LEAD SCORING CASE STUDY

Submitted By

Ms.Pragati Saxena

IMPORTING AND CHECKING OF DATA

```
# Importing dataset
lead = pd.read_csv("C:\\Users\\praga\\OneDrive\\Desktop\\logistic regression\\Leads.csv")
lead.head()
```

	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	...	Get updates on DM Content	Lead Profile	City	Asymmetrique Activity Index	Asymmetriqu Profile Inde
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	0	0.0	...	No	Select	Select	02.Medium	02.Mediu
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	674	2.5	...	No	Select	Select	02.Medium	02.Mediu
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	1532	2.0	...	No	Potential Lead	Mumbai	02.Medium	01.Hig
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	305	1.0	...	No	Select	Mumbai	02.Medium	01.Hig
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.0	1428	1.0	...	No	Select	Mumbai	02.Medium	01.Hig

5 rows × 37 columns

Step 2: Inspecting the Dataframe

```
# Let's check the dimensions of the dataf
lead.shape
```

```
(9240, 37)
```

```
# let's look at the statistical aspects of the dataframe
lead.describe()
```

WORKING WITH NULL VARIABLES

DELETING UNIQUE VARIABLES AND REPLACING 'SELECT' BY NULL.

```
#dropping Lead Number and Prospect ID since they have all unique values

lead.drop(['Prospect ID', 'Lead Number'], 1, inplace = True)
```

```
#Converting 'Select' values to NaN.

lead = lead.replace('Select', np.nan)
```

```
#checking null values in each rows

lead.isnull().sum()
```

...

```
#checking percentage of null values in each column

round(100*(lead.isnull().sum()/len(lead.index)), 2)
```

...

```
#dropping cols with more than 45% missing values

cols=lead.columns

for i in cols:
    if((100*(lead[i].isnull().sum()/len(lead.index))) >= 45):
        lead.drop(i, 1, inplace = True)
```

```
lead.info()
```

...

```
#checking null values percentage

round(100*(lead.isnull().sum()/len(lead.index)), 2)
```

...

WORKING WITH CATEGORICAL VARIABLES

Categorical variable having high percentge of null values

```
##FIRST VARIABLE : COUNTRY

#checking value counts of Country column

lead['Country'].value_counts(dropna=False)
```

...

```
# Since India is the most common occurence among the non-missing values we can impute all missing values with India

lead['Country'] = lead['Country'].replace(np.nan,'India')
```

```
lead['Country'].value_counts(dropna=False)
```

...

```
##Since India is the pnly biggest country we are dropping the col.
```

```
#creating a list of columns to be droppped

cols_to_drop=['Country']
```

CATEGORICAL VARIABLES:CITY AND SPECIALIZATION

```
#checking value counts of "City" column

lead['City'].value_counts(dropna=False)
```

...

```
#replacing null values with mumbai

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

```
lead['City'].value_counts(dropna=False)
```

...

```
## 3RD VARIABLE : Specialization

#checking value counts of Specialization column

lead['Specialization'].value_counts(dropna=False)
```

...

```
##converting null values by 'not specified'

lead['Specialization'] = lead['Specialization'].replace(np.nan, 'Not Specified')
```

```
#combining Management Specializations because they show similar trends

lead['Specialization'] = lead['Specialization'].replace(['Finance Management','Human Resource Management',
                                                          'Marketing Management','Operations Management',
                                                          'IT Projects Management','Supply Chain Management',
                                                   'Healthcare Management','Hospitality Management',
                                                          'Retail Management'] ,'Management_Specializations')
```

```
lead['Specialization'].value_counts(dropna=False)
```

...

CATEGORICAL VARIABLES:CITY AND SPECIALIZATION

```
#checking value counts of "City" column

lead['City'].value_counts(dropna=False)
```

...

```
#replacing null values with mumbai

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

```
lead['City'].value_counts(dropna=False)
```

...

```
## 3RD VARIABLE : Specialization

#checking value counts of Specialization column

lead['Specialization'].value_counts(dropna=False)
```

...

```
##converting null values by 'not specified'

lead['Specialization'] = lead['Specialization'].replace(np.nan, 'Not Specified')
```

```
#combining Management Specializations because they show similar trends

lead['Specialization'] = lead['Specialization'].replace(['Finance Management','Human Resource Management',
                                                           'Marketing Management','Operations Management',
                                                           'IT Projects Management','Supply Chain Management',
                                                    'Healthcare Management','Hospitality Management',
                                                           'Retail Management'] ,'Management_Specializations')
```

```
lead['Specialization'].value_counts(dropna=False)
```

...

CATEGORICAL VARIABLES:'What is your current occupation and 'What matters most to you in choosing a course'

```
#4TH VARIABLE : What is your current occupation

lead['What is your current occupation'].value_counts(dropna=False)
```

...

```
#imputing Nan values with mode "Unemployed"

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

```
#checking count of values
lead['What is your current occupation'].value_counts(dropna=False)
```

...

```
#5TH VARIABLE : What matters most to you in choosing a course

#checking value counts

lead['What matters most to you in choosing a course'].value_counts(dropna=False)
```

...

```
#replacing Nan values with Mode "Better Career Prospects"

lead['What matters most to you in choosing a course'] = lead['What matters most to you in choosing a course'].replace(np.nan,'Bet
```

```
#checking value counts of variable
lead['What matters most to you in choosing a course'].value_counts(dropna=False)
```

...

```
#Here again we have another Column that is worth Dropping. So we Append to the cols_to_drop List
cols_to_drop.append('What matters most to you in choosing a course')
cols_to_drop
```

...

CATEGORICAL VARIABLES:TAGS

```
#6th VARIABLE : TAGS

#checking value counts of Tag variable
lead['Tags'].value_counts(dropna=False)
```

...

```
#replacing Nan values with "Not Specified"
lead['Tags'] = lead['Tags'].replace(np.nan,'Not Specified')
```

```
#replacing tags with low frequency with "Other Tags"
lead['Tags'] = lead['Tags'].replace(['In confusion whether part time or DLP', 'in touch with EINS','Diploma holder (Not Eligible)
                                     'Approached upfront','Graduation in progress','number not provided', 'opp hangup','Still Thi
                                    'Lost to Others','Shall take in the next coming month','Lateral student','Interested in Next
                                    'Recognition issue (DEC approval)','Want to take admission but has financial problems',
                                    'University not recognized'], 'Other_Tags')

lead['Tags'] = lead['Tags'].replace(['switched off',
                                      'Already a student',
                                       'Not doing further education',
                                       'invalid number',
                                       'wrong number given',
                                       'Interested  in full time MBA'] , 'Other_Tags')
```

```
#checking value counts of Tag variable
lead['Tags'].value_counts(dropna=False)
```

CATEGORICAL VARIABLE: LEAD SOURCE

```
#7TH VARIABLE : Lead Source
```

```
#checking value counts of Lead Source column

lead['Lead Source'].value_counts(dropna=False)
```

```
...
```

```
#replacing Nan Values and combining low frequency values
lead['Lead Source'] = lead['Lead Source'].replace(np.nan,'Others')
lead['Lead Source'] = lead['Lead Source'].replace('google','Google')
lead['Lead Source'] = lead['Lead Source'].replace('Facebook','Social Media')
lead['Lead Source'] = lead['Lead Source'].replace(['bing','Click2call','Press_Release',
                                                    'youtubechannel','welearnblog_Home',
                                                    'WeLearn','blog','Pay per Click Ads',
                                                   'testone','NC_EDM'] ,'Others')
```

Result for Lead Source Variable

Maximum number of leads are generated by Google and Direct traffic. Conversion Rate of reference leads and leads through welingak website is high. To improve overall lead conversion rate, focus should be on improving lead converion of olark chat, organic search, direct traffic, and google leads and generate more leads from reference and welingak website.

CATEGORICAL VARIABLE: LAST ACTIVITY AND LAST NOTABLE ACTIVITY

```
# VARIABLE 8 : Last Activity

lead['Last Activity'].value_counts(dropna=False)
```

...

```
#replacing Nan Values and combining low frequency values

lead['Last Activity'] = lead['Last Activity'].replace(np.nan,'Others')
lead['Last Activity'] = lead['Last Activity'].replace(['Unreachable','Unsubscribed',
                                                        'Had a Phone Conversation',
                                                        'Approached upfront',
                                                        'View in browser link Clicked',
                                                        'Email Marked Spam',
                                                        'Email Received','Resubscribed to emails',
                                                         'Visited Booth in Tradeshow'],'Others')
```

```
lead['Last Activity'].value_counts(dropna=False)
```

...

```
#Check the Null Values in All Columns:
round(100*(lead.isnull().sum()/len(lead.index)), 2)
```

...

```
#Drop all rows which have Nan Values. Since the number of Dropped rows is less than 2%, it will not affect the model
lead = lead.dropna()
```

```
#Checking percentage of Null Values in All Columns:
round(100*(lead.isnull().sum()/len(lead.index)), 2)
```

...

```
#VARIABLE 8: Lead Origin
lead['Lead Origin'].value_counts(dropna=False)
```

...

```
#VARIABLE 9: LAST NOTABLE ACTIVITY
```

COLUMNS TO BE DROPPED

```python
#list of columns to be dropped
cols_to_drop
```

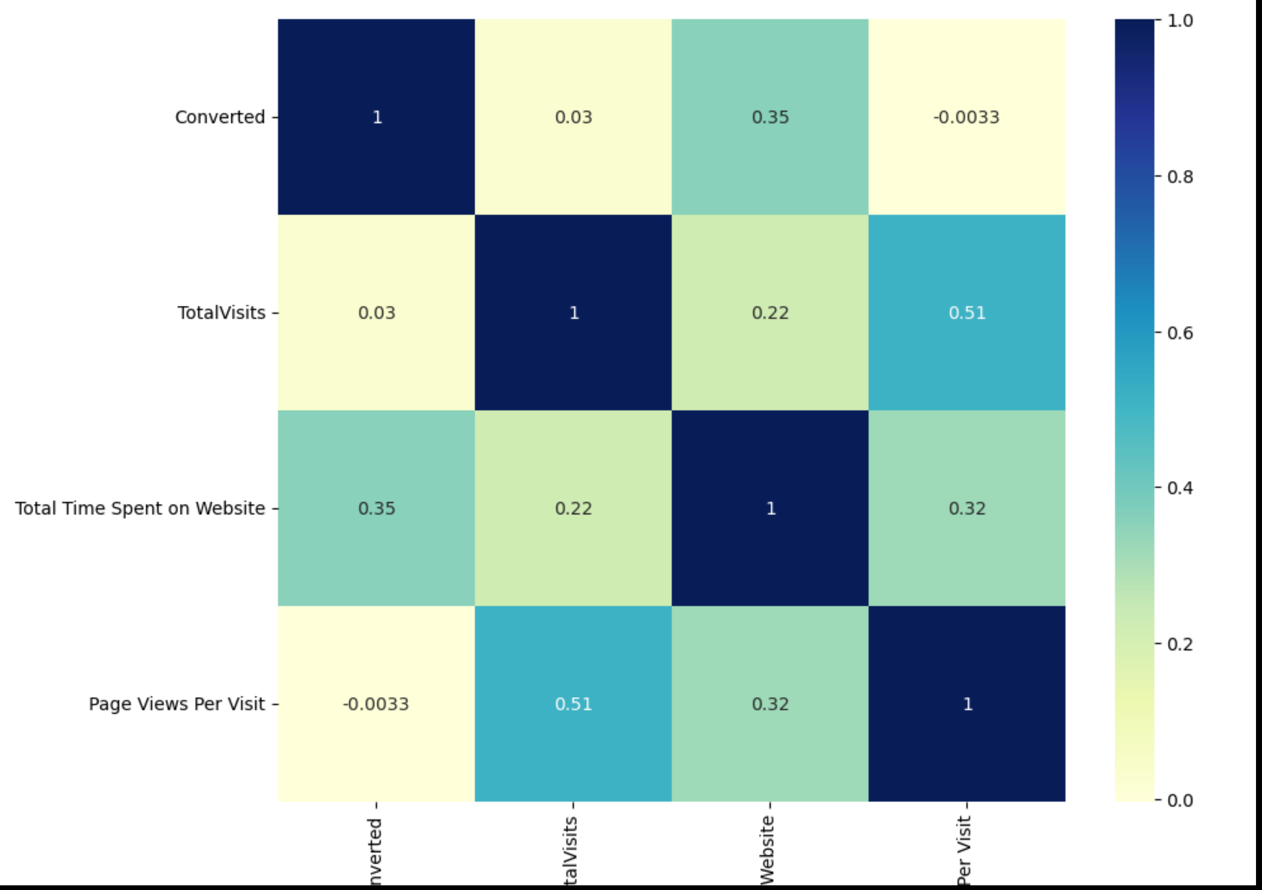
```
['Country', 'What matters most to you in choosing a course']
```

```python
#adding imbalanced columns to the list of columns to be dropped

cols_to_drop.extend(['Search','Magazine','Newspaper Article','X Education Forums','Newspaper',
                 'Digital Advertisement','Through Recommendations','Do Not Call','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'])
```

```python
#dropping columns
lead = lead.drop(cols_to_drop,1)
lead.info()
```

HEAT MAP: FIND CORRELATION BETWEEN VARIABLES

```
#Checking correlations of numeric values
# figure size
plt.figure(figsize=(10,8))

# heatmap
sns.heatmap(lead.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



REPLACING YES AND NO VALUES BY BINARY VALUES ‘0’ AND ‘1’

```python
# List of variables to map

varlist =  ['A free copy of Mastering The Interview','Do Not Email']

# Defining the map function
def binary_map(x):
    return x.map({'Yes': 1, "No": 0})

# Applying the function to the housing list
lead[varlist] = lead[varlist].apply(binary_map)
```

```python
#getting dummies and dropping the first column and adding the results to the master dataframe
dummy = pd.get_dummies(lead[['Lead Origin','What is your current occupation',
                             'City']], drop_first=True)

leads = pd.concat([lead,dummy],1)
```

```python
dummy = pd.get_dummies(leads['Specialization'], prefix  = 'Specialization')
dummy = dummy.drop(['Specialization_Not Specified'], 1)
lead = pd.concat([lead, dummy], axis = 1)
```

```python
dummy = pd.get_dummies(leads['Lead Source'], prefix  = 'Lead Source')
dummy = dummy.drop(['Lead Source_Others'], 1)
lead = pd.concat([lead, dummy], axis = 1)
```

```python
dummy = pd.get_dummies(leads['Last Activity'], prefix  = 'Last Activity')
dummy = dummy.drop(['Last Activity_Others'], 1)
lead = pd.concat([lead, dummy], axis = 1)
```

```python
dummy = pd.get_dummies(leads['Last Notable Activity'], prefix  = 'Last Notable Activity')
dummy = dummy.drop(['Last Notable Activity_Other_Notable_activity'], 1)
lead = pd.concat([lead, dummy], axis = 1)
```

```python
dummy = pd.get_dummies(leads['Tags'], prefix  = 'Tags')
dummy = dummy.drop(['Tags_Not Specified'], 1)
lead = pd.concat([lead, dummy], axis = 1)
```

SPLITTING DATA INTO TRAIN AND TEST

```
from sklearn.model_selection import train_test_split

# Putting response variable to y
y = lead['Converted']

y.head()

X=lead.drop('Converted', axis=1)
```

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

```
X_train.info()
```

SCALING OF DATA

```
#scaling numeric columns

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

num_cols=X_train.select_dtypes(include=['float64', 'int64']).columns

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])

X_train.head()
```

MODEL BUILDNG

MODEL BUILDING USING STATS

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

```
from sklearn.feature_selection import RFE
rfe = RFE(lr,n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)
```

```
rfe.support_
```

...

```
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
#list of RFE supported columns
col = X_train.columns[rfe.support_]
col
```

...

```
X_train.columns[~rfe.support_]
```

...

```
#BUILDING MODEL #1

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

PART 2

```
#BUILDING MODEL #2

X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	Converted	**No. Observations:**	6372
Model:	GLM	**Df Residuals:**	6357
Model Family:	Binomial	**Df Model:**	14
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-1263.0
Date:	Sat, 17 Feb 2024	**Deviance:**	2526.0
Time:	23:49:53	**Pearson chi2:**	8.36e+03
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.6060
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4354	0.116	-3.743	0.000	-0.663	-0.207
Total Time Spent on Website	1.0605	0.060	17.665	0.000	0.943	1.178
Lead Source_Direct Traffic	-1.4424	0.165	-8.762	0.000	-1.765	-1.120
Lead Source_Google	-1.0580	0.150	-7.038	0.000	-1.353	-0.763
Lead Source_Organic Search	-1.0774	0.190	-5.680	0.000	-1.449	-0.706
Lead Source_Welingak Website	4.3001	0.738	5.830	0.000	2.855	5.746
Last Activity_SMS Sent	1.9526	0.114	17.111	0.000	1.729	2.176
Last Notable Activity_Modified	-1.7075	0.126	-13.514	0.000	-1.955	-1.460

BUILDING MODEL 3 AND PREDICTED VALUES ON TRAIN SET

```
#BUILDING MODEL #3
X_train_sm = sm.add_constant(X_train[col])
logm3 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm3.fit()
res.summary()
```

...

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
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
```

...

```
# Getting the Predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

...

```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

```
array([0.44582497, 0.07934499, 0.02327763, 0.99180634, 0.01483375,
       0.1412266 , 0.03044004, 0.96156545, 0.52501575, 0.98955508])
```

```
y_train_pred_final = pd.DataFrame({'Converted':y_train.values, 'Converted_prob':y_train_pred})
y_train_pred_final['Prospect ID'] = y_train.index
y_train_pred_final.head()
```

CONFUSION MATRIX

CONFUSION MATRIX TO FIND OUT ACCURACY,SENSITIVITY AND SPECIFICITY ON TRAIN DATA

```
from sklearn import metrics

# Confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.Predicted )
print(confusion)
```

```
[[3767  186]
 [ 287 2132]]
```

```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.Predicted))
```

```
0.9257689893283113
```

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

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

```
0.8813559322033898
```

```
# Let us calculate specificity
TN / float(TN+FP)
```

```
0.9529471287629648
```

```
# Calculate False Postive Rate - predicting conversion when customer does not have convert
print(FP/ float(TN+FP))
```

```
0.047052871237035165
```

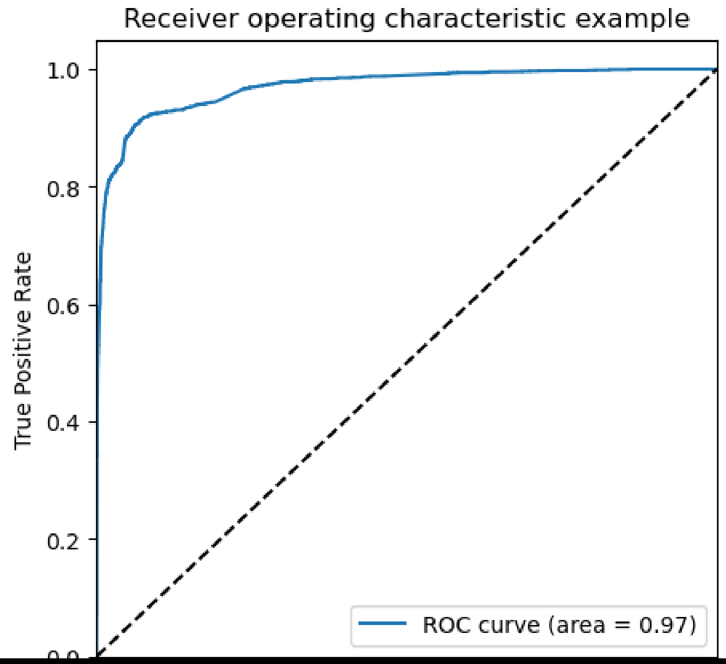
```
# Negative predictive value
print (TN / float(TN+ FN))
```

```
0.9292057227429699
```

```python
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
```

```python
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Converted, y_train_pred_final.Converted_prob, drop_intermediate = Fa
```

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



The ROC Curve should be a value close to 1. We are getting a good value of 0.97 indicating a good predictive model.

OBSERVATION

Observation:
So as we can see above the model seems to be performing well.
The ROC curve has a value of 0.97, which is very good.
We have the following values for the Train Data:

Accuracy : 92.26%
Sensitivity : 91.4%
Specificity : 92.76%
Some of the other Stats are derived below, indicating the False Positive Rate, Positive Predictive Value,Negative Predictive Valu

WORKING ON TEST DATA

Working on Test Data

```
y_test_pred = res.predict(X_test_sm)
```

```
y_test_pred[:10]
```

...

```
# Converting y_pred to a dataframe which is an array
y_pred_1 = pd.DataFrame(y_test_pred)
```

```
# Let's see the head
y_pred_1.head()
```

...

```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
```

```
# Putting CustID to index
y_test_df['Prospect ID'] = y_test_df.index
```

```
# Removing index for both dataframes to append them side by side
y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

```
# Appending y_test_df and y_pred_1
y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
```

```
y_pred_final.head()
```

...

```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 0 : 'Converted_prob'})
```

```
y_pred_final.head()
```

...

```
# Rearranging the columns
y_pred_final = y_pred_final[['Prospect ID','Converted','Converted_prob']]
y_pred_final['Lead_Score'] = y_pred_final.Converted_prob.map( lambda x: round(x*100))
```

CREATING CONFUSION MATRIX ON TEST DATA

CONFUSION MATRIX TO FIND OUT ACCURACY,SENSITIVITY AND SPECIFICITY ON TEST DATA.

```
# Let's check the overall accuracy.
metrics.accuracy_score(y_pred_final.Converted, y_pred_final.final_Predicted)
```

```
0.9274990845844013
```

```
confusion2 = metrics.confusion_matrix(y_pred_final.Converted, y_pred_final.final_Predicted )
confusion2
```

```
array([[1574,  115],
       [  83,  959]], dtype=int64)
```

```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

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

```
0.9203454894433781
```

```
# Let us calculate specificity
TN / float(TN+FP)
```

```
0.9319123741859088
```

```
precision_score(y_pred_final.Converted , y_pred_final.final_Predicted)
```

```
0.8929236499068901
```

```
recall_score(y_pred_final.Converted, y_pred_final.final_Predicted)
```

```
0.9203454894433781
```

FINAL OUTCOME

Final Result

```
Final values from train & Test:

Train Data:
Accuracy : 92.66%
Sensitivity : 91.44%
Specificity : 92.76%
Test Data:
Accuracy : 92.74%
Sensitivity : 92.03%
Specificity : 93.19%
```

THANK YOU !!!