LEAD SCORING CASE STUDY

Basic checks:

- We will check records and columns using command : df.shape
  There are 9240 records and 37 columns

- Checking the number of columns, its data type and number of NOT NULL values
  df.info()

- Checking percentage of NULL/missing values
  100*df.isnull().mean()

Data cleaning:

- Checking number of rows and columns
  df.shape

- Converting 'Select' values to NaN
  df = df.replace('Select', np.nan)

- List out the Unique values in the data set
  df.nunique()

```
Prospect ID                                    9240
Lead Number                                    9240
Lead Origin                                       5
Lead Source                                      21
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                                   18
How did you hear about X Education                9
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                                      5
City                                              6
Asymmetrique Activity Index                       3
Asymmetrique Profile Index                        3
Asymmetrique Activity Score                      12
Asymmetrique Profile Score                       10
I agree to pay the amount through cheque          1
A free copy of Mastering The Interview            2
Last Notable Activity                            16
dtype: int64
```

- Dropping the columns that have value as 1

  df.drop(["Magazine","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)

- Also the columns "Prospect ID" is continuous variables and can be dropped

  df.drop(["Prospect ID"], axis = 1, inplace = True)

- Drop the column tag as it is derived by sales team

  df.drop(["Tags"], axis = 1, inplace = True)

- making lead number as index

  df = df.set_index('Lead Number')

Missing values:

- Select percentage of missing values

  100*df.isnull().mean()

Lead Origin	0.000000
Lead Source	0.389610
Do Not Email	0.000000
Do Not Call	0.000000
Converted	0.000000
TotalVisits	1.482684
Total Time Spent on Website	0.000000
Page Views Per Visit	1.482684
Last Activity	1.114719
Country	26.634199
Specialization	36.580087
How did you hear about X Education	78.463203
What is your current occupation	29.112554
What matters most to you in choosing a course	29.318182
Search	0.000000
Newspaper Article	0.000000
X Education Forums	0.000000
Newspaper	0.000000
Digital Advertisement	0.000000
Through Recommendations	0.000000
Lead Quality	51.590909
Lead Profile	74.188312
City	39.707792
Asymmetrique Activity Index	45.649351
Asymmetrique Profile Index	45.649351
Asymmetrique Activity Score	45.649351
Asymmetrique Profile Score	45.649351
A free copy of Mastering The Interview	0.000000
Last Notable Activity	0.000000

dtype: float64

- Drop the columns that have more than 40% null/missing values

  df.drop(['How did you hear about X Education', 'Lead Quality', 'Lead Profile', 'Asymmetrique Activity Index', 'Asymmetrique Profile Index', 'Asymmetrique Activity Score', 'Asymmetrique Profile Score'],axis=1,inplace=True)

- Select percentage of missing values

  100*df.isnull().mean()

```
Lead Origin                                     0.000000
Lead Source                                     0.389610
Do Not Email                                    0.000000
Do Not Call                                     0.000000
Converted                                       0.000000
TotalVisits                                     1.482684
Total Time Spent on Website                     0.000000
Page Views Per Visit                            1.482684
Last Activity                                   1.114719
Country                                        26.634199
Specialization                                 36.580087
What is your current occupation                29.112554
What matters most to you in choosing a course  29.318182
Search                                          0.000000
Newspaper Article                               0.000000
X Education Forums                              0.000000
Newspaper                                       0.000000
Digital Advertisement                           0.000000
Through Recommendations                         0.000000
City                                           39.707792
A free copy of Mastering The Interview          0.000000
Last Notable Activity                           0.000000
dtype: float64
```

Categorical columns analysis:

- Checking different values of column Country
  ```
  df['Country'].value_counts()
  ```

- India is the most occuring country. So Null can be replaced with "India"
  ```
  df["Country"] = df["Country"].replace(np.nan,'India')
  ```

- Set value of Country as "Other Country" when country value is other than India
  ```
  df["Country"] = [ 'India' if val == 'India' else 'Outside India' for val in df["Country"] ]
  ```

- Check the percentage of Country values.
  ```
  100 * df['Country'].value_counts()/df['Country'].value_counts().sum()
  ```

- Number of rows having India is very high in country column, so dropping this column.
  ```
  df.drop(['Country'],axis=1,inplace=True)
  ```

- Checking percentage of NULL/missing values
  ```
  100*df.isnull().mean().sort_values(ascending = False)
  ```

- df['City'].value_counts()

```
Mumbai                        3222
Thane & Outskirts              752
Other Cities                   686
Other Cities of Maharashtra    457
Other Metro Cities             380
Tier II Cities                  74
Name: City, dtype: int64
```

- Mumbai is the most occuring city, we can replace missing values with Mumbai

  df['City']=df['City'].replace(np.nan,'Mumbai')

- df['City'].value_counts()

```
Mumbai                        6891
Thane & Outskirts              752
Other Cities                   686
Other Cities of Maharashtra    457
Other Metro Cities             380
Tier II Cities                  74
Name: City, dtype: int64
```

- Checking the values of 'Specialization' column

  df['Specialization'].value_counts()

```
Finance Management                  976
Human Resource Management           848
Marketing Management                838
Operations Management               503
Business Administration             403
IT Projects Management              366
Supply Chain Management             349
Banking, Investment And Insurance   338
Media and Advertising               203
Travel and Tourism                  203
International Business              178
Healthcare Management               159
Hospitality Management              114
E-COMMERCE                          112
Retail Management                   100
Rural and Agribusiness               73
E-Business                           57
Services Excellence                  40
Name: Specialization, dtype: int64
```

- As "Finance Management" is the most occuring value, replacing this with "Not Available"/missing values.

  df['Specialization']=df['Specialization'].replace(np.nan,'Finance Management')

- Checking the values of 'Specialization' column

  df['Specialization'].value_counts()

```
Finance Management                  4356
Human Resource Management            848
Marketing Management                 838
Operations Management                503
Business Administration              403
IT Projects Management               366
Supply Chain Management              349
Banking, Investment And Insurance    338
Media and Advertising                203
Travel and Tourism                   203
International Business               178
Healthcare Management                159
Hospitality Management               114
E-COMMERCE                           112
Retail Management                    100
Rural and Agribusiness                73
E-Business                            57
Services Excellence                   40
Name: Specialization, dtype: int64
```

- Check the values of column ''What matters most to you in choosing a course'

  df['What matters most to you in choosing a course'].value_counts()

```
Better Career Prospects      6528
Flexibility & Convenience       2
Other                           1
Name: What matters most to you in choosing a course, dtype: int64
```

- Drop the column "What matters most to you in choosing a course" as its single value "Better Career Prospects" occurance is very high

  df.drop('What matters most to you in choosing a course', axis = 1,inplace=True)

- Check the values of column 'What is your current occupation'

  df['What is your current occupation'].value_counts()

```
Unemployed              5600
Working Professional     706
Student                  210
Other                     16
Housewife                 10
Businessman                8
Name: What is your current occupation, dtype: int64
```

- As unemployed is the highest occuring value, replacing NULL values with this value

  df['What is your current occupation']=df['What is your current occupation'].replace(np.nan,'Unemployed')

- Check the values of column 'Last Activity'

  df['Last Activity'].value_counts()

```
Email Opened                    3437
SMS Sent                        2745
Olark Chat Conversation          973
Page Visited on Website          640
Converted to Lead                428
Email Bounced                    326
Email Link Clicked               267
Form Submitted on Website        116
Unreachable                       93
Unsubscribed                      61
Had a Phone Conversation          30
Approached upfront                 9
View in browser link Clicked       6
Email Received                     2
Email Marked Spam                  2
Resubscribed to emails             1
Visited Booth in Tradeshow         1
Name: Last Activity, dtype: int64
```

- As "Email Opened" is the highest occuring value, replacing NULL values with this value
  df['Last Activity']=df['Last Activity'].replace(np.nan,'Email Opened')

- Check the values of column 'Lead Source'
  df['Lead Source'].value_counts()

```
Google               2868
Direct Traffic       2543
Olark Chat           1755
Organic Search       1154
Reference             534
Welingak Website      142
Referral Sites        125
Facebook               55
bing                    6
google                  5
Click2call              4
Live Chat               2
Social Media            2
Press_Release           2
blog                    1
Pay per Click Ads       1
testone                 1
welearnblog_Home        1
WeLearn                 1
youtubechannel          1
NC_EDM                  1
Name: Lead Source, dtype: int64
```

- As "Google" is the highest occuring value, replacing NULL values with this value
  df['Lead Source']=df['Lead Source'].replace(np.nan,'Google')

- Also as Google and google are same, replacing google with Google
  df['Lead Source']=df['Lead Source'].replace('google','Google')

- Checking percentage of NULL/missing values

  100*df.isnull().mean().sort_values(ascending = False)

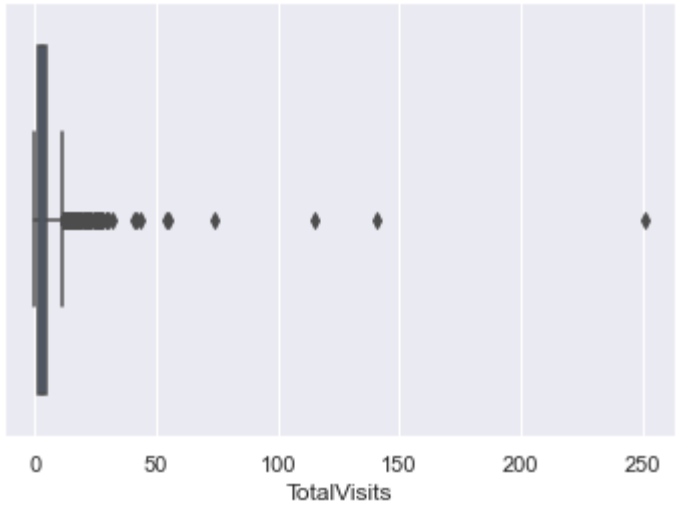
```
TotalVisits                                1.482684
Page Views Per Visit                       1.482684
Last Notable Activity                      0.000000
A free copy of Mastering The Interview     0.000000
Lead Source                                0.000000
Do Not Email                               0.000000
Do Not Call                                0.000000
Converted                                  0.000000
Total Time Spent on Website                0.000000
Last Activity                              0.000000
Specialization                             0.000000
What is your current occupation            0.000000
Search                                     0.000000
Newspaper Article                          0.000000
X Education Forums                         0.000000
Newspaper                                  0.000000
Digital Advertisement                      0.000000
Through Recommendations                    0.000000
City                                       0.000000
Lead Origin                                0.000000
dtype: float64
```

- Checking the stat for TotalVisits

  df['TotalVisits'].describe()

- Plotting box plot

  sns.boxplot(df['TotalVisits'])



- There is outlier in TotalVisits. So we need to use median to replace missing values.

  df['TotalVisits'] = df['TotalVisits'].fillna(df['TotalVisits'].median())

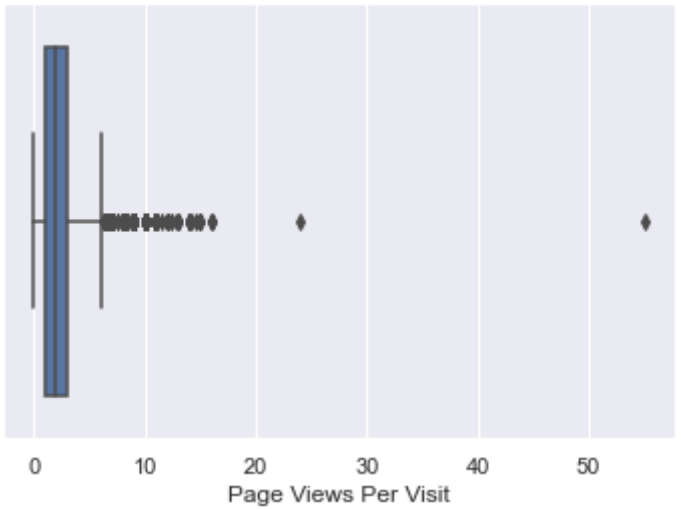
- Checking the stat for TotalVisits

  df['Page Views Per Visit'].describe()

```
count    9103.000000
mean        2.362820
std         2.161418
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        55.000000
Name: Page Views Per Visit, dtype: float64
```

- Plotting box plot

  sns.boxplot(df['Page Views Per Visit'])



- There is outlier in TotalVisits. So we need to use median to replace missing values.

```
df['Page Views Per Visit'] = df['Page Views Per Visit'].fillna(df['Page Views Per Visit'].median())
```

- Checking percentage of NULL/missing values

```
100*df.isnull().mean().sort_values(ascending = False)
```

EDA:

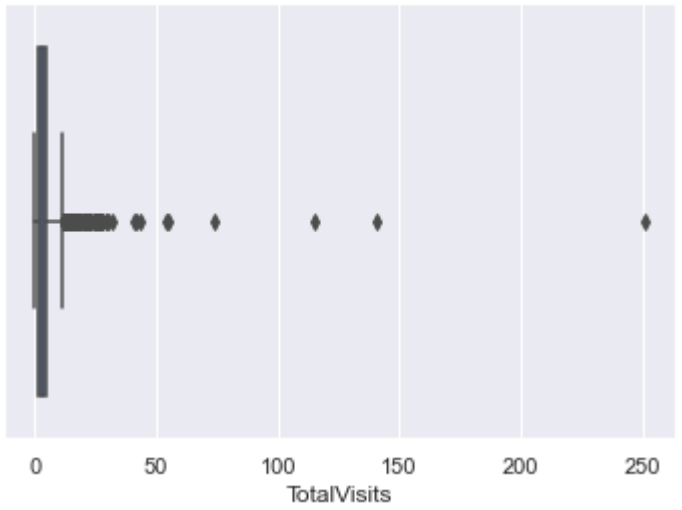
```
id_cols=["Lead Number"]
cont_cols=["TotalVisits","Total Time Spent on Website","Page Views Per Visit"]
cat_cols=["Converted","Lead Origin","Lead Source","Do Not Email","Do Not Call","Last Activity","Specialization",
      "What is your current occupation","Search","Newspaper Article","X Education Forums","Newspaper",
      "Digital Advertisement","Through Recommendations","City","A free copy of Mastering The Interview","Last Notable Activity"]
target=["Converted"]
```

Unvaried analysis:

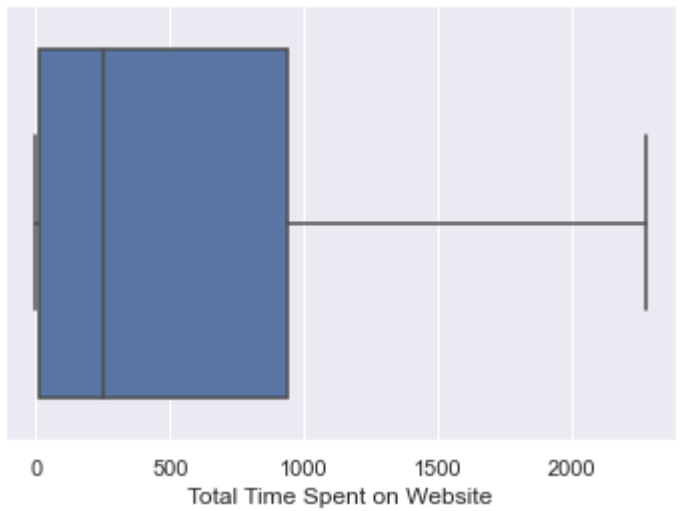
Boxplot for continuoua columns

```
for i in cont_cols:
    print(i)
    sns.boxplot(df[i])
    plt.show()
```

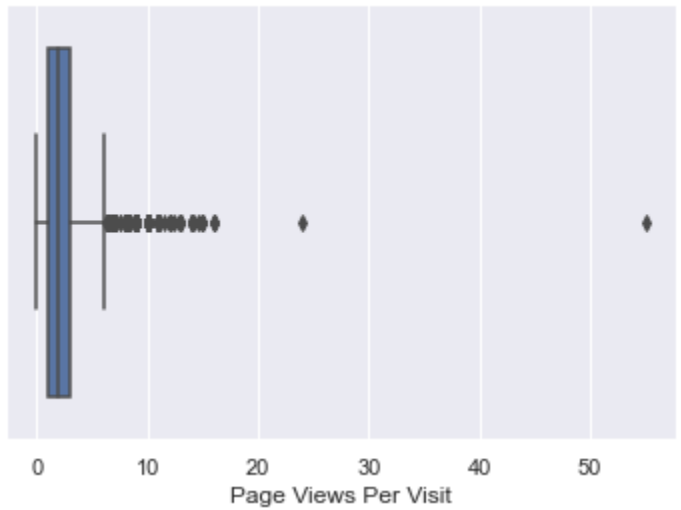
TotalVisits



Total Time Spent on Website



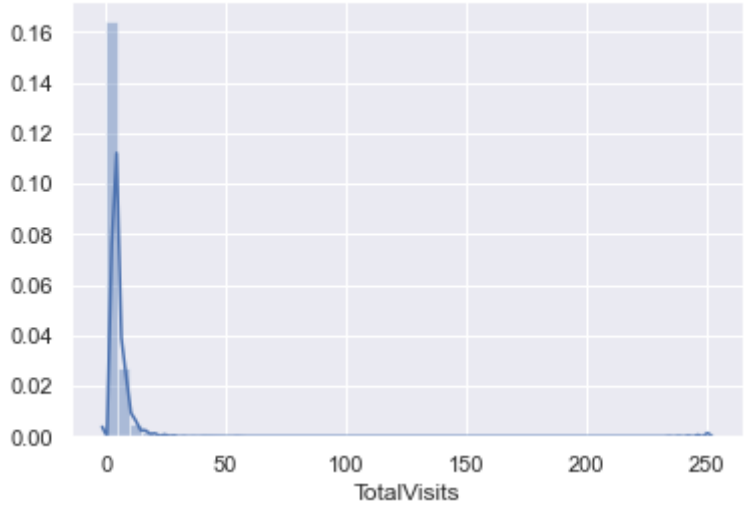
Page Views Per Visit



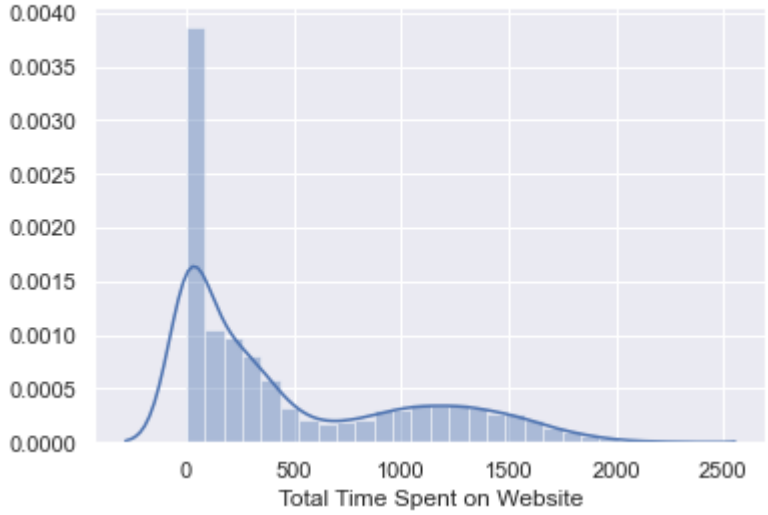
Distplot for continuos columns

```
for i in cont_cols:
    print(i)
    sns.distplot(df[i])
    plt.show()
```

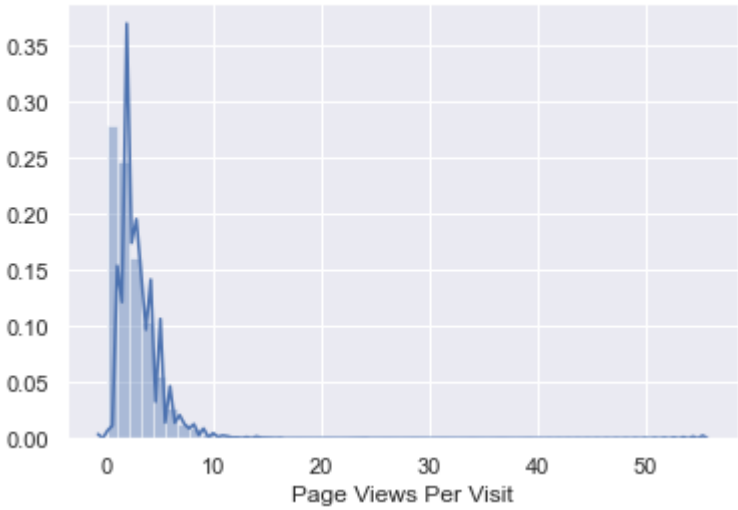
TotalVisits



Total Time Spent on Website

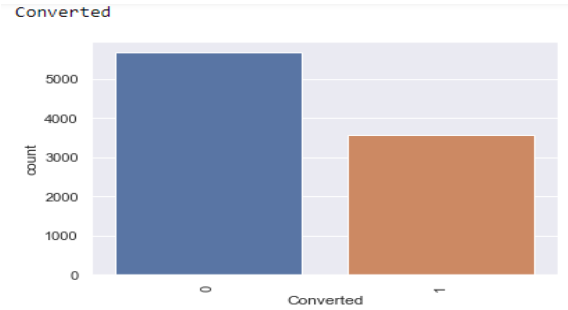


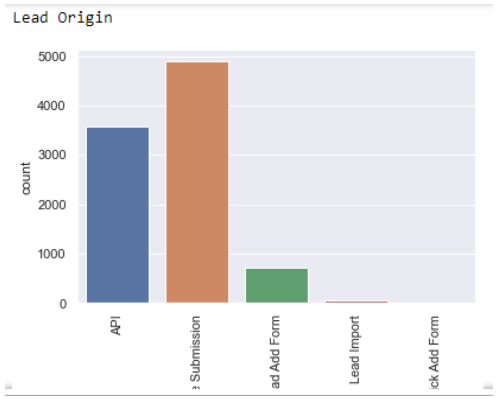
Page Views Per Visit

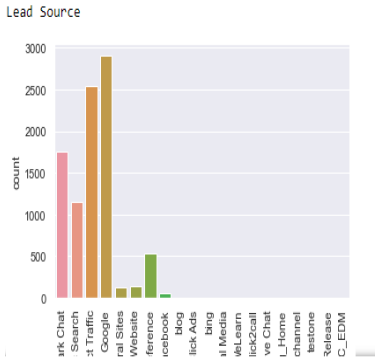


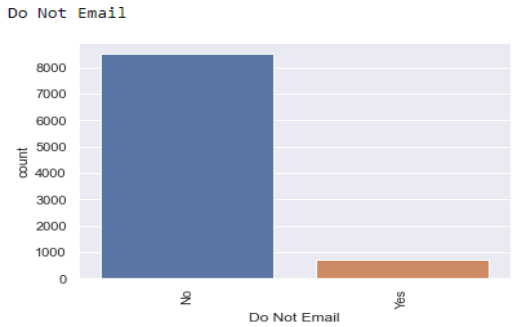
Count plot for categorical columns

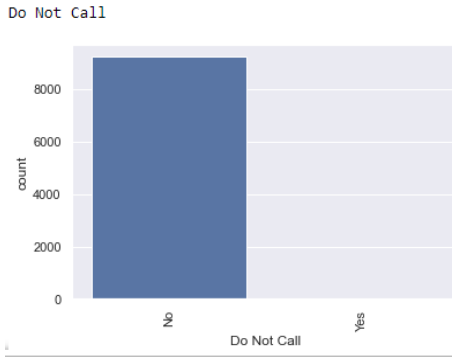
```
for i in cat_cols:
    print(i)
    sns.set()
    plt.subplots_adjust(wspace=.2,hspace=1 )
    sns.countplot(df[i]).tick_params(axis='x', rotation = 90)
    plt.show()
```

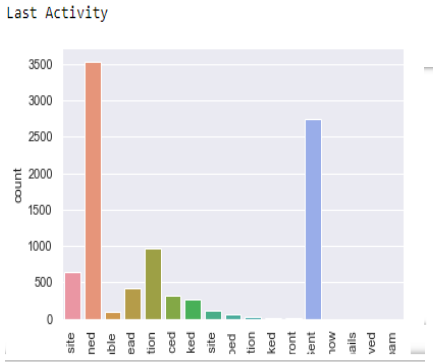


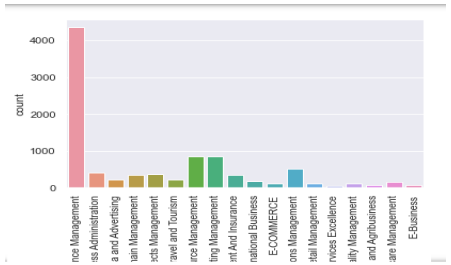


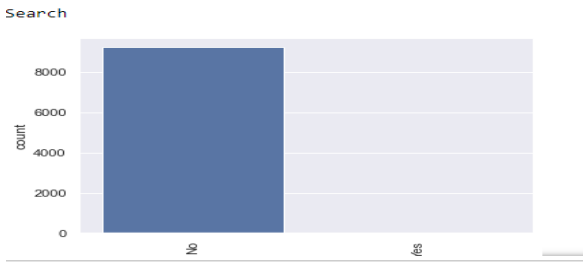


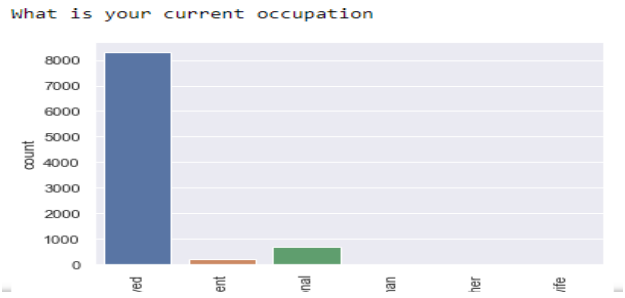








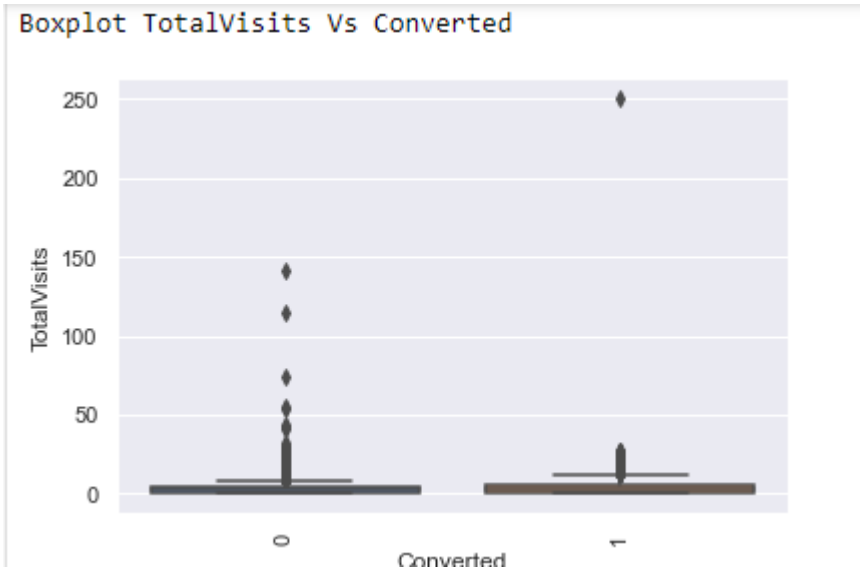




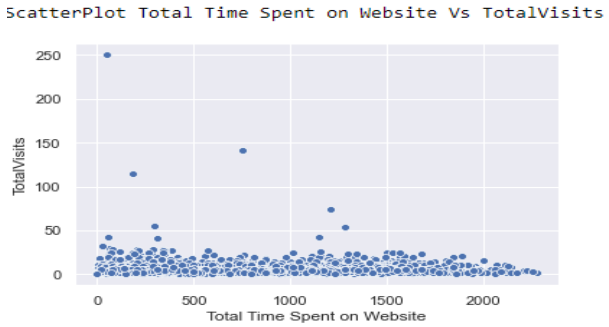
Bivarite analysis:

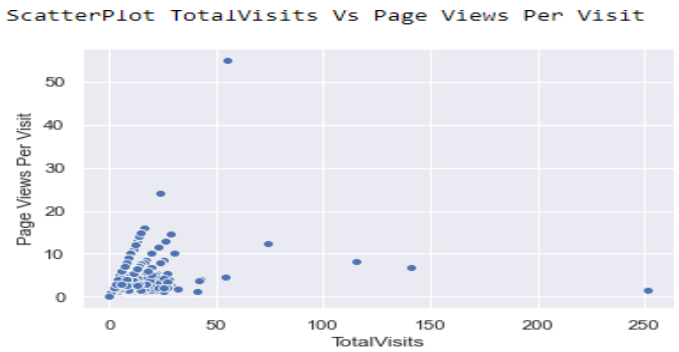
Box plot for continuous columns V/S categorical columns

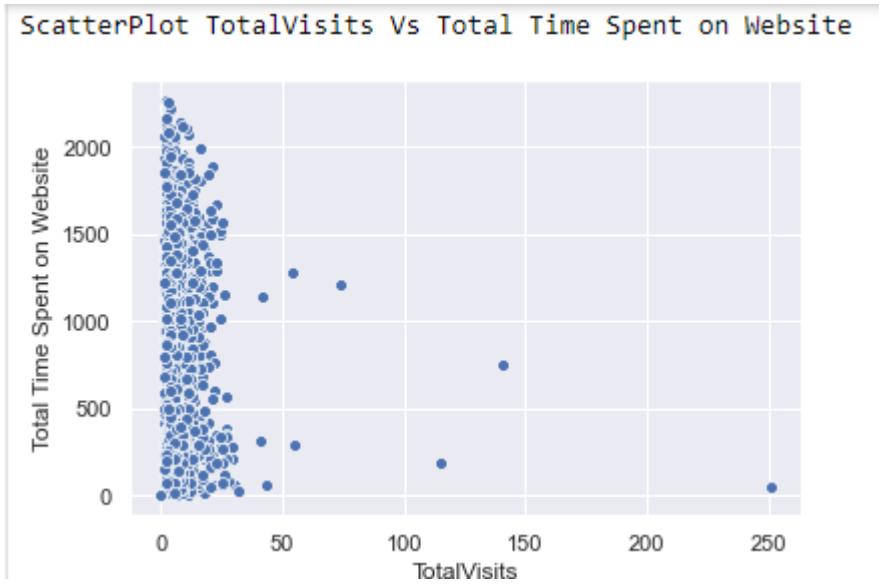
```
for i in cont_cols:
    for j in cat_cols:
        print("Boxplot",i,"Vs",j)
        sns.boxplot(df[j],df[i]).tick_params(axis='x', rotation = 90)
        plt.show()
```

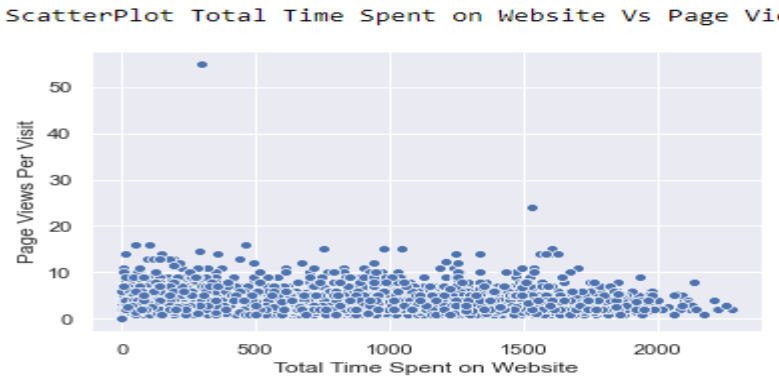


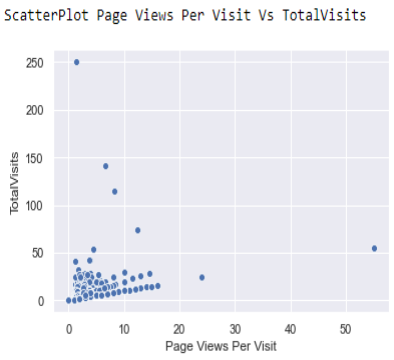
```
for i in cont_cols:
    for j in cont_cols:
        if i!=j:
            print("ScatterPlot",i,"Vs",j)
            sns.scatterplot(df[i],df[j])
            plt.show()
```

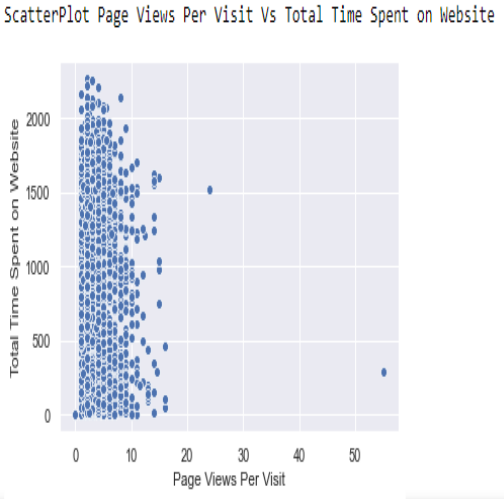






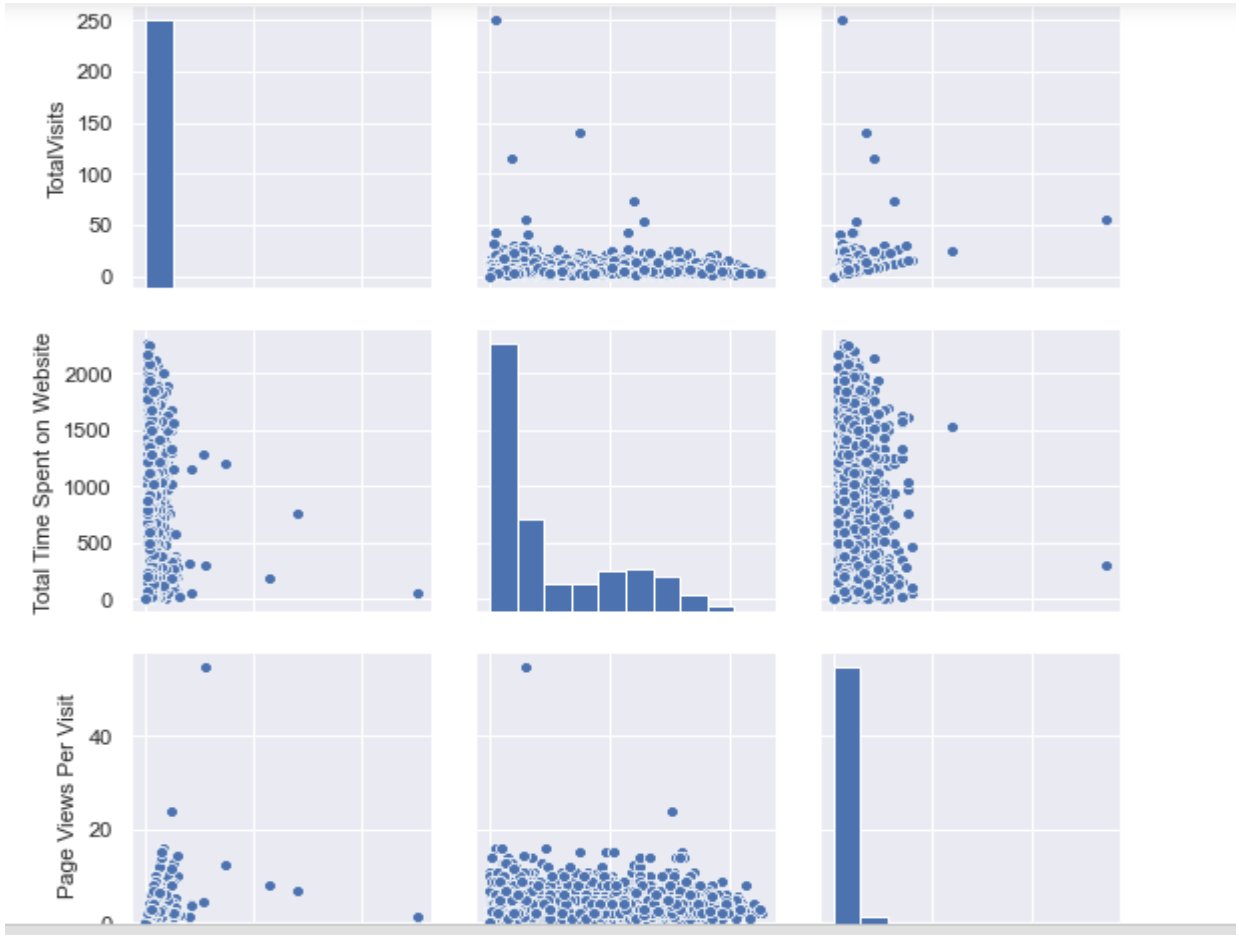






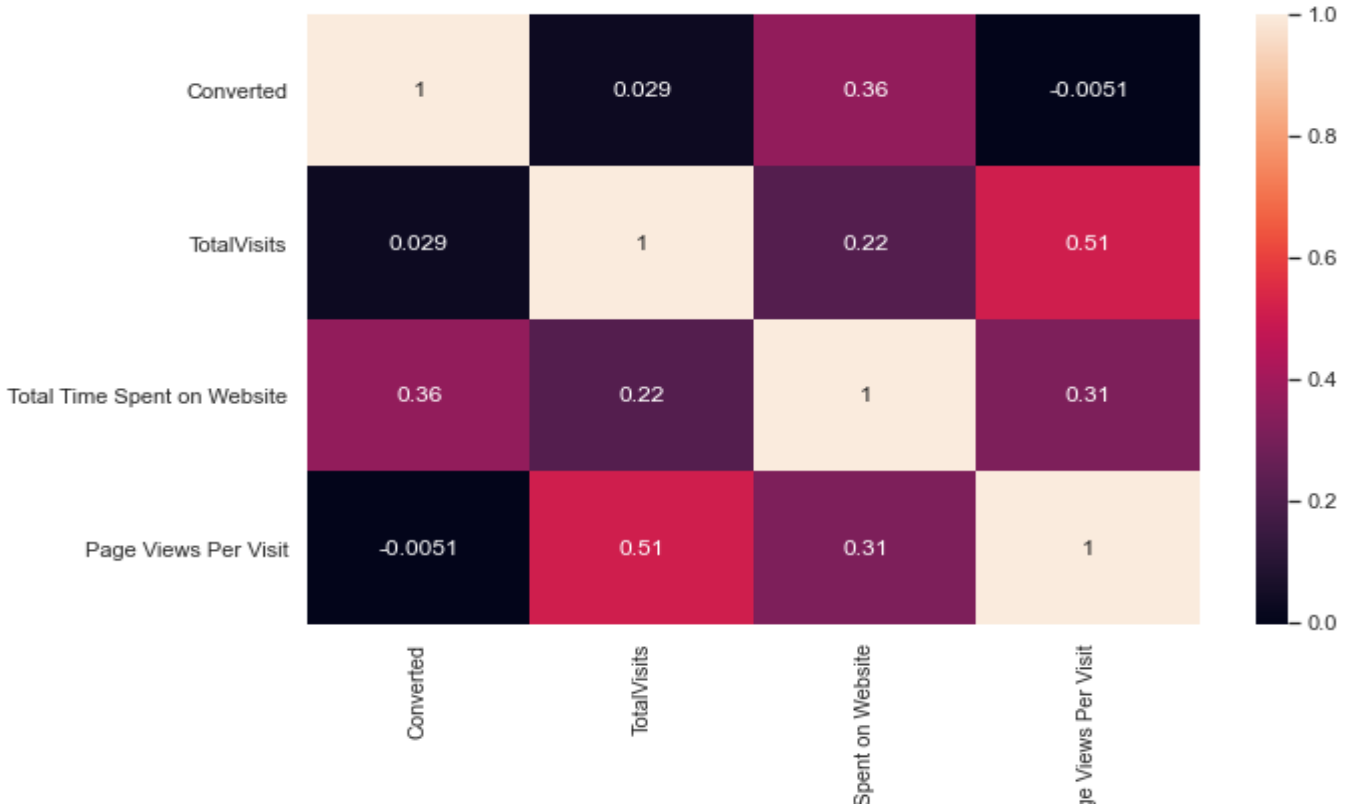
Multivariate analysis:

- sns.pairplot(df[cont_cols])



- Heat map

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
```



- As seen in the above graphs some columns are not having even data distribution, so dropping such columns.

```
df.drop(['What is your current occupation','Do Not Call','Do Not Email','Search','X Education Forums','Newspaper','Newspaper Article','Through Recommendations','Digital Advertisement'],axis = 1, inplace = True)
```

- There are some columns which have many categories, so we can categorize such less occuring categoroes as Other

first let us set Others for column 'Lead Source' whose occurance is less than 1000

```
less_occuring_category_labels_dict = dict(df['Lead Source'].value_counts() < 1000)
for key in less_occuring_category_labels_dict.keys():
    if less_occuring_category_labels_dict[key]==True:
        df['Lead Source'] = df['Lead Source'].replace(key,'Other')
```

- let us set Others for column 'Last Activity' whose occurance is less than 300

```
less_occuring_category_labels_dict = dict(df['Last Activity'].value_counts() < 300)
for key in less_occuring_category_labels_dict.keys():
    if less_occuring_category_labels_dict[key]==True:
        df['Last Activity'] = df['Last Activity'].replace(key,'Other')
```

- let us set Others for column 'Last Notable Activity' whose occurance is less than 500

```
less_occuring_category_labels_dict = dict(df['Last Notable Activity'].value_counts() < 500)
for key in less_occuring_category_labels_dict.keys():
    if less_occuring_category_labels_dict[key]==True:
        df['Last Notable Activity'] = df['Last Notable Activity'].replace(key,'Other')
```

- let us set Others for column 'Specialization' whose occurance is less than 350

```
less_occuring_category_labels_dict = dict(df['Specialization'].value_counts() < 350)
for key in less_occuring_category_labels_dict.keys():
    if less_occuring_category_labels_dict[key]==True:
        df['Specialization'] = df['Specialization'].replace(key,'Other')
```

Data preparation and features selection:

- Converting YS/NO vaues to 1/0 for column "A free copy of Mastering The Interview"

```
df['A free copy of Mastering The Interview'] = df['A free copy of Mastering The Interview'].apply(lambda x: 1 if x=='Yes' else 0)
```

- As columns "Last Notable Activity" and "Last Activity" have same values, we will drop the column "Last Notable Activity"

```
df.drop('Last Notable Activity', axis = 1, inplace = True)
```

```
dum_cols=["Lead Origin", "Lead Source", "Last Activity", "Specialization", "City"]
dum = pd.get_dummies(df[dum_cols],drop_first=True)
df.drop(dum_cols, axis = 1, inplace = True)
df = pd.concat([df,dum],axis = 1)
```

Splitting data into train and test data sets:

```
X = df.drop("Converted",axis=1)
y = df[["Converted"]]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

Scaling the columns

```
scaler = StandardScaler()
X_train[['Total Time Spent on Website','Page Views Per Visit','TotalVisits']]  = scaler.fit_transform(X_train[['Total Time Spent on Website','Page Views Per Visit','TotalVisits']])
```

Feature selection:

```
estimator = LogisticRegression()
selector = RFE(estimator,n_features_to_select = 15)
selector = selector.fit(X_train,y_train)
selector.support_
```

```
cols_to_keep = X_train.columns[selector.support_]
```

```
# Below columns should be used for modeling
cols_to_keep
```

```
Index(['Total Time Spent on Website', 'Lead Origin_Landing Page Submission',
       'Lead Origin_Lead Add Form', 'Lead Origin_Lead Import',
       'Lead Origin_Quick Add Form', 'Lead Source_Google',
       'Lead Source_Olark Chat', 'Lead Source_Other',
       'Last Activity_Email Bounced', 'Last Activity_Email Opened',
       'Last Activity_Olark Chat Conversation', 'Last Activity_Other',
       'Last Activity_Page Visited on Website', 'Last Activity_SMS Sent',
       'Specialization_Finance Management'],
      dtype='object')
```

```
# Below columns should not be used for modeling
X_train.columns[~selector.support_]
```

```
Index(['TotalVisits', 'Page Views Per Visit',
       'A free copy of Mastering The Interview', 'Lead Source_Organic Search',
       'Specialization_Human Resource Management',
       'Specialization_IT Projects Management',
       'Specialization_Marketing Management',
       'Specialization_Operations Management', 'Specialization_Other',
       'City_Other Cities', 'City_Other Cities of Maharashtra',
       'City_Other Metro Cities', 'City_Thane & Outskirts',
       'City_Tier II Cities'],
      dtype='object')
```

Model building:

Model 1:

```
X_train_sm = sm.add_constant(X_train,has_constant="add")
model1 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
result1 = model1.fit()
result1.summary()
```

Calculate the VIFs for the new model

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

	Features	VIF
7	Lead Source_Other	5.05
2	Lead Origin_Lead Add Form	4.65
9	Last Activity_Email Opened	4.04
1	Lead Origin_Landing Page Submission	3.99
13	Last Activity_SMS Sent	3.46
6	Lead Source_Olark Chat	2.79
14	Specialization_Finance Management	2.67
10	Last Activity_Olark Chat Conversation	2.16
5	Lead Source_Google	1.87
11	Last Activity_Other	1.53
12	Last Activity_Page Visited on Website	1.52
8	Last Activity_Email Bounced	1.29
3	Lead Origin_Lead Import	1.26
0	Total Time Spent on Website	1.24
4	Lead Origin_Quick Add Form	1.01

Model 2:

```
model2 = sm.GLM(y_train ,X_train_sm, family = sm.families.Binomial())
result2 = model2.fit()
result2.summary()
```

Calculate the VIFs for the new model

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

	Features	VIF
0	const	31.15
9	Last Activity_Email Opened	5.81
13	Last Activity_SMS Sent	5.35
7	Lead Source_Other	4.64
3	Lead Origin_Lead Add Form	4.29
10	Last Activity_Olark Chat Conversation	3.25
2	Lead Origin_Landing Page Submission	2.53
6	Lead Source_Olark Chat	2.41
12	Last Activity_Page Visited on Website	2.35
11	Last Activity_Other	2.24
8	Last Activity_Email Bounced	1.71
14	Specialization_Finance Management	1.64
5	Lead Source_Google	1.36
4	Lead Origin_Lead Import	1.26
1	Total Time Spent on Website	1.24

Model 3:

```
X_train_sm = X_train_sm.drop(["Lead Origin_Lead Import"],axis=1)
model3 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
result3 = model3.fit()
result3.summary()
```

Model 4:

```
model4 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
result4 = model4.fit()
result4.summary()
```

Model 5:

```
model5 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
result5 = model5.fit()
result5.summary()
```

Model evaluation:

```
y_train_pred = result5.predict(X_train_sm)
y_train_pred_final = pd.DataFrame(y_train_pred,columns=["Converted_prob"])
y_train_pred_final["Converted"] = y_train["Converted"]
y_train_pred_final['Lead Number'] = y_train.index
y_train_pred_final.reset_index(drop=True, inplace=True)
y_train_pred_final["Converted_class"]=np.where(y_train_pred_final["Converted_prob"] > 0.5,1,0)
y_train_pred_final = y_train_pred_final.dropna()
```

```
y_train_pred_final
```

	Converted_prob	Converted	Lead Number	Converted_class
0	0.248574	1	593802	0
1	0.310815	0	600305	0
2	0.285664	1	589724	0
3	0.913521	0	616844	1
4	0.072832	0	585361	0
...	...	...	...	...
7387	0.917697	1	656685	1
7388	0.974187	1	659710	1
7389	0.248574	1	588165	0
7390	0.054280	0	596447	0
7391	0.248574	0	606685	0

7392 rows × 4 columns

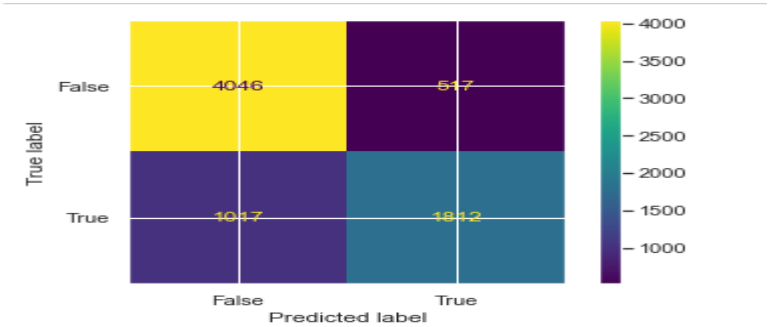
Finding the metrics like accuracy, sensitivity and specicity

```
def metrices_(converted,predicted):
    cm1 = metrics.confusion_matrix(converted,predicted)
    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])
    return accuracy,sensi,speci
```

```
acc,sensi,speci=metrices_(y_train_pred_final.Converted,y_train_pred_final.Converted_class)
print('Accuracy: {}, Sensitivity {}, specitiy {}'.format(acc,sensi,speci))
```

Finding the Confusion Matrix

```
confusion_matrix = metrics.confusion_matrix(y_train_pred_final["Converted"],y_train_pred_final["Converted_class"])
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```



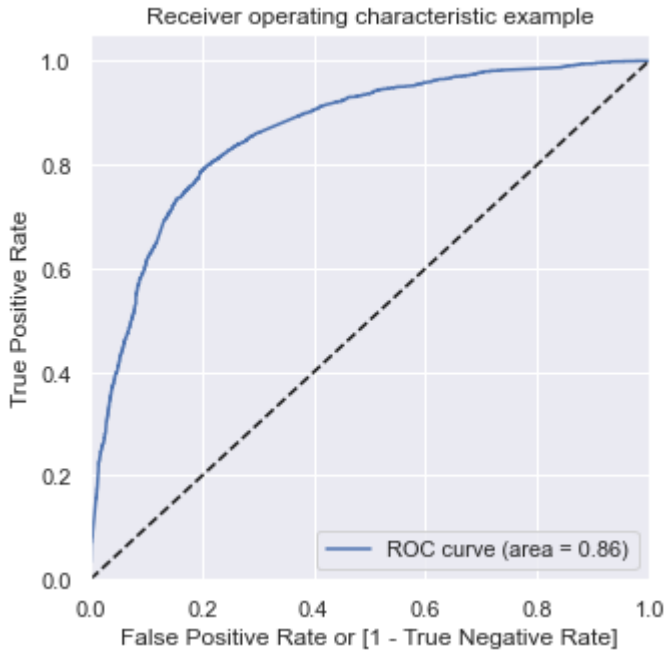
```
Finding the Confusion Matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Converted_class )
Confusion

Finding the TP, TN, FP, FN
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negative

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
```

```
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Converted, y_train_pred_final.Converted_prob,
drop_intermediate = False )

draw_roc(y_train_pred_final.Converted, y_train_pred_final.Converted_prob)
```




```
Finding the optimal Point
# columns with different probability cutoffs
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    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()
```

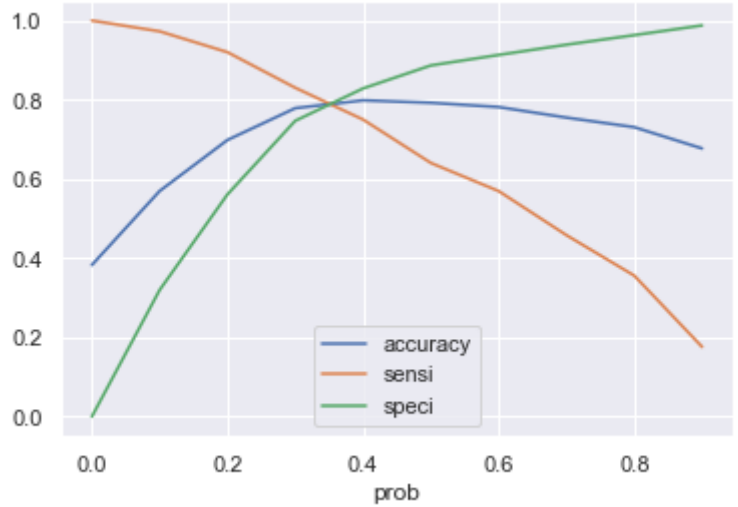
calculate accuracy sensitivity and specificity for various probability cutoffs

```
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix
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 = metrics.confusion_matrix(y_train_pred_final.Converted, train[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)
```

plot accuracy sensitivity and specificity for various probabilities.

```
plt.figure(figsize=(20,15))
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])

plt.show()
```



Metrics Accuracy, Sensitivity, Specicity
acc,sensi,speci=metrices_(y_train_pred_final.Converted, y_train_pred_final.final_predicted)
print('Accuracy: {}, Sensitivity {}, specifitiy {} '.format(acc,sensi,speci))

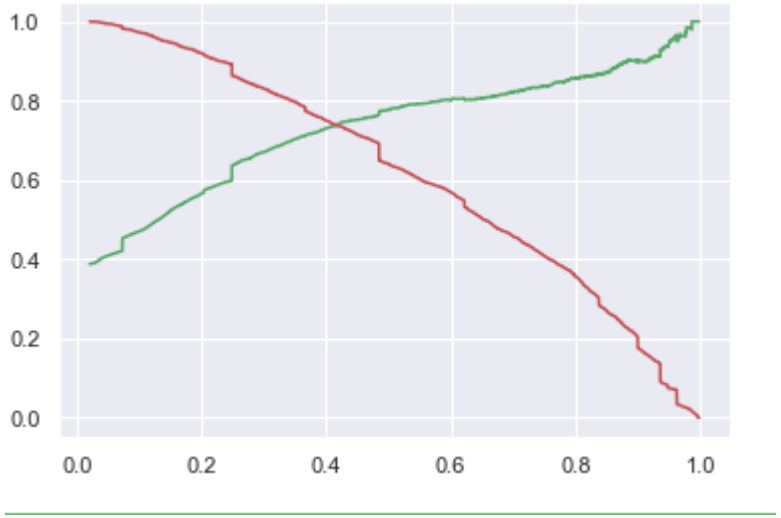
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Converted_class)
Confusion

Finding the Precision Score
precision_score(y_train_pred_final.Converted, y_train_pred_final.final_predicted)

Finding the Recall Score
recall_score(y_train_pred_final.Converted, y_train_pred_final.final_predicted)

plt.plot(thresholds, p[:-1], "g-",label='Precision')
plt.plot(thresholds, r[:-1], "r-",label='Recall')

plt.show()



Making prediction on test data:

Selecting the variables that were part of final model.

```
col1 = X_train_sm.columns

# Adding constant variable to test dataframe
X_test_lm = sm.add_constant(X_test)

X_test_lm = X_test_lm[col1]
X_test_lm.info()



Converting y_pred to a dataframe which is an array
y_pred_test = pd.DataFrame(y_test_pred)
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
y_test_df['Lead Number'] = y_test_df.index

y_pred_final['final_test_predicted'] = y_pred_final.Converted_prob.map(lambda x: 1 if x > 0.35 else 0)
```

Assigning lead score in the data frame with respect to lead number:

```
making new df with lead score
lead_scorer_df=pd.DataFrame()
df1=y_train_pred_final[['Lead Number','Converted_prob']]
df2=y_pred_final[['Lead Number','Converted_prob']]

Assigning the value to the Lead Score by Multiplying 100
lead_scorer_df['Lead Score']=lead_scorer_df['Converted_prob'].apply(lambda x: round((x*100),2))


dropping Converted_Prob
lead_scorer_df.drop('Converted_prob',1,inplace=True)
lead_scorer_df.head()
```

Conclusion:

Below are the variables that contribute most in the probability of a lead getting converted
Lead Origin_Lead Add FormLast Activity_SMS Sent
Last Activity_Email Opened
Last Activity_OtherTotal Time Spent on Website
Lead Source_Olark Chat
Last Activity_Page Visited on Website
Lead Source_Google