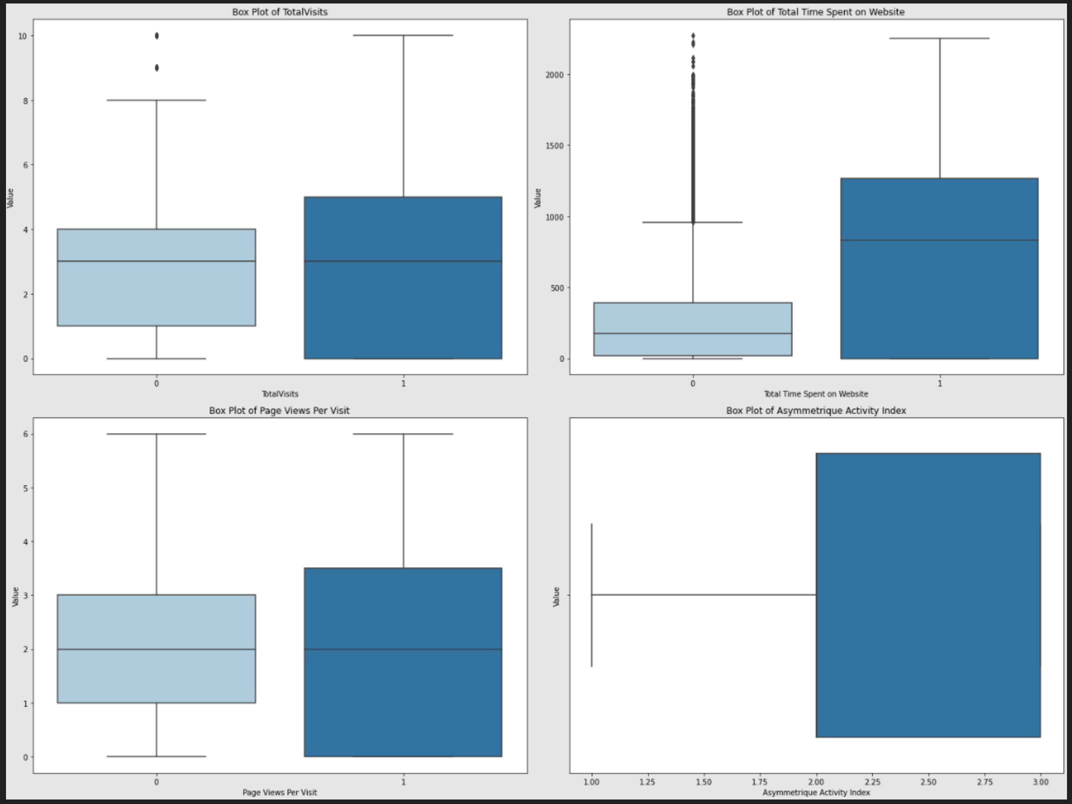






Data Imbalance of Conversion with respect to different channels





Boxplot for total visits, time spent, activity index & pages viewed.

- Dropping irrelevant columns from the dataset

Page Views Per Visit Asymmetrique Activity Index

In [103]:
```
irrelavant_columns=['Lead Number','Tags','Country','Search','Magazine','Newspaper Article','X Education Forums',
                    'Newspaper','Digital Advertisement','Through Recommendations','Receive More Updates About Our Cou
                    'Update me on Supply Chain Content','Get updates on DM Content','I agree to pay the amount throug
                    'A free copy of Mastering The Interview']
leads_score_data.drop(columns=irrelavant_columns, axis=1, inplace=True)
```

In [105]:
```
leads_score_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9074 entries, 0 to 9239
Data columns (total 19 columns):
```

In [108]:
```
# Dropping 'Prospect ID' because that column is irrelevant
leads_score_data.drop(columns="Prospect ID", axis=1, inplace=True)
```

- Getting list of categorical variables and adding dummy variables

In [110]:
```
# Getting list of columns with categorical variables
cat_cols= leads_score_data.select_dtypes(include=['object']).columns
cat_cols
```

Out[110]:
```
Index(['Lead Origin', 'Lead Source', 'Do Not Email', 'Do Not Call',
       'Last Activity', 'Specialization', 'How did you hear about X Education',
       'What is your current occupation',
       'What matters most to you in choosing a course', 'Lead Quality',
       'Lead Profile', 'City', 'Last Notable Activity'],
      dtype='object')
```

In [112]:
```
# Creating dummy for 'Lead Origin'
d = pd.get_dummies(leads_score_data['Lead Origin'], prefix='Lead Origin', drop_first = True)
leads_score_data = pd.concat([leads_score_data,d], axis=1)
leads_score_data.drop(columns="Lead Origin", axis=1, inplace=True)
leads_score_data.head()
```

Training Data

- Splitting the dataset for training & testing, fitting the training data

```
In [138]:  # Standard Scaling the numerical data 'TotalVisits', 'Total Time Spent on Website', 'Page Views Per Visit', 'Asymmetrique Act
           scaler = StandardScaler()

           leads_score_data[['TotalVisits', 'Total Time Spent on Website', 'Page Views Per Visit', 'Asymmetrique Activity Index']] = sca

           leads_score_data.head()
```

Out[138]:

	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Index	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Lead Source_Facebook	Lead Source_Google	Lead Source_Olark Chat
0	0	-1.147962	-0.885664	-1.265259	-0.626751	0	0	0	0	0	1
1	0	0.650299	0.350519	0.130693	-0.626751	0	0	0	0	0	0
2	1	-0.428657	1.924177	-0.148498	-0.626751	1	0	0	0	0	0
3	0	-0.788309	-0.326263	-0.706878	-0.626751	1	0	0	0	0	0
4	1	-0.428657	1.733431	-0.706878	-0.626751	1	0	0	0	1	0


```
In [138]:  # Standard Scaling the numerical data 'TotalVisits', 'Total Time Spent on Website', 'Page Views Per Visit', 'Asymmetrique Act
           scaler = StandardScaler()

           leads_score_data[['TotalVisits', 'Total Time Spent on Website', 'Page Views Per Visit', 'Asymmetrique Activity Index']] = sca

           leads_score_data.head()
```

Out[138]:

	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Index	Lead Origin_Landing Page Submission	Lead Origin_Lead Add Form	Lead Origin_Lead Import	Lead Source_Facebook	Lead Source_Google	Lead Source_Olark Chat
0	0	-1.147962	-0.885664	-1.265259	-0.626751	0	0	0	0	0	1
1	0	0.650299	0.350519	0.130693	-0.626751	0	0	0	0	0	0
2	1	-0.428657	1.924177	-0.148498	-0.626751	1	0	0	0	0	0
3	0	-0.788309	-0.326263	-0.706878	-0.626751	1	0	0	0	0	0
4	1	-0.428657	1.733431	-0.706878	-0.626751	1	0	0	0	1	0

- Using RFE selecting the best 16 features from the dataset to train model

```
In [143]:  # RFE to get best 16 variables fromthe list of 98 variables

           from sklearn.linear_model import LogisticRegression
           logreg = LogisticRegression(solver='liblinear')
           logreg.fit(X_train,y_train)


           from sklearn.feature_selection import RFE
           rfe = RFE(logreg,  n_features_to_select=16)          # ru
           rfe_1 = rfe.fit(X_train, y_train)
```

```
In [145]:  #list of RFE supported columns

           col = X_train.columns[rfe.support_]
           col

Out[145]:  Index(['total_time_spent_on_website', 'lead_origin_lead_add_form',
                  'lead_source_olark_chat', 'lead_source_welingak_website',
                  'do_not_email_yes', 'last_activity_other_activity',
                  'last_activity_sms_sent',
                  'what_is_your_current_occupation_working_professional',
                  'lead_quality_might_be', 'lead_quality_not_known',
                  'lead_quality_not_sure', 'lead_quality_worst',
                  'lead_profile_lateral_student', 'lead_profile_student_of_someschool',
                  'last_notable_activity_modified', 'last_notable_activity_unreachable'],
                 dtype='object')
```

Building Model

Building the model

In [149]:
```python
# Building Model for iteration #1

logm1 = sm.GLM(y_train,X_train_rfe, family = sm.families.Binomial())
res = logm1.fit()
res.summary()
```

Out[149]:

Generalized Linear Model Regression Results

Dep. Variable:	converted	**No. Observations:**	6351
Model:	GLM	**Df Residuals:**	6334
Model Family:	Binomial	**Df Model:**	16
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2149.7
Date:	Tue, 23 Jul 2024	**Deviance:**	4299.5
Time:	21:33:12	**Pearson chi2:**	6.44e+03
No. Iterations:	21		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.3787	0.135	10.203	0.000	1.114	1.644
total_time_spent_on_website	1.0983	0.045	24.192	0.000	1.009	1.187
lead_origin_lead_add_form	2.6663	0.243	10.964	0.000	2.190	3.143
lead_source_olark_chat	1.3909	0.115	12.097	0.000	1.166	1.616
lead_source_welingak_website	3.6910	0.761	4.849	0.000	2.199	5.183
do_not_email_yes	-1.3919	0.197	-7.068	0.000	-1.778	-1.006
last_activity_other_activity	1.6307	0.542	3.008	0.003	0.568	2.693
last_activity_sms_sent	1.3706	0.085	16.078	0.000	1.204	1.538
what_is_your_current_occupation_working_professional	1.7694	0.221	7.995	0.000	1.336	2.203
lead_quality_might_be	-1.5452	0.158	-9.781	0.000	-1.855	-1.236

Accuracy & Confusion Matrix of the model

In [150]:
```python
# Accuracy and confustion matrix for iteration #1
y_train_pred = res.predict(X_train_rfe).values.reshape(-1)
y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Converted_Prob':y_train_pred})
y_train_pred_final['predicted'] = y_train_pred_final.Converted_Prob.map(lambda x: 1 if x > 0.5 else 0)
y_train_pred_final.head()
# Get Confusion matrix
tn,fp,fn,tp= confusion_matrix(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final.predicted).ravel()
print('Confusion Matrix:')
print('True Negative:',tn, '    ','False Positive:',fp)
print('False Negative:',fn,'    ','True Positive:',tp, '\n')
# Checking the overall model accuracy
print('Overall model accuracy:', accuracy_score(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final.predicted))
```

```
Confusion Matrix:
True Negative: 3544      False Positive: 361
False Negative: 596      True Positive: 1850

Overall model accuracy: 0.8493150684931506
```

Calculating VIF to check multicollinearity

In [151]:
```python
# VIF for iteration #1
X_train_new = X_train_rfe
X_train_new = X_train_new.drop(['const'], axis=1)
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[151]:

	Features	VIF
9	lead_quality_not_known	1.90
6	last_activity_sms_sent	1.74
11	lead_quality_worst	1.68
8	lead_quality_might_be	1.66

- Final Model with nearly ~85% accuracy

```
In [162]:
# Accuracy and Confusion matrix for final model
tn,fp,fn,tp= confusion_matrix(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final.predicted).ravel()
print('Confusion Matrix:')
print('True Negative:',tn, '     ','False Positive:',fp)
print('False Negative:',fn,'     ','True Positive:',tp, '\n')
# Checking the overall model accuracy
print('Overall model accuracy:', accuracy_score(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final.predicted))
```

```
Confusion Matrix:
True Negative: 3542      False Positive: 363
False Negative: 594      True Positive: 1852

Overall model accuracy: 0.8493150684931506
```

- Calculating Accuracy, Specificity, Sensitivity, False Positive Rate

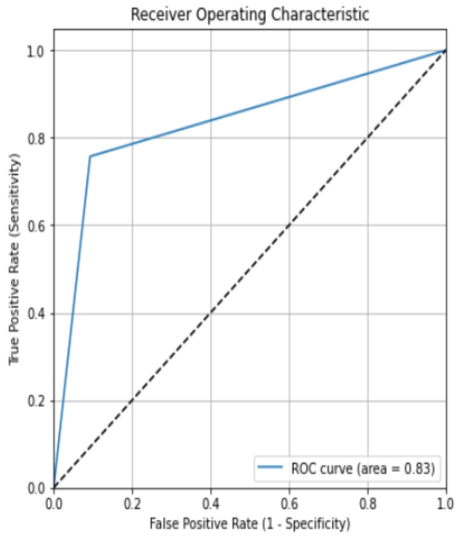
```
In [163]:
# Accuracy, Sensitivity, Specificity, False Positive Rate
print('Overall model accuracy:', accuracy_score(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final.predicted))
print('Sensitivity / Recall: ',tp / float(tp+fn))
print('Specificity: ', tn / float(tn+fp))
print('False Positive Rate: ',fp/ float(tn+fp))
```

```
Overall model accuracy: 0.8493150684931506
Sensitivity / Recall:  0.7571545380212592
Specificity:  0.9070422535211268
False Positive Rate:  0.09295774647887324
```

Model Evaluation

- Plotting ROC Curve

In [164]:
```
# Plotting ROC curve

fpr, tpr, thresholds= roc_curve(y_train_pred_final.Converted, y_train_pred_final.predicted, drop_intermediate = False )
auc_score= roc_auc_score(y_train_pred_final.Converted, y_train_pred_final.predicted)
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.grid()
plt.show()
```



- Finding Optimal Cut-off

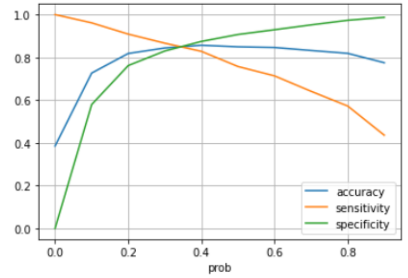
In [165]:
```
# Finding optimal cut-off

num= [float(x)/10 for x in range(10)]
for i in num:
    y_train_pred_final[i]=  y_train_pred_final.Converted_Prob.map(lambda x: 1 if x > i else 0)
y_train_pred_final.head()
```

Out[165]:

	Converted	Converted_Prob	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	0.114137	0	1	1	0	0	0	0	0	0	0	0
1	0	0.110472	0	1	1	0	0	0	0	0	0	0	0
2	0	0.223635	0	1	1	1	0	0	0	0	0	0	0
3	1	0.786635	1	1	1	1	1	1	1	1	1	0	0
4	1	0.636005	1	1	1	1	1	1	1	1	0	0	0

In [166]:
```
df= pd.DataFrame(columns = ['prob','accuracy','sensitivity','specificity'])

for i in num:
    TN,FP,FN,TP= confusion_matrix(y_true= y_train_pred_final.Converted, y_pred= y_train_pred_final[i]).ravel()
    accuracy= (TN+TP)/float(TN+FP+FN+TP)
    specificity= TN / float(TN+FP)
    sensitivity= TP / float(TP+FN)
    df.loc[i]= [i,accuracy,sensitivity,specificity]

df
```

In [167]:
```
df.plot.line(x= 'prob', y= ['accuracy','sensitivity','specificity'])
plt.grid()
plt.show()
```



In above plot, it's visible that 0.34 is the optimal point to set as cutoff probability for our model.

- Final Step of adding lead score to test data

In [179]:
```python
# Adding Lead Number in the final results dataframe for test set
y_test_pred_final= y_test_pred_final.merge(leads_score_data2['Lead Number'], how= 'left', left_index= True, right_index= True
y_test_pred_final['Lead Score']= y_test_pred_final.Converted_Prob * 100
y_test_pred_final= y_test_pred_final[['Lead Number', 'Converted', 'predicted', 'Converted_Prob','Lead Score']].sort_values(
    'Lead Score', ascending= False)
y_test_pred_final.head(10)
```

Out[179]:

	Lead Number	Converted	predicted	Converted_Prob	Lead Score
868	651281	1	1	0.999009	99.900918
140	659123	1	1	0.998850	99.884959
655	653773	1	1	0.998850	99.884959
2565	634875	1	1	0.998709	99.870935
1582	643814	1	1	0.998587	99.858708
1963	640614	1	1	0.997345	99.734513
1525	644144	1	1	0.997254	99.725410
2055	639824	1	1	0.997221	99.722081
1786	642069	1	1	0.997011	99.701112
619	654027	1	1	0.996916	99.691576

- Final model of ~85% accuracy.

In [180]:
```python
print('Model Evaluation Metrics on Test dataset')


TN,FP,FN,TP= confusion_matrix(y_true= y_test_pred_final.Converted, y_pred= y_test_pred_final.predicted).ravel()
print('Confusion Matrix:')
print('True Negative:',TN, '     ','False Positive:',FP)
print('False Negative:',FN,'     ','True Positive:',TP, '\n')


print('Overall model accuracy:', accuracy_score(y_true= y_test_pred_final.Converted, y_pred= y_test_pred_final.predicted))

print('Sensitivity / Recall: ',TP / float(TP+FN))

print('Specificity: ', TN / float(TN+FP))

print('False Positive Rate: ',FP/ float(TN+FP))
```

```
Model Evaluation Metrics on Test dataset
Confusion Matrix:
True Negative: 1574      False Positive: 160
False Negative: 263      True Positive: 726

Overall model accuracy: 0.8446566287183254
Sensitivity / Recall:  0.7340748230535895
Specificity:  0.9077277970011534
False Positive Rate:  0.0922722029988466
```

Observations

Variables like "lead_origin_lead_add_form," "lead_source_welingak_website," and "lead_source_olark_chat" positively impact lead conversion rates. Conversely, "lead_quality_worst," "lead_quality_not_sure," and "lead_quality_might_be" negatively affect conversion rates.

To reach an 80% conversion rate, X Education should focus on acquiring more leads from lead_add_form, welingak, and olark_chat, and aggressively follow up with these leads. Additionally, improving lead quality is crucial, as poor lead quality significantly reduces conversion rates. It's best to avoid leads with poor quality indicators. The sales team should prioritize leads from positively impactful sources and avoid those with negative impacts.

Thank You!