LEAD SCORE CASE STUDY

We built a Logistic regression model to assign a lead score between 0 and 100 to each of the leads which can be used by the company to target potential leads

Submitted by- Archana, Pranshu and Pulkit

STEP 1: Import Libraries and Data :

[1]:
```
# supress warnings:
import warnings
warnings.filterwarnings('ignore')

# import necessary libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# reading data:
df= pd.read_csv('Leads.csv')
df
```

[3]:
```
# Data frame inspection :

df.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
```

GETTING THE SHAPE OF THE DATA FRAME, DESCRIBING THE DATAFRAME AND CONVERTING VARIABLES

[5]:
```
df.shape
```

[5]:
```
(9240, 37)
```

[6]:
```
df.describe()
```

[6]:
```
''''''
```

Finding the unique values of the columns

[10]:
```
# checking if there are columns with unique value since it wont affect analysis so going to drop that:
df.nunique()
```

[10]:
```
Prospect ID                                    9240
,Lead Number                                    9240
,Lead Origin                                       5
,Lead Source                                      20
,Do Not Email                                      2
,Do Not Call                                       2
,Converted                                         2
,TotalVisits                                      41
,Total Time Spent on Website                    1731
,Page Views Per Visit                            114
,Last Activity                                    17
,Country                                          38
,Specialization                                   19
,How did you hear about X Education               10
,What is your current occupation                   6
,What matters most to you in choosing a course     3
,Search                                            2
,Magazine                                          1
,Newspaper Article                                 2
,X Education Forums                                2
,Newspaper                                         2
,Digital Advertisement                             2
,Through Recommendations                           2
,Receive More Updates About Our Courses            1
,Tags                                             26
,Lead Quality                                      5
,Update me on Supply Chain Content                 1
,Get updates on DM Content                         1
,Lead Profile                                      6
,City                                              7
,Asymmetrique Activity Index                       3
,Asymmetrique Profile Index                        3
,Asymmetrique Activity Score                      12
```

Caption

[12]:
```
df.columns
```

[12]:
```
Index(['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', 'Last Activity',
,       'Country', 'Specialization', 'How did you hear about X Education',
,       'What is your current occupation',
,       'What matters most to you in choosing a course', 'Search', 'Magazine',
,       'Newspaper Article', 'X Education Forums', 'Newspaper',
,       'Digital Advertisement', 'Through Recommendations',
,       'Receive More Updates About Our Courses', 'Tags', 'Lead Quality',
,       'Update me on Supply Chain Content', 'Get updates on DM Content',
,       'Lead Profile', 'City', 'Asymmetrique Activity Index',
,       'Asymmetrique Profile Index', 'Asymmetrique Activity Score',
,       'Asymmetrique Profile Score',
,       'I agree to pay the amount through cheque',
,       'A free copy of Mastering The Interview', 'Last Notable Activity'],
,      dtype='object')
```

[13]:
```
# dropping unique values of columns as it not needed:

df1= df.drop(['Magazine','Receive More Updates About Our Courses','I agree to pay the amount through cheque','Get update
```

[14]:
```
# checking percentage of null values :
round(100*(df1.isnull().sum()/len(df1.index)), 2)
```

[14]:
```
Prospect ID                                    0.00
,Lead Number                                    0.00
 Lead Origin                                    0.00
```

Caption

DROPPING THE COLUMNS WITH NA PERCENTAGE VALUE MORE THAN 40

```
df2 = df1.drop(['Prospect ID','Asymmetrique Profile Index','Asymmetrique Activity Index','Asymmetrique A
df2.head()
```

[15]: ''''''

RECHECKING THE PERCENTAGE OF MISSING VALUES NOW

[16]:
```
# from previous record of head detail,it is understandable that
```

[17]:
```
# Rechecking the percentage of missing values
round(100*(df2.isnull().sum()/len(df2.index)), 2)
```

[17]:
```
Lead Origin                                        0.00
,Lead Source                                        0.39
,Do Not Email                                       0.00
,Do Not Call                                        0.00
,Converted                                          0.00
,TotalVisits                                        1.48
,Total Time Spent on Website                        0.00
,Page Views Per Visit                               1.48
,Last Activity                                      1.11
,Country                                           26.63
,Specialization                                    15.56
,What is your current occupation                   29.11
,What matters most to you in choosing a course     29.32
,Search                                             0.00
,Newspaper Article                                  0.00
,X Education Forums                                 0.00
,Newspaper                                          0.00
,Digital Advertisement                              0.00
,Through Recommendations                            0.00
,A free copy of Mastering The Interview             0.00
```

Caption

FILLING THE NA VALUES OF THE VARIABLES WHERE WE HAVE HEAVY NA VALUES

[19]:
```
df2['Specialization'] = df2['Specialization'].fillna('abc')
df2['What matters most to you in choosing a course'] = df2['What matters most to you in choosing a course'].fillna('abc')
df2['Country'] = df2['Country'].fillna('abc')
df2['What is your current occupation'] = df2['What is your current occupation'].fillna('abc')
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 9240 entries, 0 to 9239

Data columns (total 21 columns):

 #   Column                                         Non-Null Count  Dtype

---  ------                                         --------------  -----

 0   Lead Origin                                    9240 non-null   object

 1   Lead Source                                    9204 non-null   object

 2   Do Not Email                                   9240 non-null   object

 3   Do Not Call                                    9240 non-null   object

 4   Converted                                      9240 non-null   int64

 5   TotalVisits                                    9103 non-null   float64

 6   Total Time Spent on Website                    9240 non-null   int64

 7   Page Views Per Visit                           9103 non-null   float64

 8   Last Activity                                  9137 non-null   object

 9   Country                                        9240 non-null   object
```

Caption

CHECKING PERCENT OF NULL VALUES AND LOSE IF NULL VALUES ARE REMOVED

[20]:
```
# rechecking of percentanges of null value :
round(100*(df2.isnull().sum()/len(df2.index)),2)
```

[20]:
```
Lead Origin                                        0.00
,Lead Source                                        0.39
,Do Not Email                                       0.00
,Do Not Call                                        0.00
,Converted                                          0.00
,TotalVisits                                        1.48
,Total Time Spent on Website                        0.00
,Page Views Per Visit                               1.48
,Last Activity                                      1.11
,Country                                            0.00
,Specialization                                     0.00
,What is your current occupation                    0.00
,What matters most to you in choosing a course      0.00
,Search                                             0.00
,Newspaper Article                                  0.00
,X Education Forums                                 0.00
,Newspaper                                          0.00
,Digital Advertisement                              0.00
,Through Recommendations                            0.00
,A free copy of Mastering The Interview             0.00
,Last Notable Activity                              0.00
,dtype: float64
```

[21]:
```
# Checking the percent of lose if the null values are removed
round(100*(sum(df2.isnull().sum(axis=1) > 1)/df2.shape[0]),2)
```

[21]:
```
1.48
```

[22]:
```
df3 = df2[df2.isnull().sum(axis=1) <1]
```

[23]:
```
# Code for checking number of rows left in percent
round(100*(df3.shape[0])/(df.shape[0]),2)
```

Caption

EXPLORATORY DATA ANALYSIS STARTS FROM HERE

[25]:
```
# To familiarize all the categorical values
for column in df2:
    print(df2[column].astype('category').value_counts())
    print('-----------------------------------------------------------------------------------------')
```

```
landing page submission    4886

api                        3580

lead add form               718

lead import                  55

quick add form                1

Name: Lead Origin, dtype: int64

-----------------------------------------------------------------------------------------

google               2873

direct traffic       2543
```

[26]:
```
# # relation of all categorical data with 'converted' :

plt.figure(figsize = (10,5))

plt.subplot(1,2,1)
sns.countplot(x='Lead Origin', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Lead Origin')

plt.subplot(1,2,2)
sns.countplot(x='Lead Source', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Lead Source')
plt.show()
```

Caption

[27]:
```
plt.subplot(1,2,1)
sns.countplot(x='Do Not Email', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Do Not Email')

plt.subplot(1,2,2)
sns.countplot(x='Do Not Call', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Do Not Call')
plt.show()
```

```
<Figure size 640x480 with 2 Axes>
```

[28]:
```
plt.figure(figsize = (10,5))

plt.subplot(1,2,1)
sns.countplot(x='Last Activity', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Last Activity')

plt.subplot(1,2,2)
sns.countplot(x='Country', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Country')
plt.show()
```

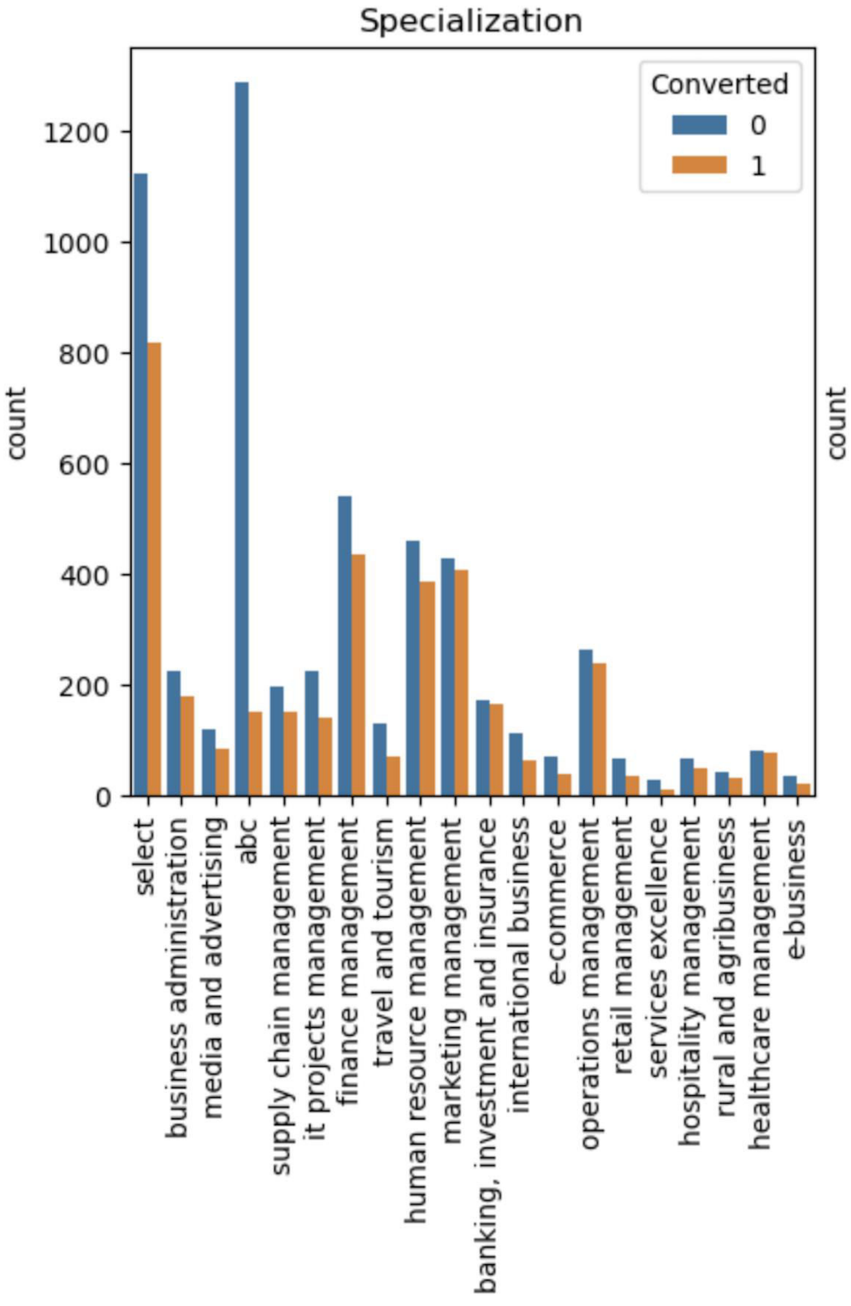
```
<Figure size 1000x500 with 2 Axes>
```

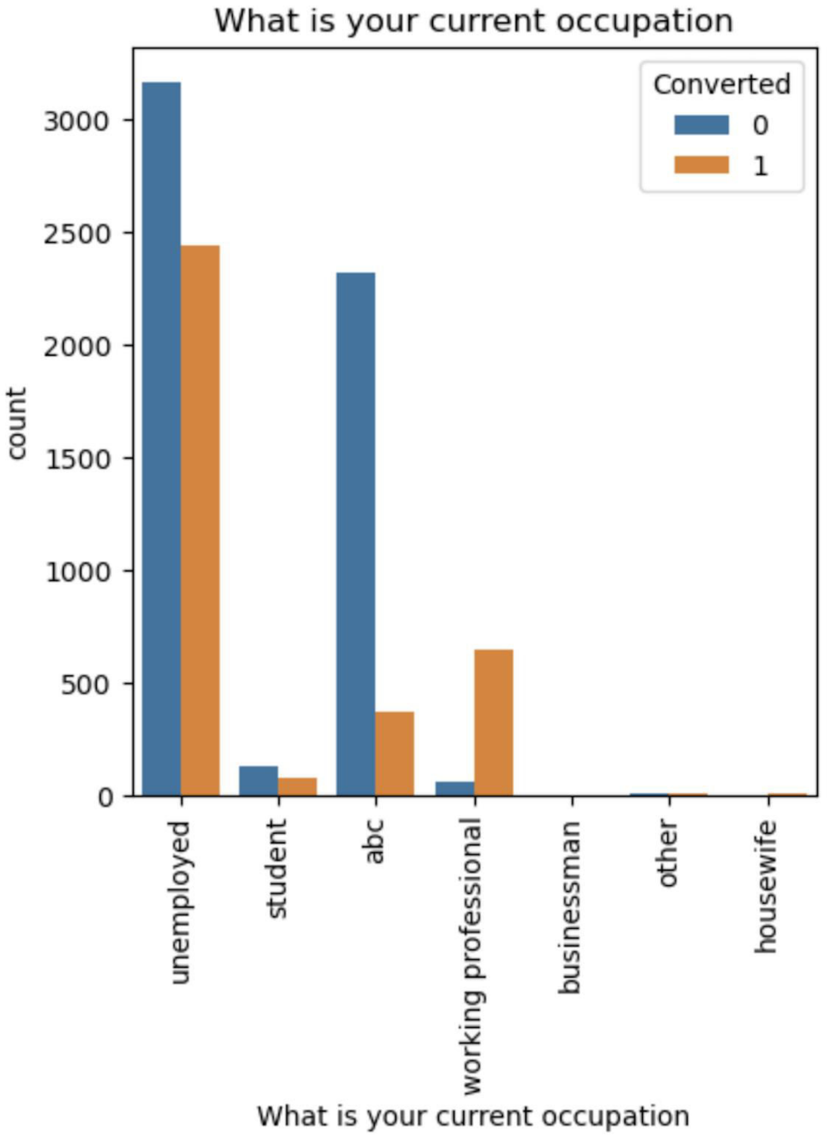
[29]:
```
plt.figure(figsize = (10,5))

plt.subplot(1,2,1)
sns.countplot(x='Specialization', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Specialization')

plt.subplot(1,2,2)
sns.countplot(x='What is your current occupation', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('What is your current occupation')
plt.show()
```

Specialization

What is your current occupation

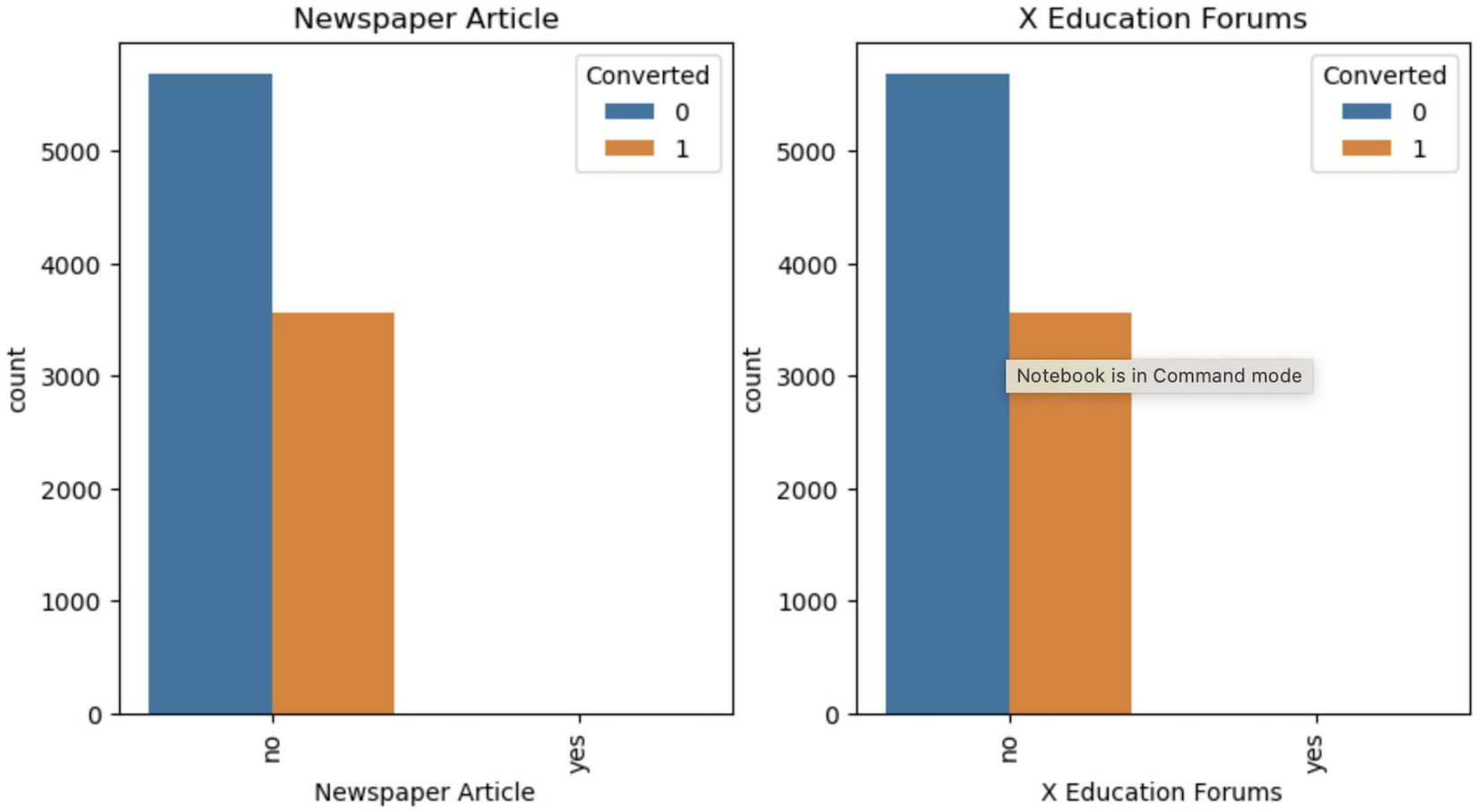
Caption


Specialization
Converted
0
1
count
0
200
400
600
800
1000
1200
select
business administration
media and advertising
abc
supply chain management
it projects management
finance management
travel and tourism
human resource management
marketing management
banking, investment and insurance
international business
e-commerce
operations management
retail management
services excellence
hospitality management
rural and agribusiness
healthcare management
e-business


What is your current occupation
Converted
0
1
count
0
500
1000
1500
2000
2500
3000
unemployed
student
abc
working professional
businessman
other
housewife
What is your current occupation

Caption

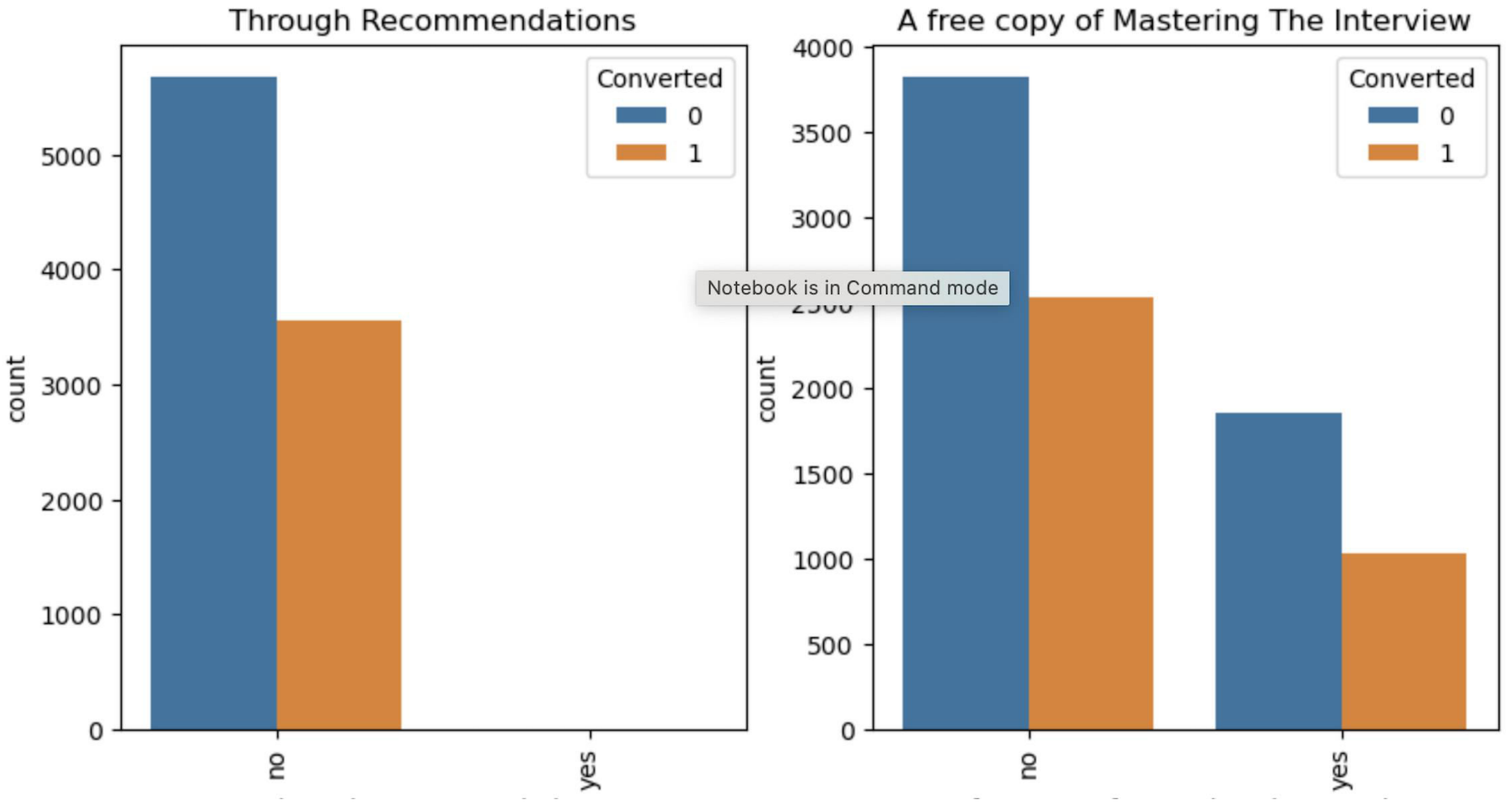
[31]:
```
plt.figure(figsize = (10,5))

plt.subplot(1,2,1)
sns.countplot(x='Newspaper Article', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Newspaper Article')

plt.subplot(1,2,2)
sns.countplot(x='X Education Forums', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('X Education Forums')
plt.show()
```



Caption

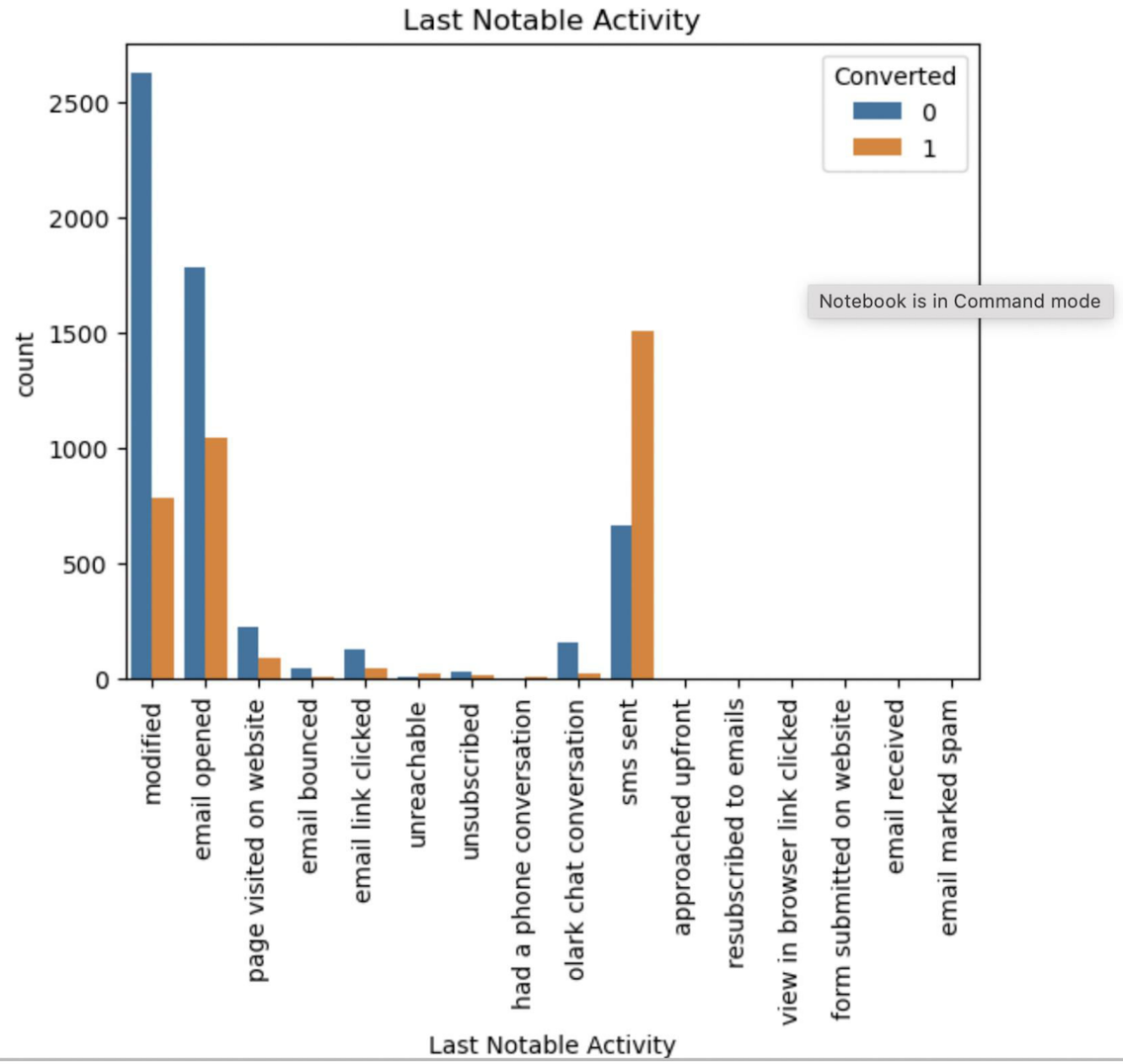
PLOTTING THE SUBPLOT OF THROUGH RECOMMENDATION AGAINST CONVERTED COLUMN

[33]:
```
plt.figure(figsize = (10,5))

plt.subplot(1,2,1)
sns.countplot(x='Through Recommendations', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Through Recommendations')

plt.subplot(1,2,2)
sns.countplot(x='A free copy of Mastering The Interview', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('A free copy of Mastering The Interview')
plt.show()
```



Caption

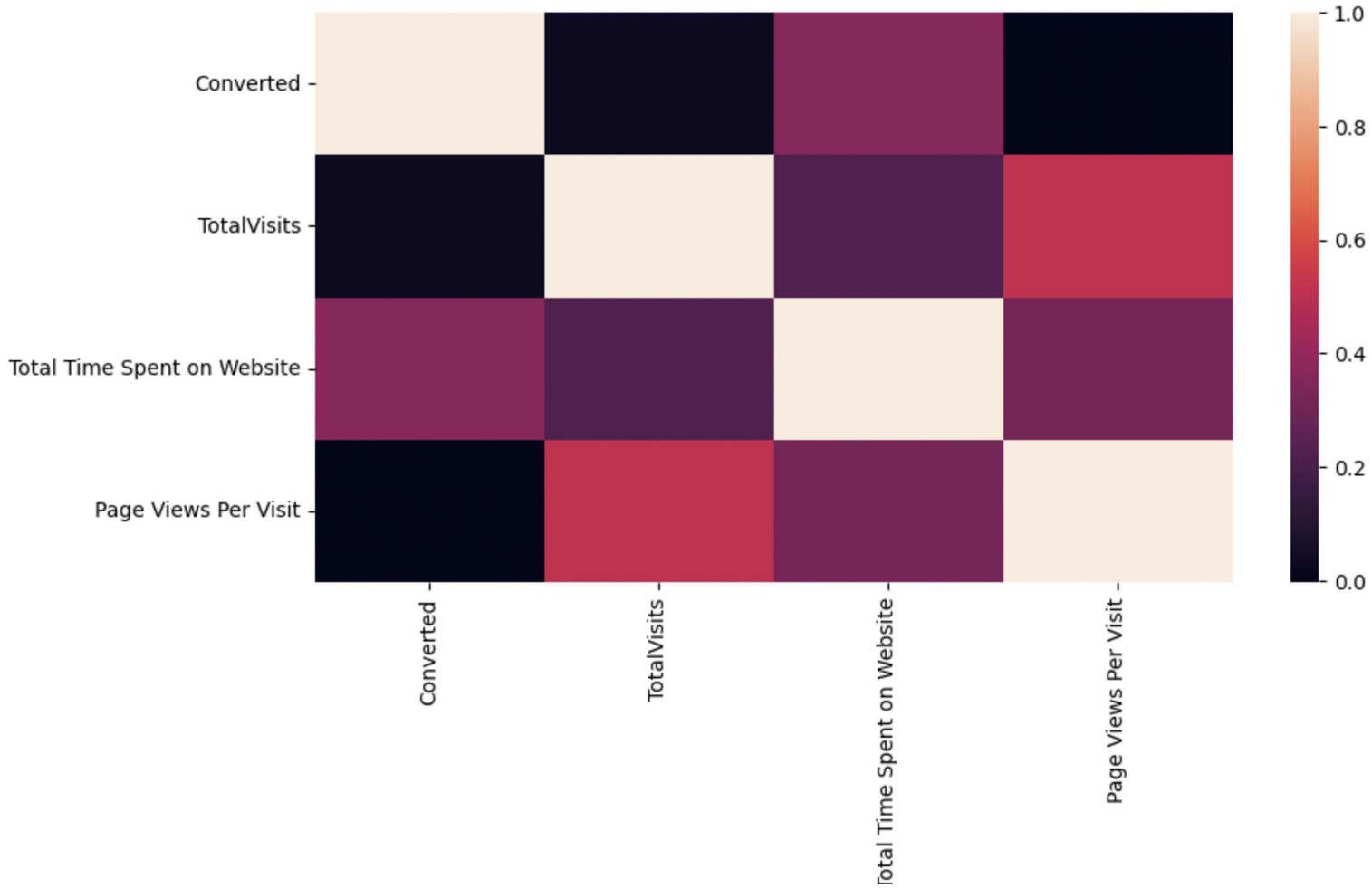
[34]:
```
sns.countplot(x='Last Notable Activity', hue='Converted', data= df2).tick_params(axis='x', rotation = 90)
plt.title('Last Notable Activity')
plt.show()
```



Caption

PLOTTING THE HEATMAP OF CORRELATION

[36]:
```
# To check the correlation among varibles
plt.figure(figsize=(10,5))
sns.heatmap(df2.corr())
plt.show()
```



Caption

[37]:
```
##It is understandable from the above EDA that there are many elements that have very little data and so will be of less relevan

numeric = df2[['TotalVisits','Total Time Spent on Website','Page Views Per Visit']]
numeric.describe(percentiles=[0.25,0.5,0.75,0.9,0.99])
```

[37]: ''''''

CREATING DUMMY VARIABLES

```
[38]: Index(['Lead Origin', 'Lead Source', 'Do Not Email', 'Do Not Call',
      ,       'Converted', 'TotalVisits', 'Total Time Spent on Website',
      ,       'Page Views Per Visit', 'Last Activity', 'Country', 'Specialization',
      ,       'What is your current occupation',
      ,       'What matters most to you in choosing a course', 'Search',
      ,       'Newspaper Article', 'X Education Forums', 'Newspaper',
      ,       'Digital Advertisement', 'Through Recommendations',
      ,       'A free copy of Mastering The Interview', 'Last Notable Activity'],
      ,      dtype='object')
```

```
[39]: df2.loc[:, df2.dtypes == 'object'].columns
```

```
[39]: Index(['Lead Origin', 'Lead Source', 'Do Not Email', 'Do Not Call',
      ,       'Last Activity', 'Country', 'Specialization',
      ,       'What is your current occupation',
      ,       'What matters most to you in choosing a course', 'Search',
      ,       'Newspaper Article', 'X Education Forums', 'Newspaper',
      ,       'Digital Advertisement', 'Through Recommendations',
      ,       'A free copy of Mastering The Interview', 'Last Notable Activity'],
      ,      dtype='object')
```

```
[40]: # Create dummy variables using the 'get_dummies'
      dummy = pd.get_dummies(df2[['Lead Origin','Specialization' ,'Lead Source', 'Do Not Email', 'Last Activity', 'What is your curren
      # Add the results to the master dataframe
      df2 = pd.concat([df2, dummy], axis=1)
      df2
```

```
[40]: ,,,
```

[42]:
```
# Import the required library
from sklearn.model_selection import train_test_split
# we specify this so that train and test data set always have the same row respectively .
```

[43]:
```
X = df2.drop(['Converted'], 1)
X.head()
```

[43]: ''''''

KEEPING THE TRAINING RATIO TO BE 70% AND THE TEST RATIO TO BE 30%

[46]:
```
# Import MinMax scaler :

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# apply Scaler() the numeric variables :
X_train[['TotalVisits', 'Page Views Per Visit', 'Total Time Spent on Website']] = scaler.fit_transform(X_train[['TotalVisits', '
X_train.head()
```

[46]: ''''''

BUILDING THE MODEL NOW

3934	0.000000	0.000000	0.000000	0

storage.

,

5 rows × 84 columns

,

Model building :

[47]:
```
# Import 'LogisticRegression'
from sklearn.linear_model import LogisticRegression
lm = LogisticRegression()
```

[48]:
```
from sklearn.feature_selection import RFE
rfe = RFE(lm,n_features_to_select= 15)            # running RFE with 15 va
rfe = rfe.fit(X_train, y_train)
```

Caption

Finding na values and replace them with mode. Moving ahead, we will calculate the RFE and fitting the model

```
X_train.isnull().sum()
```

iCloud Storage Full
To keep using iCloud Photos, iCloud Mail, and Backup, upgrade or manage your storage.

```
TotalVisits                                           99
,Total Time Spent on Website                           0
,Page Views Per Visit                                 99
,Lead Origin_landing page submission                   0
,Lead Origin_lead add form                             0
,                                                     ..
,Last Notable Activity_resubscribed to emails          0
,Last Notable Activity_sms sent                        0
,Last Notable Activity_unreachable                     0
,Last Notable Activity_unsubscribed                    0
,Last Notable Activity_view in browser link clicked    0
,Length: 84, dtype: int64
```

```
# from above we can see there are two columns haning nan values and create error while doing modeling so better to


X_train['TotalVisits'] = X_train['TotalVisits'].fillna(X_train['TotalVisits'].mode()[0])
X_train['Page Views Per Visit'] = X_train['Page Views Per Visit'].fillna(X_train['Page Views Per Visit'].mode()[0])
```

```
rfe = RFE(lm,n_features_to_select= 15)             # running RFE with 15 variables as output
rfe = rfe.fit(X_train, y_train)
```

```
# Features that have been selected by RFE
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('TotalVisits', False, 2),
, ('Total Time Spent on Website', True, 1),
```

Caption

Now, we will proceed and start building the model using Generalized Linear model. First we will add constants and then move ahead

[54]:
```
# Put all the columns selected by RFE in the variable 'col'
col = X_train.columns[rfe.support_]
```

[55]:
```
##   P value and VIF :
```

[56]:
```
# Selecting columns selected by RFE
X_train = X_train[col]
```

[57]:
```
# Importing statsmodels
import statsmodels.api as sm
```

[58]:
```
X_train_sm = sm.add_constant(X_train)
logm1 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm1.fit()
res.summary()
```

[58]: ''''''''''''''''''''''''''''

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6468
Model:	GLM	**Df Residuals:**	6452
Model Family:	Binomial	**Df Model:**	15
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2663.6
Date:	Mon, 18 Mar 2024	**Deviance:**	5327.1
Time:	08:13:00	**Pearson chi2:**	6.55e+03
No. Iterations:	21	**Pseudo R-squ. (CS):**	0.3969
Covariance Type:	nonrobust		

''

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.4108	0.095	-25.496	0.000	-2.596	-2.225

Now, we will calculate Variance inflation factor to detect multilollinearity and remove the variables that have high

[59]:
```
# Importing 'variance_inflation_factor'
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[60]:
```
# Make a VIF dataframe for all the variables present
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
```

[60]: ''''''

Going forward, we will build different models by removing columns with high vif or those that does not have relevance for model building till we get a stable model

[61]:
```
# VIF is seems to be fine but as above we can see some columns p value is high so we have to drop that columns :

X_train.drop('What is your current occupation_housewife', axis = 1, inplace = True)
```

[62]:
```
# Refit the model with the new set of features
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()
```

[62]:
```
''''''
```

PREDICTING THE MODEL AND CALCULATING THE CONVERSION RATE AND PREDICTING THE PROBABILITY OF PREDICTED ONES

[67]:
```
# Predicting the probabilities on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

[67]:
```
1871    0.195435
,6795    0.238048
,3516    0.225657
,8105    0.868399
,3934    0.089945
,4844    0.986762
,3297    0.219416
,8071    0.979638
,987     0.132565
,7423    0.921206
,dtype: float64
```

[68]:
```
# Reshaping to an array
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

[68]:
```
array([0.19543547, 0.23804822, 0.22565668, 0.86839887, 0.08994465,
,       0.98676241, 0.2194163 , 0.97963822, 0.13256498, 0.92120632])
```

[69]:
```
# Data frame with given convertion rate and probablity of predicted ones
y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Conversion_Prob':y_train_pred})
y_train_pred_final.head()
```

[69]: ,,,

Now, next step is model evaluation using Confusion matrix and ROC Curve

[71]:
```
# Importing metrics from sklearn for evaluation
from sklearn import metrics
```

[72]:
```
# Creating confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Predicted )
confusion
```

[72]:
```
array([[3539,  463],
,       [ 792, 1674]], dtype=int64)
```

[73]:
```
# Check the overall accuracy
metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.Predicted)
```

[73]:
```
0.8059678416821274
```

[74]:
```
# we can see accuracy rate is 80.59 and shows good value :
```

[75]:
```
# Substituting the value of true positive
TP = confusion[1,1]
# Substituting the value of true negatives
TN = confusion[0,0]
# Substituting the value of false positives
FP = confusion[0,1]
# Substituting the value of false negatives
FN = confusion[1,0]
```

[76]:
```
# Calculating the sensitivity
TP/(TP+FN)
```

[76]:
```
0.6788321167883211
```

[77]:
```
# Calculating the specificity
TN/(TN+FP)
```

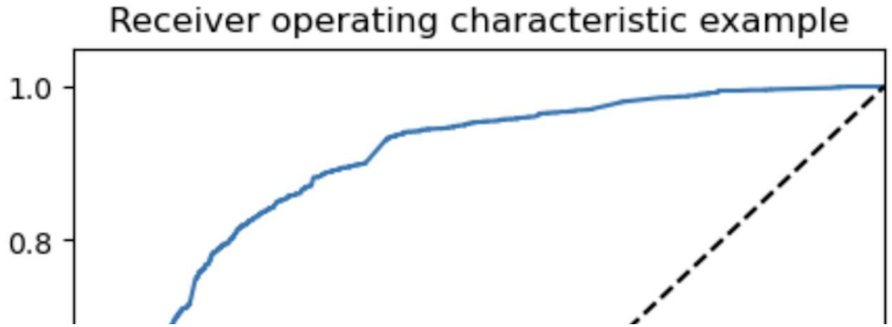
[77]:
```
0.8843078460769616
```

Caption

```
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    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 or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

[81]:
```
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Converted, y_train_pred_final.Conversion_Prob, drop_inte
```

[82]:
```
# Call the ROC function
draw_roc(y_train_pred_final.Converted, y_train_pred_final.Conversion_Prob)
```

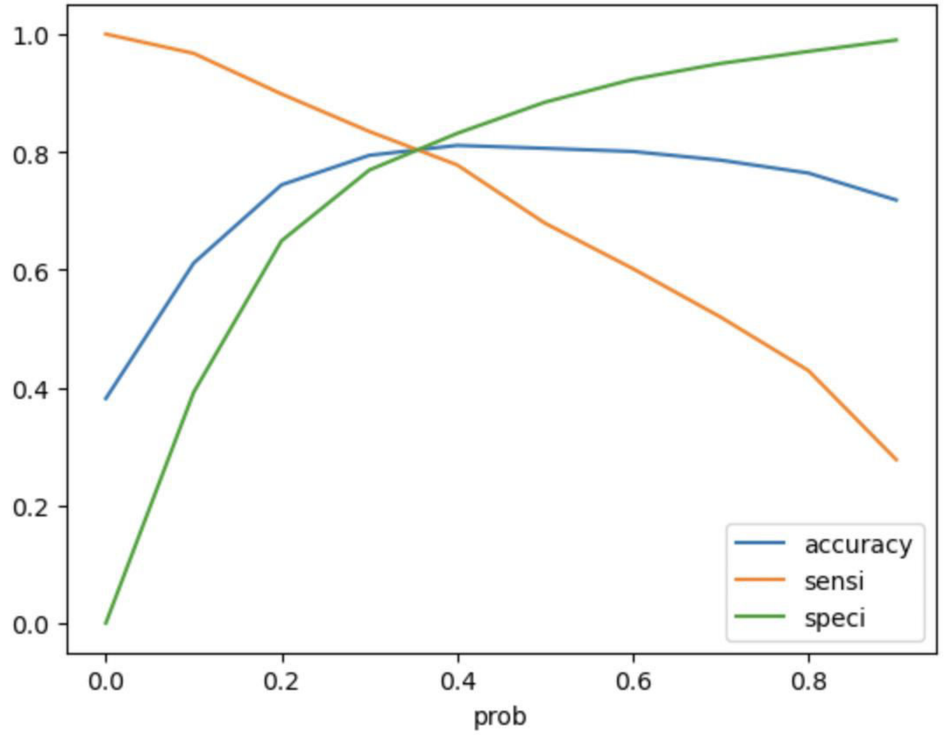


Caption

Plotting the accuracy, precision and recall that needs to be calculated using those formulas

Code Notebook Python (Pyodide)

[86]:
```
# Plotting it
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



[87]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.Conversion_Prob.map( lambda x: 1 if x > 0.35 else 0)
y_train_pred_final.head()
```

[87]: .....

Caption

Calculating the precision, recall, sensitivity and specificity using those formulas for precision

[100]:
```
# Substituting the value of true positive
TP = confusion2[1,1]
# Substituting the value of true negatives
TN = confusion2[0,0]
# Substituting the value of false positives
FP = confusion2[0,1]
# Substituting the value of false negatives
FN = confusion2[1,0]
```

[101]:
```
# Calculating the sensitivity
TP/(TP+FN)
```

[101]:
```
0.8127853881278538
```

[102]:
```
# Calculating the specificity
TN/(TN+FP)
```

[102]:
```
0.8151460942158617
```

[103]:
```
# With the current cut off as 0.35 we have accuracy, sensitivity and specificity of around 81%  .
```

precision recall : ¶

[104]:
```
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Predicted )
confusion
```

[104]:
```
array([[3539,  463],
,       [ 792, 1674]], dtype=int64)
```

[105]:
```
# Precision = TP / TP + FP
confusion[1,1]/(confusion[0,1]+confusion[1,1])
```

[105]:
```
0.7833411324286382
```

[106]:
```
#Recall = TP / TP + FN
confusion[1,1]/(confusion[1,0]+confusion[1,1])
```

Caption