LEAD SCORING CASE STUDY

LOGISTIC REGRESSION

GROUP MEMBERS:

UNNATI GARG

SINGH NIDHI KUMARI SANJEEV

KUMARI NEHA

DS C-50

CONTENTS

- Problem Statement
- Goals and Business Objective
- Python Libraries Used
- Steps to be followed for analysis
- How to implement the approach
- Analysis of Data
- Conclusion

PROBLEM STATEMENT

- An X Education sells online courses to industry professionals and need help to select the most promising leads, i.e. the leads that are most likely to convert in to paying customers
- The company requires us to build a model where in we need to assign a lead score to each of the leads such that the customers with higher lead score have a higher conversion chance and the customers with lower lead score have a lower conversion chance
- If we 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
- The CEO has given a ballpark of the target lead conversion rate to be around 80%

GOALS AND BUSINESS OBJECTIVE

- X education wants to know most promising leads. For that they want to build a Model which identifies the hot leads
- Build a logistic regression model to assign a lead score between 0 to 100 each of the leads which can be used the company to target potential leads
- Lead with higher score considered as Hot (most likely to get converted) while with lower score considered as Cold (less likely to get converted)
- Deployment of the model for the future use.

STEPS TO BE FOLLOWED FOR ANALYSIS

- Import required python libraries and dataset
- Data Cleaning
- EDA
  - Inspecting the data frame
  - Checking for missing or null values
  - Analyze the colums and handling the missing values
  - Handle outliers
- Create dummy variables
- Convert to binary variables:Yes/No to 1/0
- Group column features
- Building the Model
- Train-test Split
- Plot ROC Curve
- Find optimal cut off points
- Calculate Accuracy, Sensitivity and Specificity
- Calculate Precision and Recall
- Validating the model
- Deploying the model
- Prediction on test set
- Conclusion and Result

PYTHON LIBRARIES USED

- Numpy
- Pandas
- Matplotlib
- Seaborn
- Sklearn.model_selection import train_test_split
- Sklearn.preprocessing import StandardScaler
- Sklearn.linear_model import LogisticRegression
- Statsmodel.api
- Statsmodel.stats.outliers_influence import variance_inflation_factor
- Sklearn import metrics
- sklearn,.metrics import confusion_matrix
- sklearn,.metrics import precison_score, recall_score
- sklearn,.metrics import precison_recall_curve

DATA CLEANING

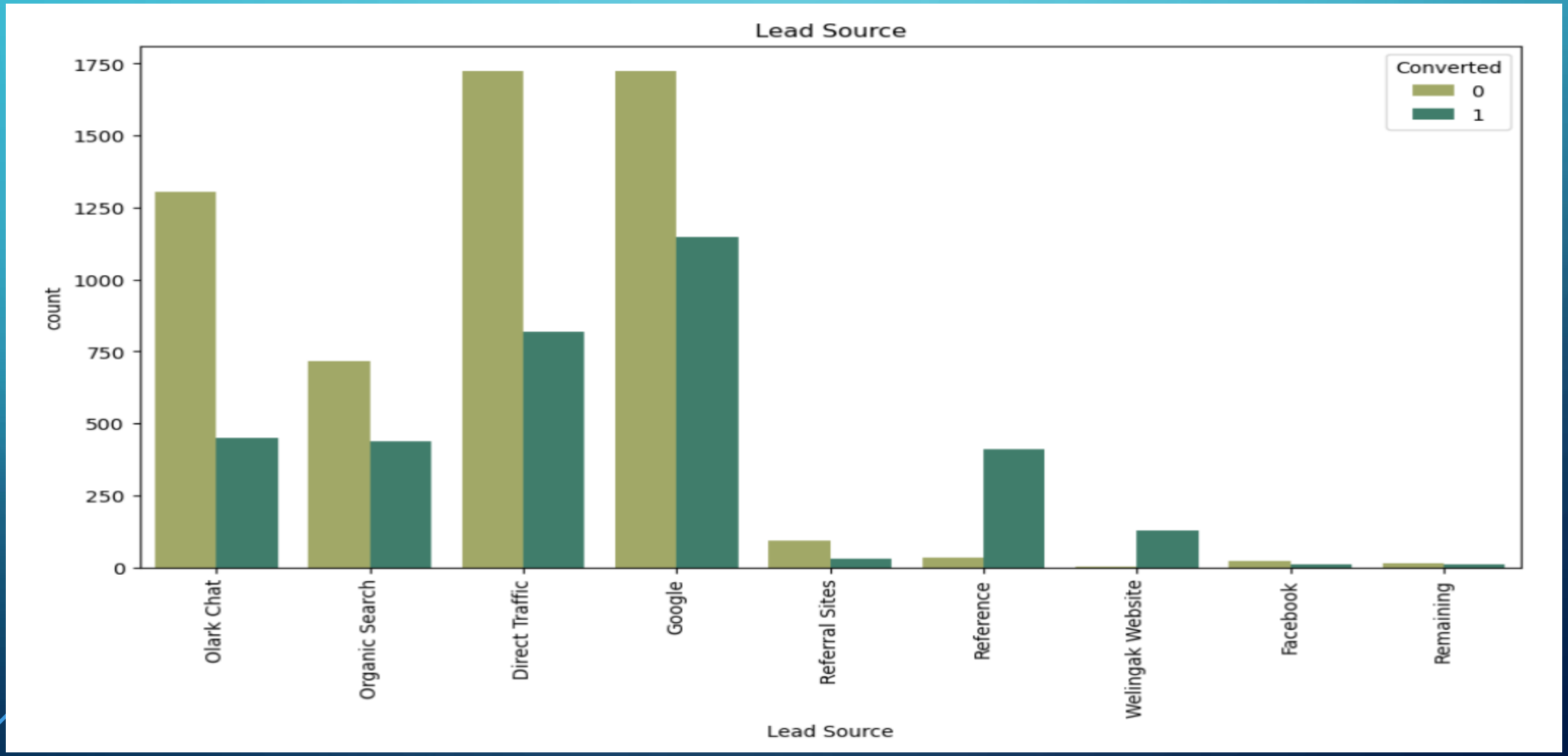
- Check number of rows and columns in the dataset
- Check for null or default values
- Replace default values with NAN
- Handling some columns with null values
- Drop null values

EDA

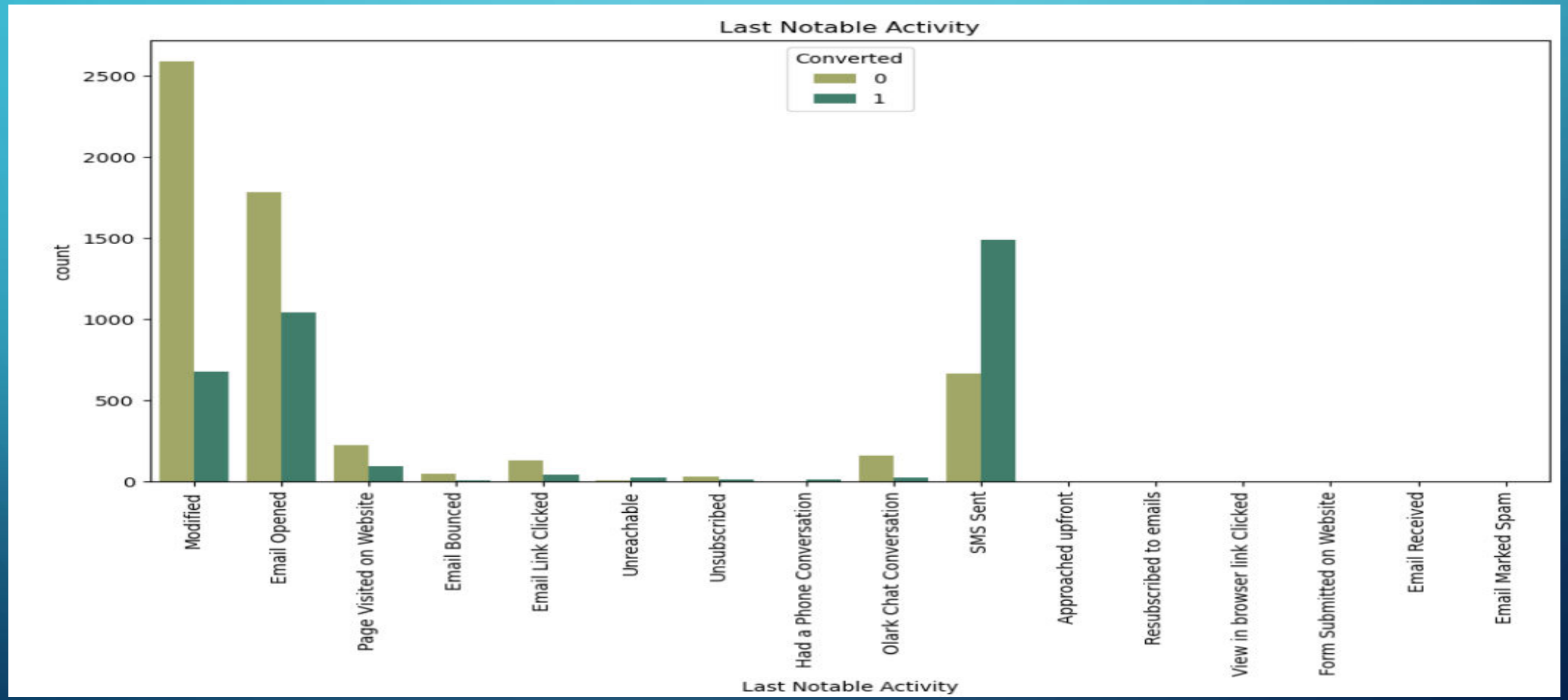
EDA

- Visualization performed with attributes of dataset
- Most of the attributes were not found useful for analysis
- Useful attributes are “Lead Score”, “Last Activity”, “Lead Origin” and “Total time spent on website”

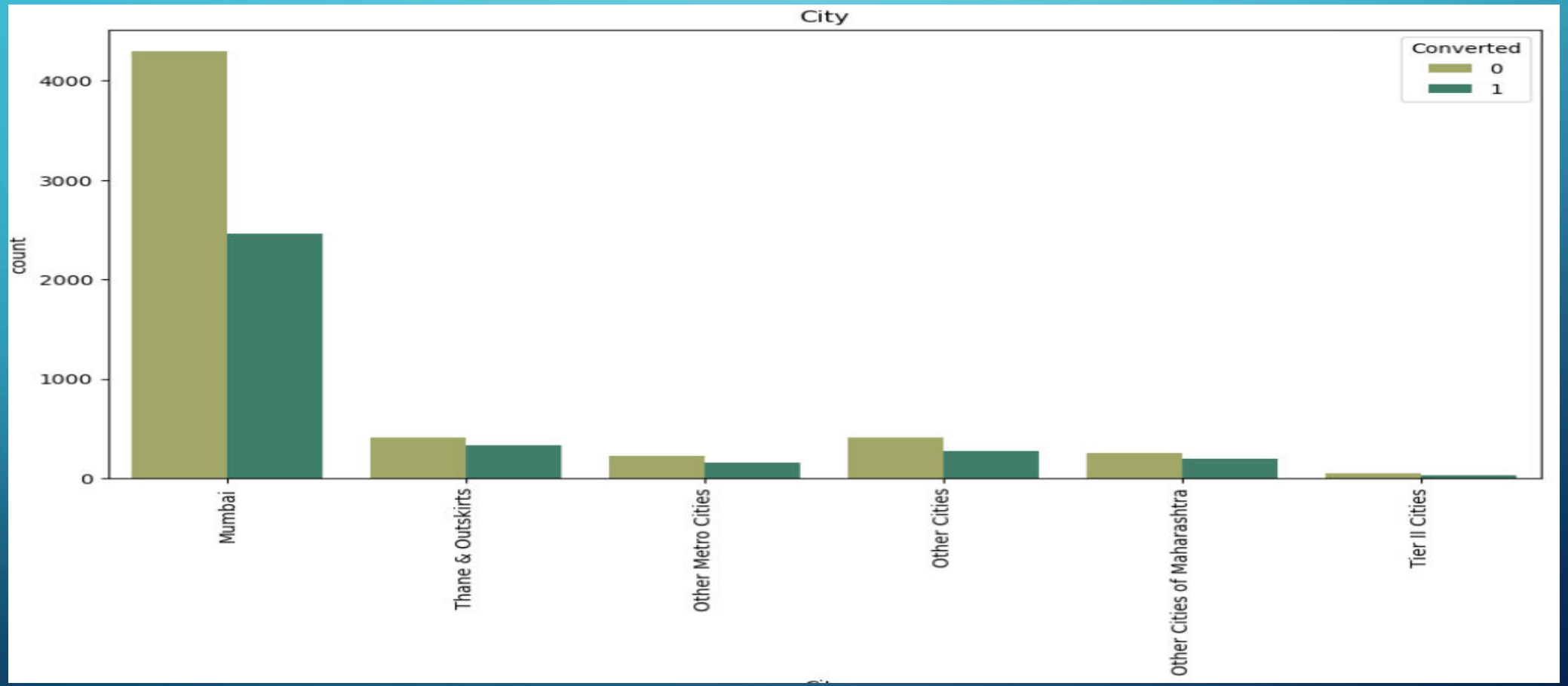
LEAD SCORE


Lead Source
Converted
0
1
count
1750
1500
1250
1000
750
500
250
0
Olark Chat
Organic Search
Direct Traffic
Google
Referral Sites
Reference
Welingak Website
Facebook
Remaining
Lead Source

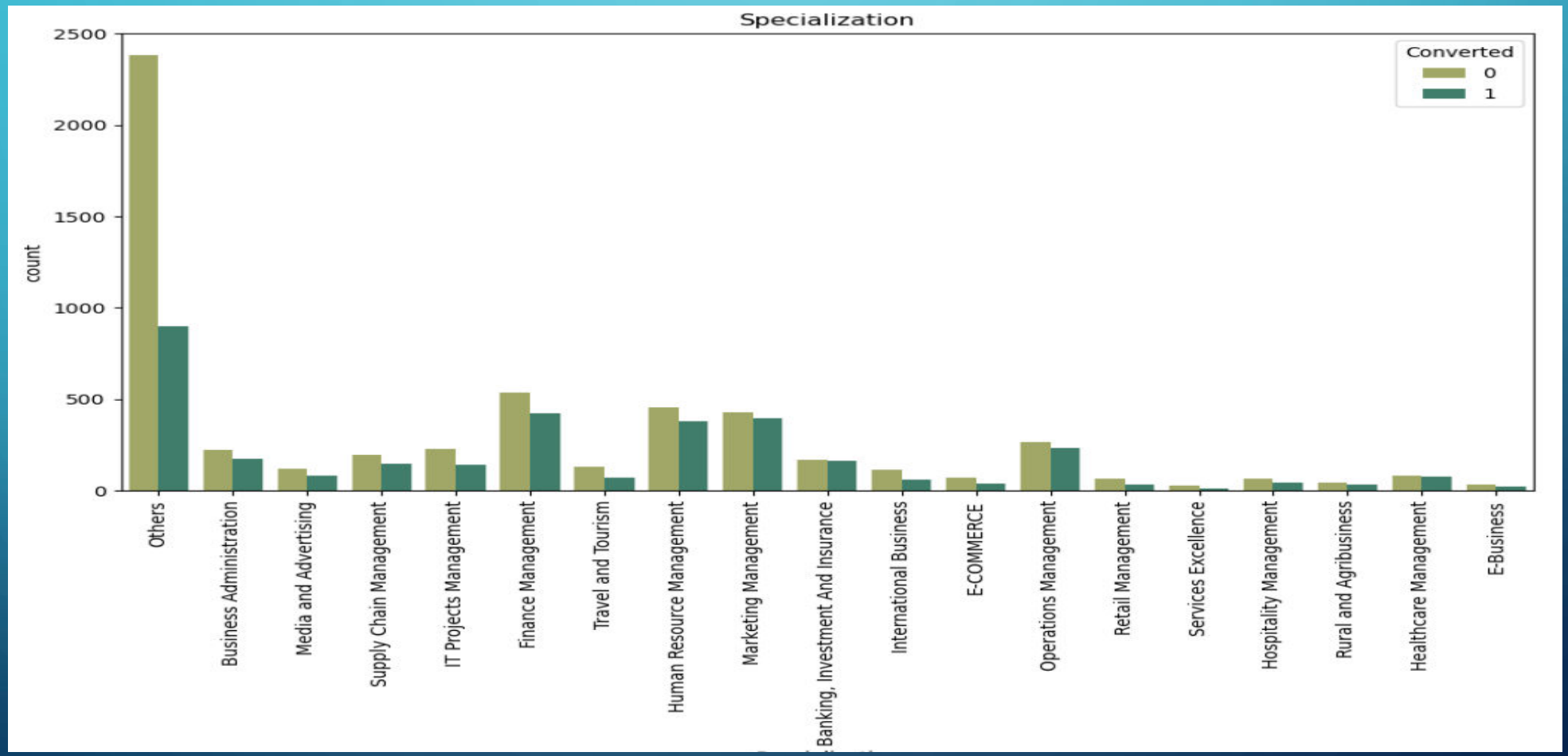
LAST NOTABLE ACTIVITY



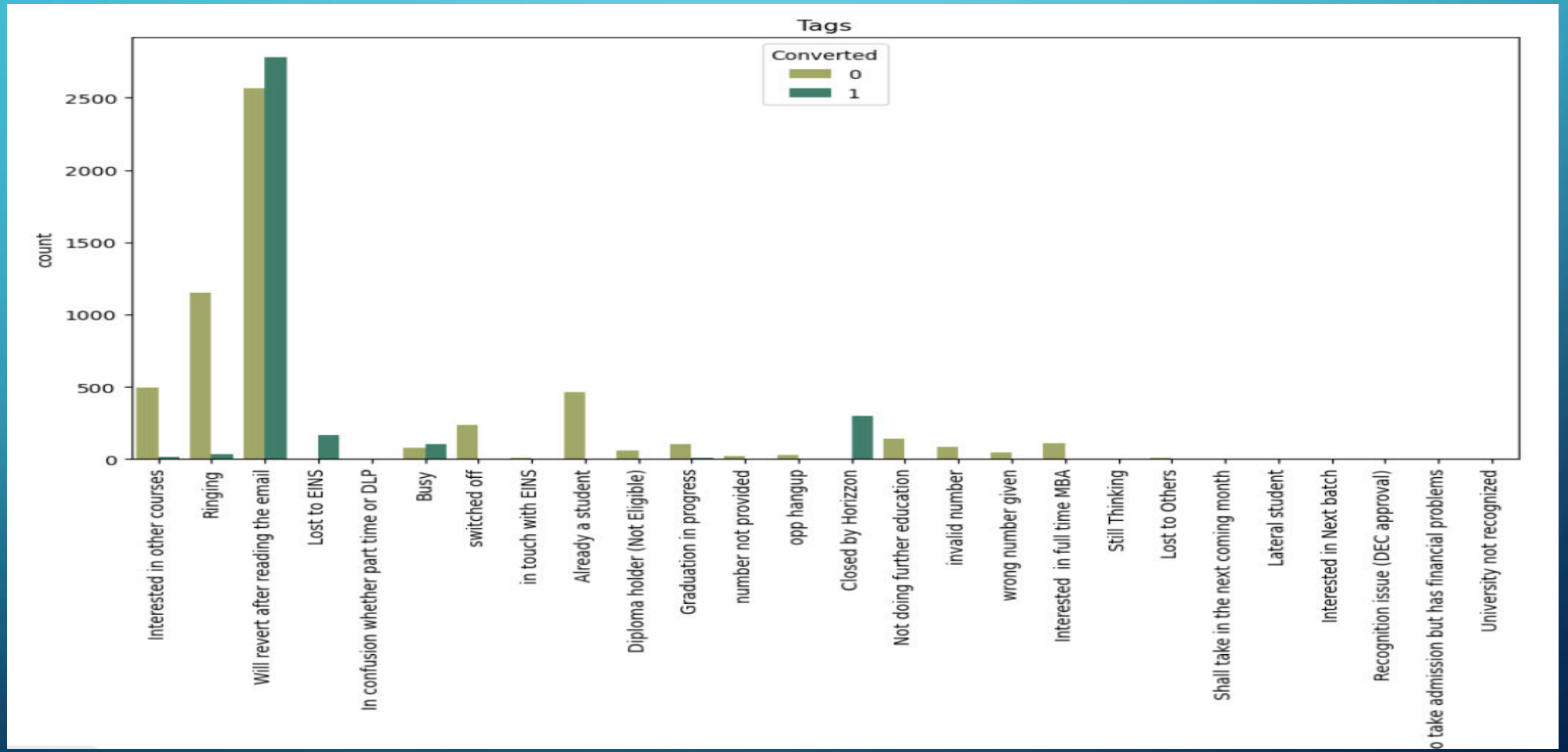
CITY


City
Converted
0
1
count
4000
3000
2000
1000
0
Mumbai
Thane & Outskirts
Other Metro Cities
Other Cities
Other Cities of Maharashtra
Tier II Cities

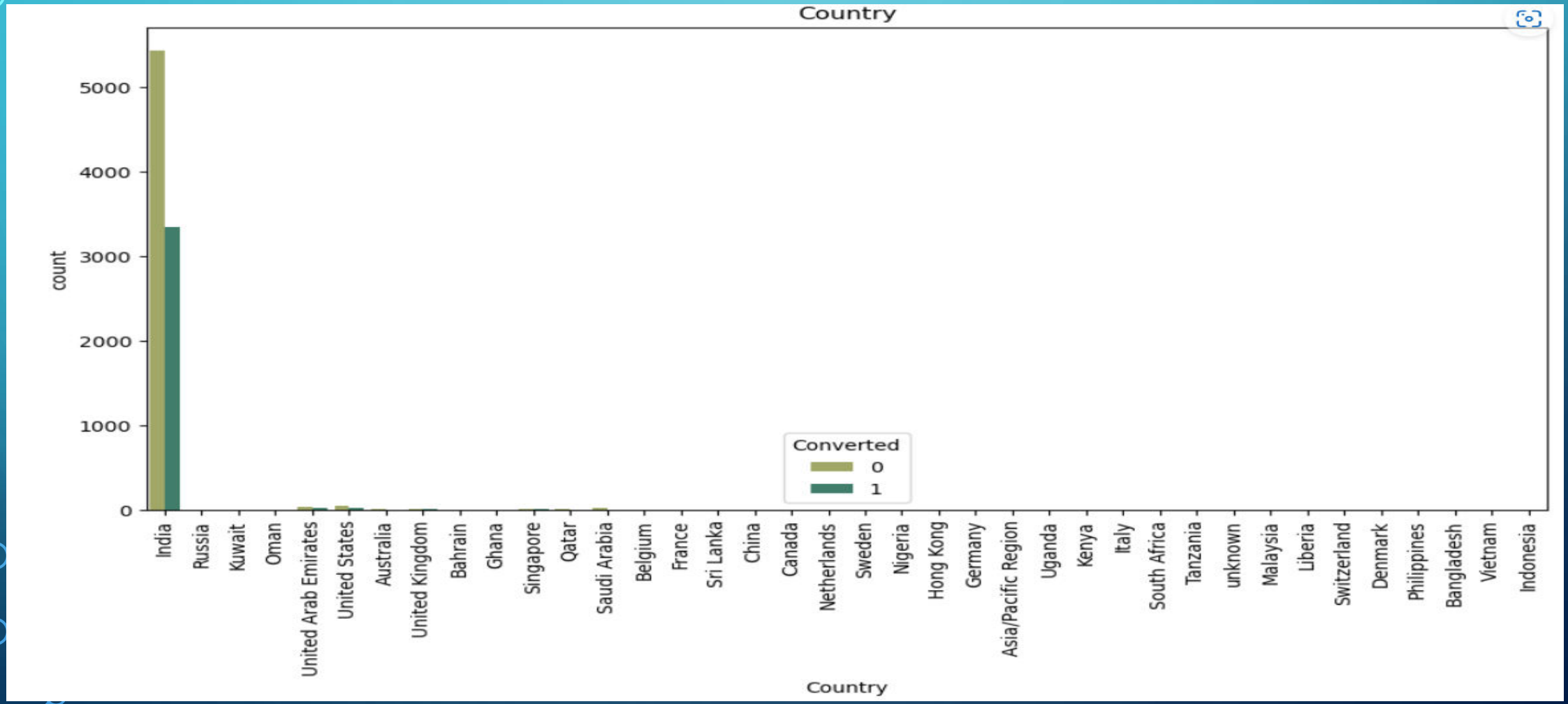
SPECIALIZATION



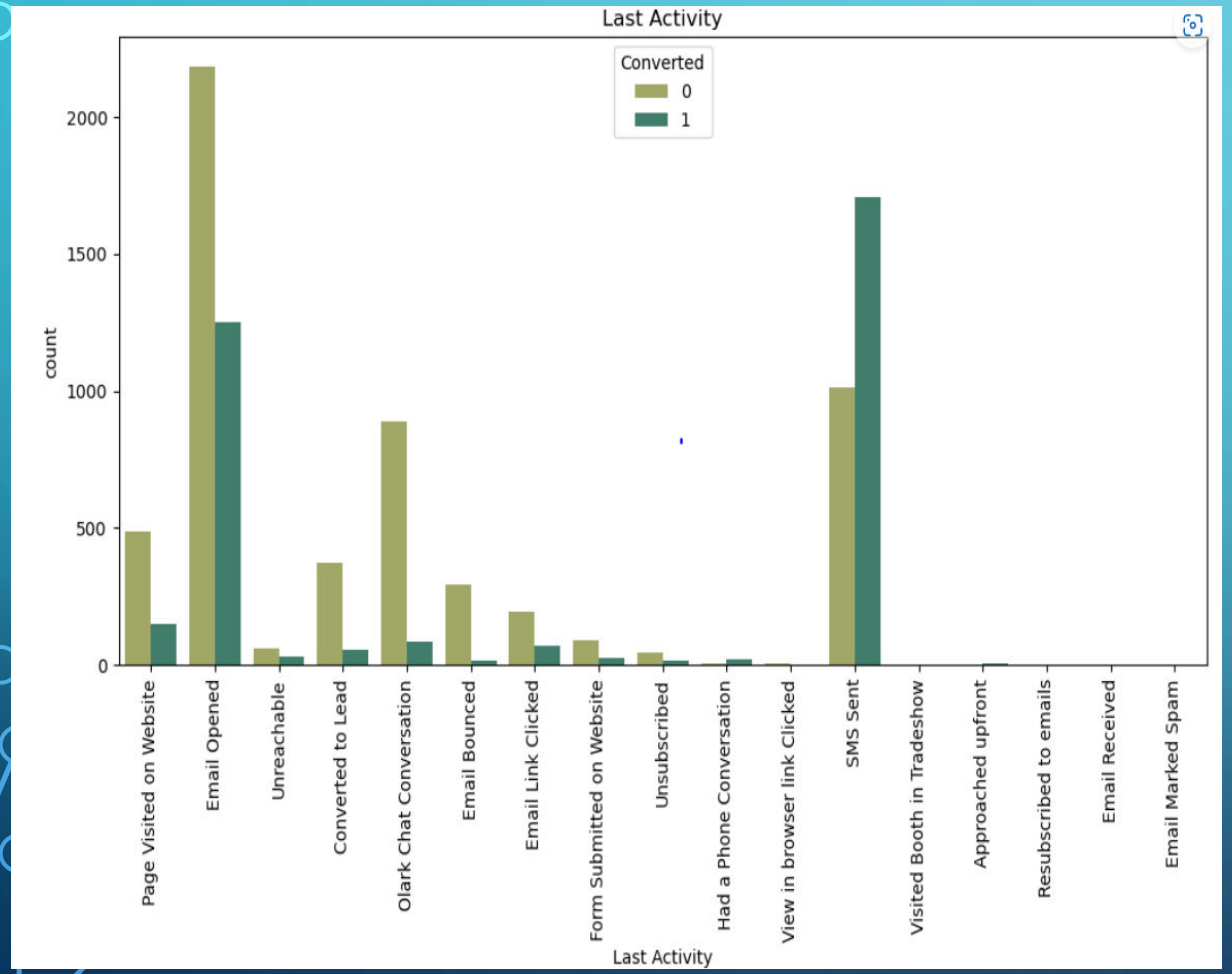
TAGS

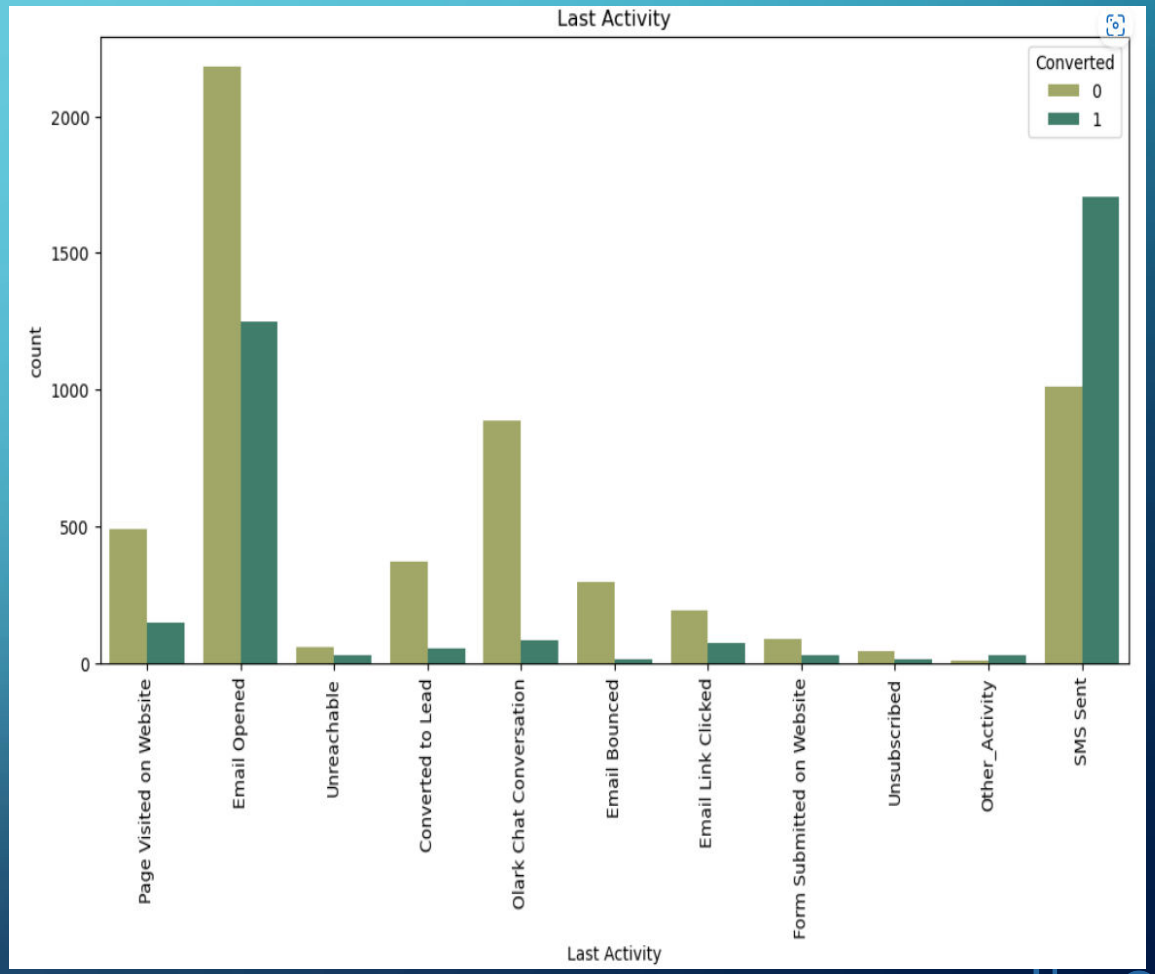

Tags
Converted
0
1
count
2500
2000
1500
1000
500
0
Interested in other courses
Ringing
Will revert after reading the email
Lost to EINS
In confusion whether part time or DLP
Busy
switched off
in touch with EINS
Already a student
Diploma holder (Not Eligible)
Graduation in progress
number not provided
opp hangup
Closed by Horizzon
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)
University not recognized

COUNTRY

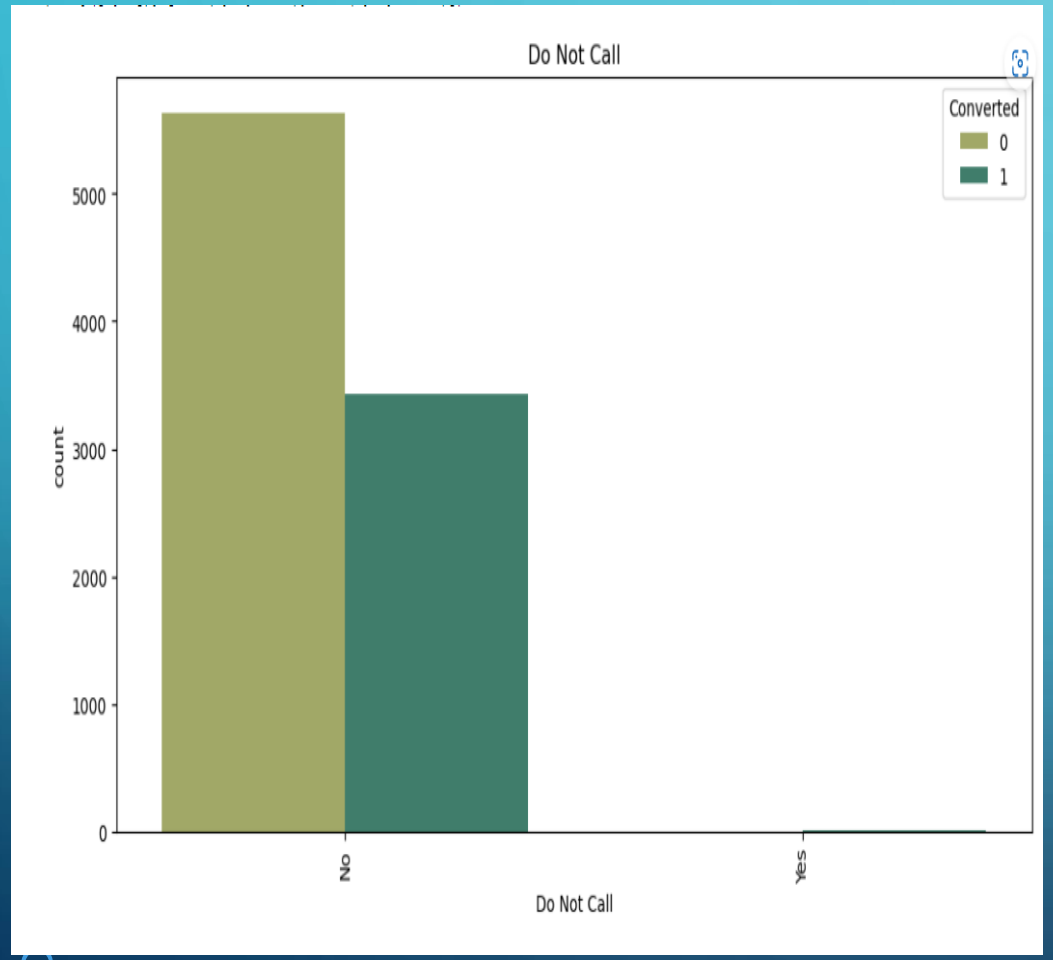

Country
count
5000
4000
3000
2000
1000
0
Converted
0
1
India
Russia
Kuwait
Oman
United Arab Emirates
United States
Australia
United Kingdom
Bahrain
Ghana
Singapore
Qatar
Saudi Arabia
Belgium
France
Sri Lanka
China
Canada
Netherlands
Sweden
Nigeria
Hong Kong
Germany
Asia/Pacific Region
Uganda
Kenya
Italy
South Africa
Tanzania
unknown
Malaysia
Liberia
Switzerland
Denmark
Philippines
Bangladesh
Vietnam
Indonesia
Country

LAST ACTIVITY - MERGE THE FIELDS WITH LESS COUNTS AS "OTHER_ACTIVITY"

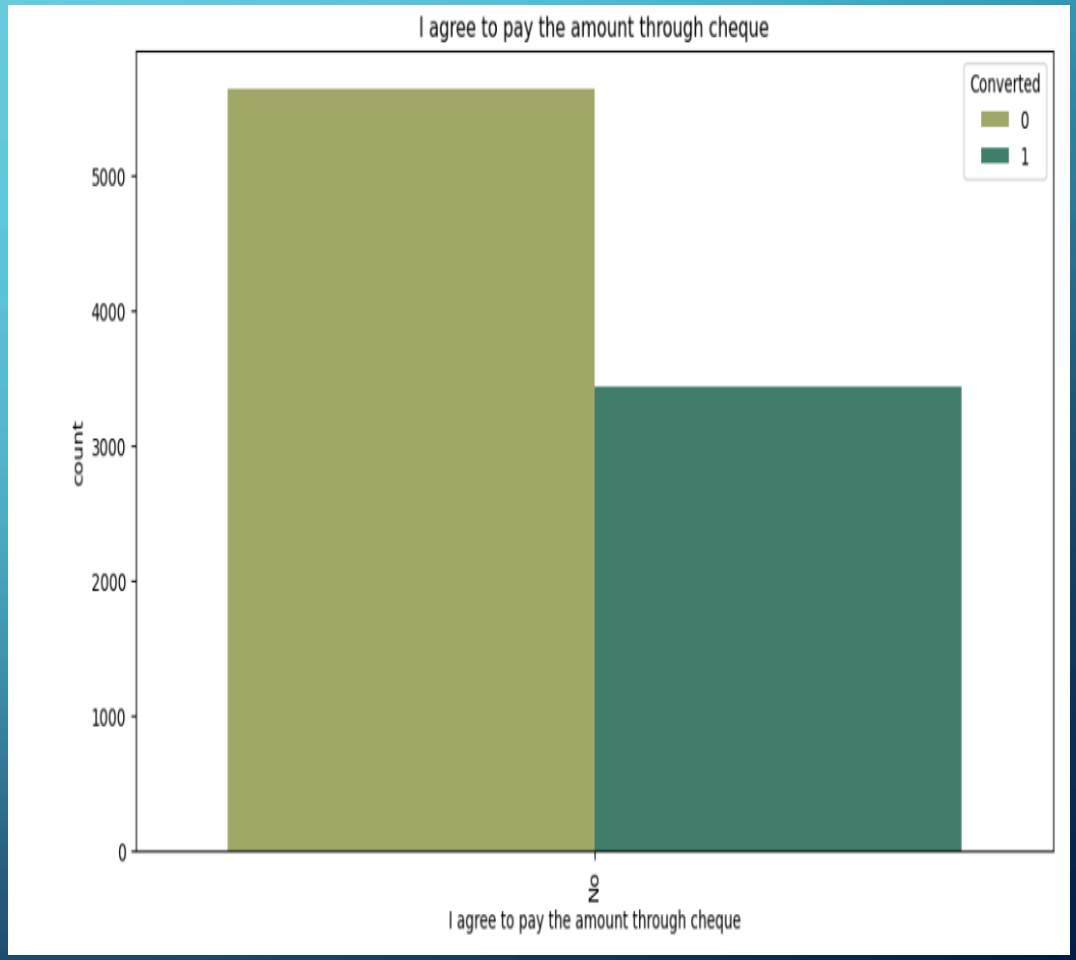




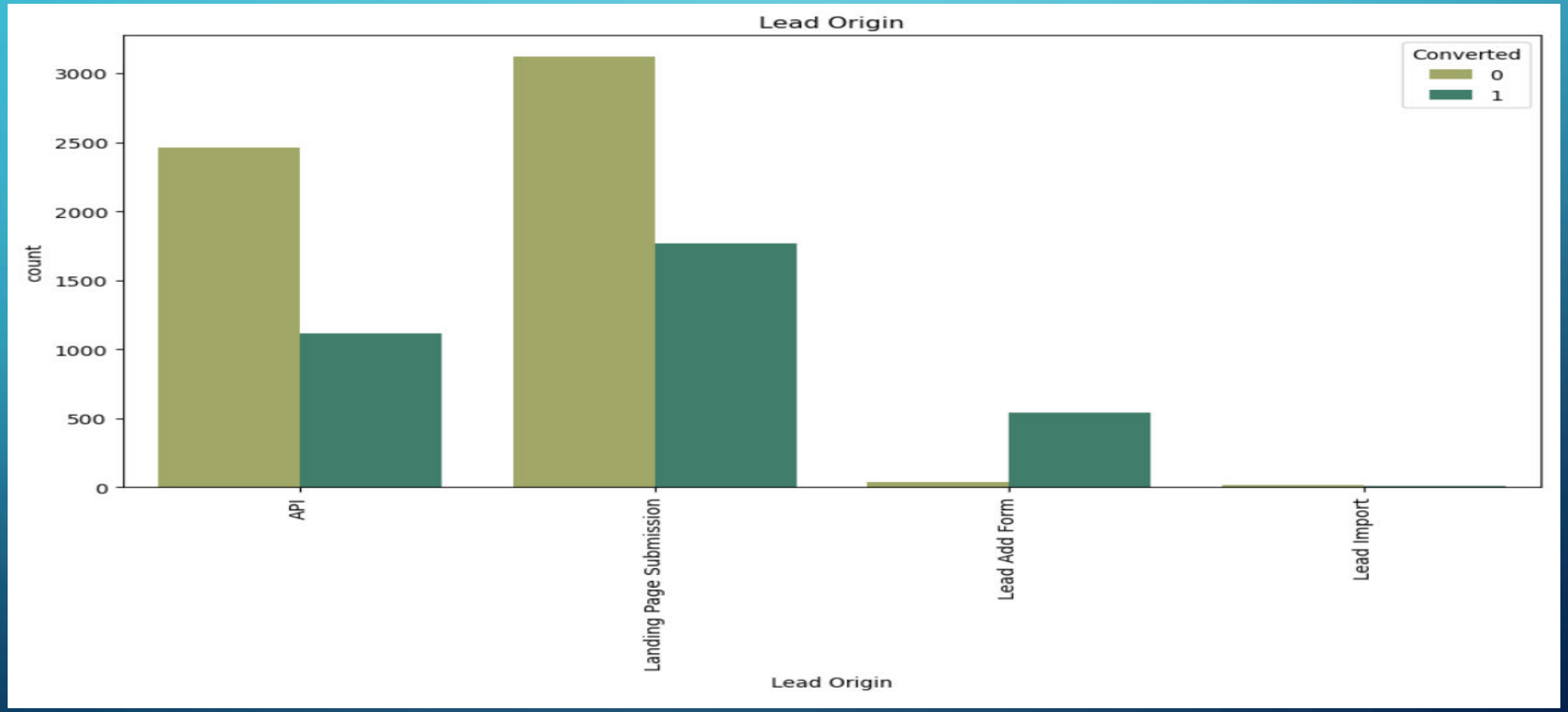
DO NOT EMAIL



I AGREE TO PAY THE AMOUNT THROUGH CHEQUE

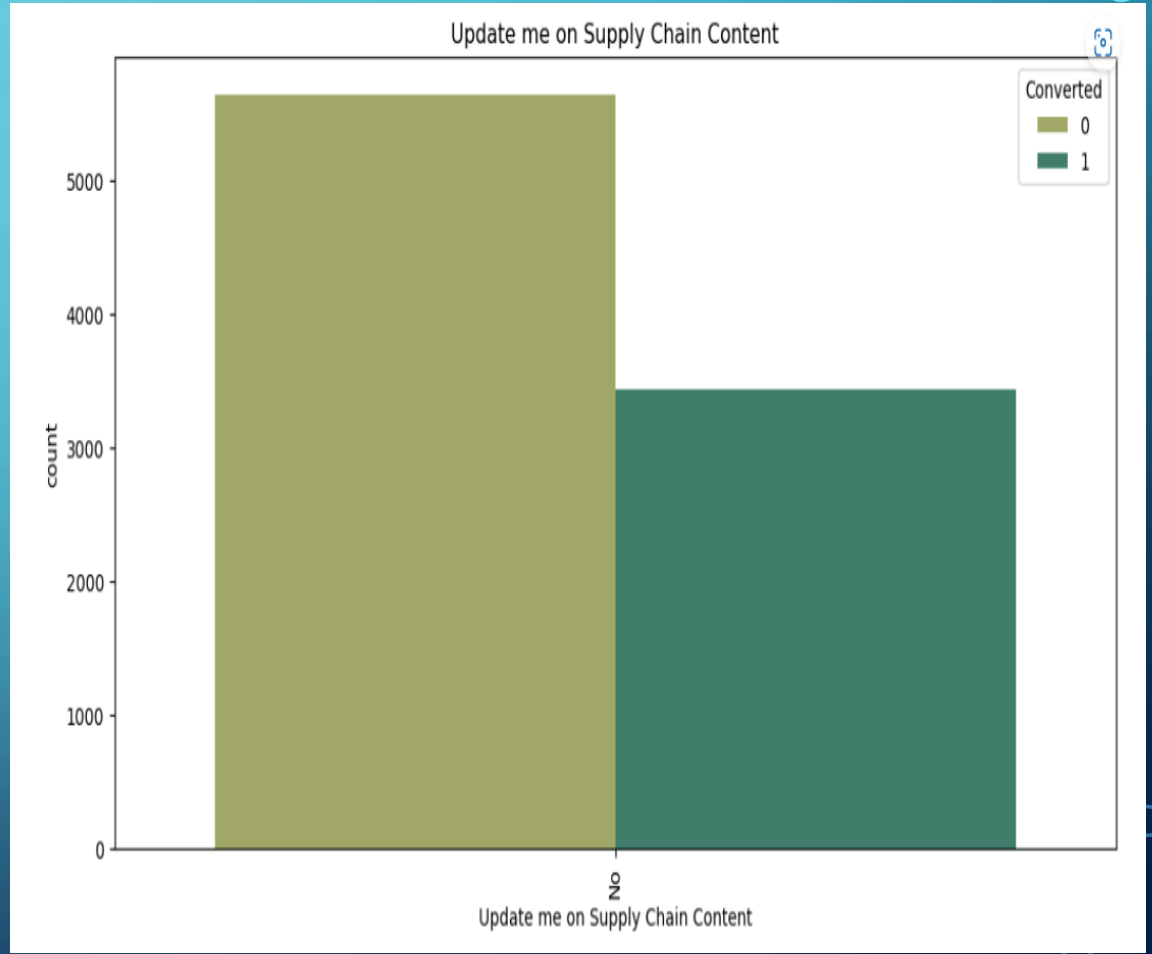


LEAD ORIGIN

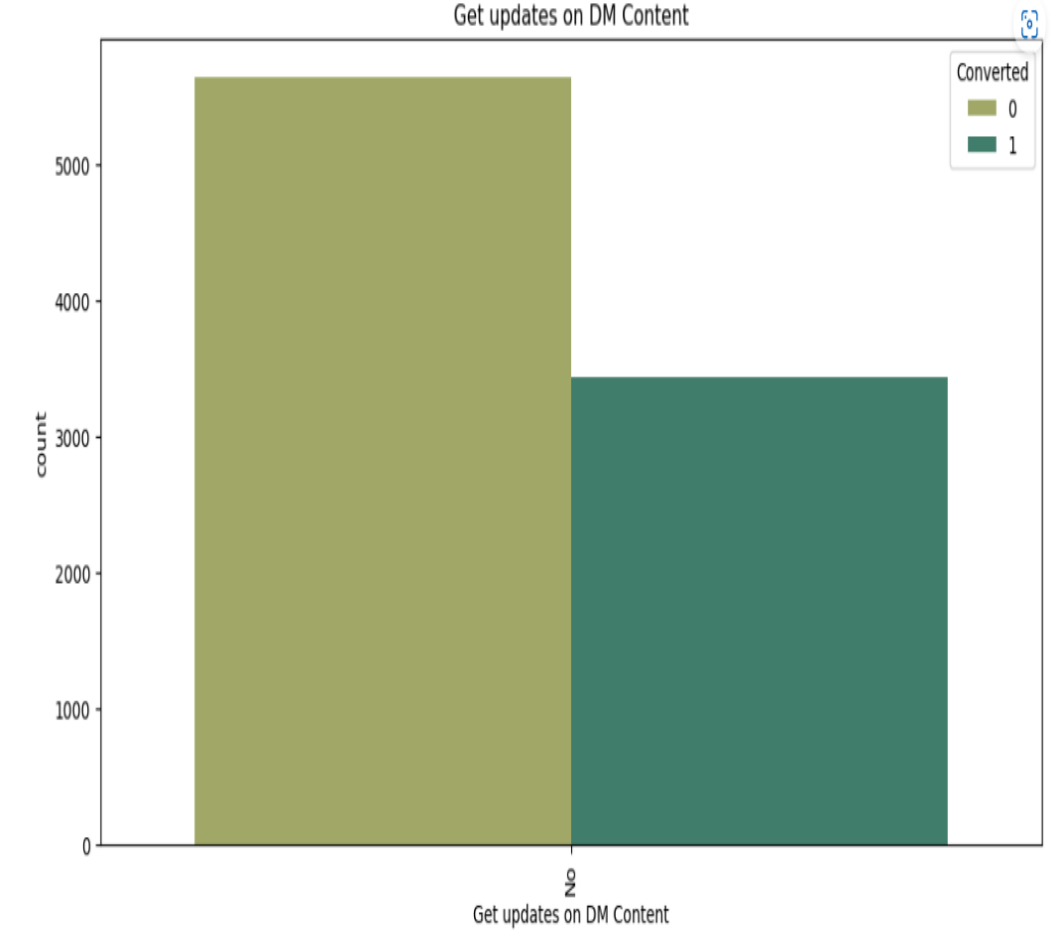

Lead Origin
Converted
0
1
count
3000
2500
2000
1500
1000
500
0
API
Landing Page Submission
Lead Add Form
Lead Import
Lead Origin

A FREE COPY OF MASTERING THE INTERVIEW

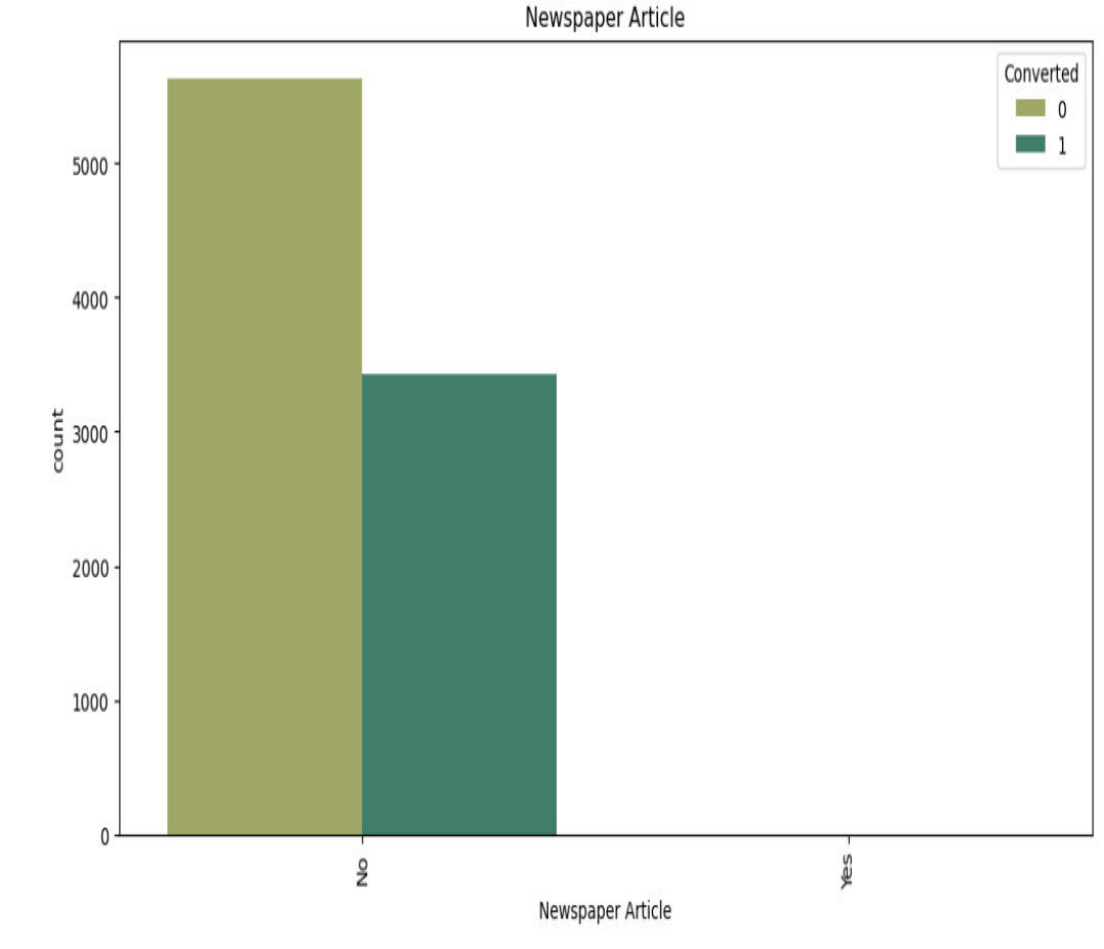
UPDATE ME ON SUPPLY CHAIN CONTENT



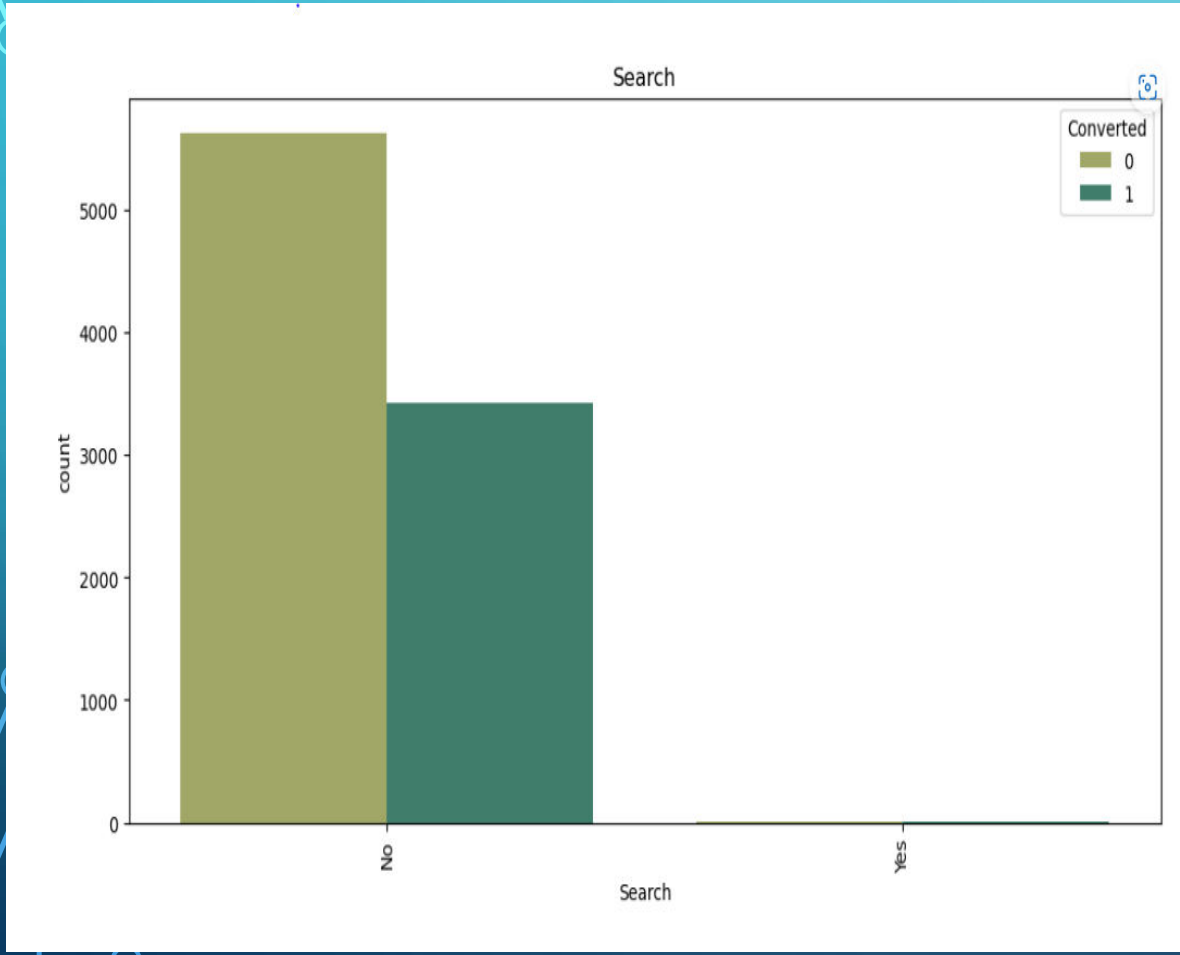
GET UPDATES ON DM CONTENT



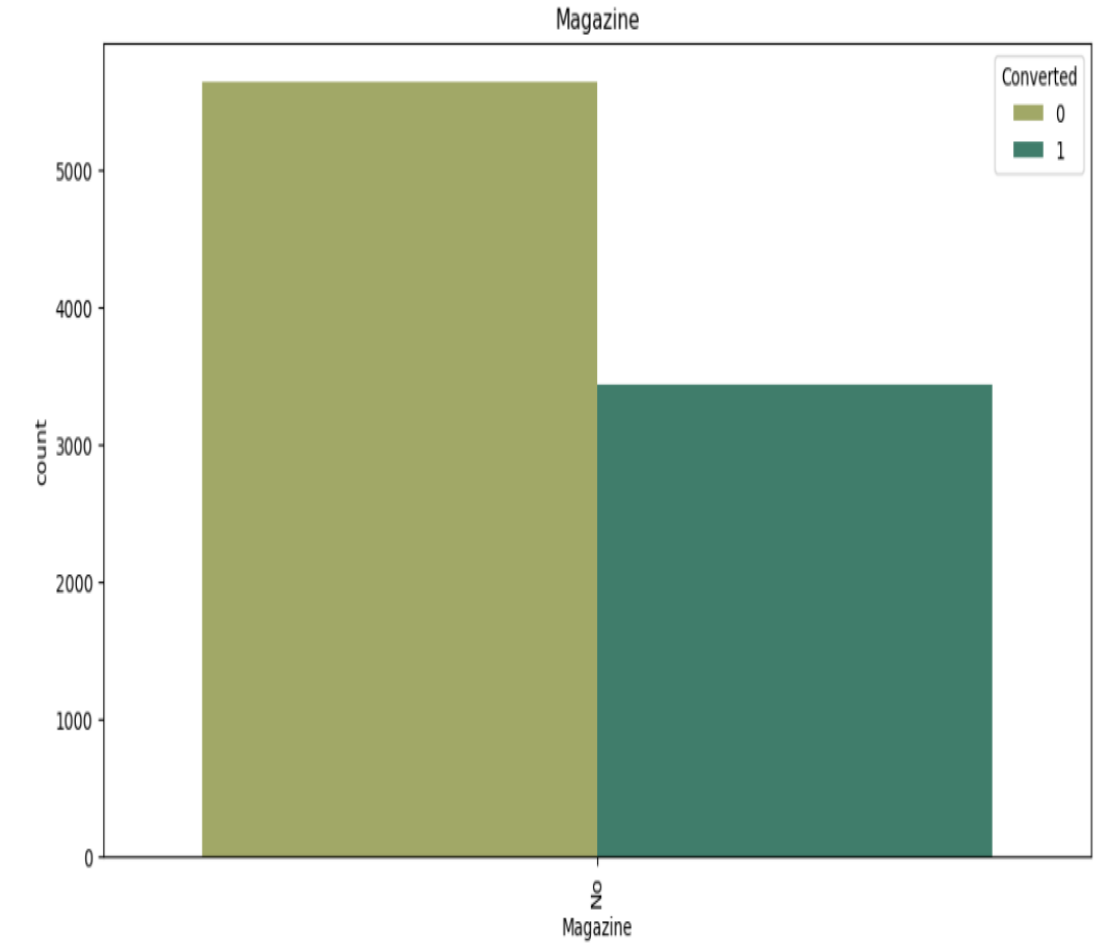
NEWSPAPER ARTICLE



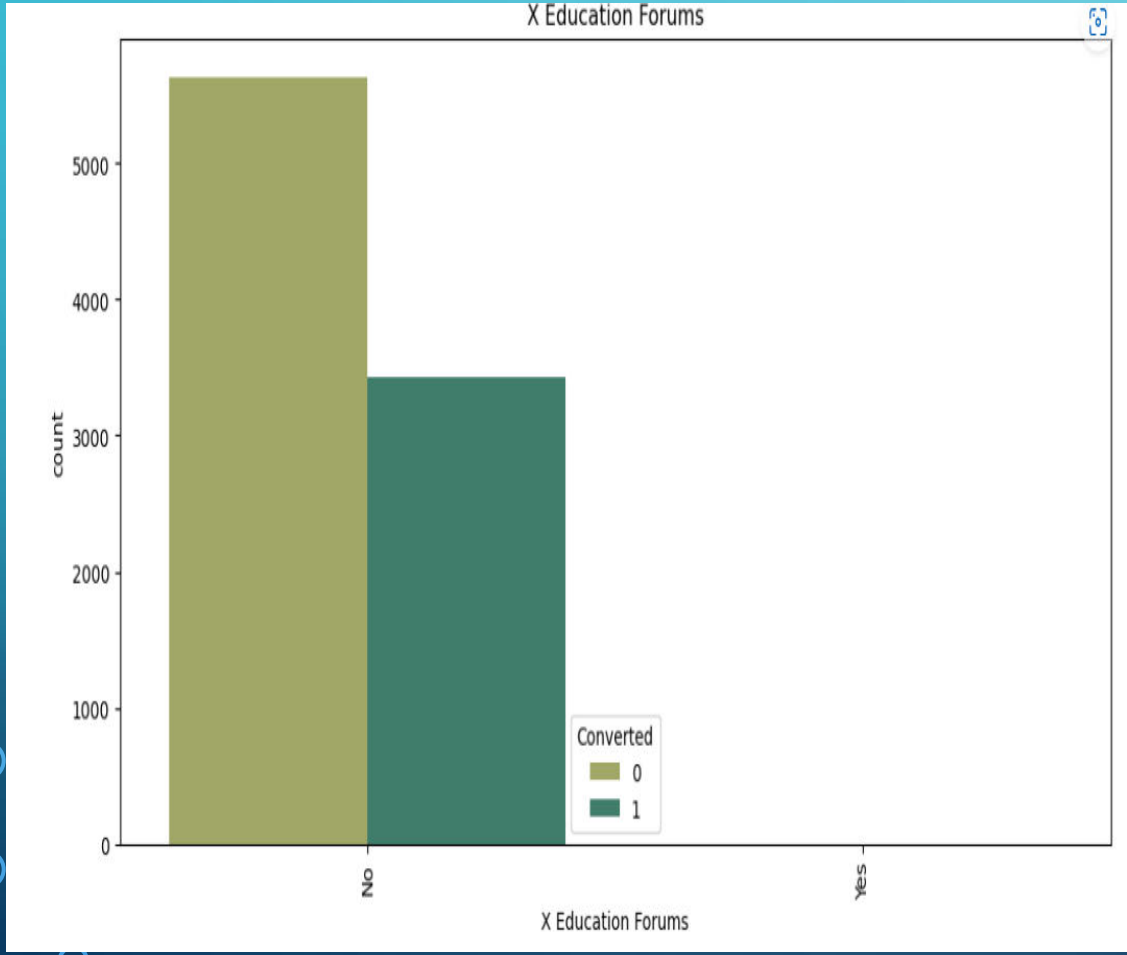
SEARCH



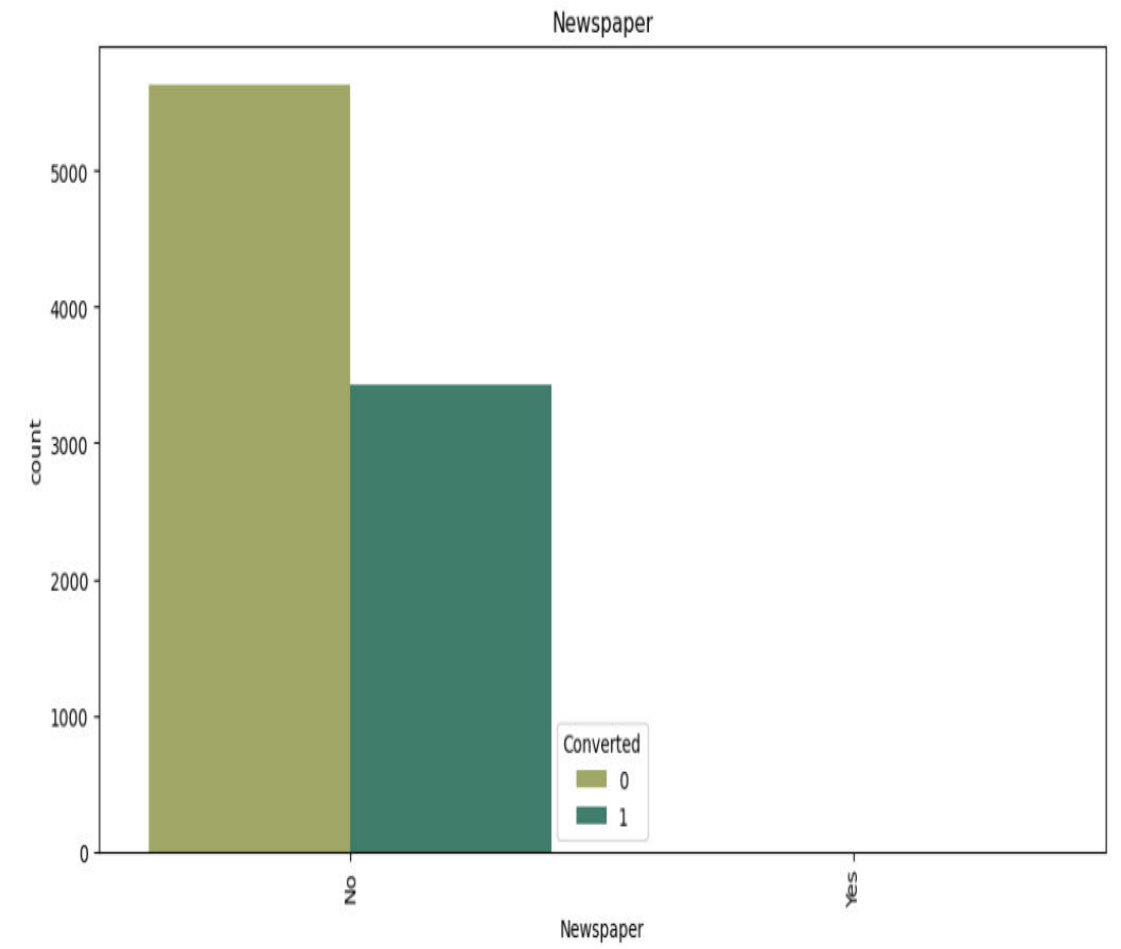
MAGAZINE



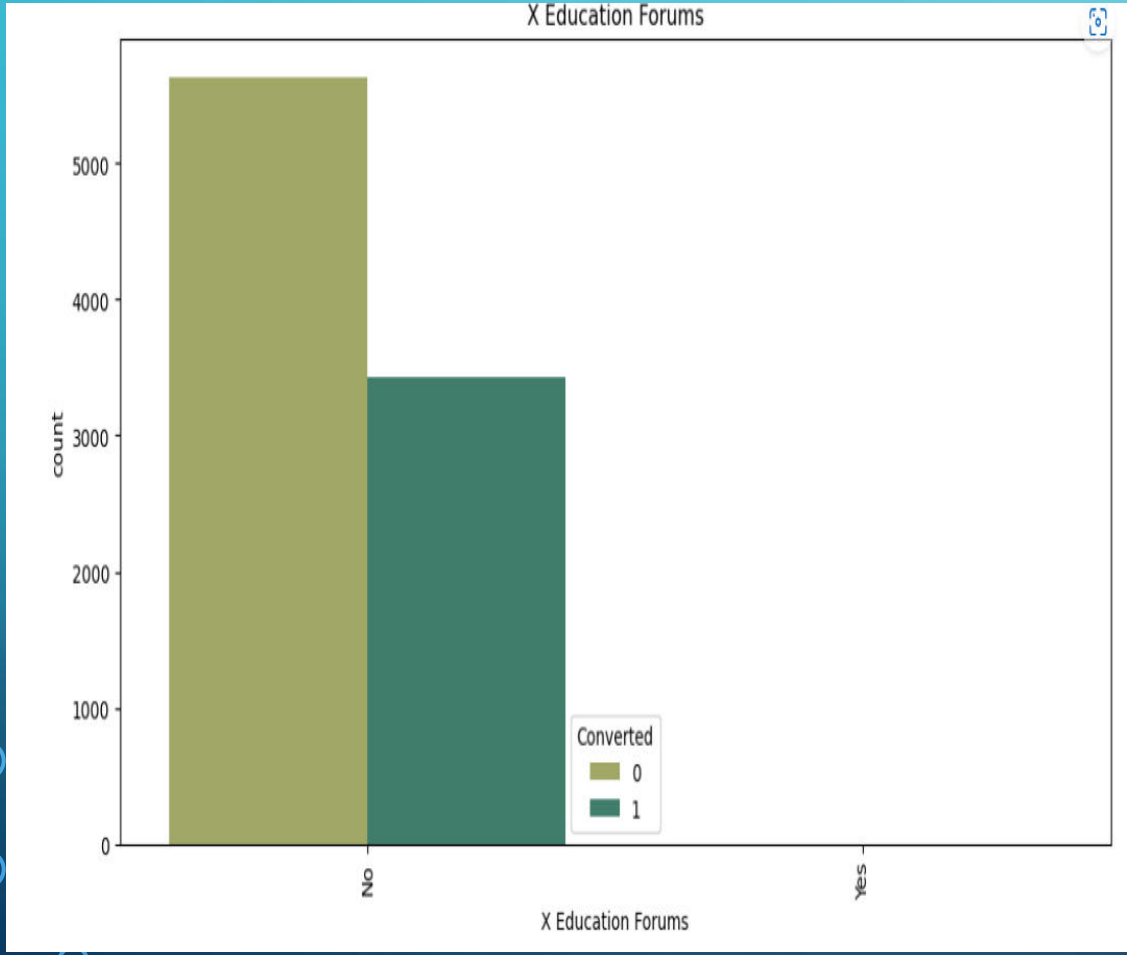
X EDUCATION FORUMS



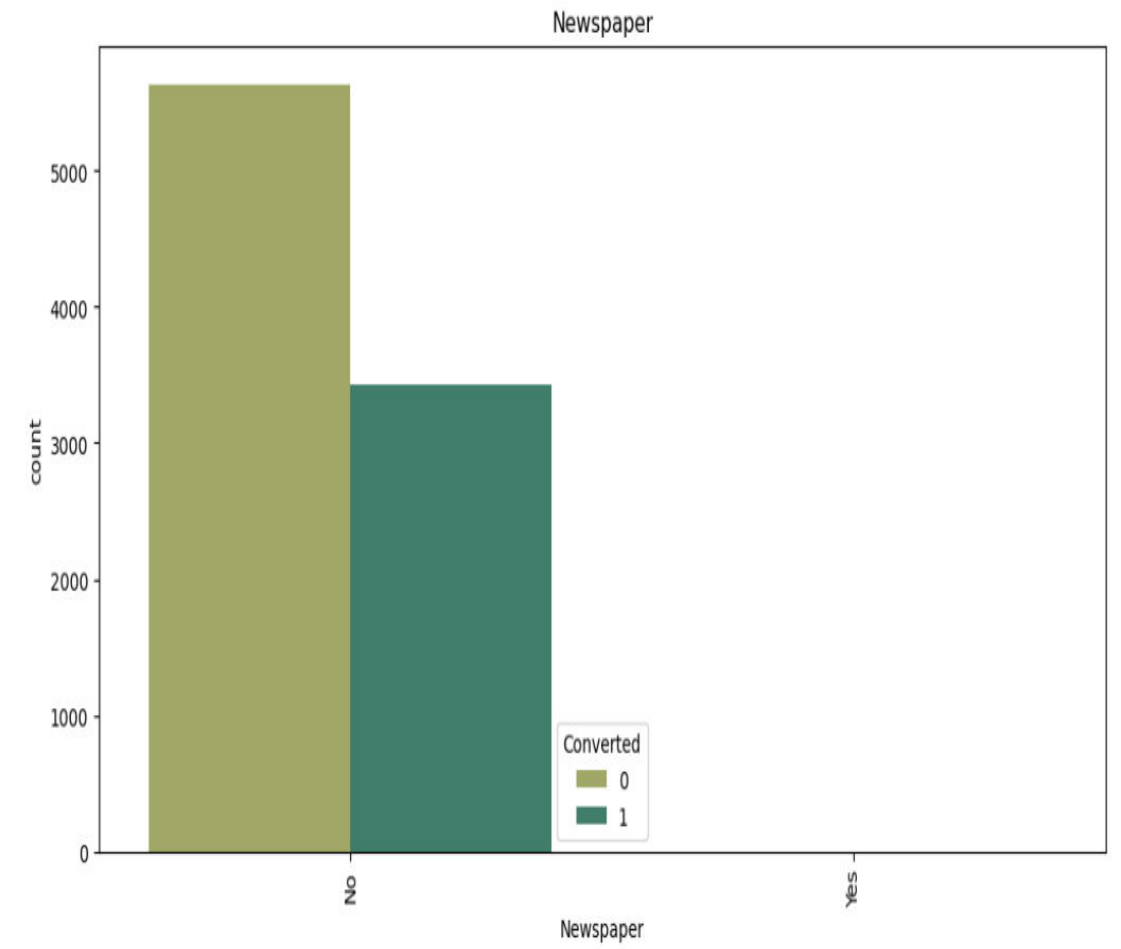
NEWSPAPER



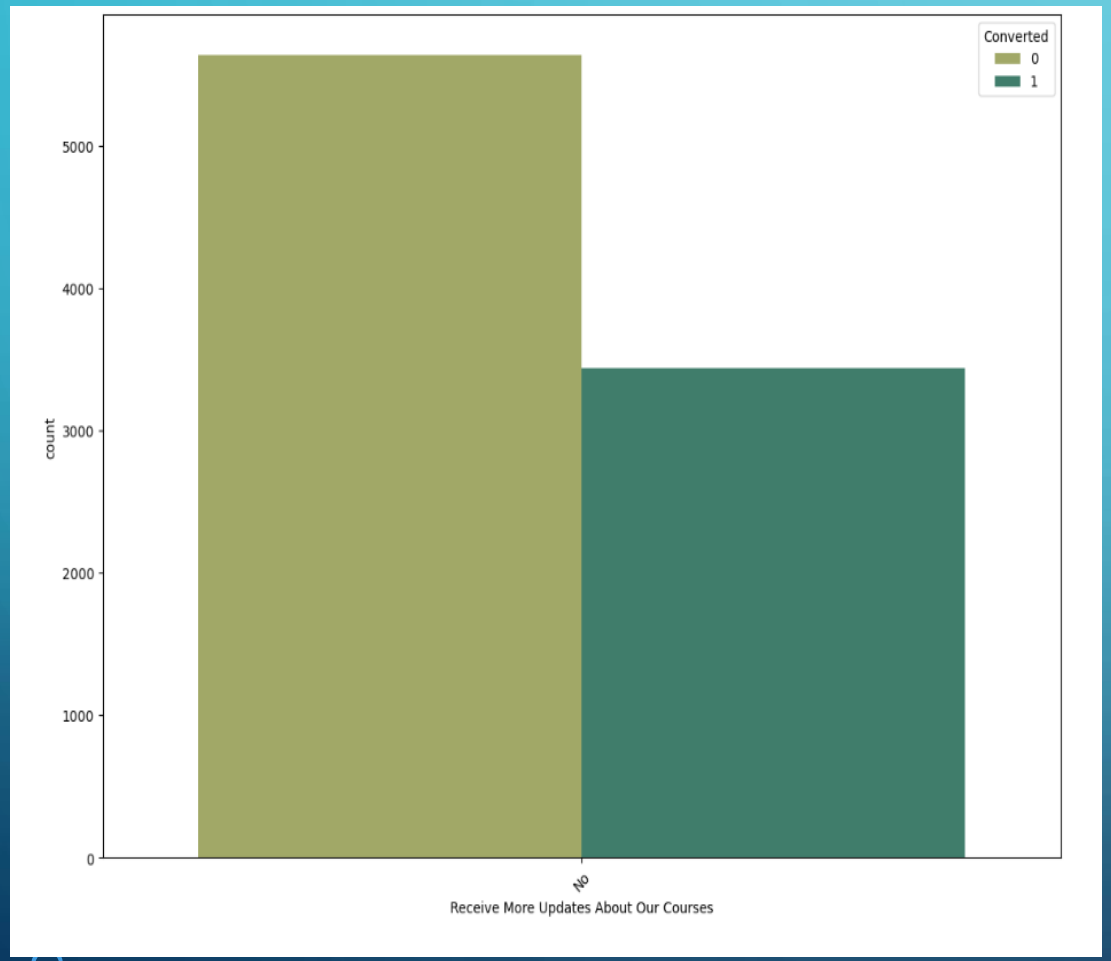
X EDUCATION FORUMS



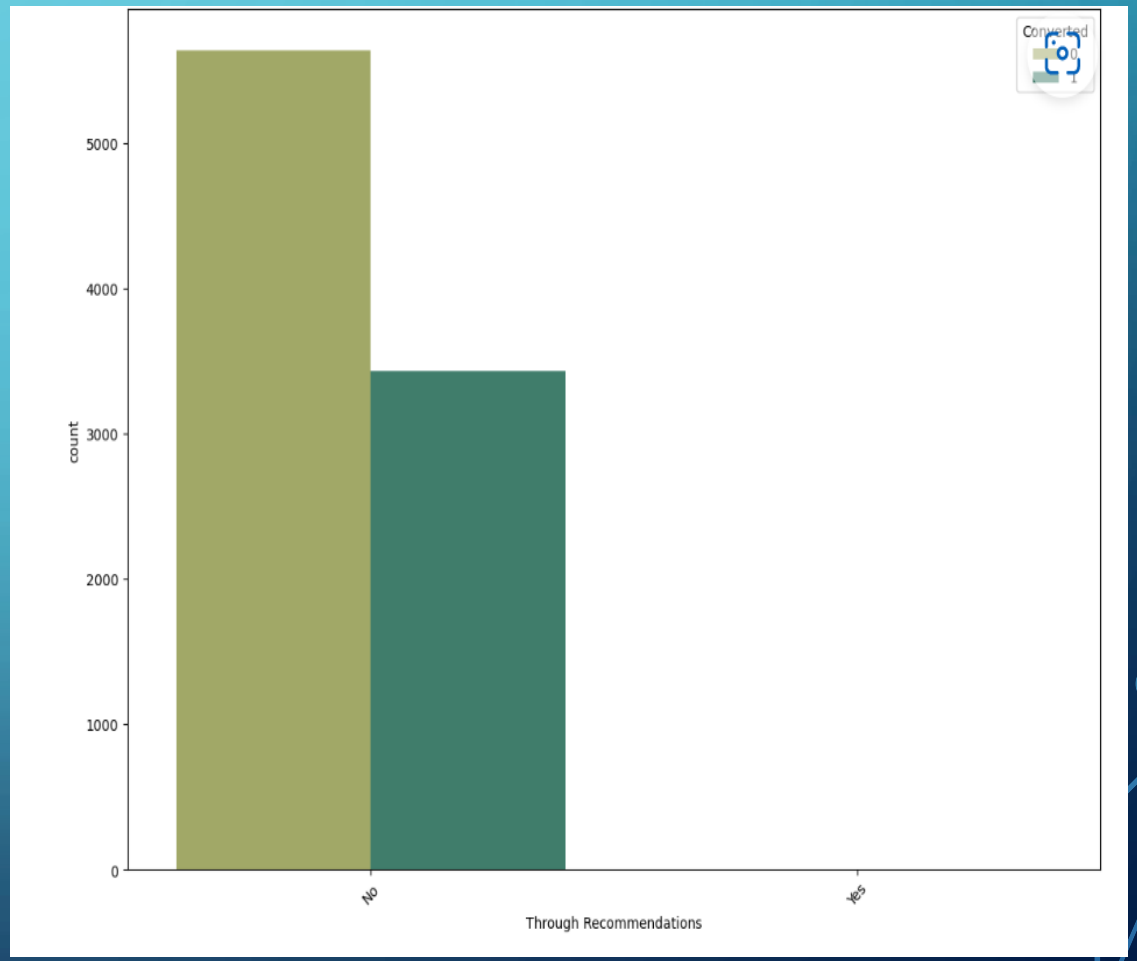
NEWSPAPER



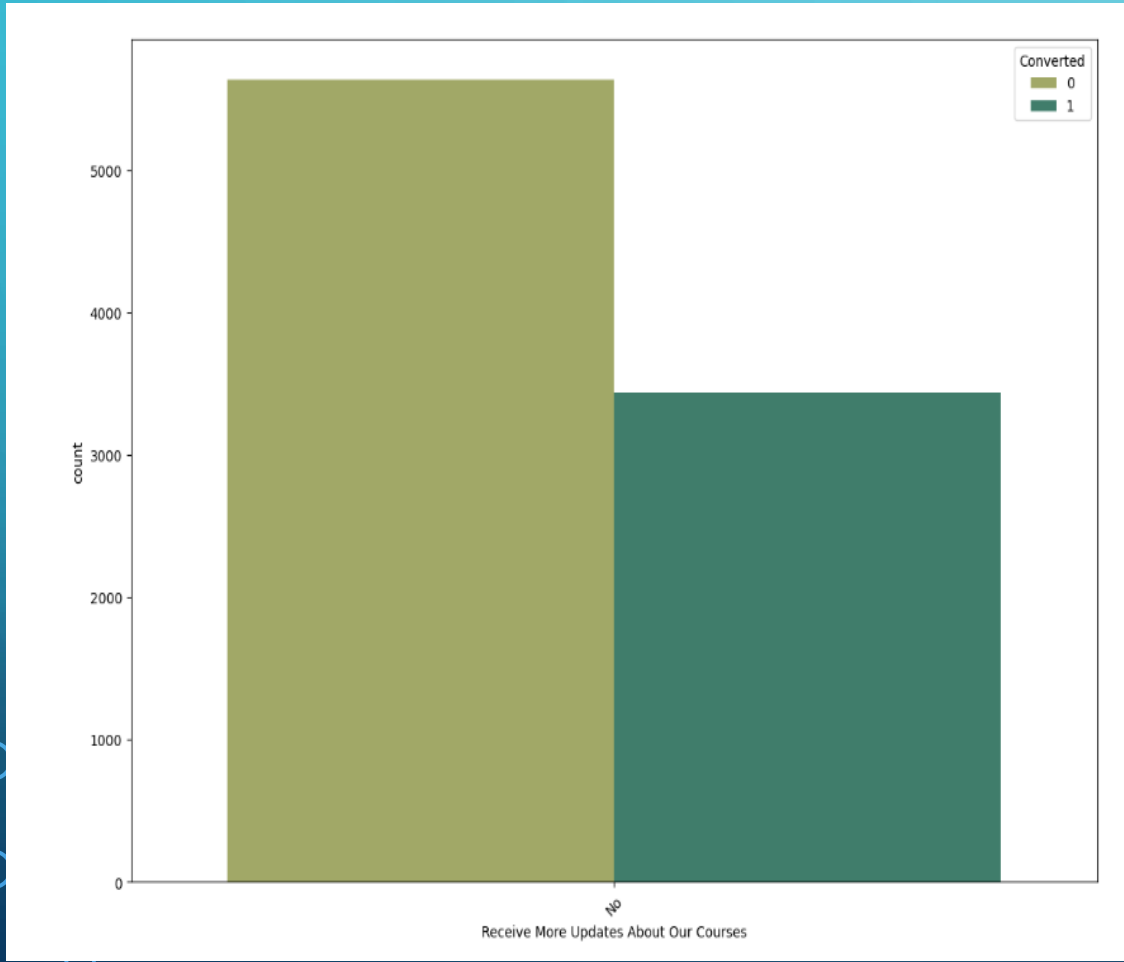
RECEIVE MORE UPDATES ABOUT OUR COURSES



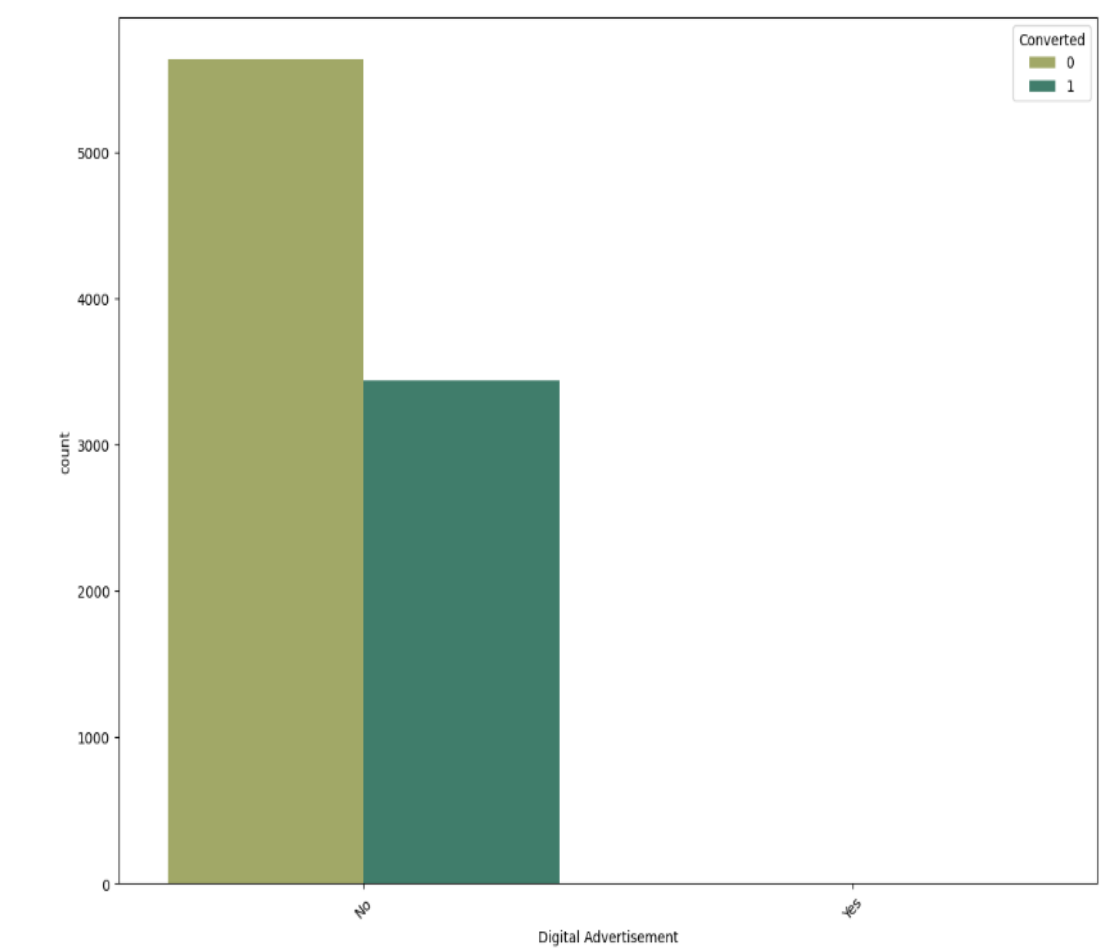
THROUGH RECOMMENDATIONS



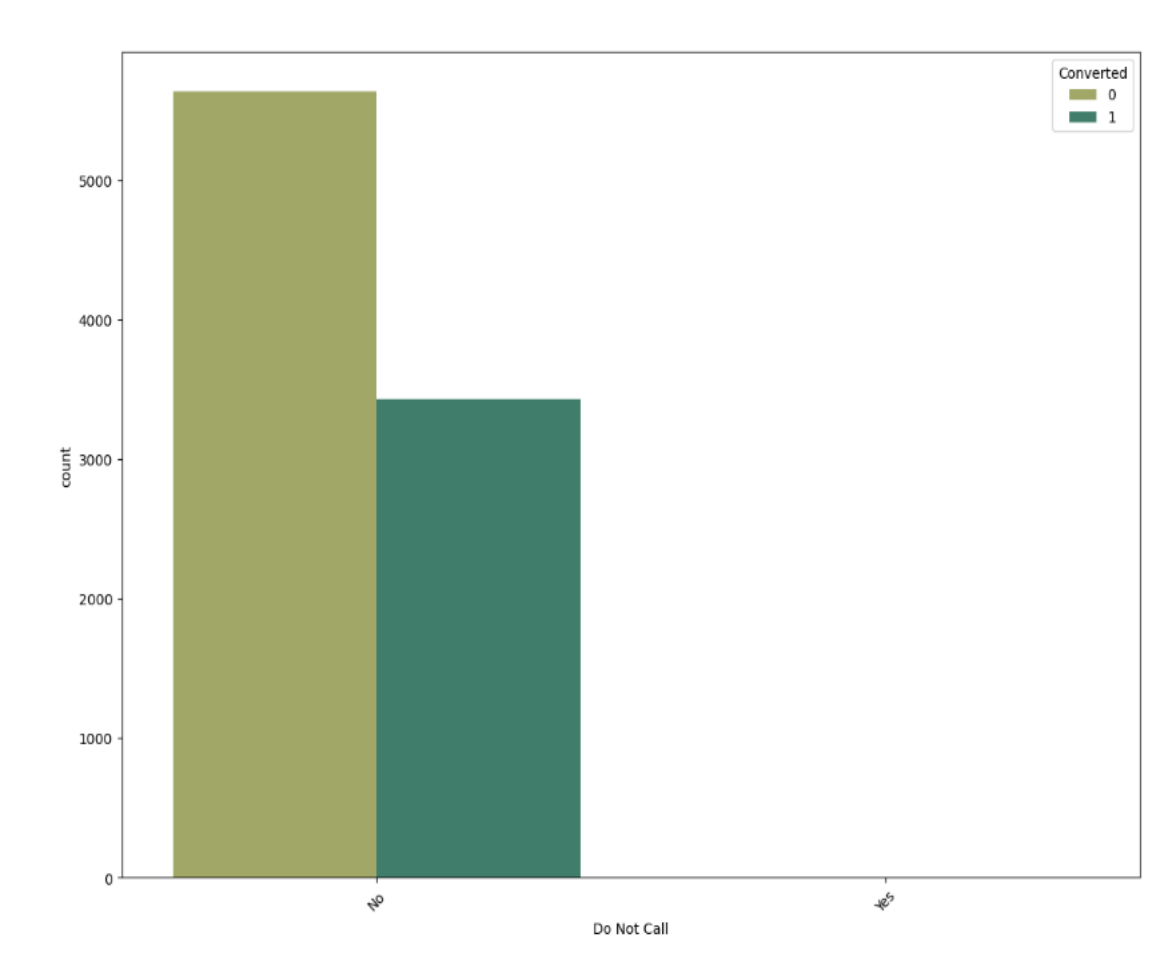
RECEIVE MORE UPDATES ABOUT OUR COURSES



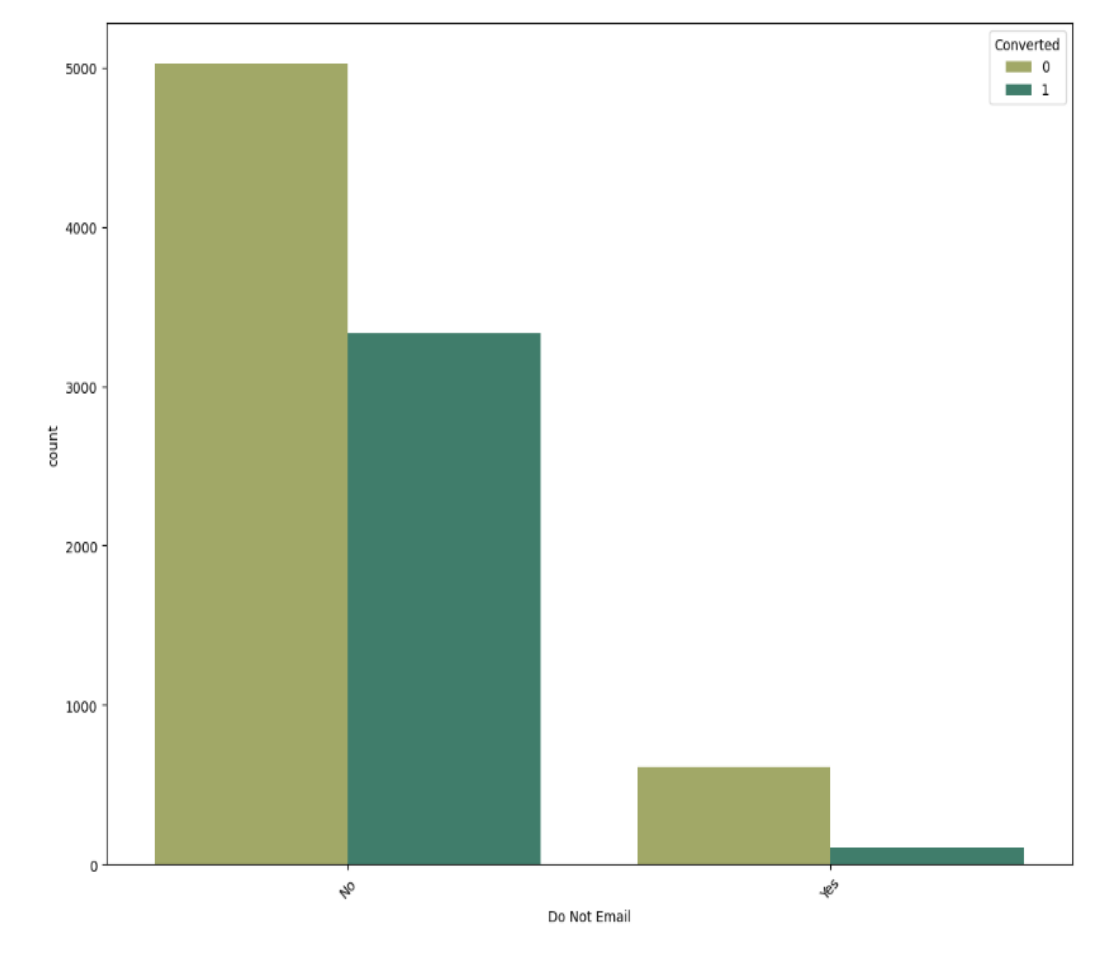
DIGITAL ADVERTISEMENT



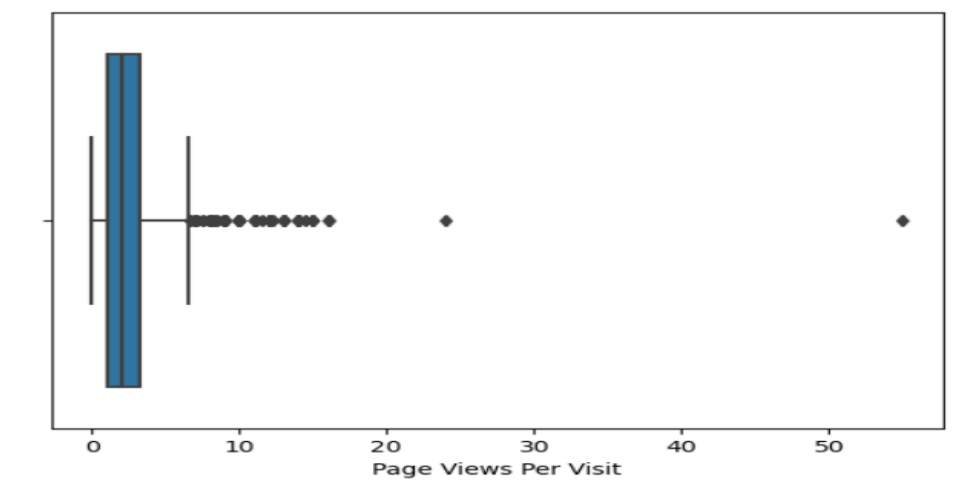
DO NOT CALL

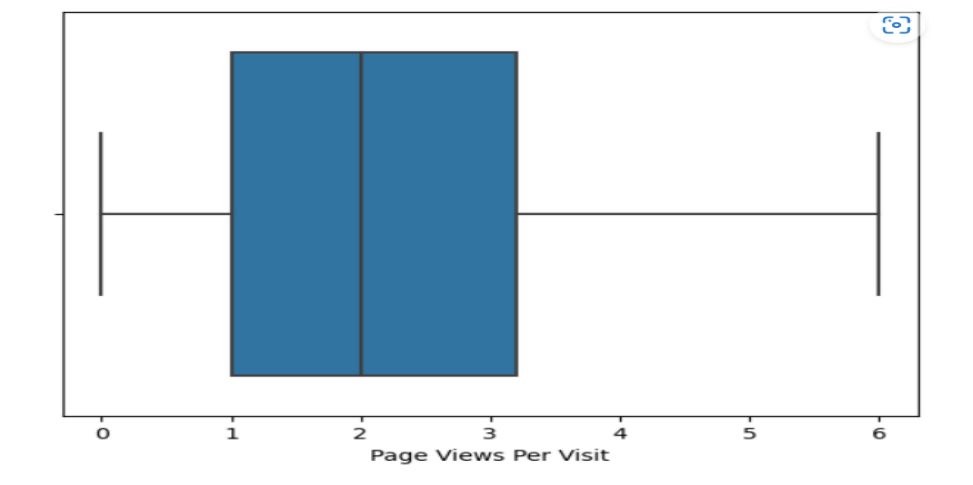


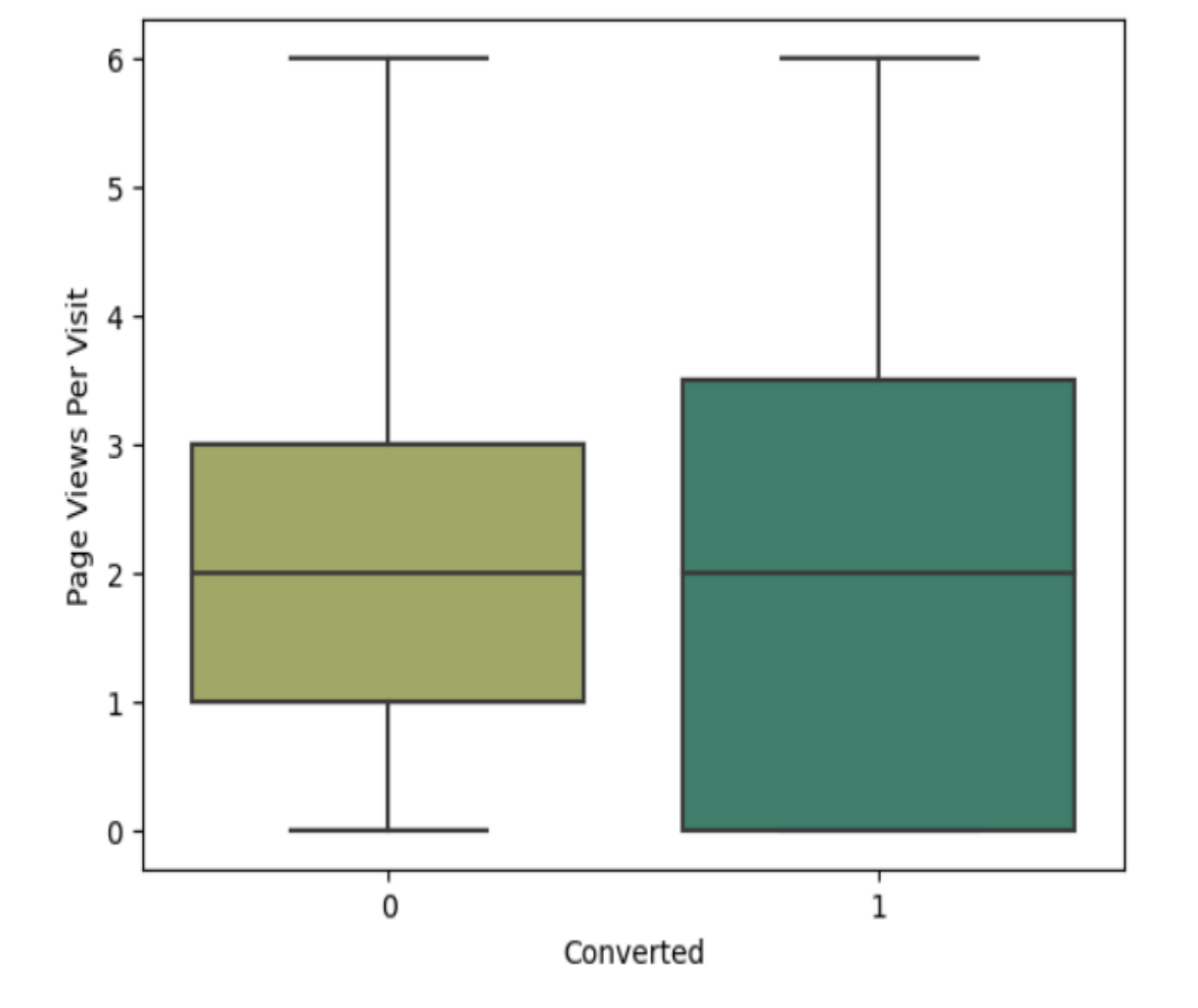
DO NOT EMAIL



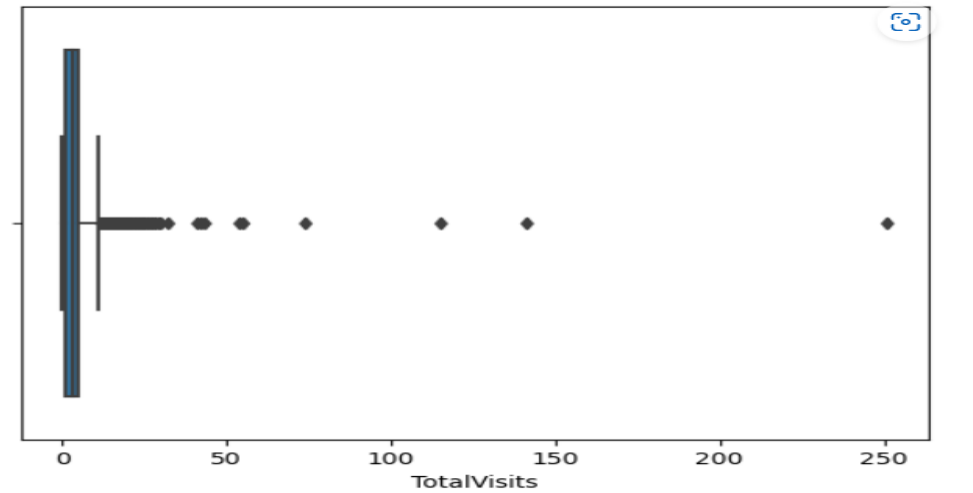
PAGE VIEWS PER VISIT – REMOVE OUTLIERS

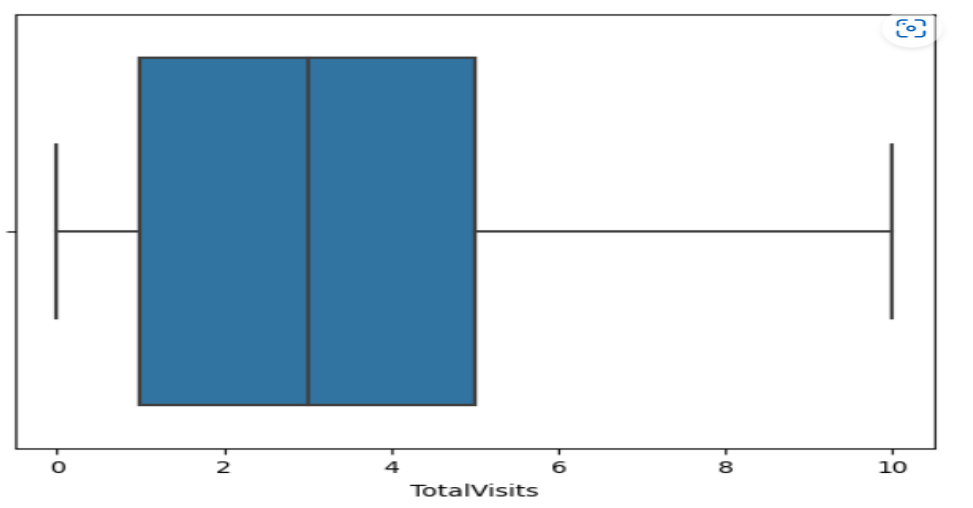


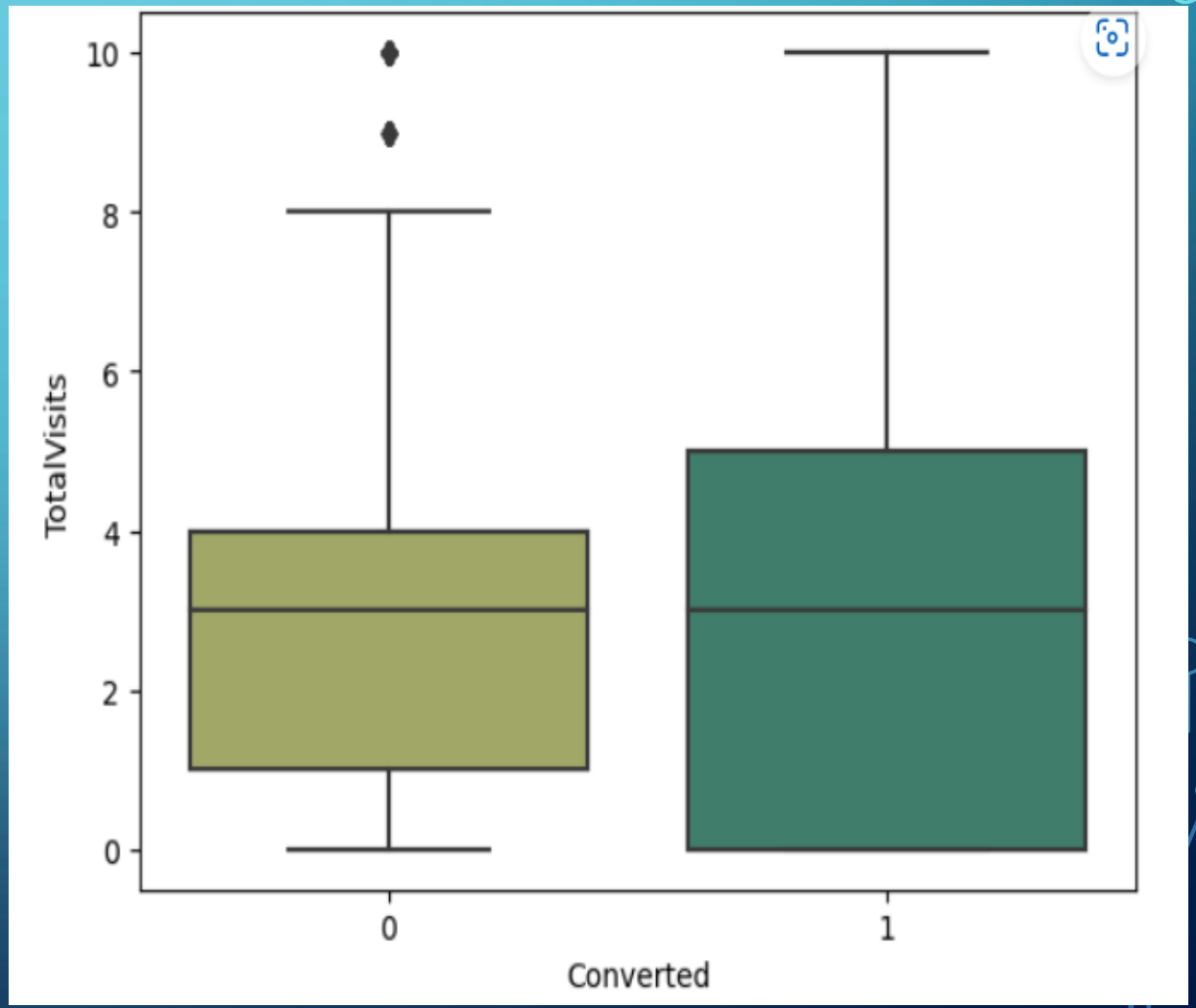




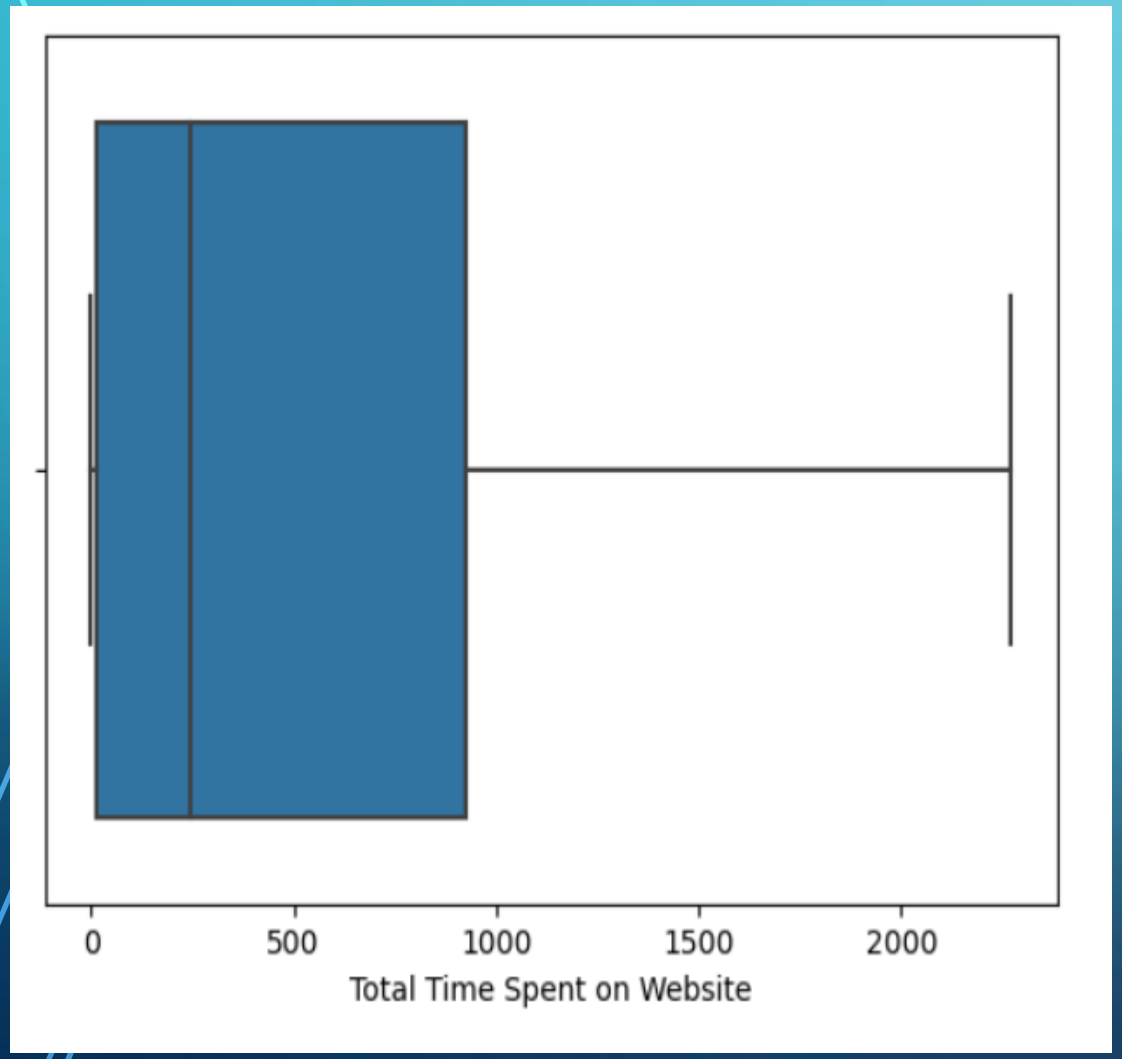
TOTAL VISITS – REMOVE OUTLIERS

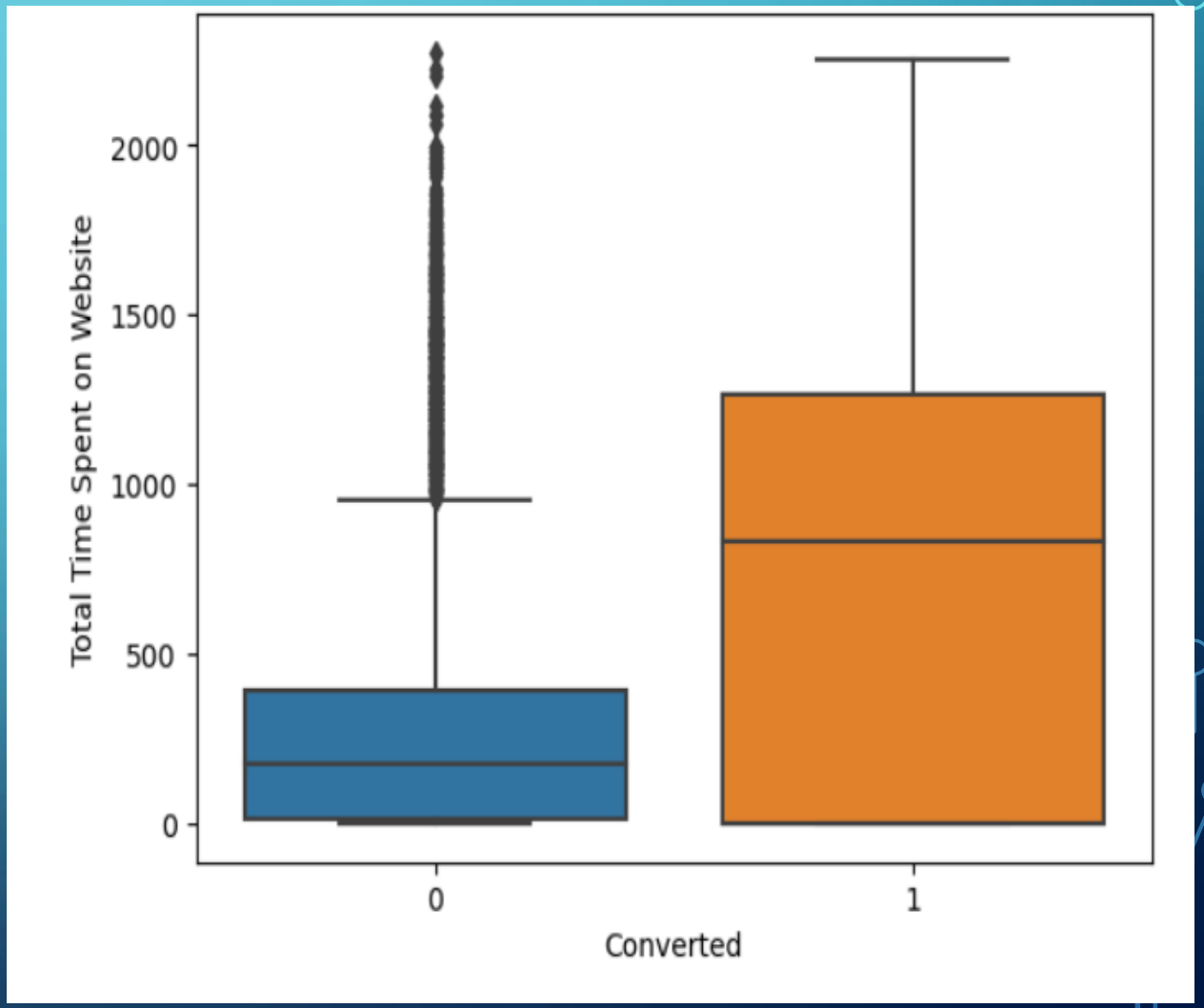






TOTAL TIME SPENT ON WEBSITE





DATA PREPARATION FOR MODELLING

DATA PREPARATION FOR MODELING

- Binary conversion to 1 and 0

```
vars =  ['Do Not Email', 'Do Not Call']

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

df[vars] = df[vars].apply(binary_map)
```

- Dummy Variables creation for categorical variables

```
# Dummy Varibles creation categorical variables
dummy_vari= pd.get_dummies(df[['Lead Origin', 'Lead Source', 'Last Activity', 'Specialization','What is your current occupat
                               'City','Last Notable Activity']], drop_first=True)
dummy_vari.head()
```

- Dropping variables for which dummies created

```
df = df.drop(['Lead Origin', 'Lead Source', 'Last Activity', 'Specialization','What is your current occupation',
                            'City','Last Notable Activity'], axis = 1)
```

MODEL BUILDING

MODEL BUILDING

- Splitting the data in to training and testing dataset
- Perform regression on train – test split. Chosen 70:30 ratio
- Use RFE for feature selection
- Building model by removing the variables whose p-value and VIF is greater than 0.05 and 5 respectively
- For creation for model steps were followed by removing the attributes with high values to almost zero
- With vif ranging below 3 total 14 models were made and 14th model was considered as the final model. Also, with 0 p-value and VIF value below 3
- Prediction on test data set
- Calculate Accuracy, Sensitivity and Specificity

- Spilt the data for training and testing and start model training

```
X = df.drop(['Prospect ID','Converted'], axis=1)
y = df['Converted']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

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

RFE Modeling

```
]:  1 from sklearn.feature_selection import RFE
```

```
]:  1 logreg = LogisticRegression()
    2 rfe = RFE(logreg,step=20)
    3 rfe = rfe.fit(X_train, y_train)
```

```
]:  1 rfe.support_
```

```
1 cols = X_train.columns[rfe.support_]
2 cols
```

Building the Model

```
:  1 X_train_sm = sm.add_constant(X_train[cols])
   2 logm1 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
   3 result = logm1.fit()
   4 result.summary()
```

```
:  1 col1 = cols.drop('What is your current occupation_Housewife')
```

Rebuilding the model

```
X_train_sm = sm.add_constant(X_train[col1])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

```
col1 = col1.drop('Last Notable Activity_Had a Phone Conversation')
```

VIF modeling

```
vif = pd.DataFrame()
vif['Features'] = X_train[col1].columns
vif['VIF'] = [variance_inflation_factor(X_train[col1].values, i) for i in range(X_train[col1].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Drop high VIF values

```
col1 = col1.drop('What is your current occupation_Unemployed')
```

Rebuilding the model

```
X_train_sm = sm.add_constant(X_train[col1])
logm9 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm9.fit()
res.summary()
```

Rebuilding the model ¶

```
X_train_sm = sm.add_constant(X_train[col1])
logm14 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm14.fit()
res.summary()
```

Check VIF again ¶

```
vif = pd.DataFrame()
vif['Features'] = X_train[col1].columns
vif['VIF'] = [variance_inflation_factor(X_train[col1].values, i) for i in range(X_train[col1].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

	Features	VIF
10	Specialization_Others	2.16
3	Lead Source_Olark Chat	2.03
12	Last Notable Activity_Modified	1.78
2	Lead Origin_Landing Page Submission	1.70
6	Last Activity_Olark Chat Conversation	1.59
8	Last Activity_SMS Sent	1.57
1	Total Time Spent on Website	1.29
4	Lead Source_Reference	1.24
0	Do Not Email	1.21
11	What is your current occupation_Working Profes...	1.19
5	Lead Source_Welingak Website	1.09
9	Last Activity_Unsubscribed	1.08
7	Last Activity_Other_Activity	1.01

From the above, can observed that the p values for columns are now 0 and VIF values are not too high. Hence model number 14 can be considered as the final model for evaluation

PREDICTION ON TRAINING 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]
```

```
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()
```

Consider the cut off point to be 0.5

```
y_train_pred_final['predicted'] = y_train_pred_final.Converted_prob.map(lambda x: 1 if x > 0.5 else 0)
y_train_pred_final.head()
```

Check the Confusion Matrix

```
con = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.predicted )
print(con)
```

Check the Accuraccy

```
print('Accuracy :',metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.predicted))
```

```
Accuracy : 0.8170366871358841
```

Check TP TN FP FN and calculate Sensitivity, Specificity, False Positive Rate, Positive Predictive Value, Negative Predictive Value

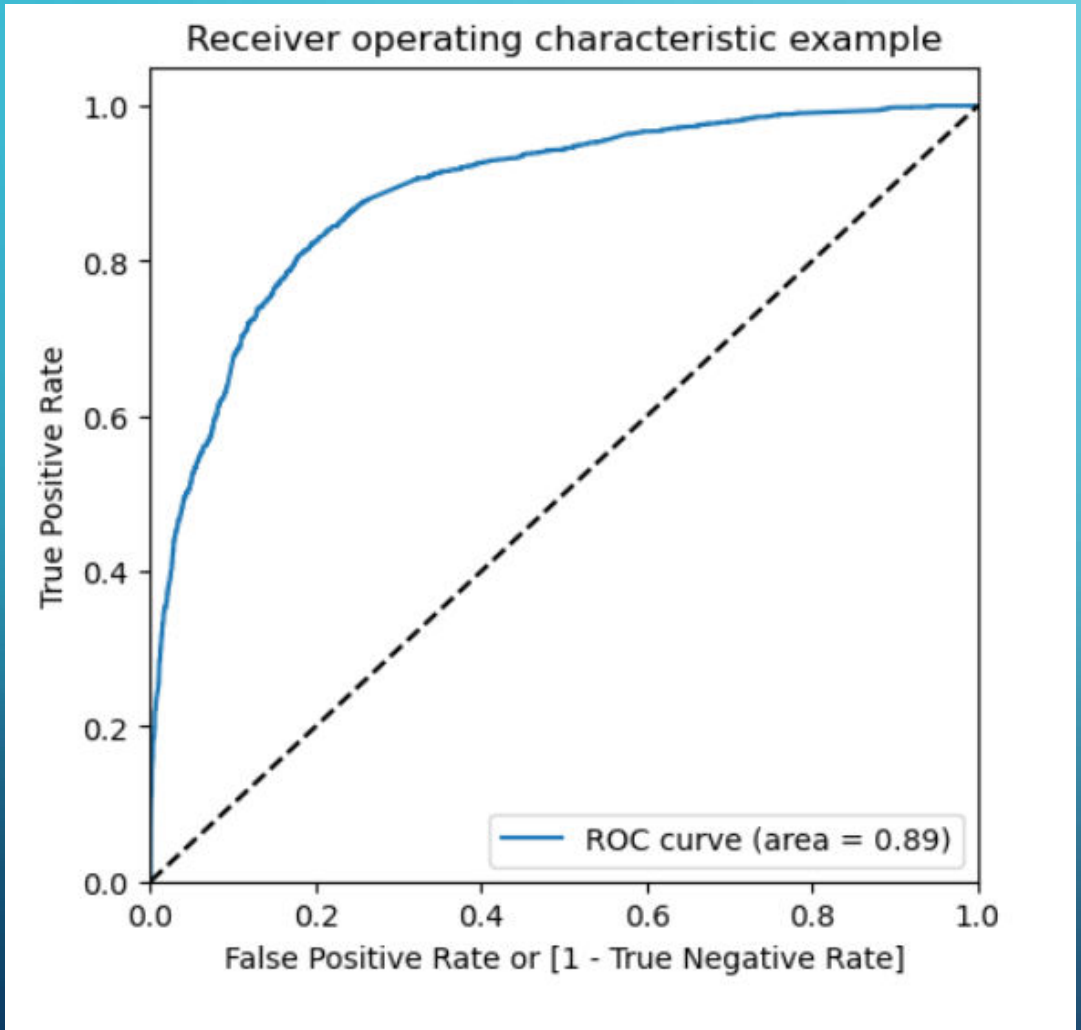
```
# true positive
TP = con[1,1]
# true negatives
TN = con[0,0]
# false positives
FP = con[0,1]
# false negatives
FN = con[1,0]
```

```
print("Sensitivity : ",TP / float(TP+FN)*100)
print("Specificity : ",TN / float(TN+FP)*100)
print("False Positive Rate :",FP/ float(TN+FP)*100)
print("Positive Predictive Value :",TP / float(TP+FP)*100)
print ("Negative predictive value :",TN / float(TN+ FN)*100)
```

1. Our model specificity is determined to be good (88.6%) but our sensitivity is just 70.7%. As a result, this will need to be resolved.

2. Our model sensitivity was 70.7%, mostly as a result of the arbitrary cut-off point of 0.5 that we used. The ROC curve will be used to optimise this cut-off point in order to obtain a respectable value for sensitivity.

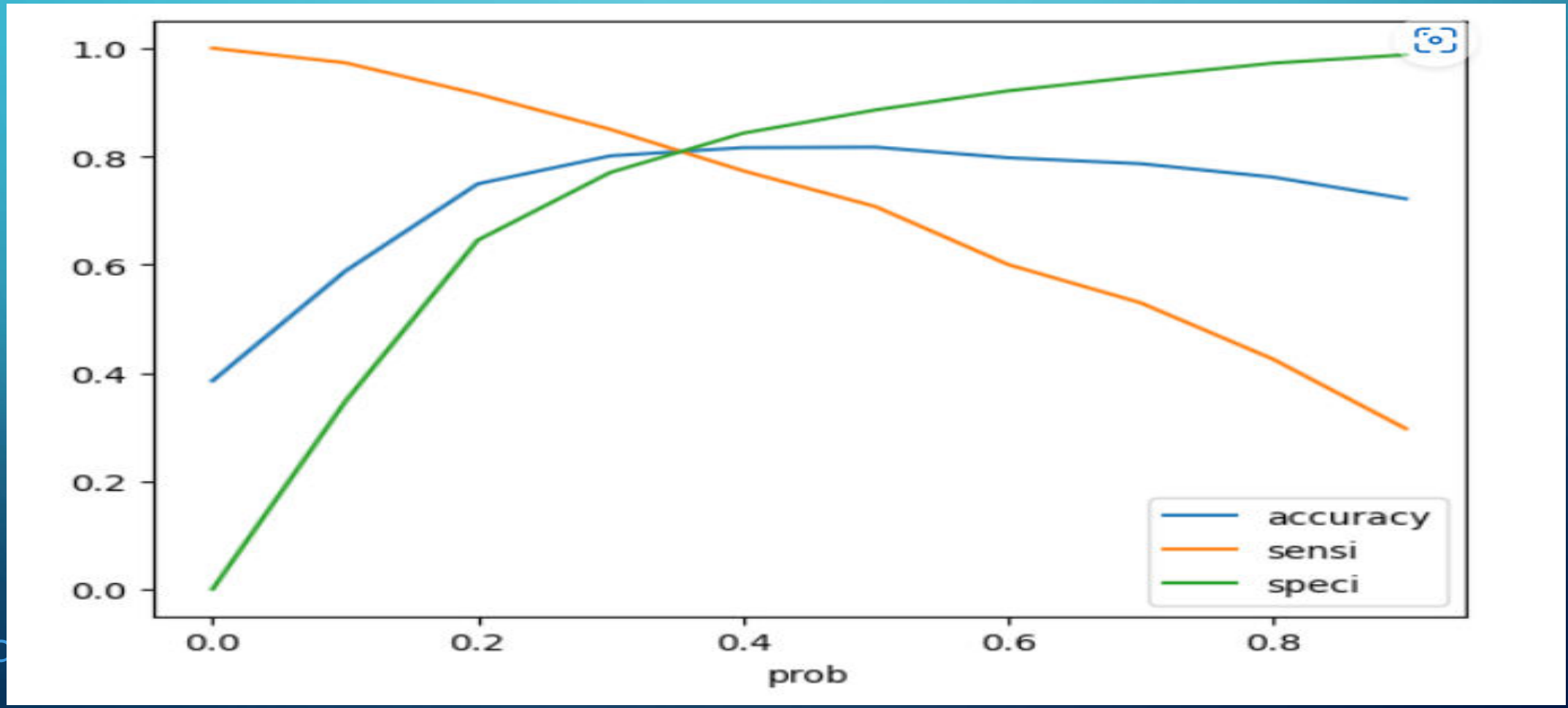
ROC CURVE




```python
def roc_c( 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
```

```
]:  1 numbers = [float(x)/10 for x in range(10)]
    2 for i in numbers:
    3     y_train_pred_final[i]= y_train_pred_final.Converted_prob.map(lambda x: 1 if x > i else 0)
    4 y_train_pred_final.head()
```



Now, Lets Evaluate the model again

```
print("Accuracy :",metrics.accuracy_score(y_train_pred_final.Converted, y_train_pred_final.final_predicted)*100)
```

```
Accuracy : 81.10533774208785
```

```
confusion_mat = metrics.confusion_matrix(y_train_pred_final.Converted, y_train_pred_final.final_predicted )
confusion_mat
```

```
array([[3148,  757],
       [ 443, 2003]], dtype=int64)
```

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

```
print("Sensitivity : ",TP / float(TP+FN)*100)
print("Specificity : ",TN / float(TN+FP)*100)
print("False Positive Rate :",FP/ float(TN+FP)*100)
print("Positive Predictive Value :",TP / float(TP+FP)*100)
print ("Negative predictive value :",TN / float(TN+ FN)*100)
```

```
Sensitivity :  81.88879803761243
Specificity :  80.6145966709347
False Positive Rate : 19.3854033290653
Positive Predictive Value : 72.57246376811595
Negative predictive value : 87.66360345307713
```

From the above Sensitivity and Specificity are 81.9% and 80.6% respectively.

PREDICTION ON TEST SET

```
X_test[['TotalVisits','Total Time Spent on Website','Page Views Per Visit']] = scaler.transform(X_test[['TotalVisits',
                                                                                                        'Total Time Spent on
                                                                                                        'Page Views Per Visi
```

```
X_test = X_test[col1]
X_test.head()
```

```
y_test_df = pd.DataFrame(y_test)
```

```
y_test_df['ID'] = y_test_df.index
y_pred1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

Appending y_test_df and y_pred1

```
y_predict = pd.concat([y_test_df, y_pred1],axis=1)
y_predict
```

```
y_predict= y_predict.rename(columns={ 0 : 'Converted_probality'})
y_predict = y_predict.reindex(columns=['ID','Converted','Converted_probality'])
```

```
y_predict['final_predicted'] = y_predict.Converted_probality.map(lambda x: 1 if x > 0.34 else 0)
```

```
y_predict
```

```
print("Accuracy :",metrics.accuracy_score(y_predict.Converted, y_predict.final_predicted)*100)
```

```
Accuracy : 80.42600073448402
```

```
confusion = metrics.confusion_matrix(y_predict.Converted, y_predict.final_predicted )
confusion
```

```
array([[1394,  340],
       [ 193,  796]], dtype=int64)
```

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

```
print("Sensitivity : ",TP / float(TP+FN)*100)
print("Specificity : ",TN / float(TN+FP)*100)
print("False Positive Rate :",FP/ float(TN+FP)*100)
print("Positive Predictive Value :",TP / float(TP+FP)*100)
print ("Negative predictive value :",TN / float(TN+ FN)*100)
```

```
Sensitivity :  80.48533872598584
Specificity :  80.3921568627451
False Positive Rate : 19.607843137254903
Positive Predictive Value : 70.07042253521126
Negative predictive value : 87.83868935097668
```

Assinging the score to final Test data

```
y_predict['Lead_Score'] = y_predict.Converted_probality.map( lambda x: round(x*100))
y_predict.head()
```

	ID	Converted	Converted_probality	final_predicted	Lead_Score
0	3271	0	0.129393	0	13
1	1490	1	0.968762	1	97
2	7936	0	0.111751	0	11
3	4216	1	0.804505	1	80
4	3830	0	0.132210	0	13

MODEL EVALUATION

MODEL EVALUATION

TRAIN DATA

- ACCURACY : 81.1 %
- SENSITIVITY : 81.8 %
- SPECIFICITY : 80.6 %

TEST DATA

- ACCURACY : 80.4 %
- SENSITIVITY : 80.5 %
- SPECIFICITY : 80.4 %

CONCLUSION

CONCLUSION

- It was found that the variables that mattered the most for the potential buyers are following:
  - What matters most to you in choosing a course
  - Tags will revert after reading the email
  - Last Notable Activity modified
  - Tags: Ringing and Other
  - Last Activity: SMS sent and Olark chat conversation
  - Occupation: Working Professional
  - Lead Origin lead import

CONCLUSION

- Keeping the above in mind, the X Education can flourish as they have very high chance to get almost all the potential buyers to change their mind and buy their courses
- Thus, we have succeeded in achieving our objective of estimating the desired lead conversion rate to be somewhere around 80%. We should be able to provide the CEO confidence to make wise decisions based on this model in order to achieve a higher lead conversion rate of 80% because the model appears to anticipate the conversion rate quite effectively

THANK YOU