LEAD SCORING CASE STUDY

USING LOGISTIC REGRESSION MODEL

Submitted by :

- Vinod Yadav
- Vignesh Kumar
- Ujjwal Verma

OUTLINE

- ○ Executive Summary
- ○ Introduction
- ○ Methodology
- ○ Results
- ○ Conclusion

EXECUTIVE SUMMARY

SUMMARY OF METHODOLOGIES

- Data Collection
- Exploratory Data Analysis
- Identifying Categorical Variables and Creating Dummy Variables
- Model Building Using Logistic Regression
- Prediction on Test Dataset
- Conclusion

SUMMARY OF RESULTS

- Data Analysis along with Interactive Visualizations
- Conclusion and recommendations

INTRODUCTION

PROBLEM STATEMENT

- An education company named X Education sells online courses to industry professionals. On any given day, many professionals who are interested in the courses land on their website and browse for courses.

- Company markets its courses on several websites and search engines like Google. Once these people land on the website, they might browse the courses or fill up a form for the course or watch some videos. When these people fill up a form providing their email address or phone number, they are classified to be a lead. Moreover, the company also gets leads through past referrals. Once these leads are acquired, employees from the sales team start making calls, writing emails, etc. Through this process, some of the leads get converted while most do not. The typical lead conversion rate at X education is around 30%.

- Although X Education gets a lot of leads, its lead conversion rate is very poor. For example, if, say, they acquire 100 leads in a day, only about 30 of them are converted. To make this process more efficient, the company wishes to identify the most potential leads, also known as ‘Hot Leads’. If they successfully identify this set of leads, the lead conversion rate should go up as the sales team will now be focusing more on communicating with the potential leads rather than making calls to everyone.

INTRODUCTION

GOALS OF THE CASE STUDY:

- To build 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.

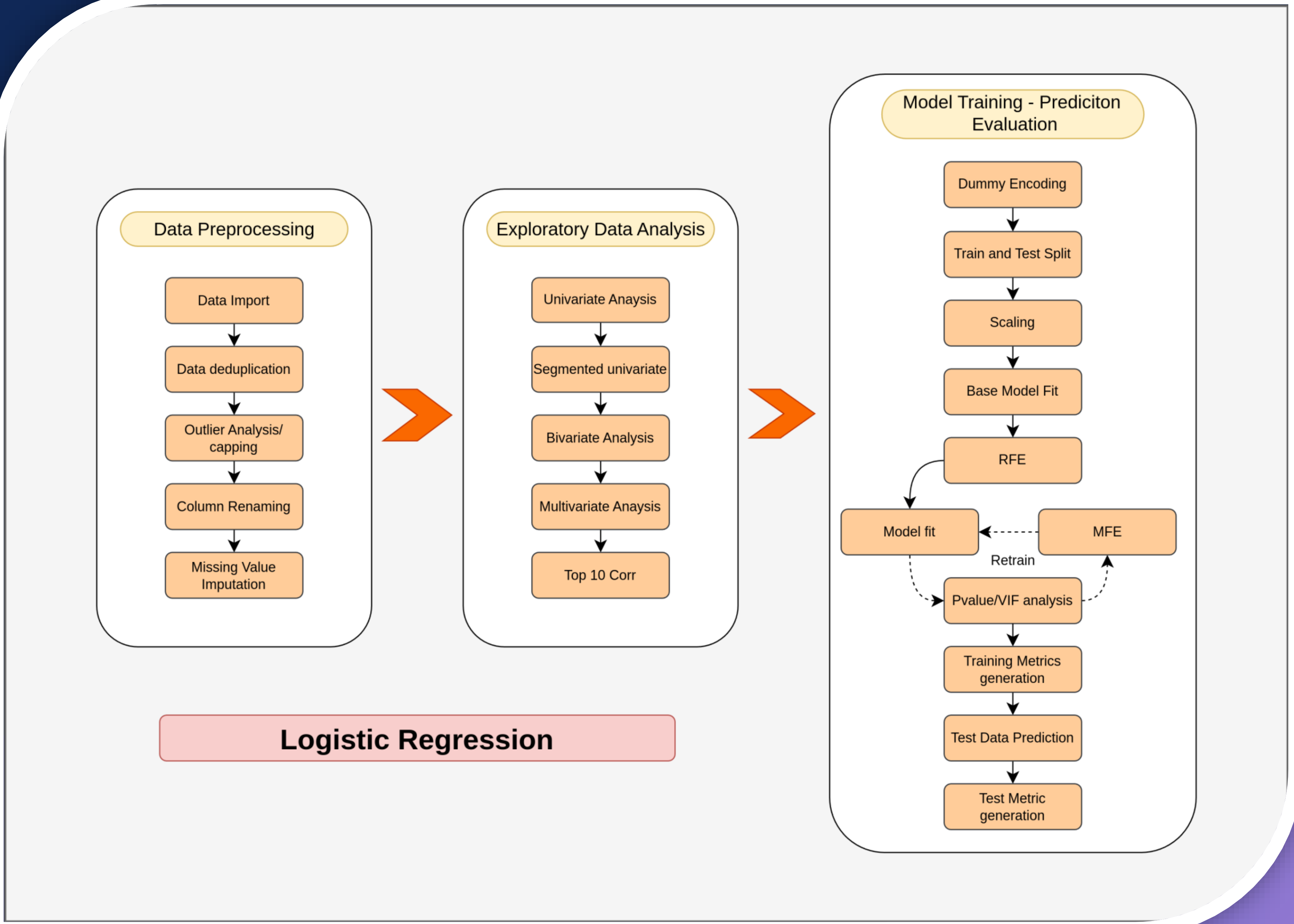
- To adjust to if the company's requirement changes in the future so you will need to handle these as well.

METHODOLOGY

GOALS OF THE CASE STUDY:

- Data Preprocessing
- Data Visualization
- Model Training
- Metrics Comparison
- Prediction on Test data
- Conclusion

LOGISTIC REGRESSION OVERVIEW



Data Preprocessing

- Data Import
- Data deduplication
- Outlier Analysis/ capping
- Column Renaming
- Missing Value Imputation

Exploratory Data Analysis

- Univariate Anaysis
- Segmented univariate
- Bivariate Analysis
- Multivariate Anaysis
- Top 10 Corr

Model Training - Prediciton Evaluation

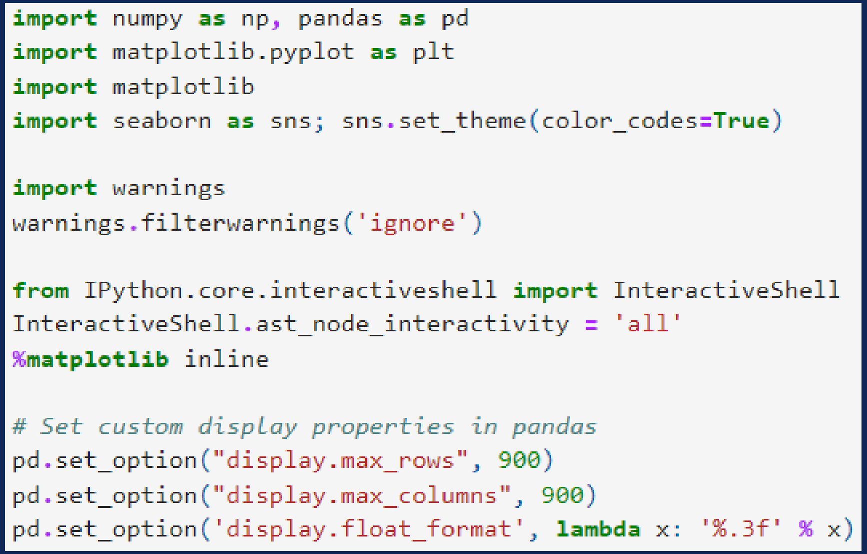
- Dummy Encoding
- Train and Test Split
- Scaling
- Base Model Fit
- RFE
- Model fit
- MFE
- Retrain
- Pvalue/VIF analysis
- Training Metrics generation
- Test Data Prediction
- Test Metric generation

Logistic Regression

STEPS IN LOGISTIC REGRESSION MODEL

DATA PREPROCESSING

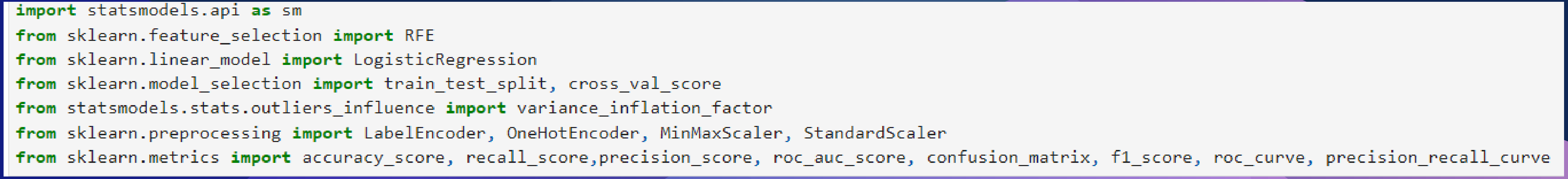
1. PACKAGE IMPORTS AND DATA INITIALIZATION



```
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns; sns.set_theme(color_codes=True)

import warnings
warnings.filterwarnings('ignore')

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
%matplotlib inline

# Set custom display properties in pandas
pd.set_option("display.max_rows", 900)
pd.set_option("display.max_columns", 900)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

With this code we have imported all libraries numpy, pandas, matplotlib, seaborn.

With below mentioned code We have imported necessary machine learning packages (sklearn, statsmodel) for performing logistic regression



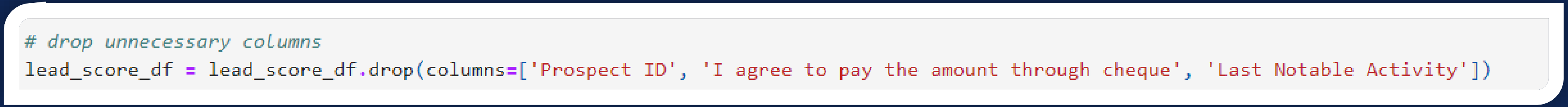
```
import statsmodels.api as sm
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler, StandardScaler
from sklearn.metrics import accuracy_score, recall_score,precision_score, roc_auc_score, confusion_matrix, f1_score, roc_curve, precision_recall_curve
```

DATA PREPROCESSING

WE HAVE CREATED CUSTOM FUNCTIONS FOR PREPROCESSING AND EDA

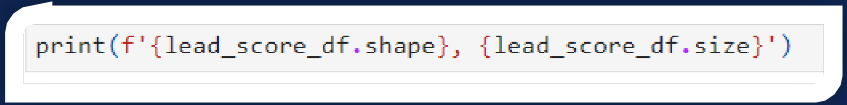
DEF CLASSIFY_FEATURE_DTYPE(DF, COLS):
DEF SHOW_STATS(DF, COLS):
DEF CHECK_COLS_NULL_PCT(DF):
DEF UNIVARIATE_PLOTS(DF, COLS, TARGET=NONE, FTYPE=NONE, L_DICT = NONE):
DEF GET_EXTREMEVAL_THRESHLD(DF, FIND_OUTLIER=FALSE)

WITH THE BELOW MENTIONED CODE WE HAVE DROP UNNECESSARY COLUMNS

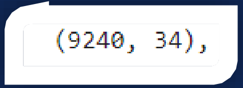


```
# drop unnecessary columns
lead_score_df = lead_score_df.drop(columns=['Prospect ID', 'I agree to pay the amount through cheque', 'Last Notable Activity'])
```

#BELOW MENTIONED CODE WILL GIVE US THE SHAPE AND SIZE OF THE DATA FRAME

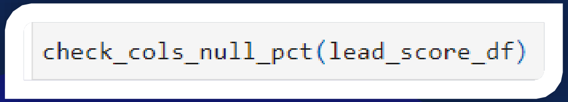


```
print(f'{lead_score_df.shape}, {lead_score_df.size}')
```

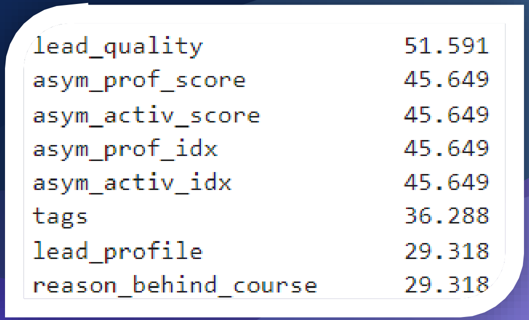


```
(9240, 34),
```

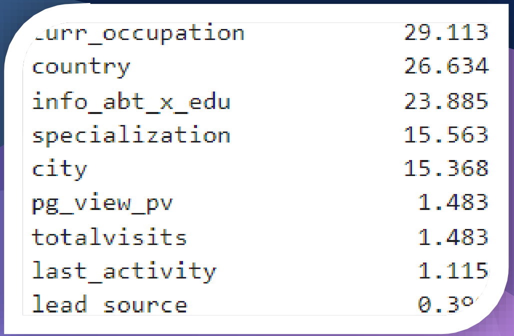
# WITH THE BELOW MENTIONED CODE WE HAVE# CHECKED NULL VAL PERCENTAGE
# AFTER CHECKING THE NULL VALUE PERCENTAGE FOR ALL THE FEATURES
# WE COULD SEE THAT THERE ARE MANY FEATURES THAT HAVE MORE THAN 40% OF NON VALUES



```
check_cols_null_pct(lead_score_df)
```



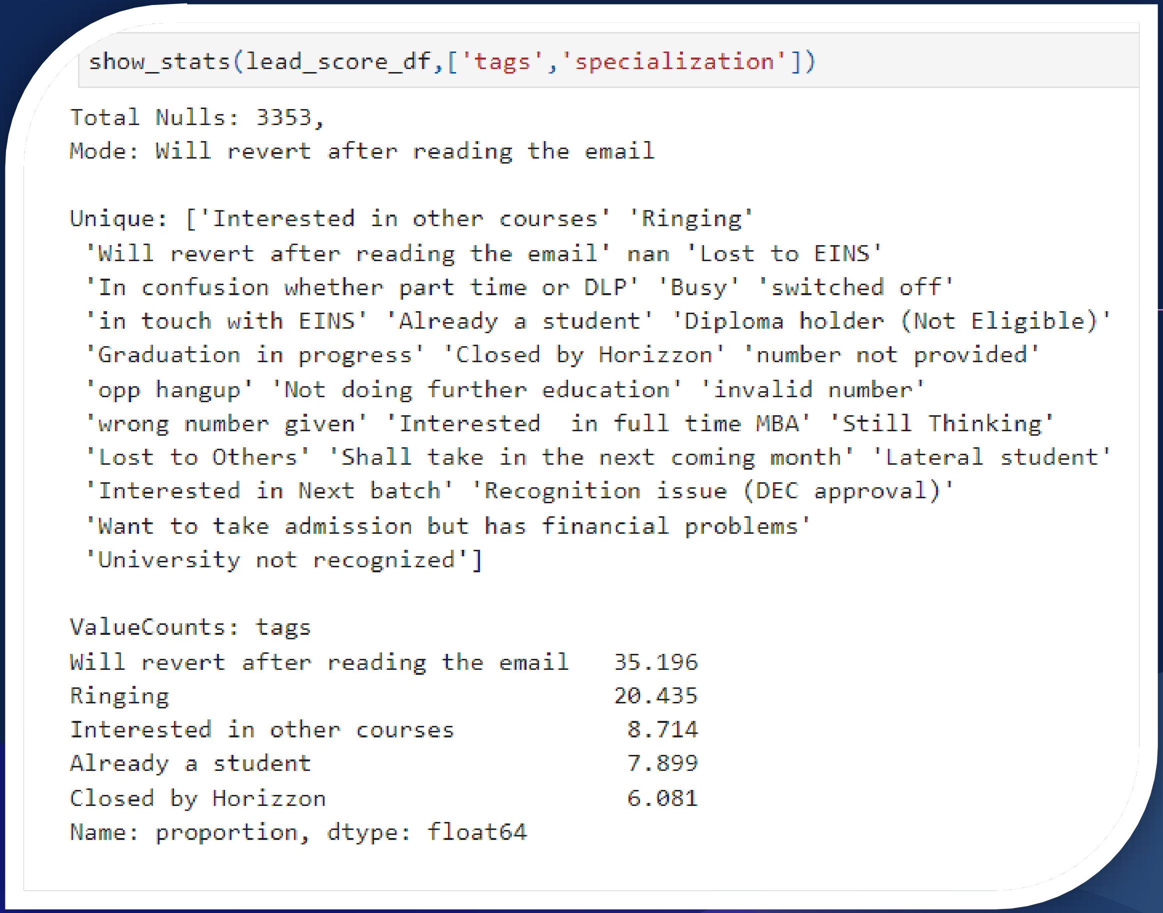
```
lead_quality             51.591
asym_prof_score          45.649
asym_activ_score         45.649
asym_prof_idx            45.649
asym_activ_idx           45.649
tags                     36.288
lead_profile             29.318
reason_behind_course     29.318
```



```
urr_occupation           29.113
country                  26.634
info_abt_x_edu           23.885
specialization           15.563
city                     15.368
pg_view_pv                1.483
totalvisits               1.483
last_activity             1.115
lead source               0.3
```

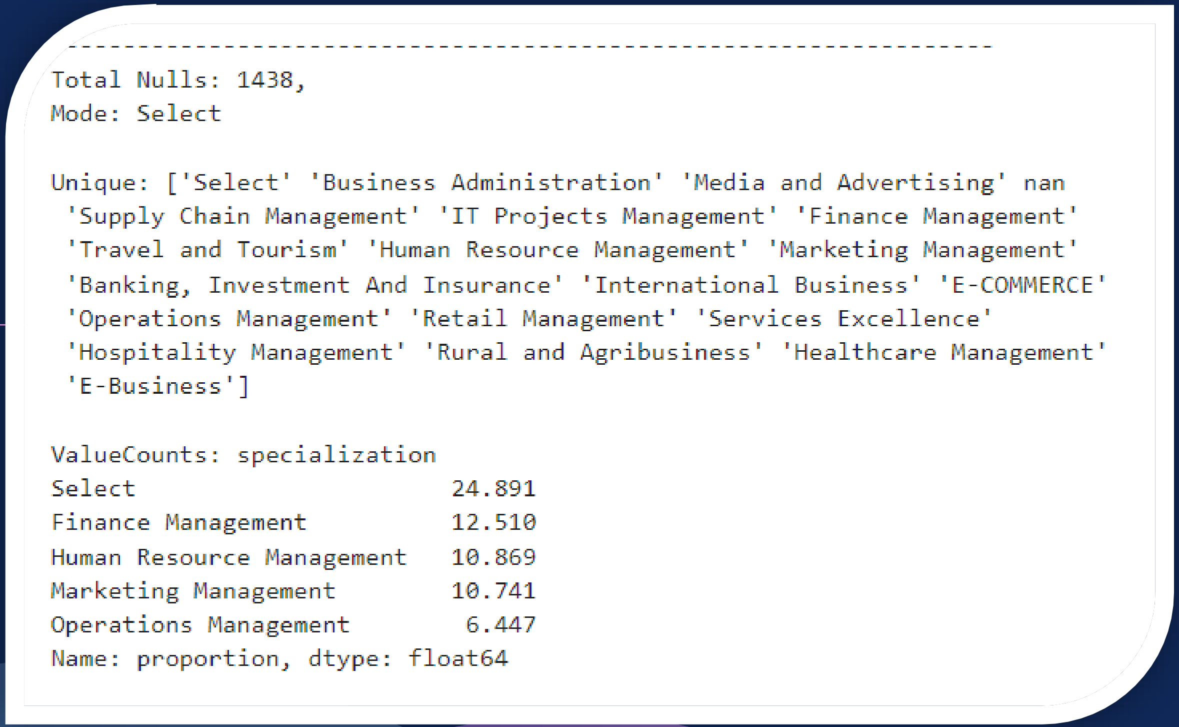
DATA PREPROCESSING

IN TAGS AND AND SPECIALIZATION WE HAVE REPLACED SELECT AND NAN VALUES WITH UNKNOWN, AND REMOVE THE UNKNOWN VALUES AFTER DOING DUMMIFICATION



```
show_stats(lead_score_df,['tags','specialization'])
```

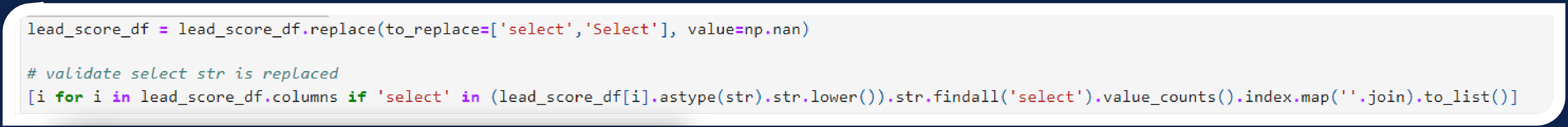
```
Total Nulls: 3353,
Mode: Will revert after reading the email

Unique: ['Interested in other courses' 'Ringing'
 'Will revert after reading the email' nan 'Lost to EINS'
 'In confusion whether part time or DLP' 'Busy' 'switched off'
 'in touch with EINS' 'Already a student' 'Diploma holder (Not Eligible)'
 'Graduation in progress' 'Closed by Horizzon' 'number not provided'
 'opp hangup' 'Not doing further education' 'invalid number'
 'wrong number given' 'Interested  in full time MBA' 'Still Thinking'
 'Lost to Others' 'Shall take in the next coming month' 'Lateral student'
 'Interested in Next batch' 'Recognition issue (DEC approval)'
 'Want to take admission but has financial problems'
 'University not recognized']

ValueCounts: tags
Will revert after reading the email    35.196
Ringing                                20.435
Interested in other courses             8.714
Already a student                       7.899
Closed by Horizzon                      6.081
Name: proportion, dtype: float64
```



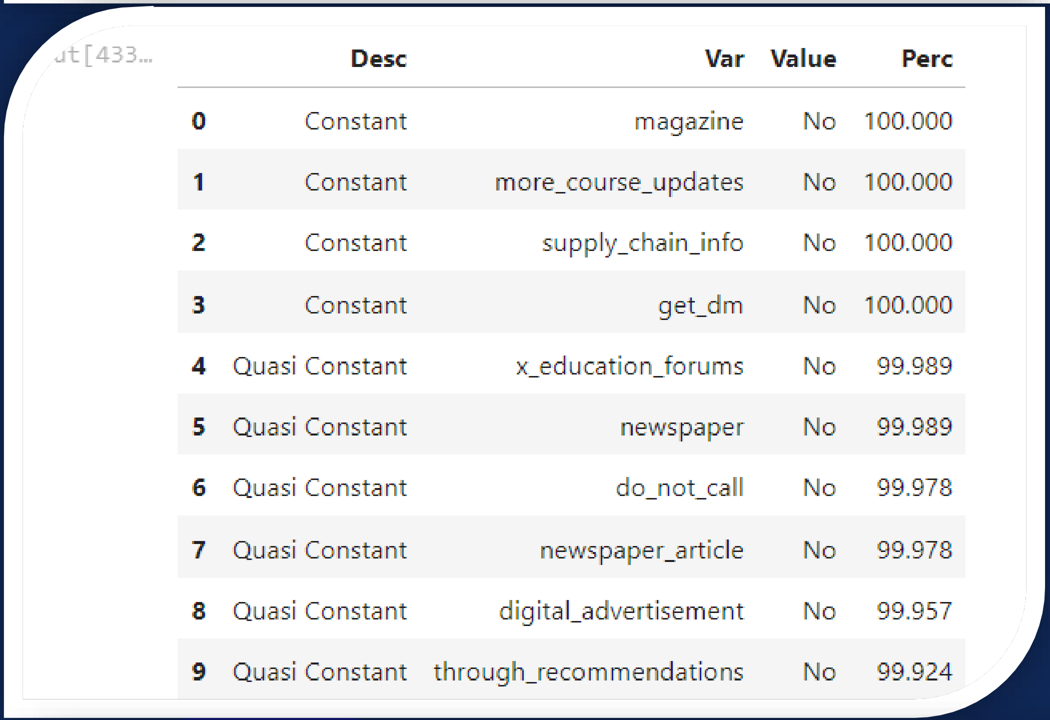
```
-------------------------------------------------------------------------
Total Nulls: 1438,
Mode: Select

Unique: ['Select' 'Business Administration' 'Media and Advertising' nan
 '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']

ValueCounts: specialization
Select                        24.891
Finance Management            12.510
Human Resource Management     10.869
Marketing Management          10.741
Operations Management          6.447
Name: proportion, dtype: float64
```

DATA PREPROCESSING

BELOW MENTIONED CODE HAS REPLACE SELECT STRING WITH NAN



```
lead_score_df = lead_score_df.replace(to_replace=['select','Select'], value=np.nan)

# validate select str is replaced
[i for i in lead_score_df.columns if 'select' in (lead_score_df[i].astype(str).str.lower()).str.findall('select').value_counts().index.map(''.join).to_list()]
```



ut[433…

	Desc	Var	Value	Perc
0	Constant	magazine	No	100.000
1	Constant	more_course_updates	No	100.000
2	Constant	supply_chain_info	No	100.000
3	Constant	get_dm	No	100.000
4	Quasi Constant	x_education_forums	No	99.989
5	Quasi Constant	newspaper	No	99.989
6	Quasi Constant	do_not_call	No	99.978
7	Quasi Constant	newspaper_article	No	99.978
8	Quasi Constant	digital_advertisement	No	99.957
9	Quasi Constant	through_recommendations	No	99.924

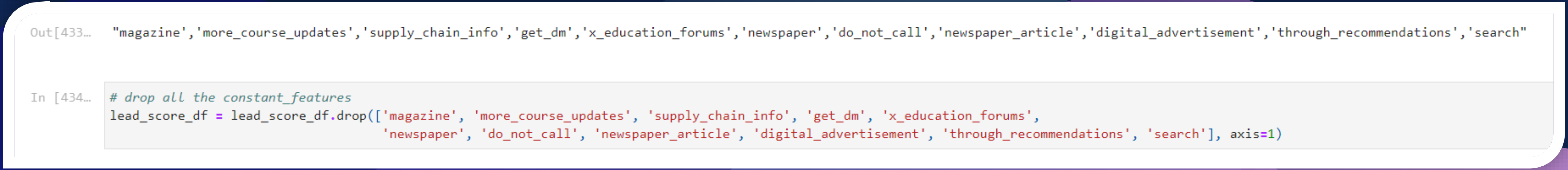
WE HAVE CHECKED CONSTANT FEATURES THAT HAS ONLY ONE VALUES

IN THE GIVEN DATA SET THERE ARE A LOT OF FEATURES THAT HAVE ONLY SINGLE VALUE AS A CATEGORY

THESE ARE CALLED AS CONSTANT FEATURES AND THESE FEATURES ARE OF LITTLE RELEVANCE FOR THE MACHINE LEARNING MODEL HENCE WE HAVE DROPPED THOSE FEATURES WHICH HAVE CONSTANT FEATURE IDENTIFICATION

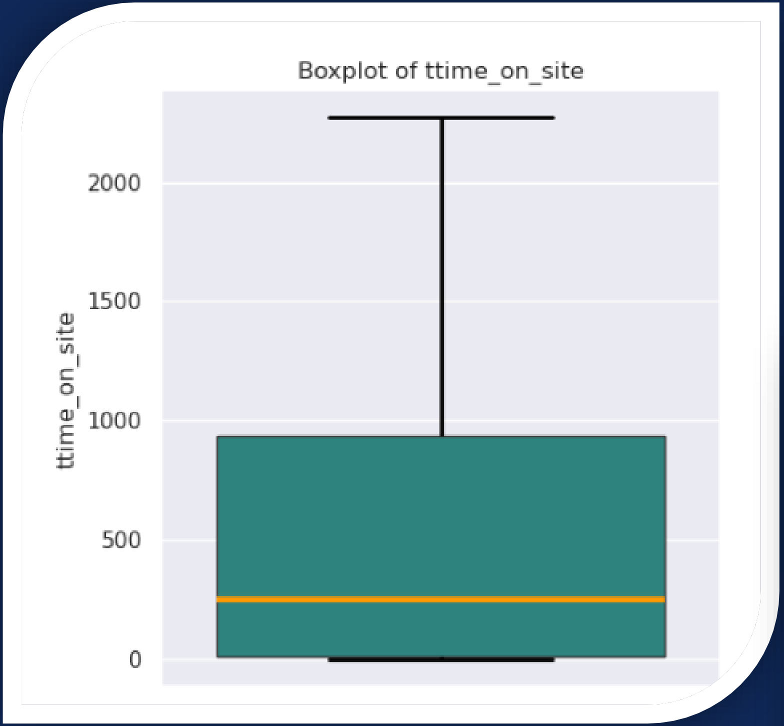
IMPUTE MISSING CATEGORICAL VALUES USING MODE, IF A PARTICULAR VALUE IN THAT COLUMN HAS HIGHER FREQUENCY SAY > 50%

AFTER COMPLETING EDA WE GOT FOUR MORE COLUMNS WHICH ARE HAVING LESS THAN 2 % OF NULL VALUES , SO WE WILL DROP THE ROWS FROM THOSE COLUMNS (‘TOTALVISITS’,’PG_VIEW_PG’,’LAST_ACTIVITY’,’LEAD_SOURCE’)

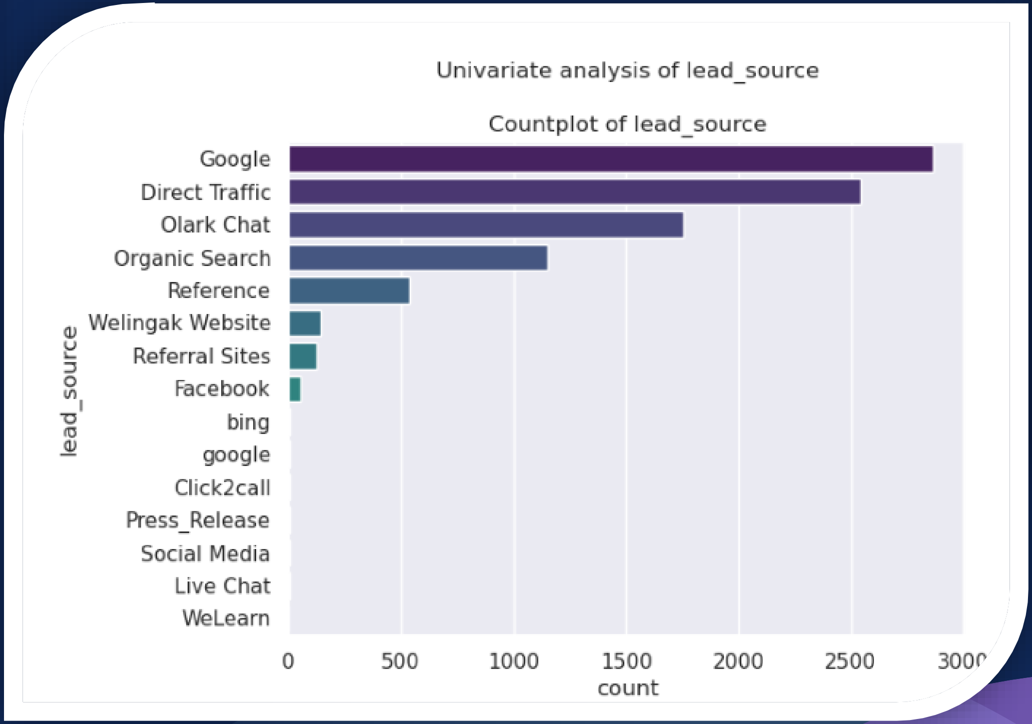


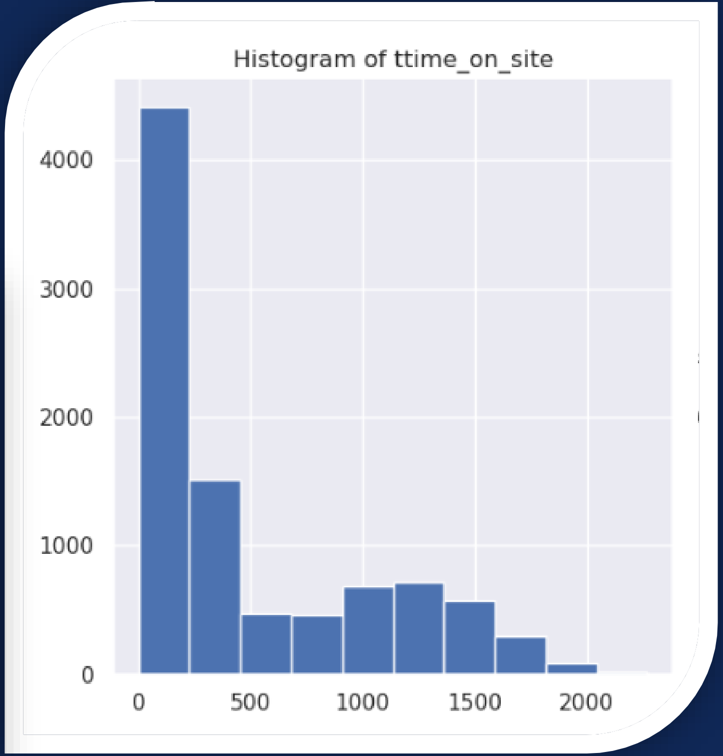
```
Out[433…  "magazine','more_course_updates','supply_chain_info','get_dm','x_education_forums','newspaper','do_not_call','newspaper_article','digital_advertisement','through_recommendations','search"

In [434…  # drop all the constant_features
          lead_score_df = lead_score_df.drop(['magazine', 'more_course_updates', 'supply_chain_info', 'get_dm', 'x_education_forums',
                                              'newspaper', 'do_not_call', 'newspaper_article', 'digital_advertisement', 'through_recommendations', 'search'], axis=1)
```

DATA VISUALIZATION



UNIVARIATE PLOTS

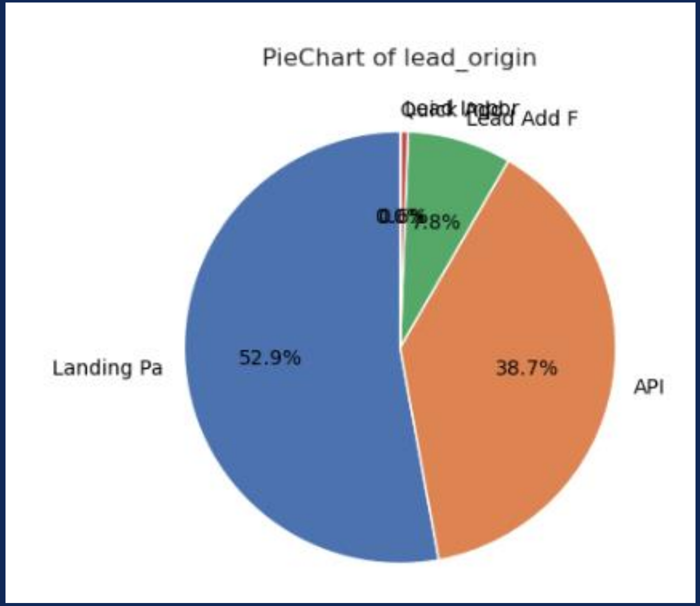




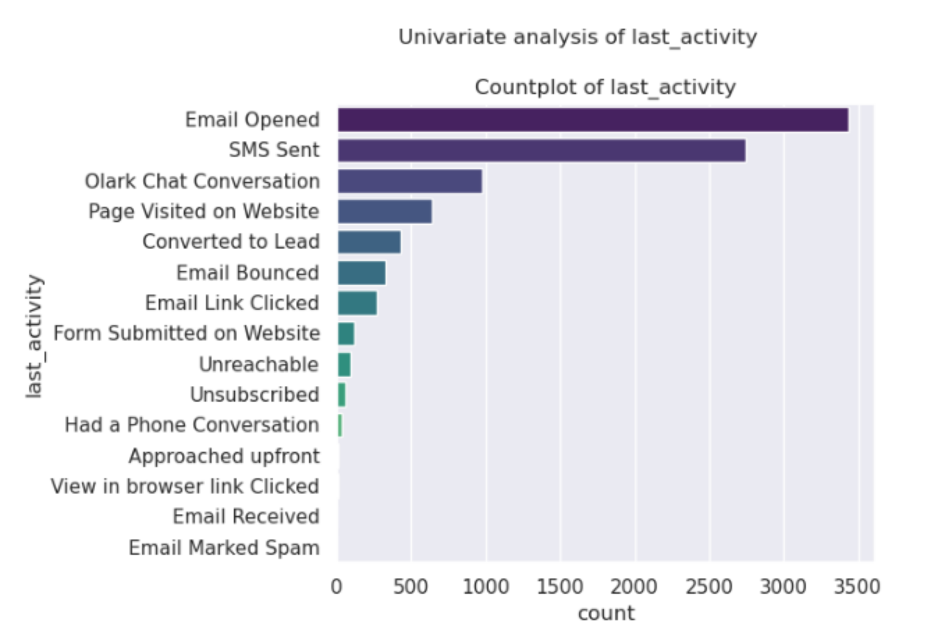
WHEN IT COMES TO LEAD SOURCE 30% ARE FROM GOOGLE, 27% ARE FROM DIRECT TRAFFIC, 19% ARE FROM OLARK

DATA VISUALIZATION

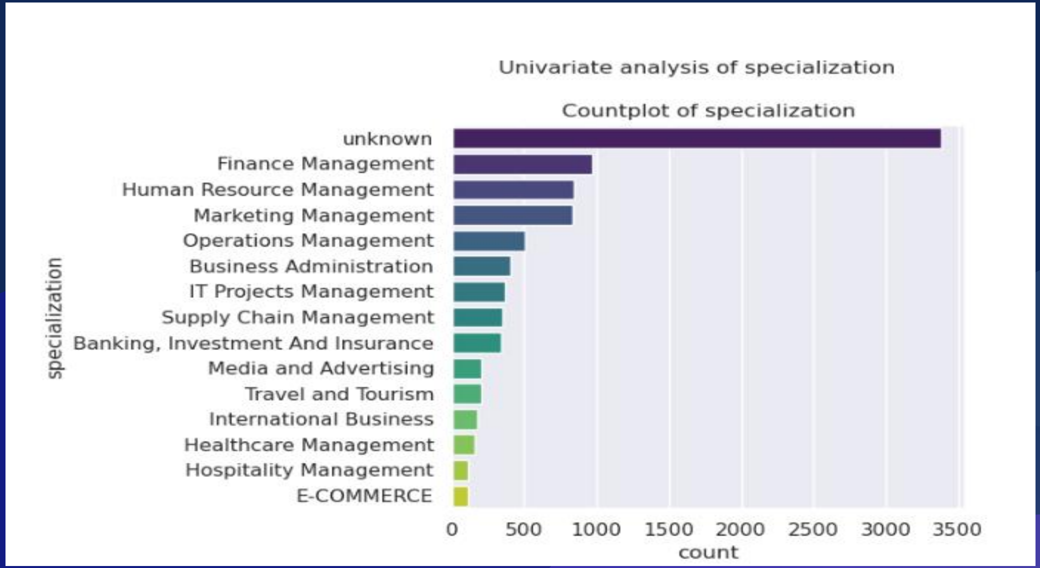
UNIVARIATE PLOTS



WHEN IT COMES TO LEAD ORIGIN CONVERSION RATES, LEAD_ADD_FORM HAS HIGHER CONVERSION RATES, HOLISTICALLY ALL HAVE SIMILAR PROBABILITY RATE



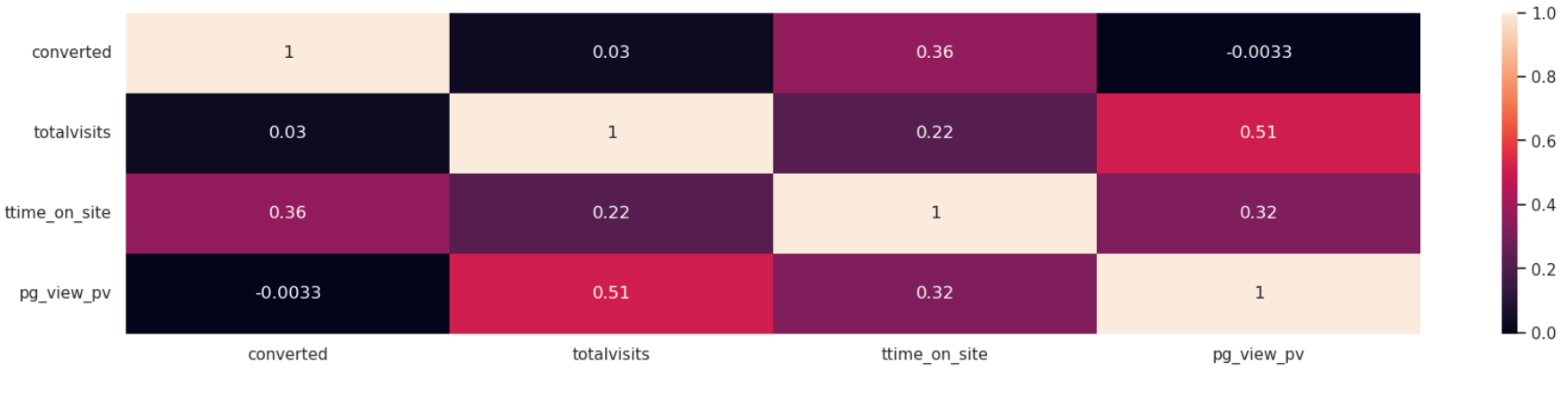
WTHE LAST ACTIVITY FEATURE MAJORITY OF THE USERS, 38% OF USERS HAVE EMAIL OPENED FOLLOWED BY SMS SENT, THEREFORE WE CAN SAY THAT MAJORITY OF THE USERS ARE ACTIVE ON EMAIL CONVERSATIONS



AMONG THE EMPLOYED USERS MOST OF THE INTERESTED USERS HAVE FINANCE MANAGEMENT AS A SPECIALIZATION, FOLLOWED BY HUMAN RESOURCE MANAGEMENT AND MARKETING MANAGEMENT, ALMOST ALL SPECIALIZATION HAS A SIMILAR CONVERSION RATE.

DATA VISUALIZATION

MULTIVARIATE PLOT

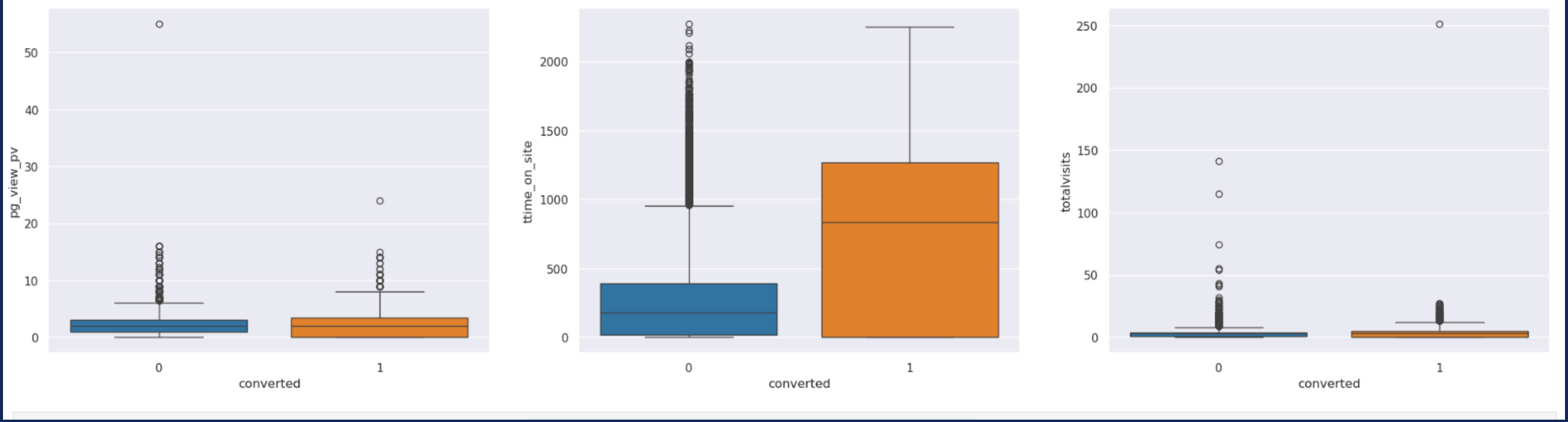


	converted	totalvisits	ttime_on_site	pg_view_pv
converted	1	0.03	0.36	-0.0033
totalvisits	0.03	1	0.22	0.51
ttime_on_site	0.36	0.22	1	0.32
pg_view_pv	-0.0033	0.51	0.32	1

THERE ARE NOT MUCH CORRELATIONS BETWEEN VARIABLES

DATA VISUALIZATION

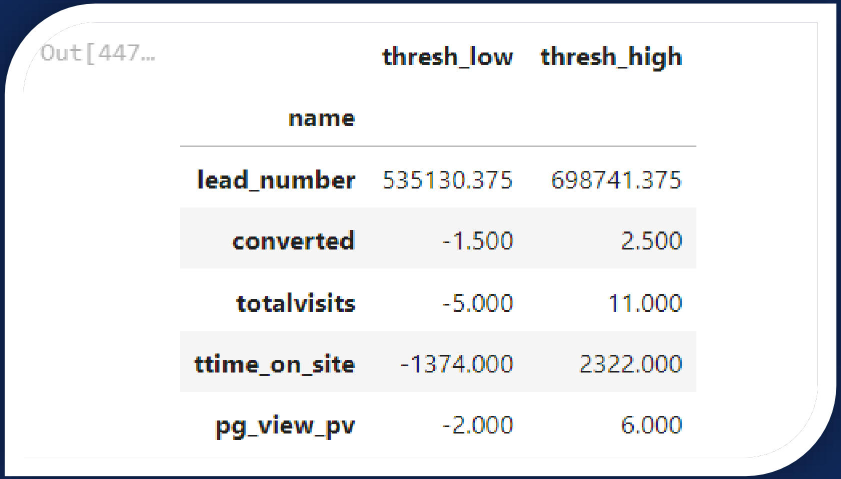
BIVARIATE - MULTIVARIATE PLOTS



THERE ARE NOT MUCH DIFFERENCE BETWEEN CONVERTED PG_VIEW_PV AND TOTAL VISISTS , WHEREAS CONVERTED LEADS USE TO SPENT MORE TIME ON SITE

DATA PREPROCESSING PART 2

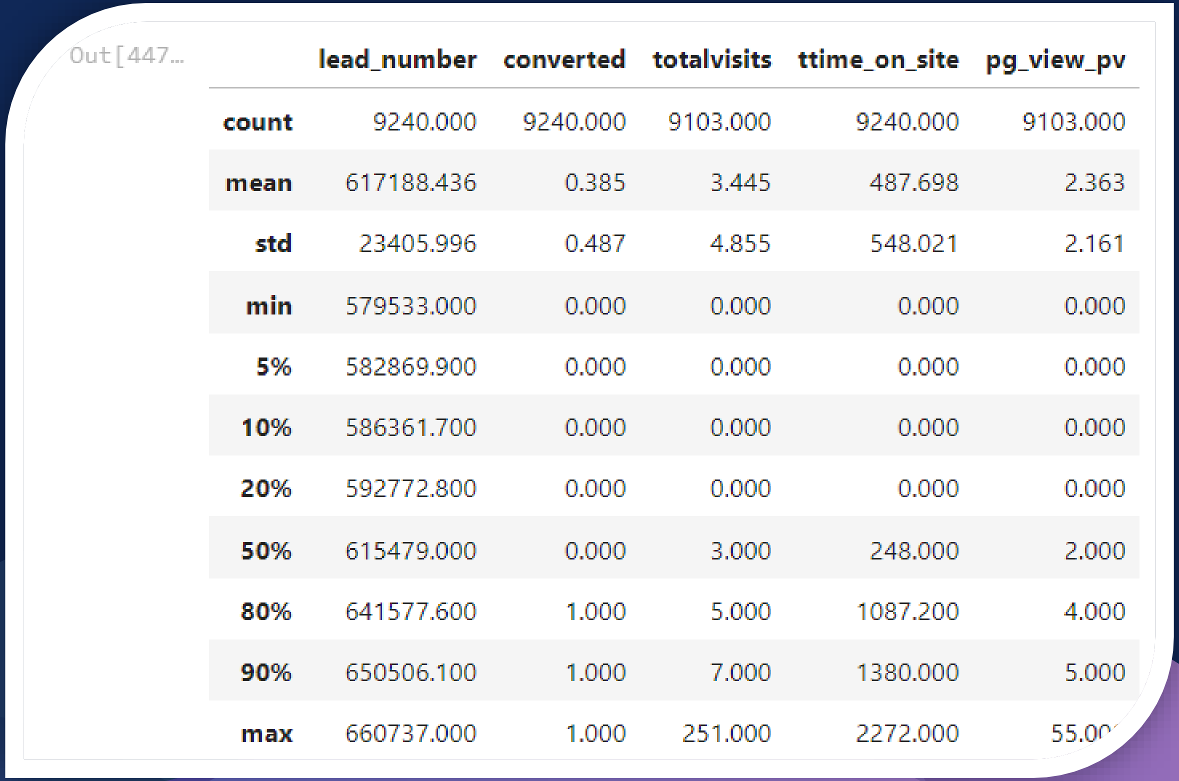
OUTLIER ANALYSIS AND CAPPING



Out[447...

name	thresh_low	thresh_high
lead_number	535130.375	698741.375
converted	-1.500	2.500
totalvisits	-5.000	11.000
ttime_on_site	-1374.000	2322.000
pg_view_pv	-2.000	6.000

AFTER COMPLETING OUTLIER ANALYSIS , WE DID CAPING FOR OUTLIERS AND THE STATS ARE SHOWN BELOW

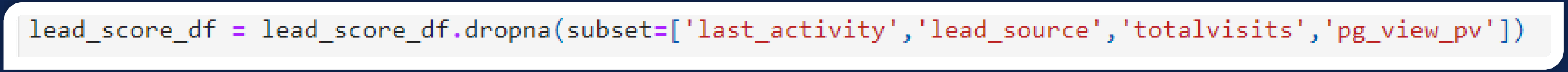


Out[447...

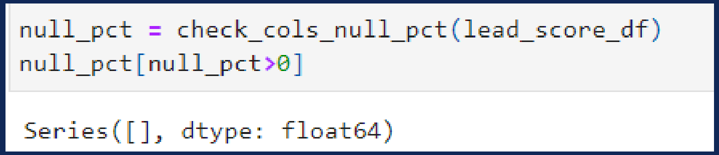
	lead_number	converted	totalvisits	ttime_on_site	pg_view_pv
count	9240.000	9240.000	9103.000	9240.000	9103.000
mean	617188.436	0.385	3.445	487.698	2.363
std	23405.996	0.487	4.855	548.021	2.161
min	579533.000	0.000	0.000	0.000	0.000
5%	582869.900	0.000	0.000	0.000	0.000
10%	586361.700	0.000	0.000	0.000	0.000
20%	592772.800	0.000	0.000	0.000	0.000
50%	615479.000	0.000	3.000	248.000	2.000
80%	641577.600	1.000	5.000	1087.200	4.000
90%	650506.100	1.000	7.000	1380.000	5.000
max	660737.000	1.000	251.000	2272.000	55.0

DATA PREPROCESSING PART 2

AFTER COMPLETING EDA WE GOT FOUR MORE COLUMNS WHICH ARE HAVING LESS THAN 2 % OF NULL VALUES , SO WE WILL DROP THE ROWS FROM THOSE COLUMNS ('TOTALVISITS','PG_VIEW_PG','LAST_ACTIVITY','LEAD_SOURCE')



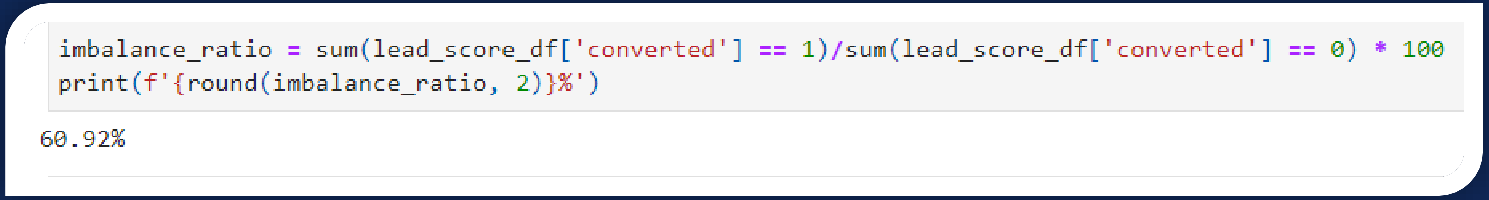
```
lead_score_df = lead_score_df.dropna(subset=['last_activity','lead_source','totalvisits','pg_view_pv'])
```



```
null_pct = check_cols_null_pct(lead_score_df)
null_pct[null_pct>0]

Series([], dtype: float64)
```

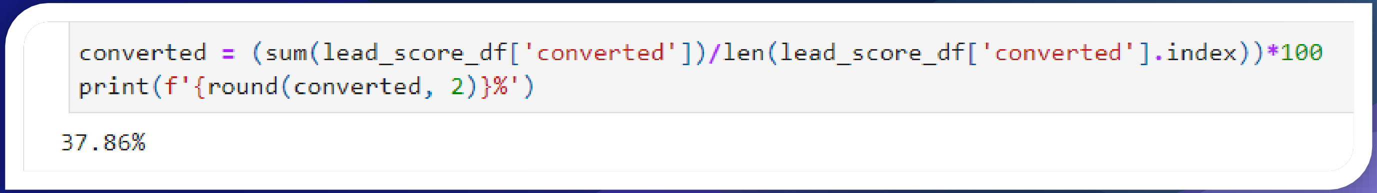
THERE ARE NO NULL VALUE COLUMN LEFT

DATA IMBALANCE & CONVERSION RATIO



```
imbalance_ratio = sum(lead_score_df['converted'] == 1)/sum(lead_score_df['converted'] == 0) * 100
print(f'{round(imbalance_ratio, 2)}%')

60.92%
```

FROM THE TARGET VARIABLE WE HAVE FOUND OUT THE IMBALANCE RATIOS AROUND 60 THEREFORE WE DECIDE NOT TO REBALANCE

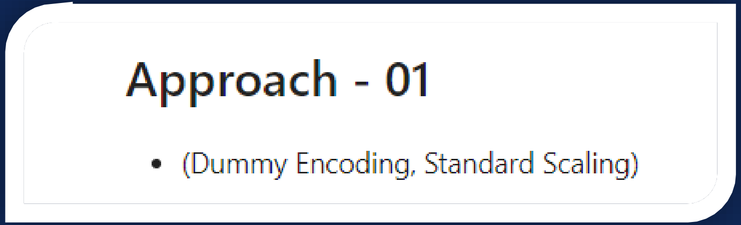


```
converted = (sum(lead_score_df['converted'])/len(lead_score_df['converted'].index))*100
print(f'{round(converted, 2)}%')

37.86%
```

FROM THE TARGET VARIABLE THE CONVERSION RATIO IS AROUND 38 IT SHOWS THAT THERE IS A VERY HIGH PROBABILITY OF FAILURE IN CONVERSION

MODEL TRAINING

WE HAVE PERFORMED DUMMY ENCODING
WE HAVE USED CUSTOM FUNCTIONS FOR MODEL TRAINING



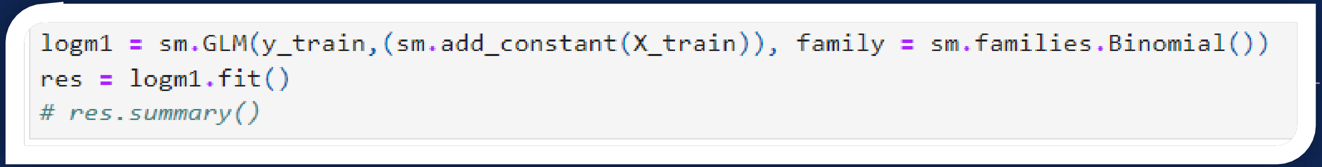
Approach - 01

- (Dummy Encoding, Standard Scaling)

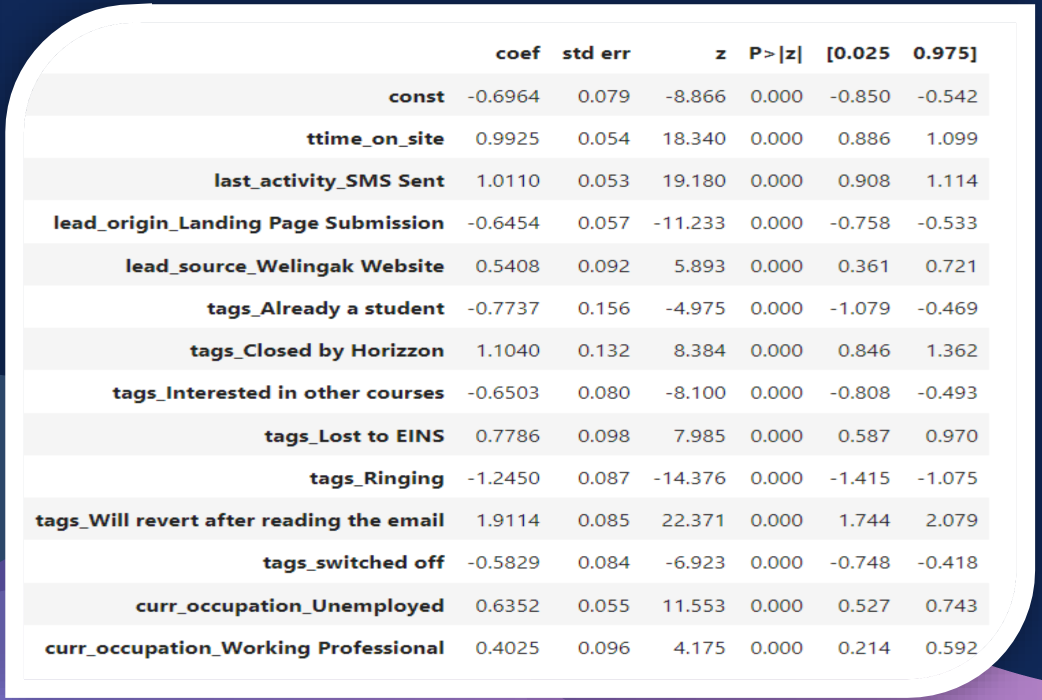
WE COMPLETED FOLLOWING STEPS IN PROCESS OF MODEL BUILDING

TRAIN AND TEST SPLIT
FEATURE SCALING
MODEL BUILDING
BASE MODEL
RFE - RECURSIVE FEATURE ELIMINATION

BASE MODEL



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

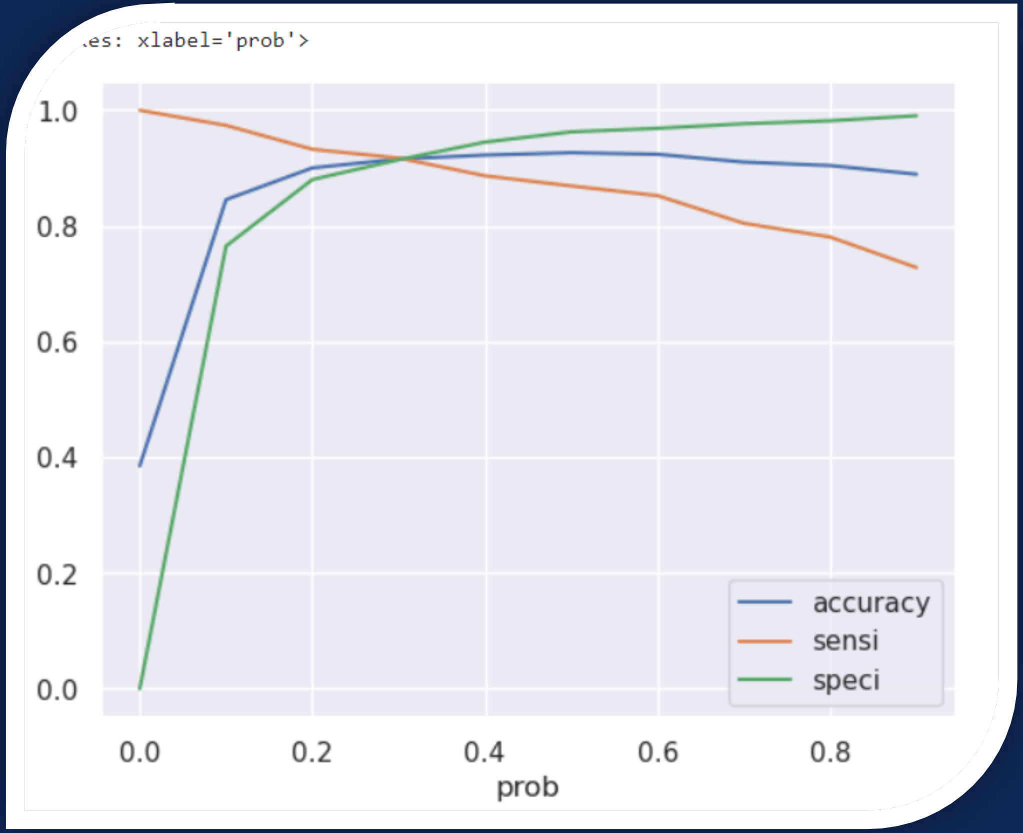


	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.6964	0.079	-8.866	0.000	-0.850	-0.542
ttime_on_site	0.9925	0.054	18.340	0.000	0.886	1.099
last_activity_SMS Sent	1.0110	0.053	19.180	0.000	0.908	1.114
lead_origin_Landing Page Submission	-0.6454	0.057	-11.233	0.000	-0.758	-0.533
lead_source_Welingak Website	0.5408	0.092	5.893	0.000	0.361	0.721
tags_Already a student	-0.7737	0.156	-4.975	0.000	-1.079	-0.469
tags_Closed by Horizzon	1.1040	0.132	8.384	0.000	0.846	1.362
tags_Interested in other courses	-0.6503	0.080	-8.100	0.000	-0.808	-0.493
tags_Lost to EINS	0.7786	0.098	7.985	0.000	0.587	0.970
tags_Ringing	-1.2450	0.087	-14.376	0.000	-1.415	-1.075
tags_Will revert after reading the email	1.9114	0.085	22.371	0.000	1.744	2.079
tags_switched off	-0.5829	0.084	-6.923	0.000	-0.748	-0.418
curr_occupation_Unemployed	0.6352	0.055	11.553	0.000	0.527	0.743
curr_occupation_Working Professional	0.4025	0.096	4.175	0.000	0.214	0.592

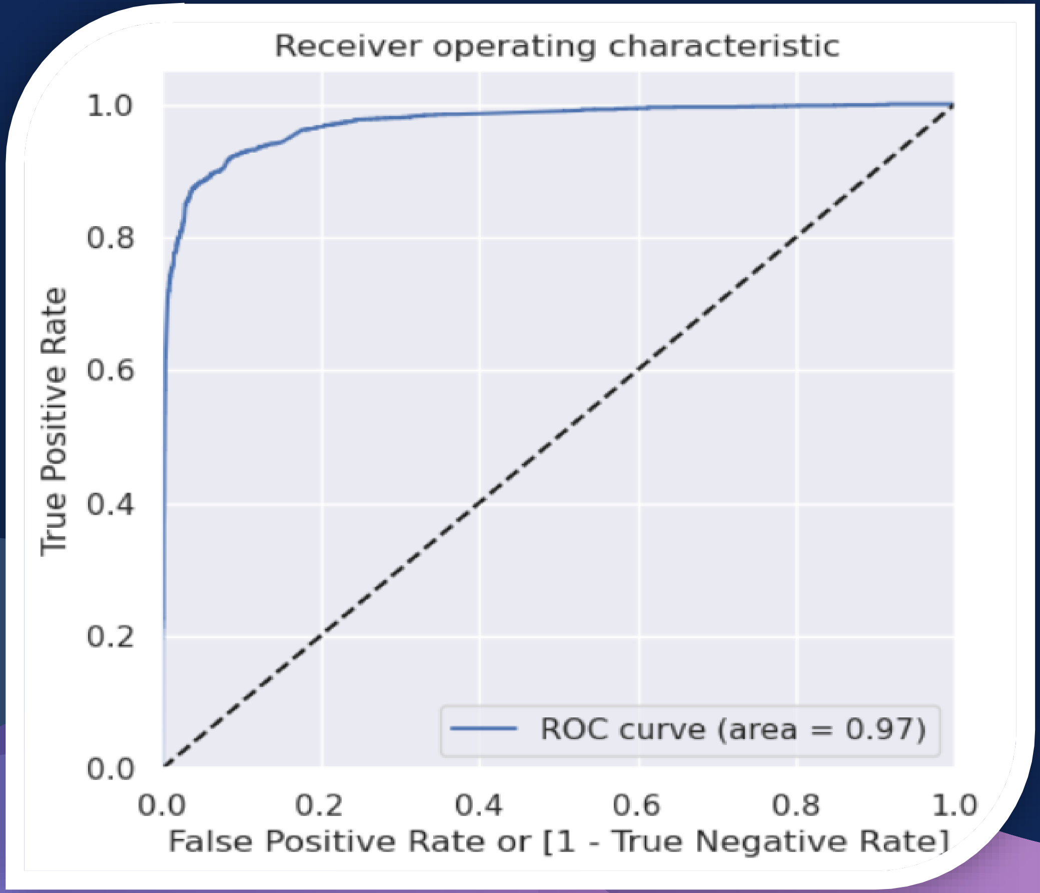
LOGISTIC REGRESSION MODEL : THIS OUR FINAL MODEL STATS

METRICS COMPARISON

WE HAVE PLOTTED ACCURACY SENSITIVITY AND SPECIFICITY FOR VARIOUS PROBABILITIES.



WE CAN SEE THAT OPTIMAL VALUE (CUT OFF VALUE) IS 0.30

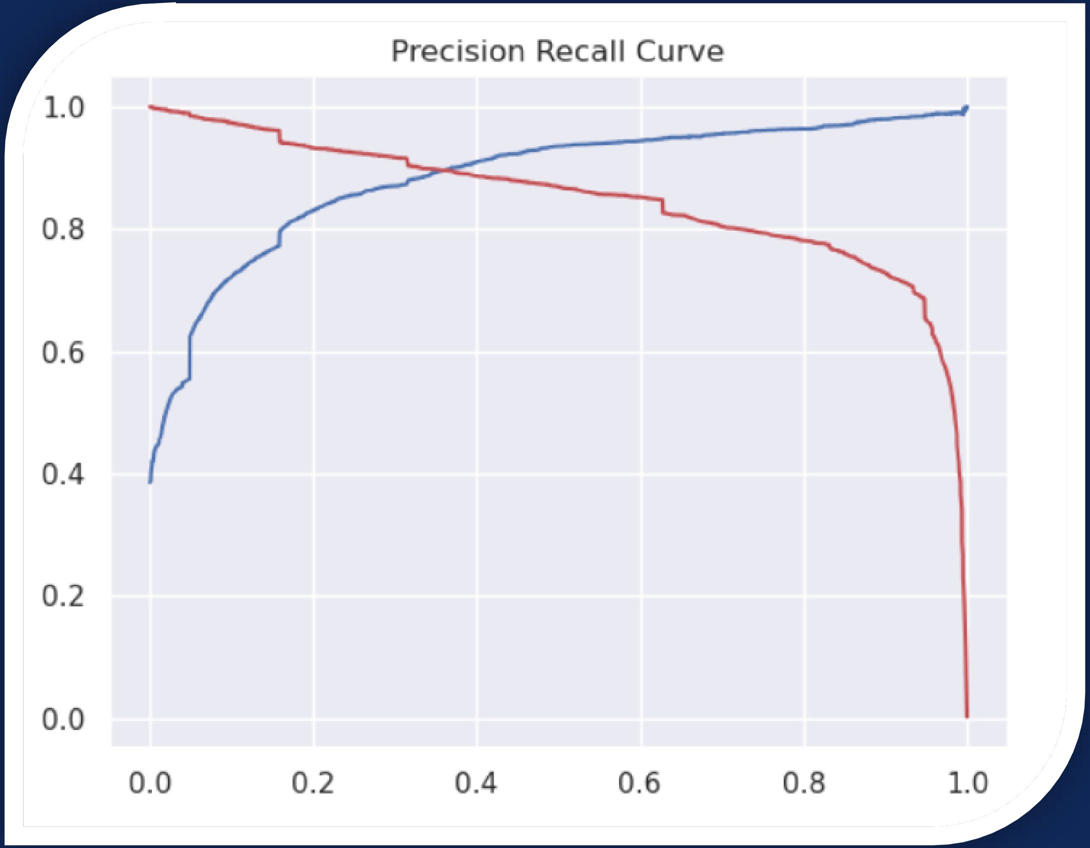


ROC CURVE AND PRECISION - RECALL CURVE

THE AREA UNDER THE CURVE OF THE ROC IS 0.97.

METRICS COMPARISON

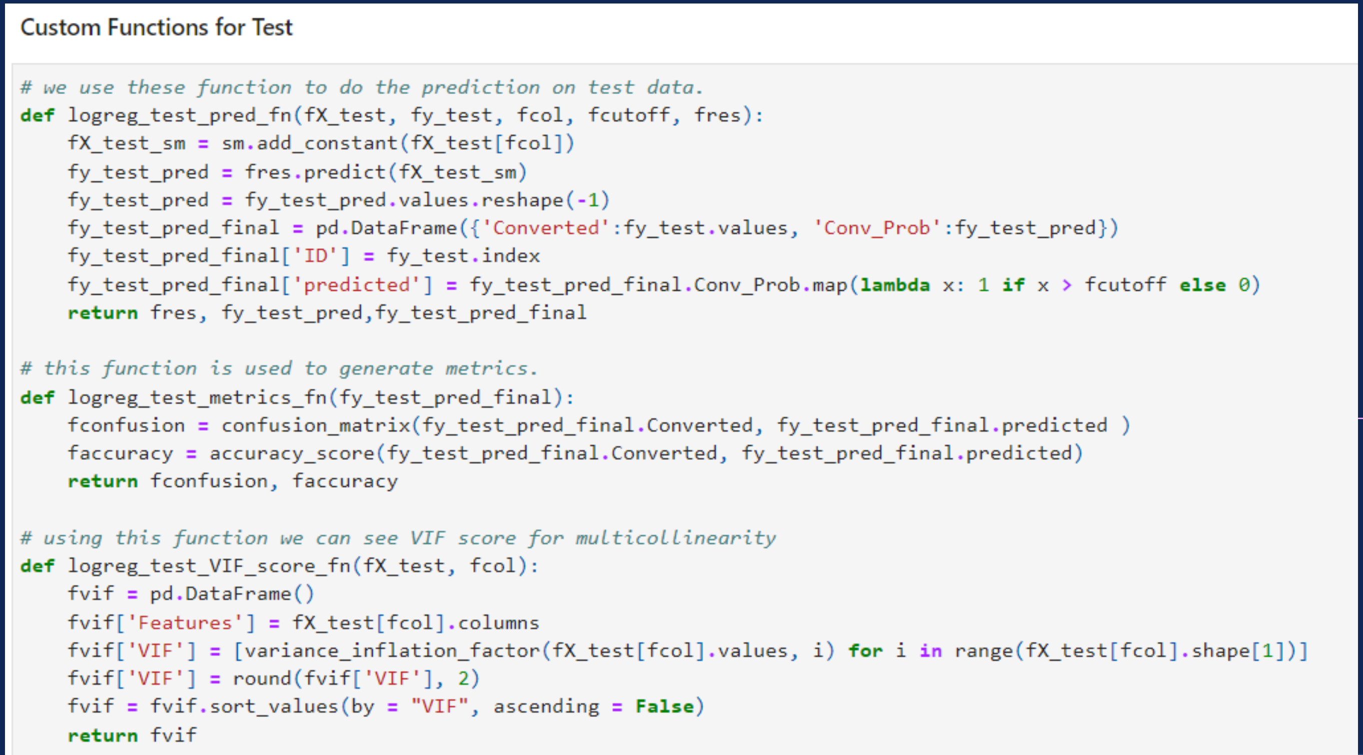
WE HAVE PLOTTED ACCURACY SENSITIVITY AND SPECIFICITY FOR VARIOUS PROBABILITIES.



AS PRECISION RECALL IS HAVING HIGH VALUE THAN SENSITIVITY AND SPECIFICITY, SO WE USED THIS CUTOFF TO FIND OUT ACCURACY..

PREDICTION ON TEST DATA

MODEL VALIDATION ON TEST DATA



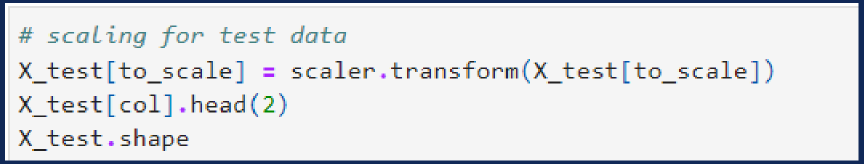
Custom Functions for Test

```
# we use these function to do the prediction on test data.
def logreg_test_pred_fn(fX_test, fy_test, fcol, fcutoff, fres):
    fX_test_sm = sm.add_constant(fX_test[fcol])
    fy_test_pred = fres.predict(fX_test_sm)
    fy_test_pred = fy_test_pred.values.reshape(-1)
    fy_test_pred_final = pd.DataFrame({'Converted':fy_test.values, 'Conv_Prob':fy_test_pred})
    fy_test_pred_final['ID'] = fy_test.index
    fy_test_pred_final['predicted'] = fy_test_pred_final.Conv_Prob.map(lambda x: 1 if x > fcutoff else 0)
    return fres, fy_test_pred,fy_test_pred_final

# this function is used to generate metrics.
def logreg_test_metrics_fn(fy_test_pred_final):
    fconfusion = confusion_matrix(fy_test_pred_final.Converted, fy_test_pred_final.predicted )
    faccuracy = accuracy_score(fy_test_pred_final.Converted, fy_test_pred_final.predicted)
    return fconfusion, faccuracy

# using this function we can see VIF score for multicollinearity
def logreg_test_VIF_score_fn(fX_test, fcol):
    fvif = pd.DataFrame()
    fvif['Features'] = fX_test[fcol].columns
    fvif['VIF'] = [variance_inflation_factor(fX_test[fcol].values, i) for i in range(fX_test[fcol].shape[1])]
    fvif['VIF'] = round(fvif['VIF'], 2)
    fvif = fvif.sort_values(by = "VIF", ascending = False)
    return fvif
```

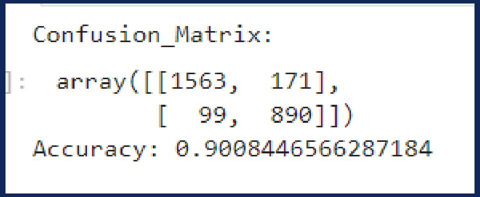
WE HAVE CREATED USER DEFINED FUNCTION WHICH WILL GIVE US PREDICTION VIF AND MATRIX ON TEST DATA SET

PREDICTION ON TEST DATA

MODEL VALIDATION ON TEST DATA

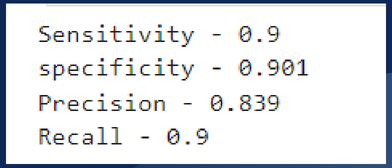


```
# scaling for test data
X_test[to_scale] = scaler.transform(X_test[to_scale])
X_test[col].head(2)
X_test.shape
```



```
Confusion_Matrix:
]:  array([[1563,  171],
           [  99,  890]])
Accuracy: 0.9008446566287184
```

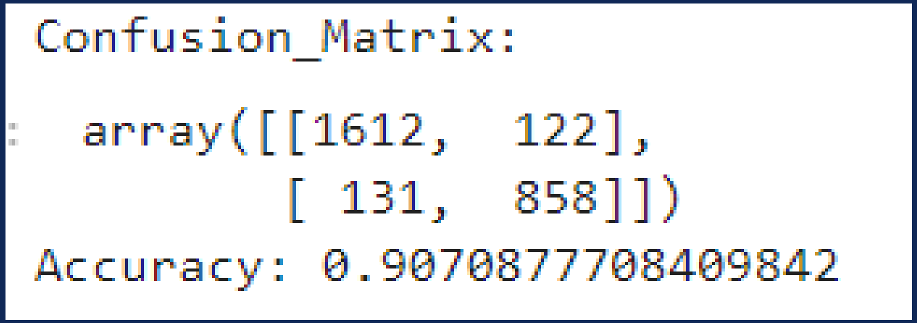
USING CUTOFF 0.30 WE CALCULATED SENSITIVITY SPECIFICITY,ACCURACY , CONFUSION MATRIX , PRECISION AND RECALL.



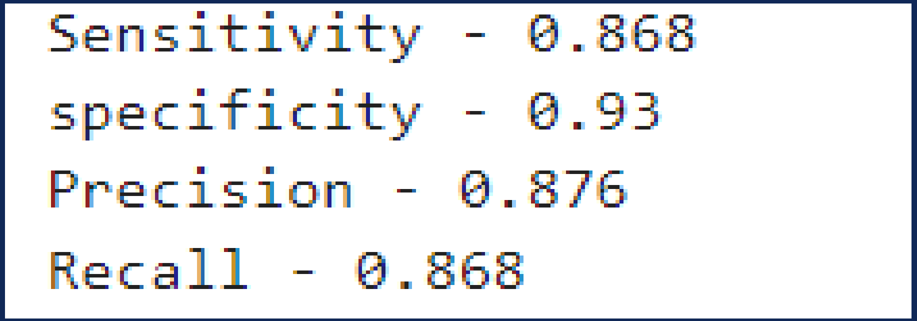
```
Sensitivity - 0.9
specificity - 0.901
Precision - 0.839
Recall - 0.9
```

CONCLUSION

MODEL VALIDATION ON TEST DATA



```
Confusion_Matrix:

:  array([[1612,  122],
          [ 131,  858]])
Accuracy: 0.9070877708409842
```



```
Sensitivity - 0.868
specificity - 0.93
Precision - 0.876
Recall - 0.868
```

THE OVERALL ACCURACY FOR APPROACH 01 IS **~90%,** THE PRECISION RECALL CURVE PROVIDES A HIGHER CUTOFF VALUE COMPARED TO SENSITIVITY AND SPECIFICITY THE METRICS SENSITIVITY PRECISION ARE IN THE RANGE THE OF **86 - 87%** WHILE SPECIFICITY IS **93%**

THANK YOU

- Vinod Yadav
- Vignesh Kumar
- Ujjwal Verma