CAPSTONE PROJECT ON
EMAIL CAMPAIGN EFFECTIVENESS PREDICTION

CHAPTER 1- OBJECTIVES AND INTRODUCTION

1.1 OBJECTIVE :

- The main objective is to create a machine learning model to characterize the mail and track the mail that is ignored; read; acknowledged by the reader.
- In addition to the ML Model prediction, we also analysed what all features can help us in getting the Email status to be not ignored by the customers.

1.2 INTRODUCTION:

Most of the small to medium business owners are making effective use of Gmail-based Email marketing Strategies for offline targeting of converting their prospective customers into leads so that they stay with them in Business.

In order to help the business grow with the Email Marketing Strategies, we are trying to find all the features that are important for an Email to not get Ignored.
Many of the times we do not tend to read an Email due to a number of reasons. Some of it can be no proper structure of the email, too many direct links and images in a single email and may be too long emails!!

Since, here we are trying to predict if the email is going to be read, ignored and acknowledged, we are basically trying different machine learning algorithms like SVM, KNN, Random Forest, XGBoost, CatBoost etc. and on the basis of the training and testing results we will get our model which will perform best in out case.

The next chapters have the following sections which are the steps involved for solving the Email Campaign Effectiveness Prediction,
Section 1 - Data collection
Section 2 - Data preparation
Section 3 - Exploratory data analysis
Section 4 - Feature Engineering
Section 5 - Handling Imbalanced dataset
Section 6 - Working different models and Evaluating model

CHAPTER 2 - DATA COLLECTION AND DATA PREPARATION

2.1 Data Summary:

The data has the following features:

Email_ID - This column contains the email ids of individuals.

Email_type - Email type contains 2 categories 1 and 2. We can assume that the types are like promotional email or important email.

Subject_Hotness_Score - It is the email effectiveness score.

Email_Source - It represents the source of the email like sales or marketing or product type email.

Email_Campaign_Type - Campaign type

Total_Past_Communications - This column contains the previous mails from the same source.

Customer_Location - Categorical data which explains the different demographics of the customers.

Time_Email_sent_Category - It has 3 categories 1,2 and 3 which may give us morning,evening and night time slots.

Word_Count - It contains the no.of words contained in the mail.

Total_Links - Total links from the mail.

Total_Images - The banner images from the promotional email.

Email_Status - It is the target variable which contains the characterization of the mail that is ignored; read; acknowledged by the reader.

2.2 Data Collection

To proceed with the problem dealing first we will load our dataset that is given to us in .csv file into a dataframe.

Mount the drive and load the csv file into a dataframe.

```
# Mounting Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#Path of data in drive
data_path='/content/drive/MyDrive/AlmaBetter/Capstone_Project/Capstone Project II/data_email_campaign.csv'
#Reading csv file
email_data=pd.read_csv(data_path)
```

After mounting the drive the next step is to import the required libraries. Python has a wide number of libraries which makes the work easier. Here pandas, numpy, matlplotlib, seaborn, math,sklearn etc.

CHAPTER 3- EXPLORATORY DATA ANALYSIS

The primary goal of EDA is to support in the analysis of data prior to making any conclusions. It may aid in the detection of apparent errors, as well as a deeper understanding of data patterns, the detection of outliers or anomalous events, and the discovery of interesting relationships between variables**.**

3.1 Duplication:

In our further analysis we found that the dataset has no duplicate entries.

3.2 Categorical Data:

Out of all the features we have a total of 6 categorical feature variables.

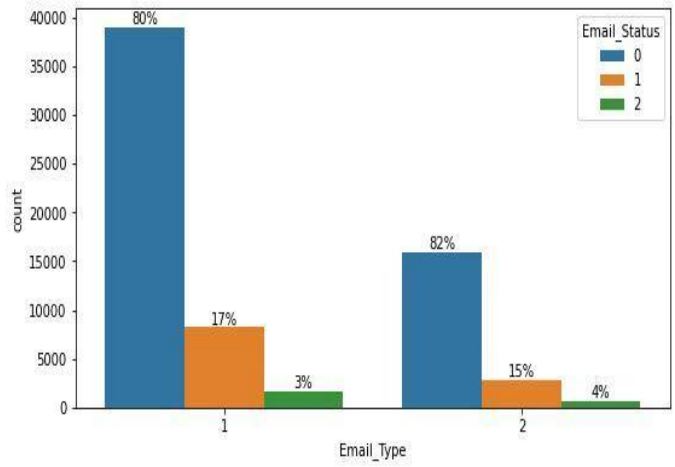
- Email_Type
- Email_source_Type
- Customer_location
- Email_Campaign_type
- Time_Email_Sent_Catagory
- Email_Satus

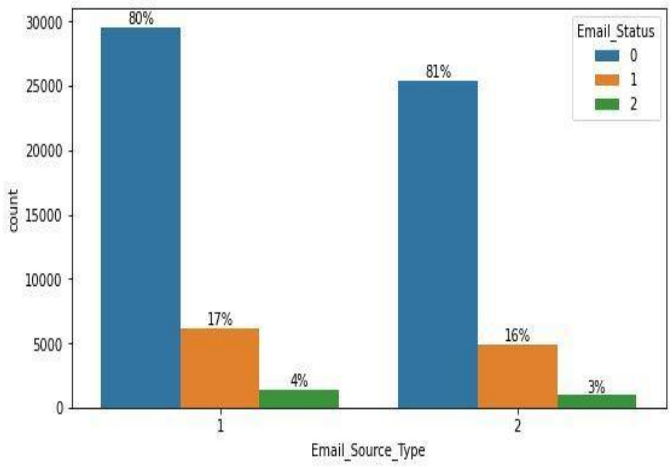
Out of these categorical variables, Email_Status is our target variable which shows 3 different statuses as “Ignored”, “Read” and “Acknowledged”. The different values in each categorical variable is shown below.

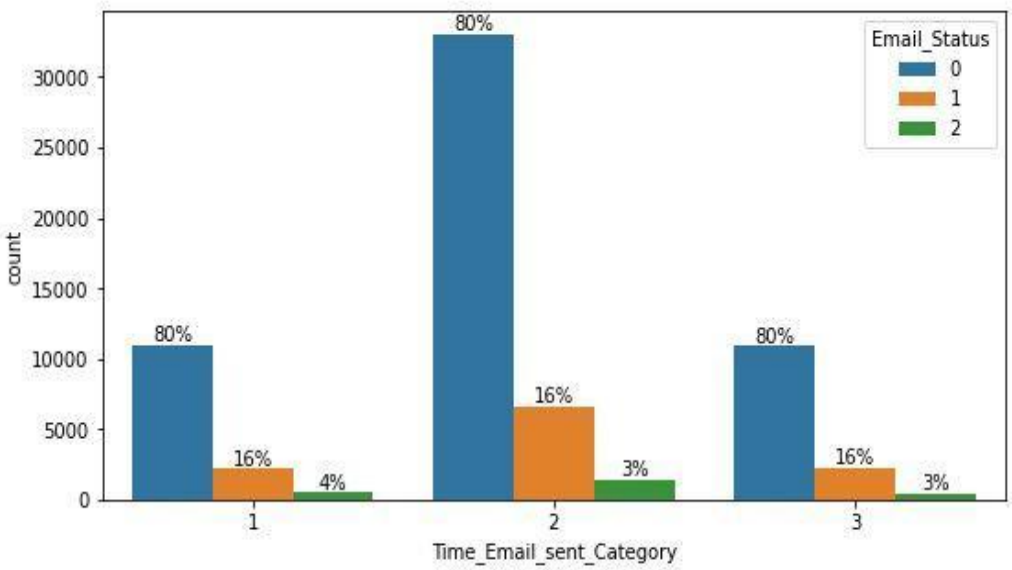
column	values	values_count_incna	values_count_nona	num_miss
Email_ID	[EMA00081000034500, EMA00081000045360, EMA0008...	68353	68353	0
Email_Type	[1, 2]	2	2	0
Email_Source_Type	[2, 1]	2	2	0
Customer_Location	[E, nan, B, C, G, D, F, A]	8	7	11595
Email_Campaign_Type	[2, 3, 1]	3	3	0
Time_Email_sent_Category	[1, 2, 3]	3	3	0
Email_Status	[0, 1, 2]	3	3	0

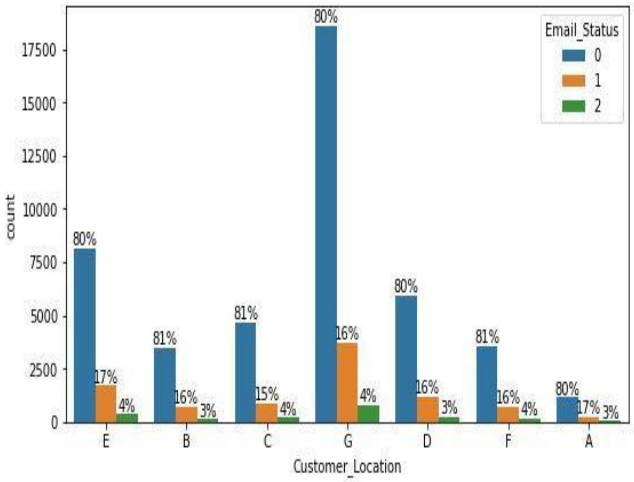
Let's check the impact of these categorical variables on our target variable.
The below graph plots show the dependence of our target variable “Email_Staus” on the different categorical features and their distribution.

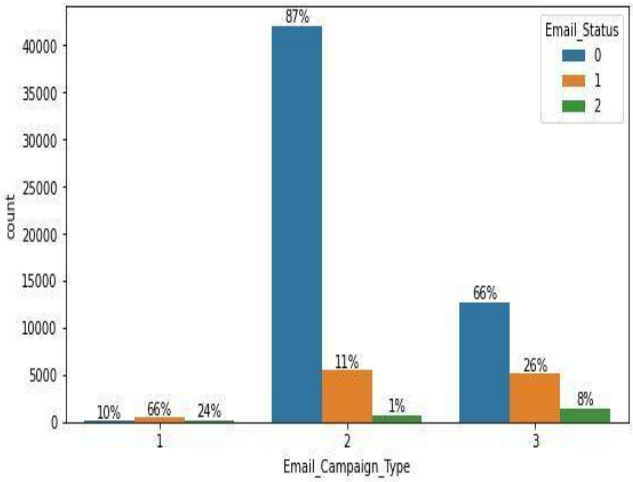
```python
def barPerc(df,xVar,ax):
    ''' barPerc(): Add percentage for hues to bar plots
    args:
        df: pandas dataframe
        xVar: (string) X variable
        ax: Axes object (for Seaborn Countplot)
    '''
    # 1. how many X categories
    ##   check for NaN and remove
    numX=len([x for x in df[xVar].unique() if x==x])
    # 2. The bars are created in hue order, organize them
    bars = ax.patches
    ## 2a. For each X variable
    for ind in range(numX):
        ## 2b. Get every hue bar
        ##     ex. 7 X categories, 3 hues =>
        ##    [0, 8, 16] are hue bars for 1st X category
        hueBars=bars[ind:][::numX]
        ## 2c. Get the total height (for percentages)
        total = sum([x.get_height() for x in hueBars])
        # 3. Print the percentage on the bars
        for bar in hueBars:
            ax.text(bar.get_x() + bar.get_width()/2.,
                    bar.get_height(),
                    f'{bar.get_height()/total:.0%}',
                    ha="center",va="bottom")
```











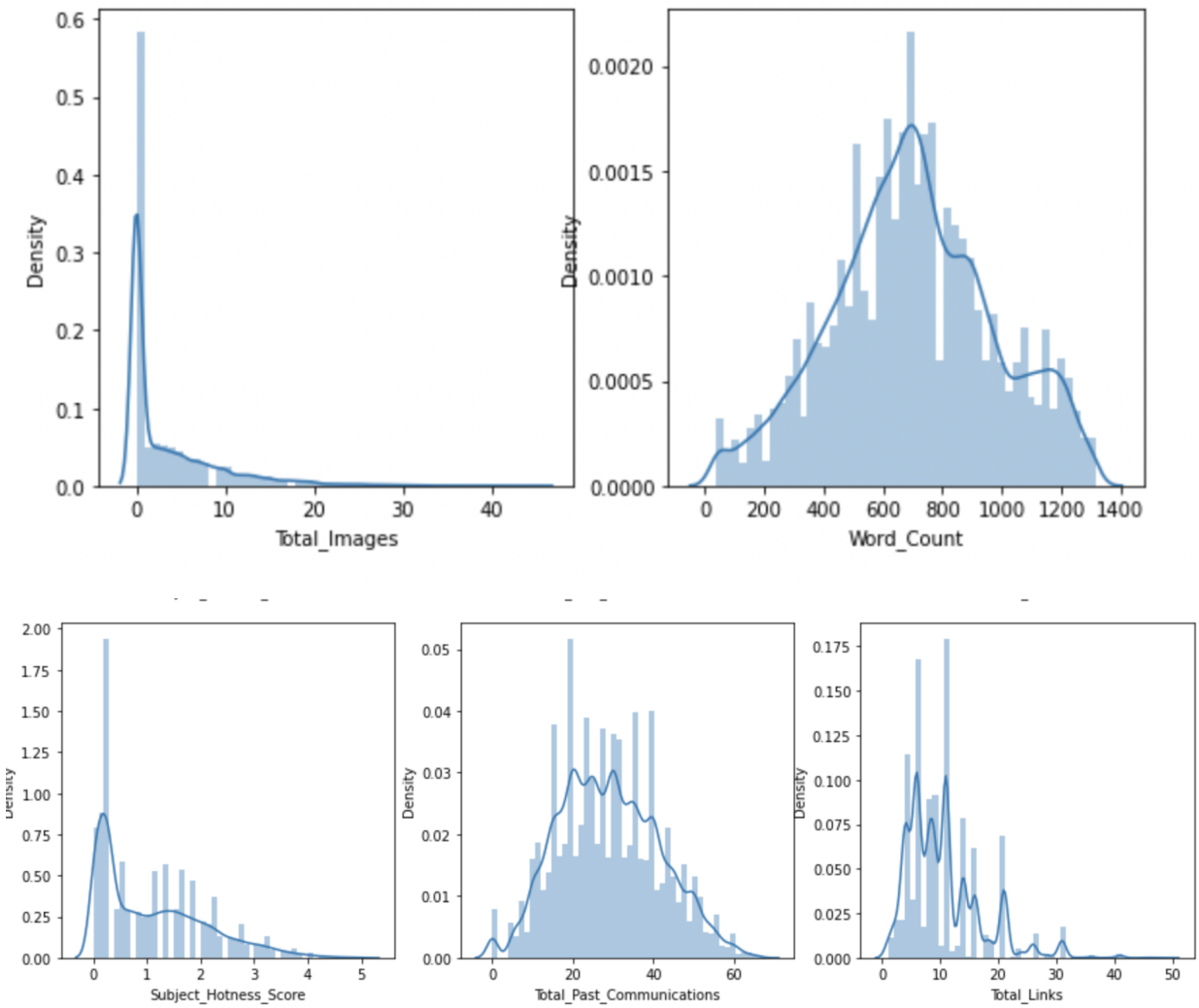
From the graphs above we infer that our target variable, i.e., Email_Status does not depend upon Time_Email_sent_catagory.

3.3 Continuous data:

We can see that there are a total of 5 numerical variables in the dataset namely,

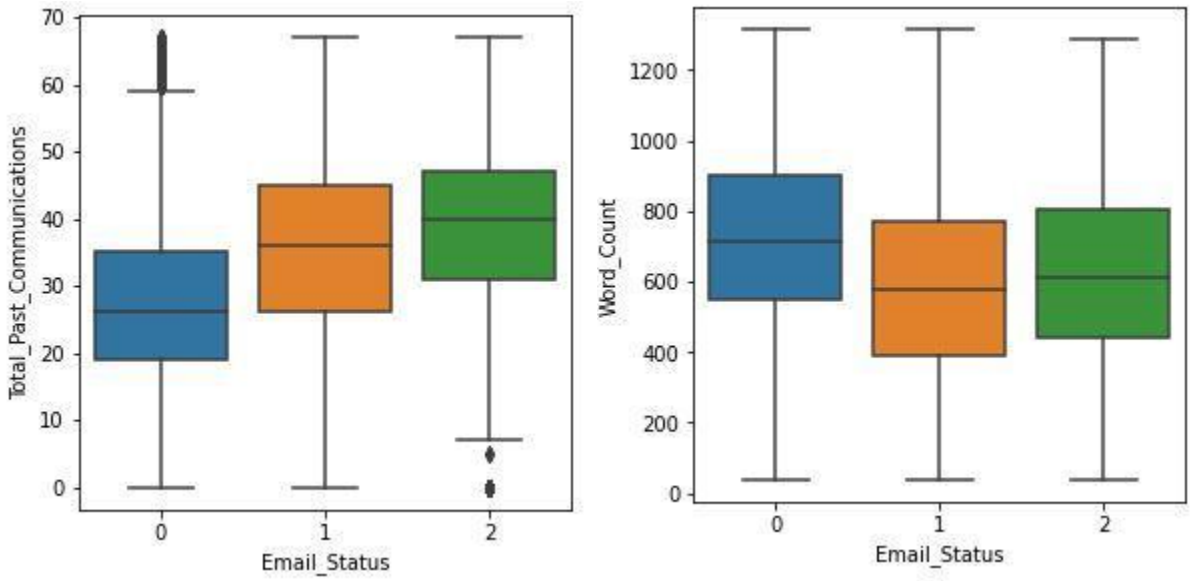
- Word_Count
- Total_Images
- Total_Links
- Total_Past_Communications
- Subject_Hotness_Score

Let us first try and understand each of these features through their distribution plots given below



We observe that 2 features i.e. “ Word Count” and “Total_Past Communications” follow a normal distribution. The rest of the features were highly skewed to the left.

Continuous features w.r.t. Email_Status:



From the Box-plot above, we find that:

- As the number of Total_Past_Communication is increasing, the chances of Email getting ignored is decreasing.
- As the word_count increases beyond the 600 mark we see that there is a high possibility of that email being ignored. The ideal mark is 400-600. No one is interested in reading long emails !

3.4 Missing data:

Once you have raw data, you must deal with issues such as missing data and ensure that the data is prepared for forecasting models in such a way that it is amenable to them.

There were missing values in some columns in our dataset. The following code snippet gave us the idea about missing values in different columns.

```
email_data.isnull().sum()
```

```
1 email_data.isnull().sum()
```

```
Email_ID                        0
Email_Type                      0
Subject_Hotness_Score           0
Email_Source_Type               0
Customer_Location           11595
Email_Campaign_Type             0
Total_Past_Communications    6825
Time_Email_sent_Category        0
Word_Count                      0
Total_Links                  2201
Total_Images                 1677
Email_Status                    0
dtype: int64
```

From the above data we realise that 4 features have null values.

- Customer_Location
- Total_past_communications
- Total_Links
- Total_Images

We will be handling it in the upcoming Data Cleaning section.

3.5 Data Cleaning:

Customer Location feature:

We have already seen in our missing values analysis that the Customer_Location feature has the most number of missing values (11595 missing values). Also, in categorical data analysis, after plotting the frequency graph of different values of Customer_location with respect to the Email_status category we found that the percentage ratio of Email being Ignored, Read or Acknowledged is the same irrespective of the location.

- The Customer_Location feature does not affect Email_Status
- We can drop Customer_Locaton feature

CHAPTER 4 : FEATURE ENGINEERING

Handling missing values :

Values that are reported as missing may be due to a variety of factors. These lack of answers would be considered missing values. The researcher may leave the data or do data imputation to replace them. Suppose the number of cases of missing values is extremely small; then, an expert researcher may drop or omit those values from the analysis. But here in our case we had a lot of data points which were having missing values in different feature columns.

4.1 Imputing Values:

Since, we have missing values in the following feature variables, we need to impute them.

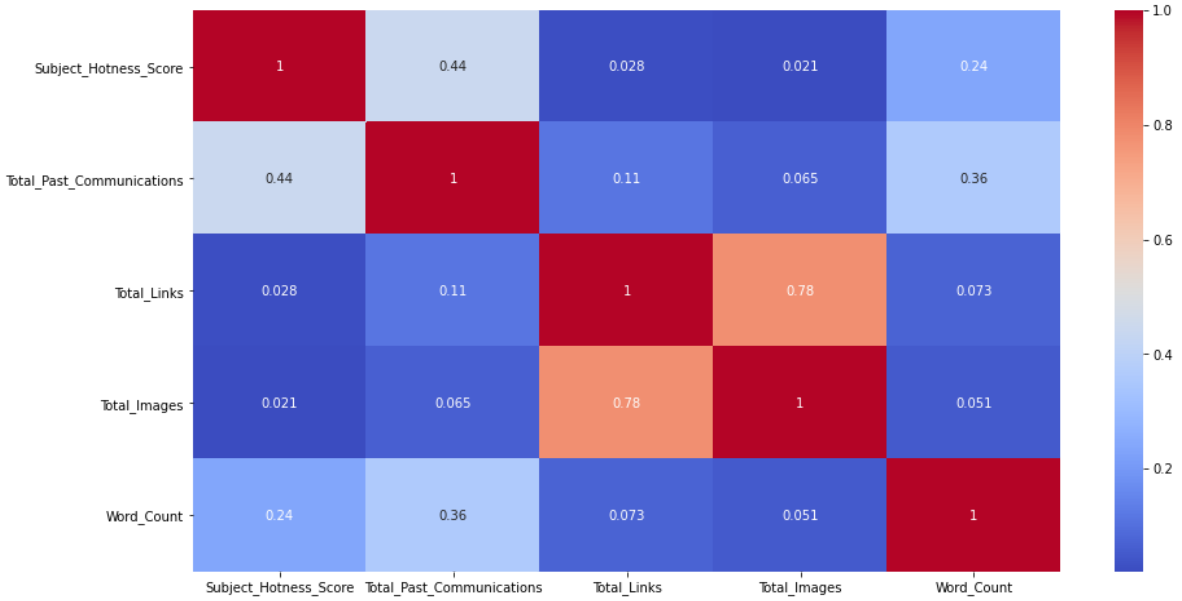
- Total_past_communications
- Total_Links
- Total_Images

Total_Past_Communication:
From the continuous data analysis part we get that the graph of Total_past_Communications follows approximately Normal Distribution. So, we decided to impute the missing values in this column by the mean of the values.

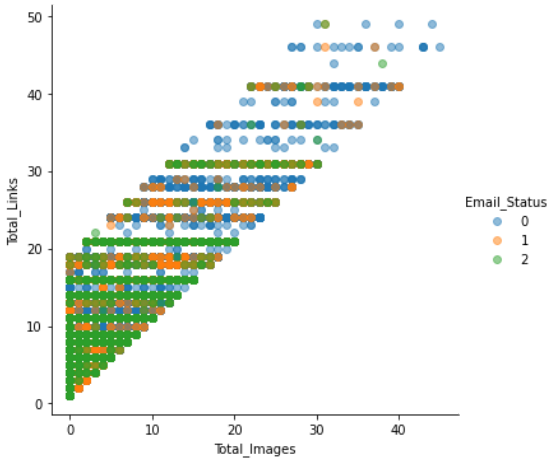
Total_Links and Total_Images:
From the continuous data analysis part we get that the graph of Total_Links & Total_Images is skewed towards the left. So, we decided to impute the missing values in these columns by the mode of the values.

4.2 Correlation Understanding:

The study of how variables are correlated is called correlation analysis. This can be done both for numerical data as well as categorical data. The most common correlation coefficient is the Pearson Correlation Coefficient. It’s used to test for linear relationships between data. For the scope of the project we try to plot a correlation matrix in order to better understand the relationship between the numerical variables.



So we can observe that the correlation score is 0.78 for Total_Images and Total_Links which on a scale of (-1,1) can be understood as high positive correlation. To understand whether this relation holds true, we can plot the selected variables using scatter plots



Hence we can consolidate the values for both of the above mentioned features

VIF Factor:

Variance inflation factor (*VIF*) quantifies how much the variance is inflated. A VIF of 1 means that there is no correlation among the jth predictor and the remaining predictor variables, and hence the variance of bj is not inflated at all. The general rule of thumb is that VIFs exceeding 4 warrant further investigation, while VIFs exceeding 10 are signs of serious multicollinearity requiring correction.

Below, we can observe that all the VIF values are within appreciable limits. Hence we can utilize these variables for the modelling.

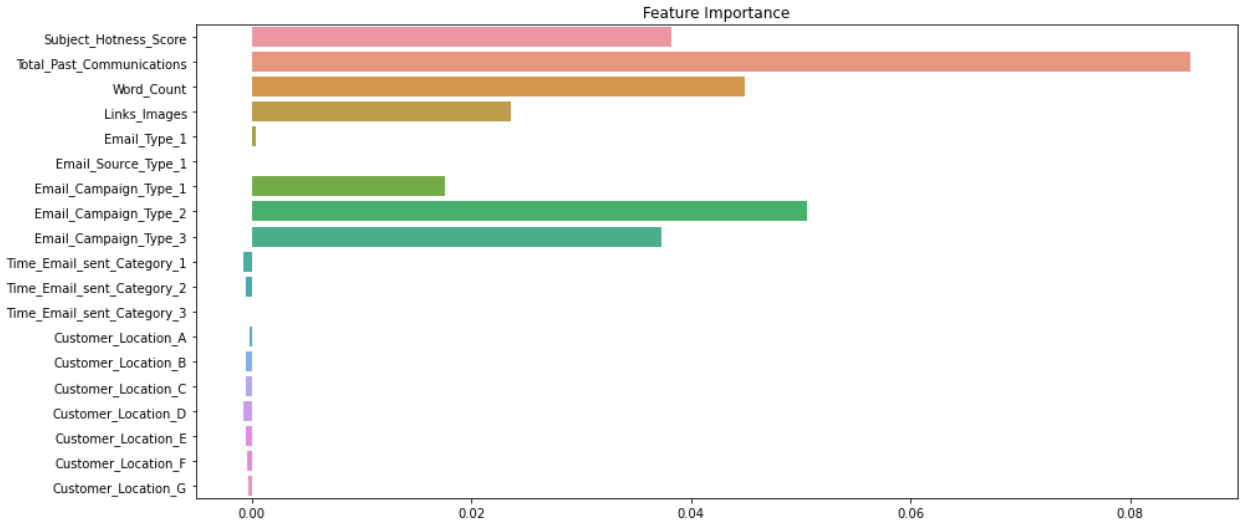
	variables	VIF
0	Subject_Hotness_Score	1.734531
1	Total_Past_Communications	3.430879
2	Word_Count	3.687067
3	Links_Images	2.629047

4.3 Feature Importance:

Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way. It is commonly used in the construction of decision trees from a training dataset, by evaluating the information gain for each variable, and selecting the variable that maximizes the information gain, which in turn minimizes the entropy and best splits the dataset into groups for effective classification.

$$\text{Entropy} = -p \log_2 p - q \log_2 q$$

Hence we can compute the information gain for each of the variables which can be visualized as given below.




```python
def comp_feature_information_gain(df, target, descriptive_feature):
    """
    This function calculates information gain for splitting on
    a particular descriptive feature for a given dataset
    and a given impurity criteria.
    Supported split criterion: 'entropy'
    """
    target_entropy = compute_impurity(df[target])

    # we define two lists below:
    # entropy_list to store the entropy of each partition
    # weight_list to store the relative number of observations in each partition
    entropy_list = list()
    weight_list = list()

    # loop over each level of the descriptive feature
    # to partition the dataset with respect to that level
    # and compute the entropy and the weight of the level's partition
    for level in df[descriptive_feature].unique():
        df_feature_level = df[df[descriptive_feature] == level]
        entropy_level = compute_impurity(df_feature_level[target])
        entropy_list.append(round(entropy_level, 3))
        weight_level = len(df_feature_level) / len(df)
        weight_list.append(round(weight_level, 3))

    feature_remaining_impurity = np.sum(np.array(entropy_list) * np.array(weight_list))


    information_gain = target_entropy - feature_remaining_impurity


    return(information_gain)
```

We can observe that Time_Email_sent_Category_1, Time_Email_sent_Category_2, Time_Email_sent_Category_3 and Customer_Location have very less importance. We can drop this feature. Total_Past_Communication and Word_Count are highly important features.

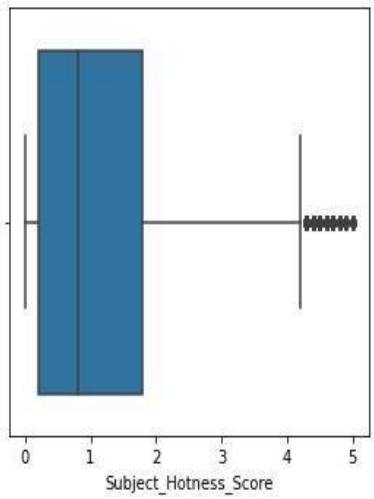
4.4 Outlier Treatment:

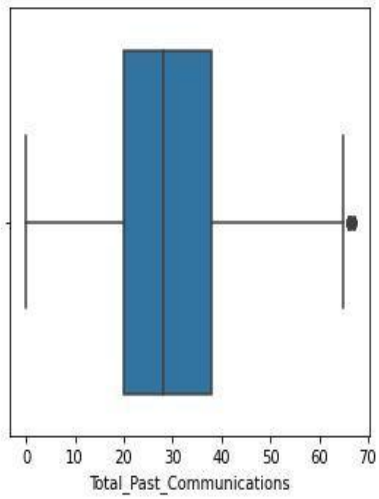
An outlier is an observation of a data point that lies an abnormal distance from other values in a given population. It is an abnormal observation during the Data Analysis stage, that data point lies far away from other values.

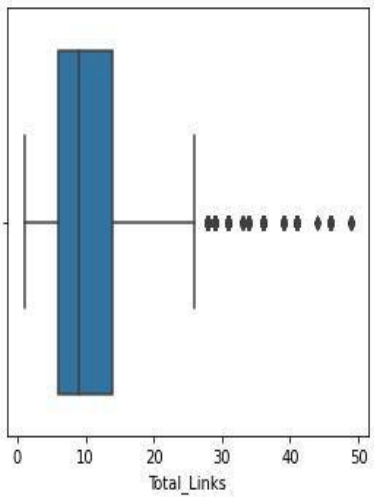
What method to use for finding outliers?

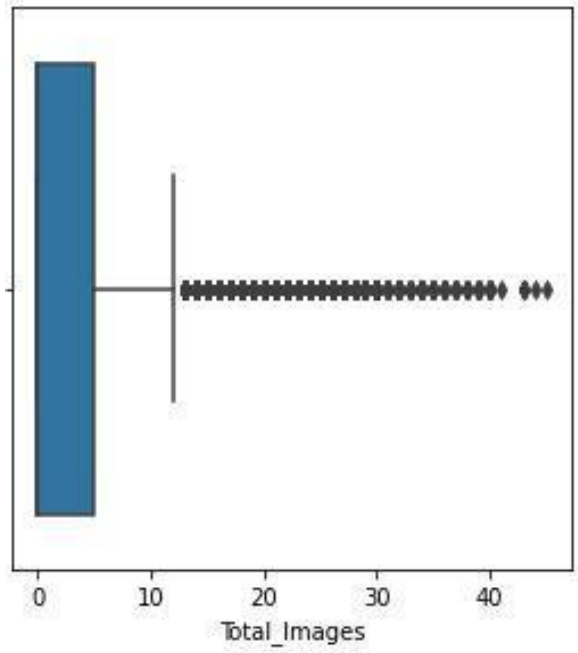
Univariate method: I believe you're familiar with Univariate analysis, playing around one variable/feature from the given data set. Here to look at the Outlier we're going to apply the BOX plot to understand the nature of the Outlier and where it is exactly.

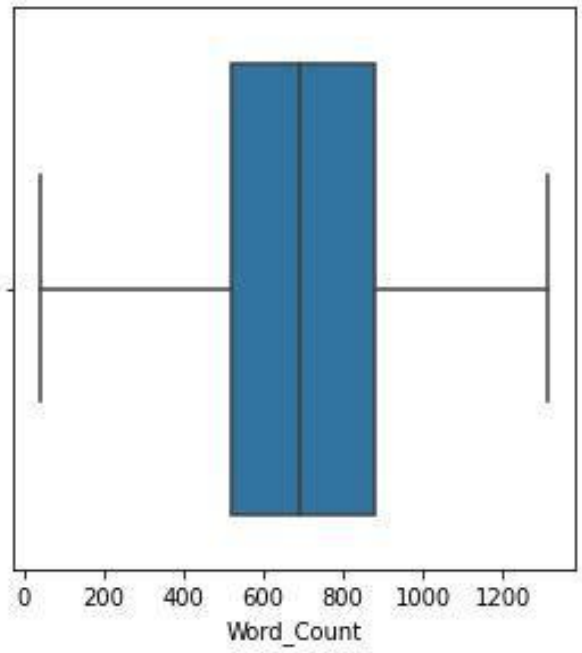
Now, let's deep dive into the data distribution and find if we have outliers or not. There are outliers in Subject_Hotness_Score,Total_Links and Total_Images.
Now let's see how to treat outliers. Except for the Word_Count column all other numeric columns have outliers. Since our dependent variable is highly imbalanced so before dropping outliers we should check that it will not delete more than 5% of the minority class which is Email_Status =1,2.

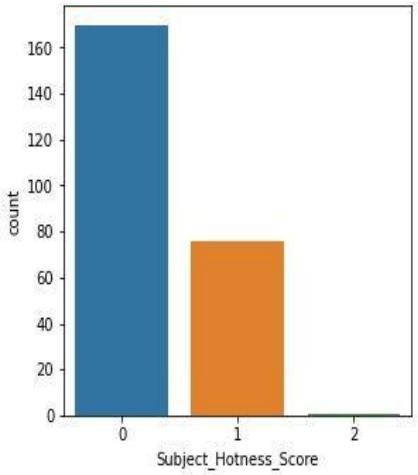


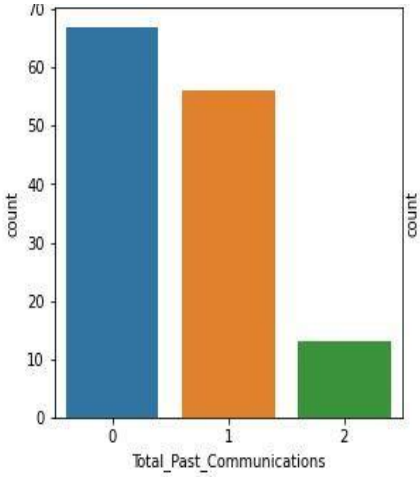


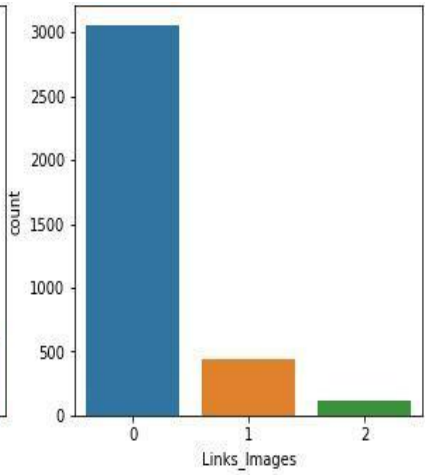












Having a look at the distribution of our outliers, it was understood that close to 5% of data was being removed from minority class.
Hence decided against removing the outliers. This problem can be solved through normalization and choosing boosted trees for our modelling which are robust to outliers.

4.5 Scaling:

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Here we used MinMaxScaler for normalization.

4.6 One-Hot Encoding:

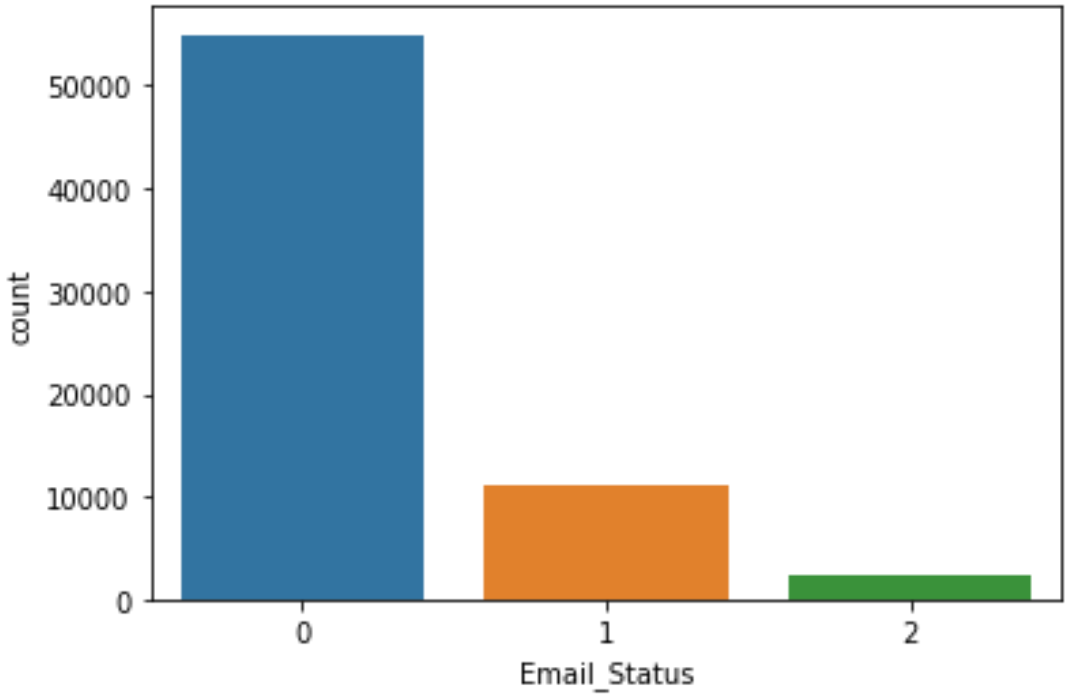
Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves.

This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application. A very good way to solve this problem for nominal data is One Hot Encoding. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

CHAPTER 5: HANDLING IMBALANCE

Looking at the Target variable Email_Status there is an observation of imbalance of dataset. Imbalanced classification refers to a classification predictive modeling problem where the number of examples in the training dataset for each class label is not balanced. Close to 80% data is of class 0, 16% data is of class 1 and 4% is of class 2.



5.1 Problems with imbalance:

One of the main challenges faced by the utility industry today is electricity theft. Electricity theft is the third largest form of theft worldwide. Utility companies are increasingly turning towards advanced analytics and machine learning algorithms to identify consumption patterns that indicate theft. However, one of the biggest stumbling blocks is the humongous data and its distribution. Fraudulent transactions are significantly lower than normal healthy transactions i.e. accounting it to around 1-2 % of the total number of observations. The ask is to improve identification of the rare minority class as opposed to achieving higher overall accuracy. Machine Learning algorithms tend to produce unsatisfactory classifiers when faced with imbalanced

datasets. For any imbalanced data set, if the event to be predicted belongs to the minority class and the event rate is less than 5%, it is usually referred to as a rare event

To handle imbalance there are primarily 2 ways:

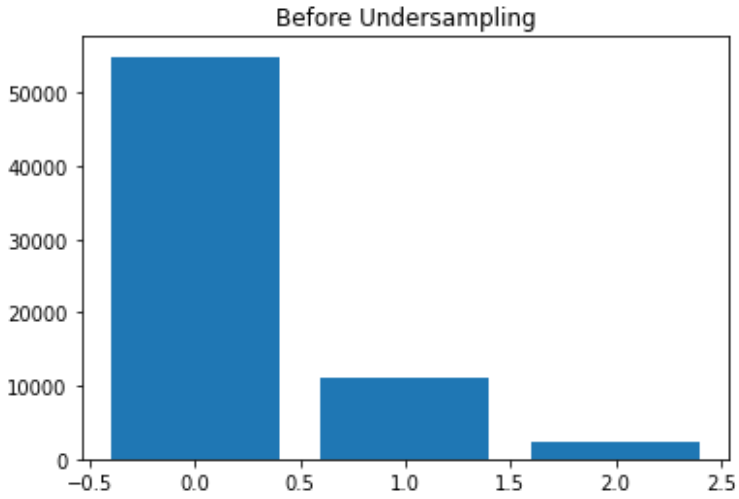
- UnderSampling
- OverSampling

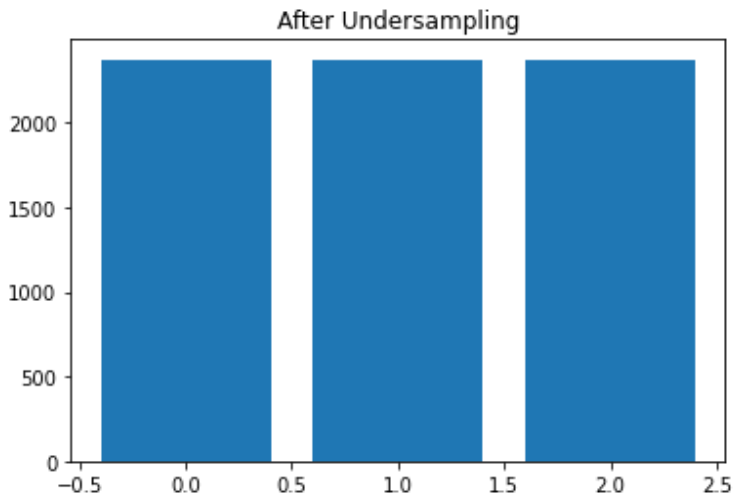
5.2 UnderSampling:

Undersampling can be defined as removing some observations of the majority class. This is done until the majority and minority class is balanced out.

Undersampling can be a good choice when you have a ton of data -think millions of rows. But a drawback to undersampling is that we are removing information that may be valuable. For the scope of the project, the technique used was Random UnderSampler and it created balanced dataset of 2373 records.

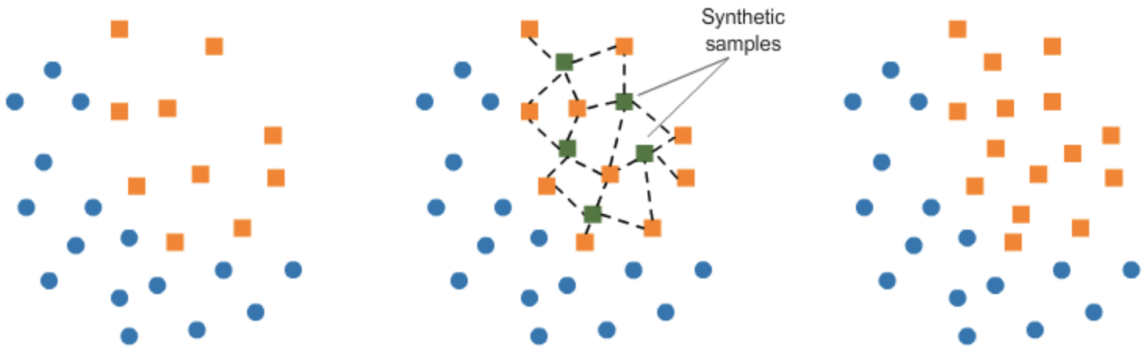
Created baseline models with undersampled data and it was observed that they underperformed primarily due to loss of information.



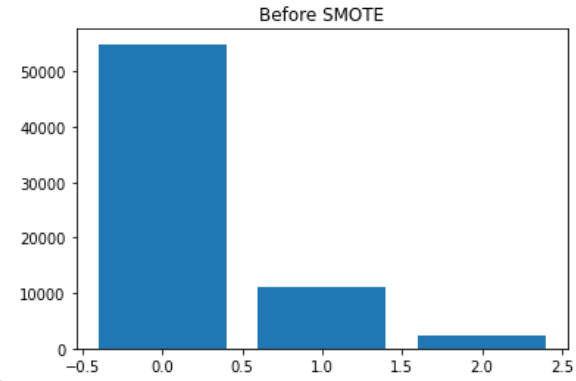


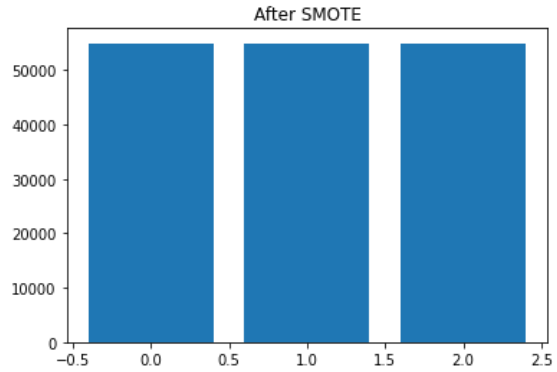
5.3 OverSampling:

The technique used is SMOTE and it generates synthetic data for the minority class. SMOTE (Synthetic Minority Oversampling Technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.



For the scope of the project, the technique used was SMOTE and it created balanced dataset of 54941 records.


Before SMOTE
50000
40000
30000
20000
10000
0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

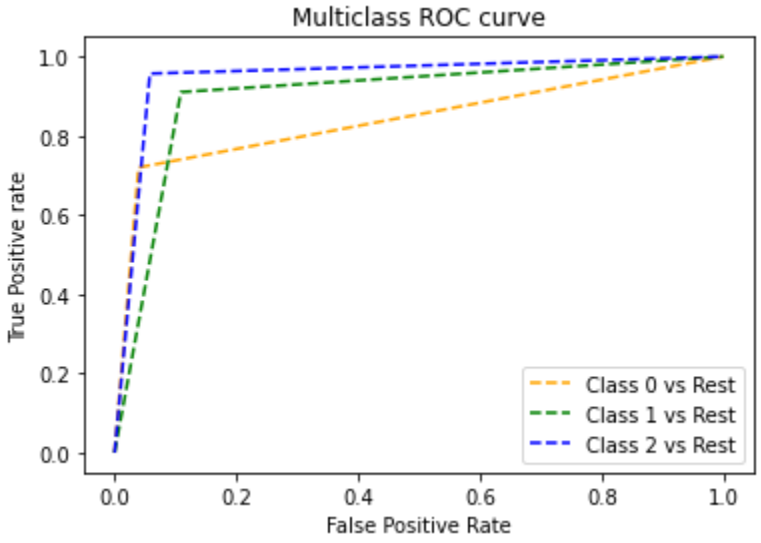

After SMOTE
50000
40000
30000
20000
10000
0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

CHAPTER 6: MODELLING AND EVALUATION

6.1 KNN -:

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

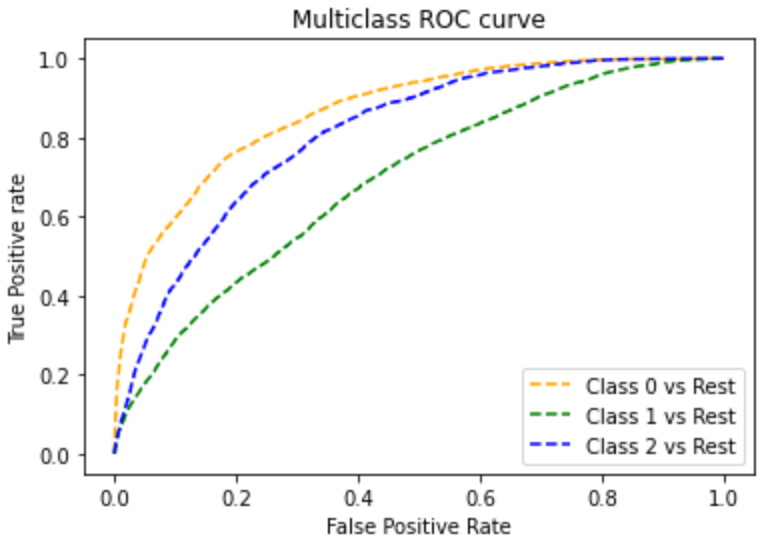
F1_Score- Train_Set :0.99 , Test_Set:0.85



6.2 Random Forest -:

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

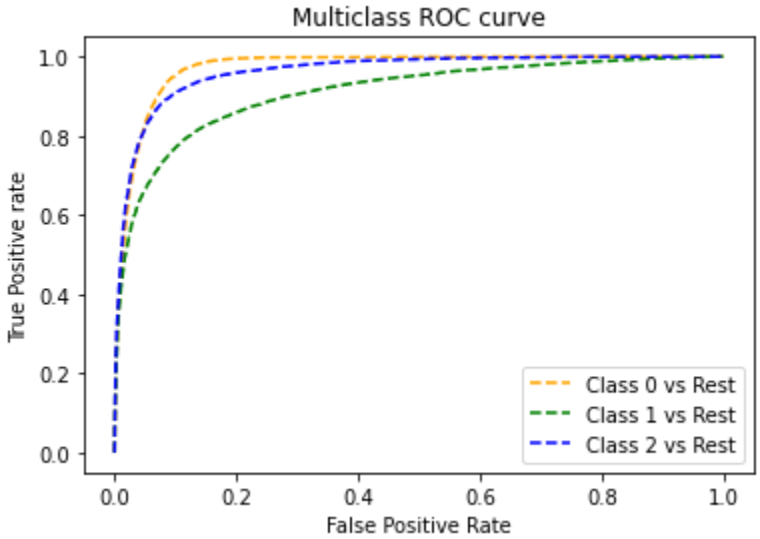
F1_Score- Train_Set :0.99 , Test_Set:0.85



6.3 XGBoost-:

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

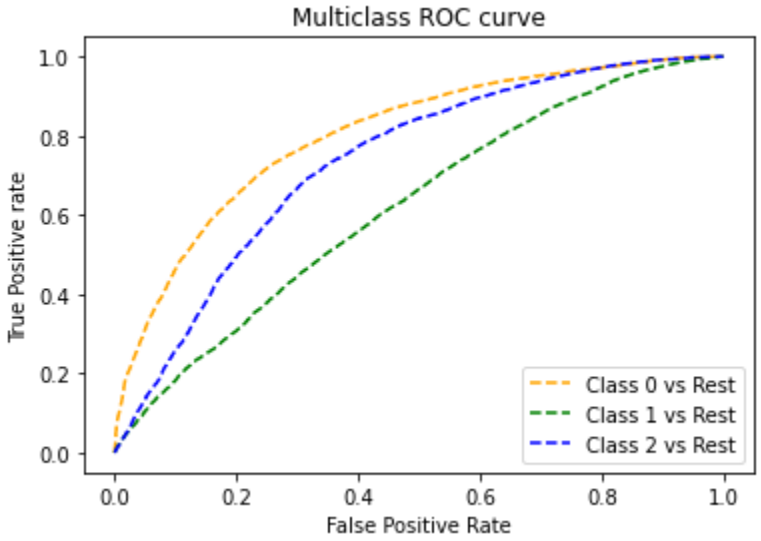
F1_Score- Train_Set :0.89 , Test_Set:0.81



6.4 Logistic Regression-:

In statistics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc.

F1_Score- Train_Set :0.50 , Test_Set:0.51



CONCLUSION

- In EDA, we observed that Email_Campaign_Type was the most important feature. If your Email_Campaign_Type was 1, there is a 90% likelihood of your Email to be read/acknowledged.

- It was observed that both Time_Email_Sent and Customer_Location were insignificant in determining the Email_status. The ratio of the Email_Status was the same irrespective of the demographic location or the time frame the emails were sent on.

- As the word_count increases beyond the 600 mark we see that there is a high possibility of that email being ignored. The ideal mark is 400-600. No one is interested in reading long emails !

- For modelling, it was observed that for imbalance handling Oversampling i.e. SMOTE worked way better than undersampling as the latter resulted in a lot of loss of information.

- Based on the metrics, XGBoost Classifier worked the best, giving a train score of 89% and test score of 81% for F1 score.

CHALLENGES

- Choosing the appropriate technique to handle the imbalance in data was quite challenging as it was a tradeoff b/w information loss vs risk of overfitting.
- Overfitting was another major challenge during the modelling process.
- Understanding what features are most important and what features to avoid was a difficult task.
- Decision making on missing value imputations and outlier treatment was quite challenging as well.