Email Campaign Effectiveness Prediction

By Vithika Karan
Data Science Trainee
AlmaBetter, Bangalore

Abstract:

Email advertising is the act of sending promotional emails to customers in mass quantities. It commonly is to generate income or leads and it can include advertising. Most importantly, email marketing allows businesses to build relationships with leads, new customers and past customers. It's a way to communicate directly to the customers in their inbox, at a time that is convenient for them. With the right messaging tone and strategies, emails are one of the most important marketing channels.

Email campaign effectiveness is a way of analyzing the kind of email campaigns being run by businesses in order to carry out their marketing and promotional agendas and hence they need to know how well the campaign is working.

The work here characterizes and predicts the emails if they are going to be ignored; read; acknowledged on the basis of the various features related to the emails in the dataset and makes recommendations to lower the number of ignored emails.

Keywords: EDA, Correlation, XGBoost, Random Forest, MultiClass Classification

Problem Statement:

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. 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. Data columns are self-explanatory.

Introduction:

The competition in the commercial world is so tough these days. The customers have so many options to go to, it is important to keep good relations with your customers to stay on top of the game. We see a lot of marketing strategies around us and almost half of our time we are engaging with different kinds of promotional schemes and advertising. One of the major digital marketing strategies of businesses involves the use of emails. Email Marketing can be summarized as a marketing technique in which businesses stay connected with their customers through emails, making them aware about their new products, updates, and important notices related to the products they are using.

Talking from a business's point of view, they'd think that the emails they are creating are special, even the best and people are going to be excited about these promotions but this is not necessarily the case. They are just one of the many emails they are getting every single day. We all subscribe to many different kinds of businesses through emails simply because it's required to do so these days, sometimes to get digital receipts of the things we bought or to get digital information about the businesses to stay updated. But many times we are not interested in reading those kinds of emails due to a number of reasons - to name a few would be- no proper structure, too many images, too many links inside the mail, complex vocabulary used or simply too long emails.

In this problem statement, we will be trying to create machine learning models that characterize and predict whether the mail is ignored, read or acknowledged by the reader. In addition to this, we will be trying to analyze and find all the features that are important for an email to not get ignored and based on that some recommendations are made.

Approach:

The approach followed here is to first check the sanctity of the data and then understand the features involved. The events followed were in our approach:

- **Understanding the Data**
- **Data cleaning and preprocessing-** finding null values and imputing them with appropriate values.
- **Exploratory data analysis-** of categorical and continuous variables against our target variable.
- **Data manipulation-** feature selection and engineering, handling multicollinearity with the help of VIF scores, feature scaling and encoding.
- **Handling Class Imbalance-** our dataset was highly imbalance with 80% majority, strategy was to splitting the stratified dataset and undersampling and oversampling with SMOTE on the train sets only so that our test set remains unknown to the models
- **Modeling**- worked on an evaluation code which was frequently used to evaluate the same models on undersampled and oversampled data in one go, logistic regression, decision trees, random forest, KNN and XGB were run to evaluate the results and then concluded on the basis of model performance and some recommendations were made to improve the numbers of read and acknowledged emails.

Understanding the Data:

First step involved is understanding the data and getting answers to some basic questions like; What is the data about? How many rows or observations are there in it? How many features are there in it? What are the data types? Are there any missing values? And anything that could be relevant and useful to our investigation. Let’s just understand the dataset first and the terms involved before proceeding further.
Our dataset consists of 68353 observations (i.e. rows) and 12 features (columns) about the emails. The data types were of integer, float and object in nature.

Let’s define the features involved:

- **Email Id** - It contains the email id's of the customers/individuals
- **Email Type** - There are two categories 1 and 2. We can think of them as marketing emails or important updates, notices like emails regarding the business.
- **Subject Hotness Score** - It is the email's

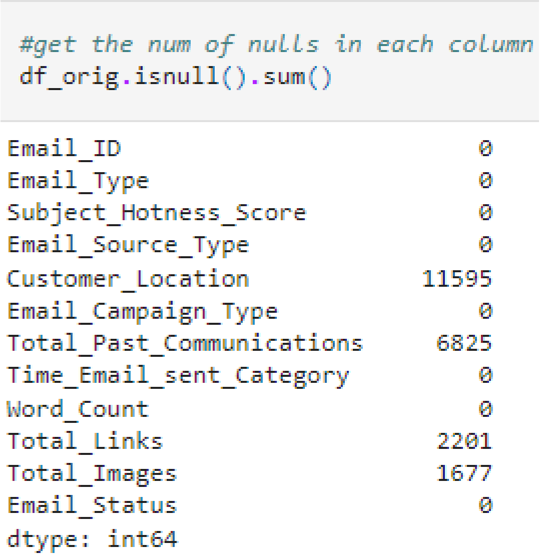
subject's score on the basis of how good and effective the content is.

- **Email Source** - It represents the source of the email like sales and marketing or important admin mails related to the product.
- **Email Campaign Type** - The campaign type of the email.
- **Total Past Communications** - This column contains the total previous mails from the same source, the number of communications had.
- **Customer Location** - Contains demographical data of the customer, the location where the customer resides.
- **Time Email sent Category** - It has three categories 1,2 and 3; the time of the day when the email was sent, we can think of it as morning, evening and night time slots.
- **Word Count** - The number of words contained in the email.
- **Total links** - Number of links in the email.
- **Total Images** - Number of images in the email.
- **Email Status** - Our target variable which contains whether the mail was ignored, read, acknowledged by the reader.

Data Cleaning and Preprocessing:

Handling missing values is an important skill in the data analysis process. If there are very few missing values compared to the size of the dataset, we may choose to drop rows that have missing values. Otherwise, it is better to replace them with appropriate values.

It is necessary to check and handle these values before feeding it to the models, so as to obtain good insights on what the data is trying to say and make great characterisation and prediction which will in turn help improve the business's content.



```
#get the num of nulls in each column
df_orig.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
```

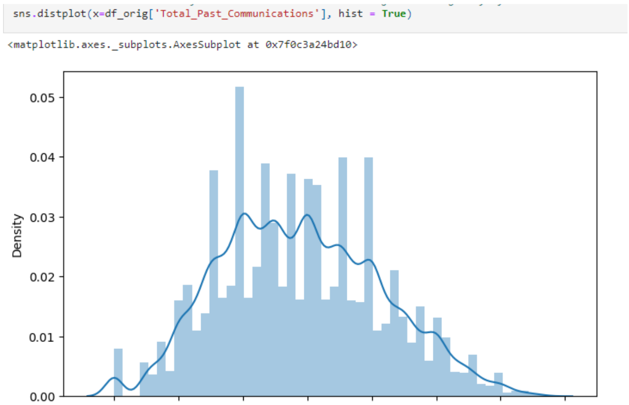
The dataset had a lot of nulls in the following columns:

- Customer Location
- Total Past Communications
- Total Links
- Total Images

But particular customer locations had a lot of them. Since it is a categorical column and it is difficult to just impute them with our understanding of where the customer's location is, it was important to see how much it affected our target variable, whether a particular location has anything to do with it or it is not correlated at all. If a particular location influences the target variables and aids in getting it ignored or otherwise, it should be filled on a condition (on Email Status) row wise.

For the rest of the continuous variables, the distribution was plotted to get an idea on how to impute these variables.

Here’s the distribution plot of Total Past Communications:

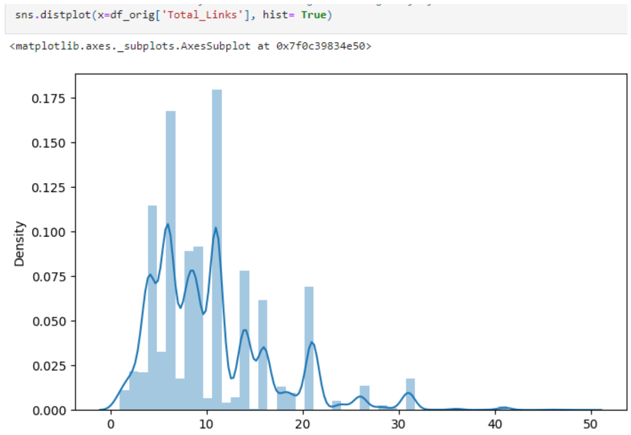


```
sns.distplot(x=df_orig['Total_Past_Communications'], hist = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c3a24bd10>
```

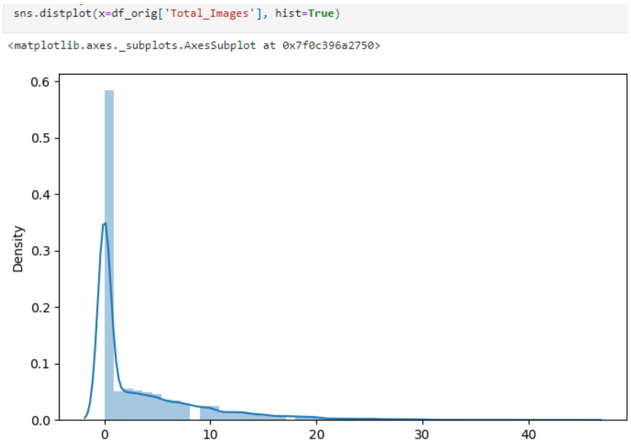
The plot showed kind of a normal distribution for Total Past Communications and that indicates the data is centered around the mean and hence mean of the column was imputed in the missing values of that column.

Going further, let's see the distribution of Total Links and Total Images.



```
sns.distplot(x=df_orig['Total_Links'], hist= True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c39834e50>
```



```
sns.distplot(x=df_orig['Total_Images'], hist=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c396a2750>
```

The distribution plot of both Total Links and Total Images are skewed to the right. Right skewed distributions occur when the long tail is on the right side of the distribution also called as positive skewed distribution which essentially suggests that there are positive outliers far along which influences the mean.

It seems like most of the values of the Total Links in the column are between 0-10 and the number of images in most of the emails seems to be 0 or fewer than 3-4. Consequently, the longer tail in an asymmetrical distribution pulls the mean away from the most common values. The mean is greater than the median. The mean overestimates the most common values in the distribution and hence mode (value with highest frequency) is used in these cases, it is more robust to outlier effect and the same is done here.

Exploratory Data Analysis:

Exploratory data analysis is a crucial part of data analysis. It involves exploring and analyzing the dataset given to find out patterns, trends and conclusions to make better decisions related to the data, often using statistical graphics and other data visualization tools to summarize the results. The visualization tools involved in our investigation are python libraries- matplotlib and seaborn.

The goal here is to explore the relationships of different variables with "Email Status" to see what factors might be contributing to ignored emails and then be able to correctly characterize the three of them.

Approach:

There are two kinds of features in the dataset: Categorical and Non Categorical Variables.

Categorical- A categorical variable is a variable that can take on one of a limited, and usually fixed, number of possible values putting a a particular category to the observation.

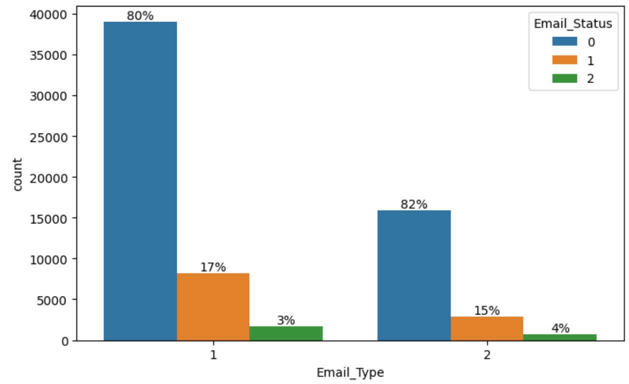
Non Categorical- A non categorical or continuous variable is a variable whose value is obtained by measuring, i.e., one which can take on an uncountable set of values.

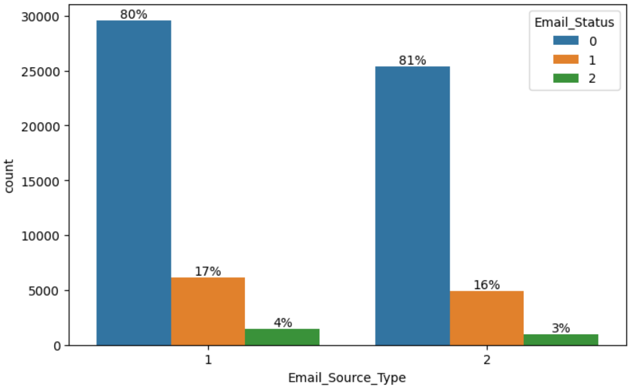
Both of them are analyzed separately.

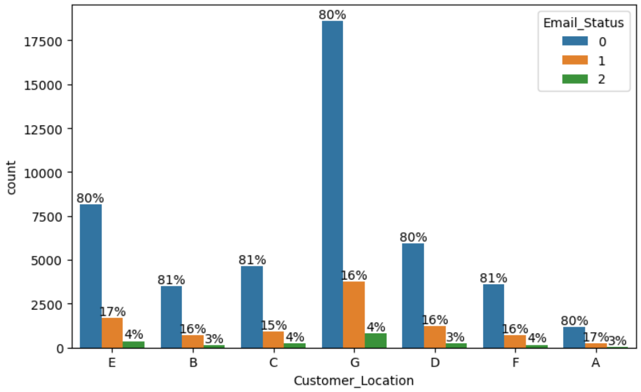
Categorical data is usually analyzed through count plots in accordance with the target variable and that is what is done here too.

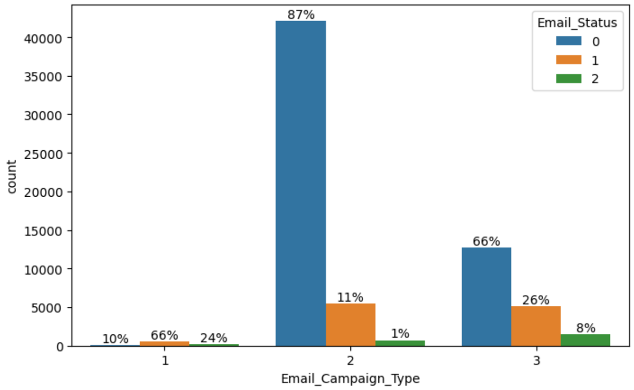
On the other hand Numeric or Continuous variables were analyzed through distribution plots and box plots to get useful insights.

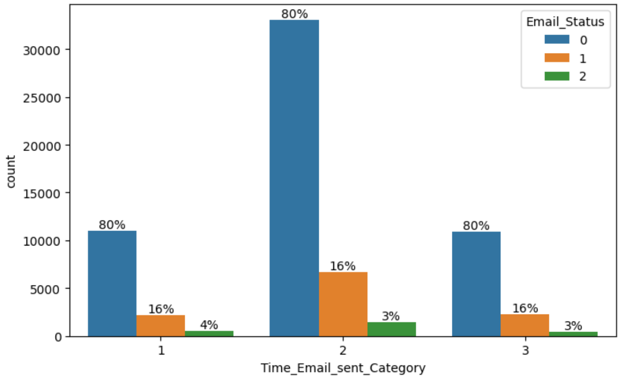
Categorical Insights:







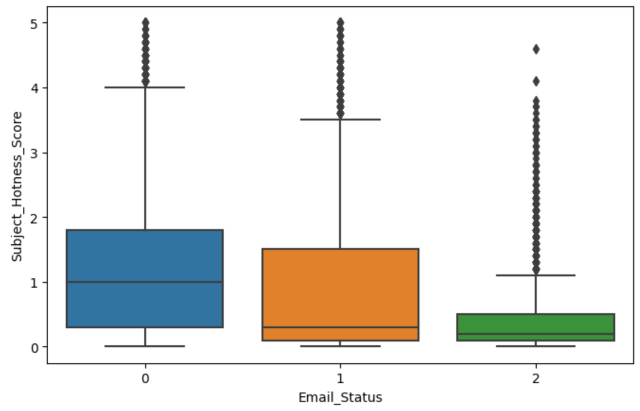


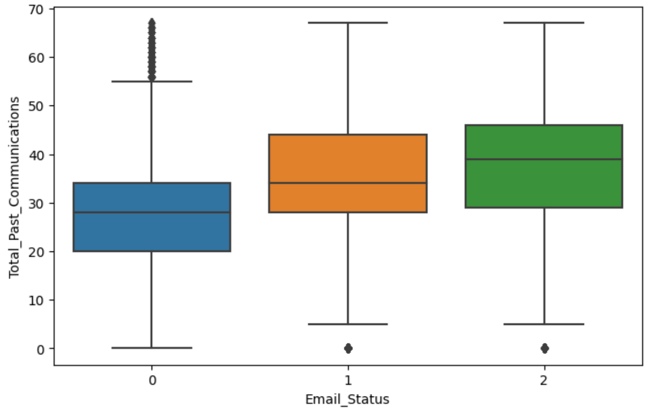


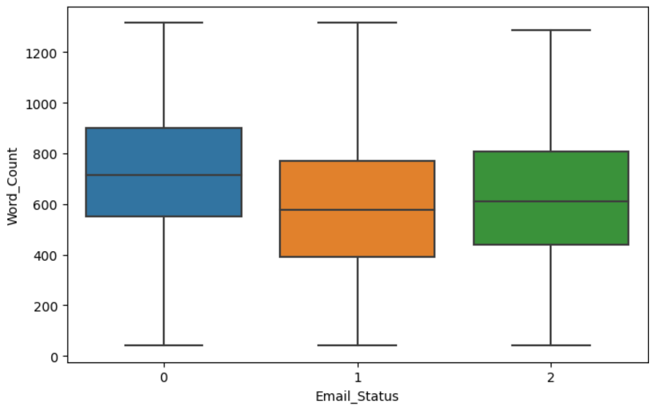
- The email type 1 which may be considered as promotional emails are sent more than email type 2 and hence are read and acknowledged more than the other type otherwise the proportion of ignored, read, acknowledged emails are kind of same in both email types.

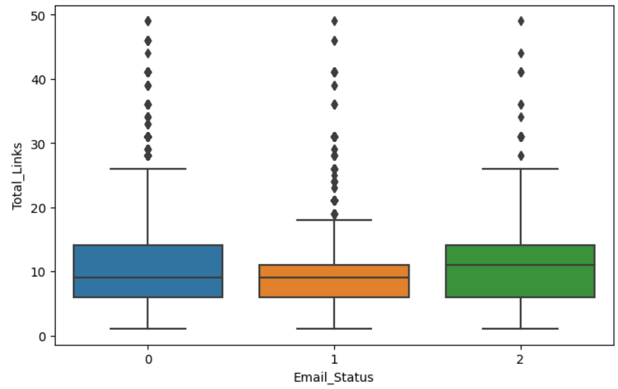
- Email source type shows kind of a similar pattern for both the categories.
- In the customer location feature we can find that irrespective of the location, the percentage ratio of emails being ignored, read and acknowledged are kind of similar. It does not exclusively influence our target variable. It would be better to not consider location as a factor in people ignoring, reading or acknowledging our emails. Other factors should be responsible for why people are ignoring the emails not location.
- In the Email Campaign Type feature, it seems like in campaign type 1 very few emails were sent but have a very high likelihood of getting read. Most emails were sent under email campaign type 2 and most ignored. Seems like campaign 3 was a success as even when less number of emails were sent under campaign 3, more emails were read and acknowledged.
- If we consider 1 and 3 as morning and night category in time email sent feature, it is obvious to think 2 as middle of the day and as expected there were more emails sent under 2nd category than either of the others, sending emails in the middle of the day could lead to reading and opening the email as people are generally working at that time and they frequently check up their emails, but it cannot be considered as the major factor in leading to acknowledge emails.

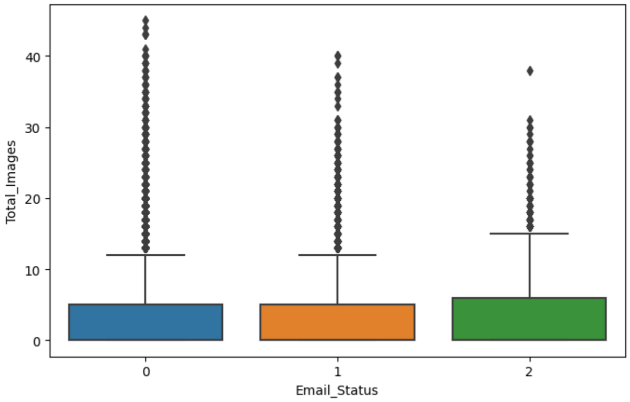
Continuous Insights:







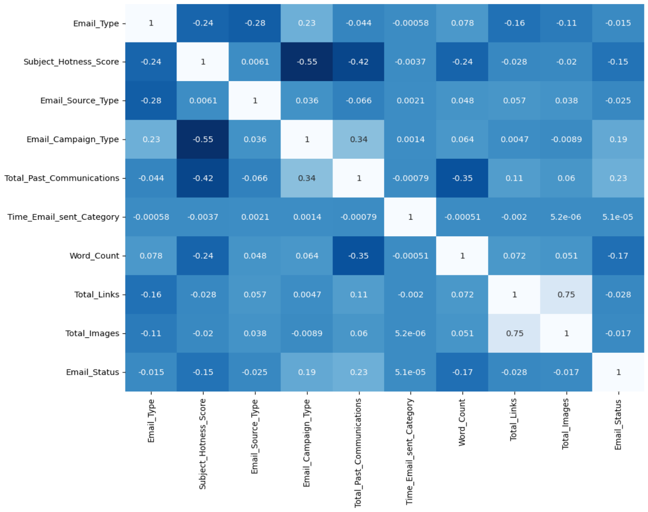




- Earlier the distribution plots of Total Links, Total Images and Total Past Communications were mentioned. Here it's observable through box plots as well that the Word Count just as Total Past Communications has kind of a normal distribution. All of the rest are rightly skewed which indicates the presence of outliers. When the median is closer to the box's lower values and the upper whisker is longer, it's a right-skewed distribution. Notice how the longer tail extends into the higher values and the dots outside the whiskers show the presence of a lot of outliers.
- Analyzing total past communications, we can see that the more the number of previous emails, the more it leads to read and acknowledged emails. This is just about making connections with your customers.
- The more the words in an email, the more it has a tendency to get ignored. Too lengthy emails are getting ignored.
- The median is kind of similar in all of the three cases in total links feature with a number of outliers.
- More images were there in ignored emails.

Correlation:

Correlation is a statistical term used to measure the degree in which two variables move in relation to each other. A perfect positive correlation means that the correlation coefficient is exactly 1. This implies that as one variable moves, either up or down, the other moves in the same direction. A perfect negative correlation means that two variables move in opposite directions, while a zero correlation implies no linear relationship at all.



Correlation matrix justifies our earlier observations. Email Campaign Type and Total past communication shows positive correlation with emails being read and acknowledged. Word Count and Subject Hotness score are the most negative amongst others. There's multicollinearity involved in Email Campaign Type, Total past communication and Total links, Total Images among others and it's important to get rid of it for a proper analysis.

Data Manipulation:

Data manipulation involves manipulating and changing our dataset before feeding it to various classification machine learning models. This involves keeping important features handling multicollinearity in the dataset, outlier treatment and creating dummy variables if necessary.

Multicollinearity:

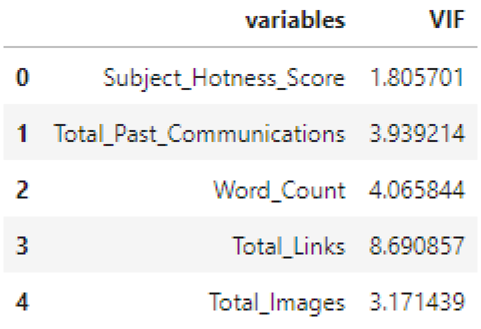
Multicollinearity occurs when two or more independent continuous variables are highly correlated with one another in classification or regression models. This means that an independent variable can be predicted by another independent variable and they both also predict the dependent variable. Multicollinearity makes it harder for the models to interpret the coefficients of individual variables or the role of them in predicting and hence in turn can exaggerate their roles and misclassify sometimes as well.

We can quantify multicollinearity using Variance Inflation Factors (VIF).

VIF determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable. VIF score of an independent variable represents how well the variable is explained by other variables.

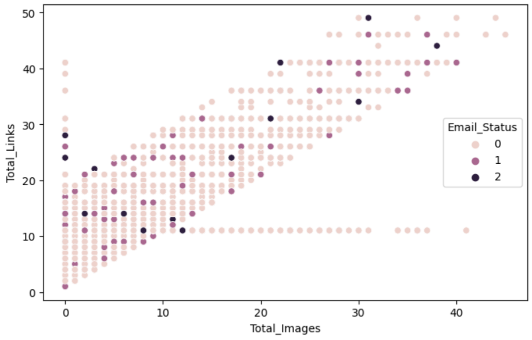
VIF = (1/(1-R^2))

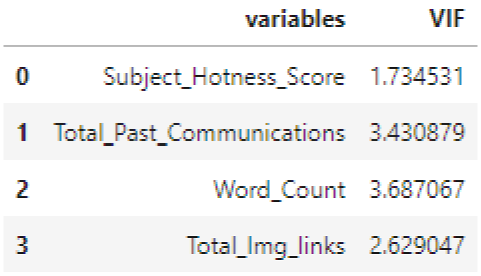
R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In general, the higher the R-squared, the better the model fits your data.The more the value of R^2 is closer to 1 the more, VIF score tends to infinity. VIF starts with 1 and denotes that the variable has no correlation at all. VIF more than 5-10 can be considered as a serious case of multicollinearity and can affect prediction models.



	variables	VIF
0	Subject_Hotness_Score	1.805701
1	Total_Past_Communications	3.939214
2	Word_Count	4.065844
3	Total_Links	8.690857
4	Total_Images	3.171439

The VIF results showed a high value for Total links and upon creating a scatter plot of Total Links and Total Images, it gave a linear relationship with some outliers, the essential step would be to combine both of these up and then check the VIF scores again and it was under check.

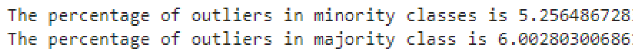




	variables	VIF
0	Subject_Hotness_Score	1.734531
1	Total_Past_Communications	3.430879
2	Word_Count	3.687067
3	Total_Img_links	2.629047

Outliers:

With the help of box-plots, we earlier saw that besides Word Count all our other continuous variables have outliers, but deleting them would lead to loss of information as our target variable is highly imbalanced we need to make sure that we aren't deleting more than 5% of information or data related to the minority class.



```
The percentage of outliers in minority classes is 5.256486728
The percentage of outliers in majority class is 6.00280300686
```

It is more than 5% information of our minority class. This dataset already has a high class imbalance issue, and if deleted this much amount of information from minority class will lead to a lack of information issue for predicting models and hence these were not deleted. They are going to affect the models either way.

Feature Scaling:

.Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is done to prevent biased nature of machine learning algorithms towards features with greater values and scale. The two techniques are:

Normalization: is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling. [0,1]

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Standardization: is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. [-1,1]

$$X' = \frac{X - \mu}{\sigma}$$

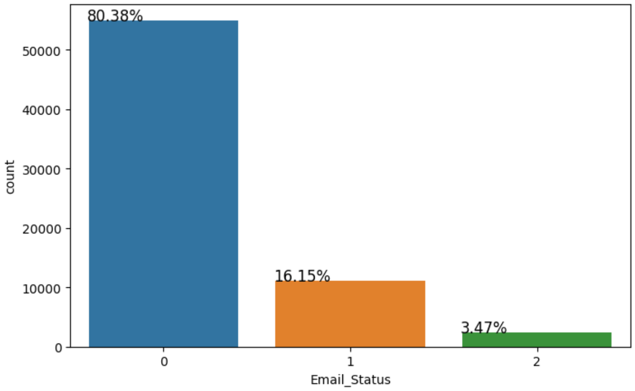
Normalization of the continuous variables was done here.

One hot encoding:

For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. We have categorical data integers encoded with us, but assuming a natural order and allowing this data to the model may result in poor performance. 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.

Handling Class Imbalance:

In the exploratory data analysis, we saw clearly that the number of emails being ignored was a lot more than being read and acknowledged. This imbalance in the class can lead to biased classification towards ignored emails.



Only 3% of observations are classified as acknowledged emails and 80% are ignored emails. This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called

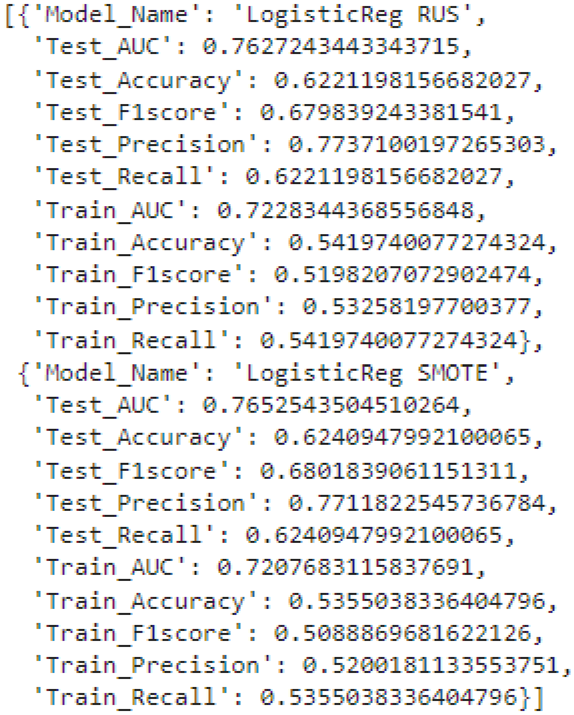
undersampling, and to duplicate examples from the minority class, called oversampling.
This project involves both of these techniques and compares the end result.

- Random undersampling deletes examples from the majority class and can result in losing information invaluable to a model.
- Oversampling is achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. One technique for this is Synthetic Minority Oversampling Technique, or SMOTE for short. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

The train test split was done before applying any resampling technique so that the test-dataset remains unknown to the models. Resampling of the train dataset was first done by Random undersampling and then by SMOTE.
Before balancing, it was made sure the train split has class distribution as same as the main dataset by using stratify while splitting.
The strategy here is to develop a model evaluation function which takes in both undersampled and oversampled data to evaluate and predict results and visualize model evaluation metrics for both of them.

Modeling:

Logistic Regression: Logistic Regression is a classification algorithm that predicts the probability of an outcome that can have only two values. Multinomial logistic regression is an extension of logistic regression that adds native support for multi-class classification problems. Instead, the multinomial logistic regression algorithm is a model that involves changing the loss function to cross-entropy loss and predicting probability distribution to a multinomial probability distribution to natively support multi-class classification problems.

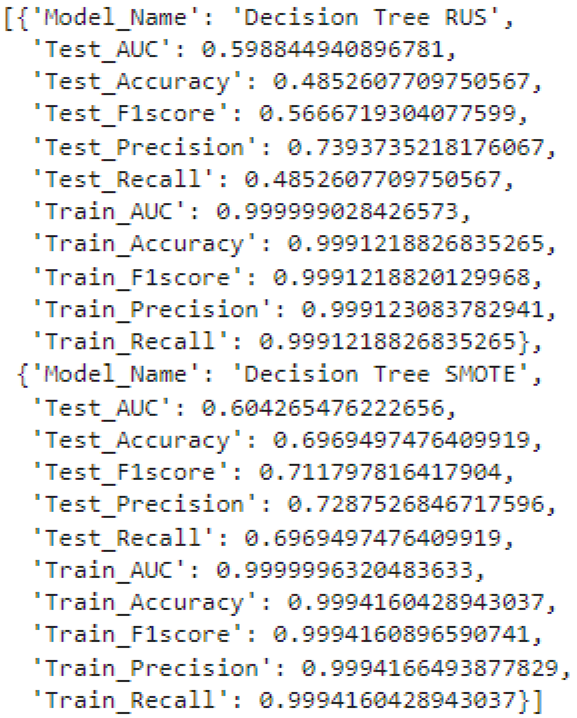


```
[{'Model_Name': 'LogisticReg RUS',
  'Test_AUC': 0.7627243443343715,
  'Test_Accuracy': 0.6221198156682027,
  'Test_F1score': 0.679839243381541,
  'Test_Precision': 0.7737100197265303,
  'Test_Recall': 0.6221198156682027,
  'Train_AUC': 0.7228344368556848,
  'Train_Accuracy': 0.5419740077274324,
  'Train_F1score': 0.5198207072902474,
  'Train_Precision': 0.53258197700377,
  'Train_Recall': 0.5419740077274324},
 {'Model_Name': 'LogisticReg SMOTE',
  'Test_AUC': 0.7652543504510264,
  'Test_Accuracy': 0.6240947992100065,
  'Test_F1score': 0.6801839061151311,
  'Test_Precision': 0.7711822545736784,
  'Test_Recall': 0.6240947992100065,
  'Train_AUC': 0.7207683115837691,
  'Train_Accuracy': 0.5355038336404796,
  'Train_F1score': 0.5088869681622126,
  'Train_Precision': 0.5200181133553751,
  'Train_Recall': 0.5355038336404796}]
```

Decision Trees: Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. Decision trees use the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Clearly Decision Tree models were overfitting.

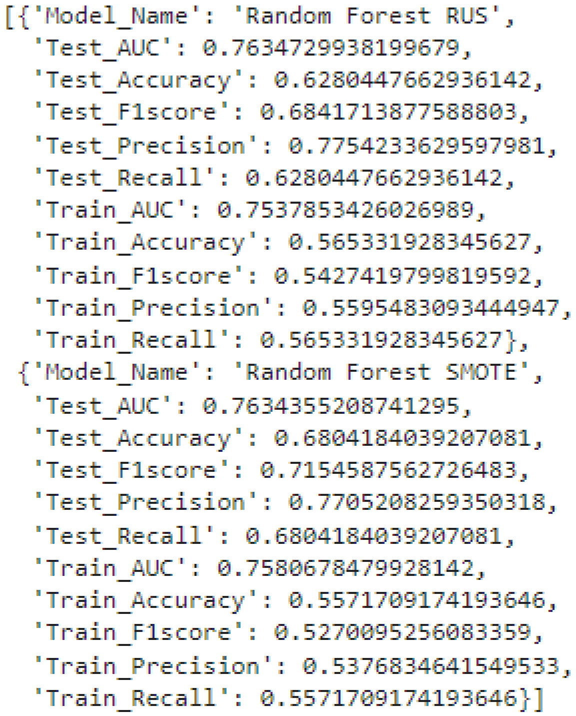
Both the datasets, whether undersampled or oversampled with SMOTE worked really well on train data but not on test data.



```
[{'Model_Name': 'Decision Tree RUS',
  'Test_AUC': 0.598844940896781,
  'Test_Accuracy': 0.4852607709750567,
  'Test_F1score': 0.5666719304077599,
  'Test_Precision': 0.7393735218176067,
  'Test_Recall': 0.4852607709750567,
  'Train_AUC': 0.999999028426573,
  'Train_Accuracy': 0.9991218826835265,
  'Train_F1score': 0.9991218820129968,
  'Train_Precision': 0.999123083782941,
  'Train_Recall': 0.9991218826835265},
 {'Model_Name': 'Decision Tree SMOTE',
  'Test_AUC': 0.604265476222656,
  'Test_Accuracy': 0.6969497476409919,
  'Test_F1score': 0.711797816417904,
  'Test_Precision': 0.7287526846717596,
  'Test_Recall': 0.6969497476409919,
  'Train_AUC': 0.9999996320483633,
  'Train_Accuracy': 0.9994160428943037,
  'Train_F1score': 0.9994160896590741,
  'Train_Precision': 0.9994166493877829,
  'Train_Recall': 0.9994160428943037}]
```

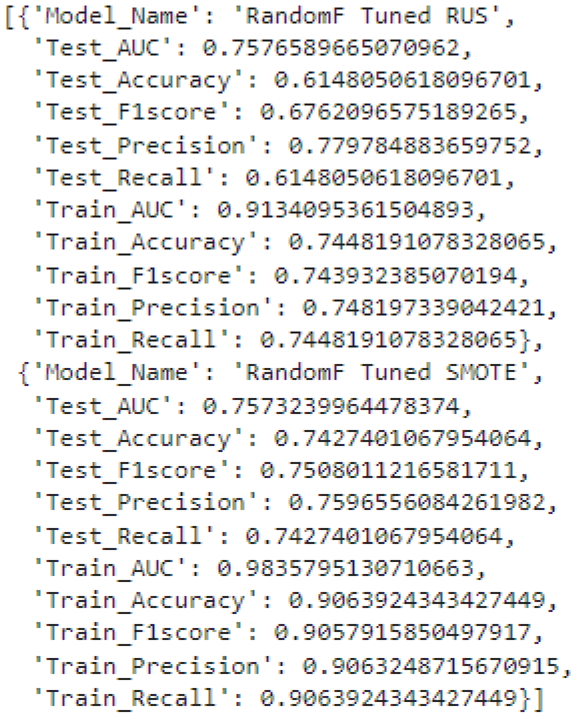
Random Forest: Random forests are an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.
To prevent overfitting, a random forest model was built. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

We got better results with SMOTE and decided to get a hyperparameter tuned model as well and then the tuned SMOTE version gave the best results till now with a good F1 score and AUC ROC.



```
[{'Model_Name': 'Random Forest RUS',
  'Test_AUC': 0.7634729938199679,
  'Test_Accuracy': 0.6280447662936142,
  'Test_F1score': 0.6841713877588803,
  'Test_Precision': 0.7754233629597981,
  'Test_Recall': 0.6280447662936142,
  'Train_AUC': 0.7537853426026989,
  'Train_Accuracy': 0.565331928345627,
  'Train_F1score': 0.5427419799819592,
  'Train_Precision': 0.5595483093444947,
  'Train_Recall': 0.565331928345627},
 {'Model_Name': 'Random Forest SMOTE',
  'Test_AUC': 0.7634355208741295,
  'Test_Accuracy': 0.6804184039207081,
  'Test_F1score': 0.7154587562726483,
  'Test_Precision': 0.7705208259350318,
  'Test_Recall': 0.6804184039207081,
  'Train_AUC': 0.7580678479928142,
  'Train_Accuracy': 0.5571709174193646,
  'Train_F1score': 0.5270095256083359,
  'Train_Precision': 0.5376834641549533,
  'Train_Recall': 0.5571709174193646}]
```

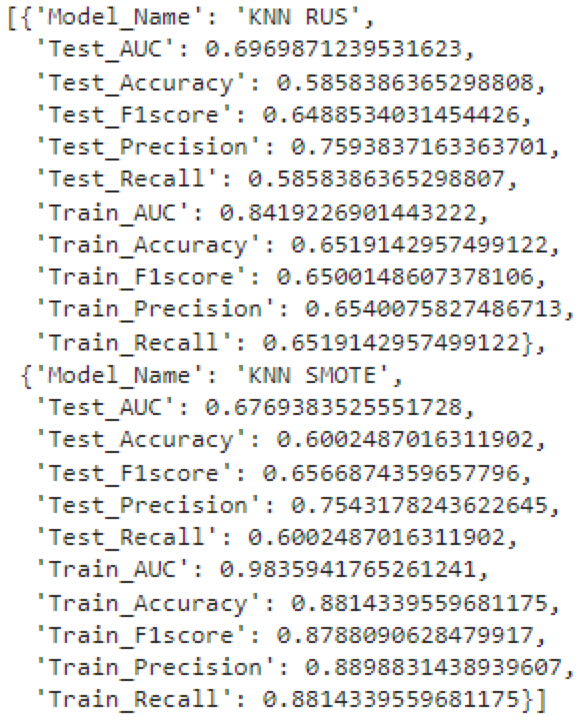
Random Forest Hyperparameter Tuned:



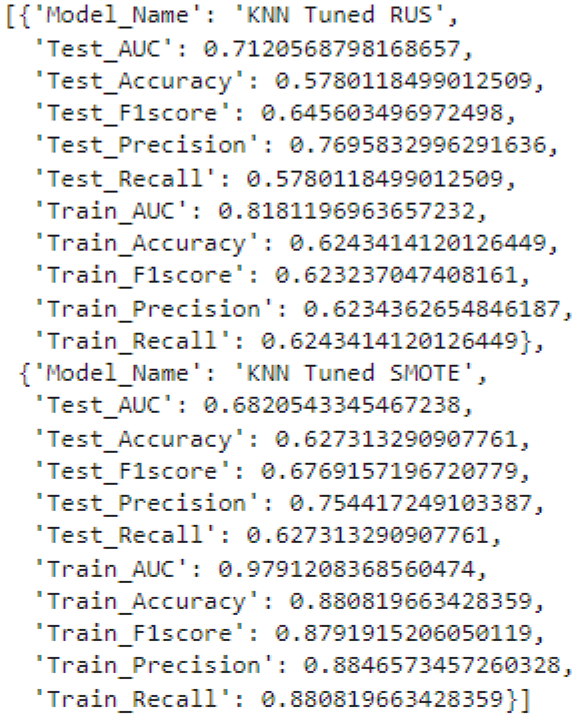
```
[{'Model_Name': 'RandomF Tuned RUS',
  'Test_AUC': 0.7576589665070962,
  'Test_Accuracy': 0.6148050618096701,
  'Test_F1score': 0.6762096575189265,
  'Test_Precision': 0.779784883659752,
  'Test_Recall': 0.6148050618096701,
  'Train_AUC': 0.9134095361504893,
  'Train_Accuracy': 0.7448191078328065,
  'Train_F1score': 0.743932385070194,
  'Train_Precision': 0.748197339042421,
  'Train_Recall': 0.7448191078328065},
 {'Model_Name': 'RandomF Tuned SMOTE',
  'Test_AUC': 0.7573239964478374,
  'Test_Accuracy': 0.7427401067954064,
  'Test_F1score': 0.7508011216581711,
  'Test_Precision': 0.7596556084261982,
  'Test_Recall': 0.7427401067954064,
  'Train_AUC': 0.9835795130710663,
  'Train_Accuracy': 0.9063924343427449,
  'Train_F1score': 0.9057915850497917,
  'Train_Precision': 0.9063248715670915,
  'Train_Recall': 0.9063924343427449}]
```

KNN Classification: K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN uses the concept of similarity in terms of distance.

We modeled through KNN classifiers and the results were worse, test recall with 0.59 indicated that there was a high number of false negatives involved and it made sense. Earlier we did not get rid of the outliers because more than 5% of minority data were outliers and this model evaluates on the basis of similarity. We tried to tune the hyperparameters and it did not make much of a difference.



```
[{'Model_Name': 'KNN RUS',
  'Test_AUC': 0.6969871239531623,
  'Test_Accuracy': 0.5858386365298808,
  'Test_F1score': 0.6488534031454426,
  'Test_Precision': 0.7593837163363701,
  'Test_Recall': 0.5858386365298807,
  'Train_AUC': 0.8419226901443222,
  'Train_Accuracy': 0.6519142957499122,
  'Train_F1score': 0.6500148607378106,
  'Train_Precision': 0.6540075827486713,
  'Train_Recall': 0.6519142957499122},
 {'Model_Name': 'KNN SMOTE',
  'Test_AUC': 0.6769383525551728,
  'Test_Accuracy': 0.6002487016311902,
  'Test_F1score': 0.6566874359657796,
  'Test_Precision': 0.7543178243622645,
  'Test_Recall': 0.6002487016311902,
  'Train_AUC': 0.9835941765261241,
  'Train_Accuracy': 0.8814339559681175,
  'Train_F1score': 0.8788090628479917,
  'Train_Precision': 0.8898831438939607,
  'Train_Recall': 0.8814339559681175}]
```



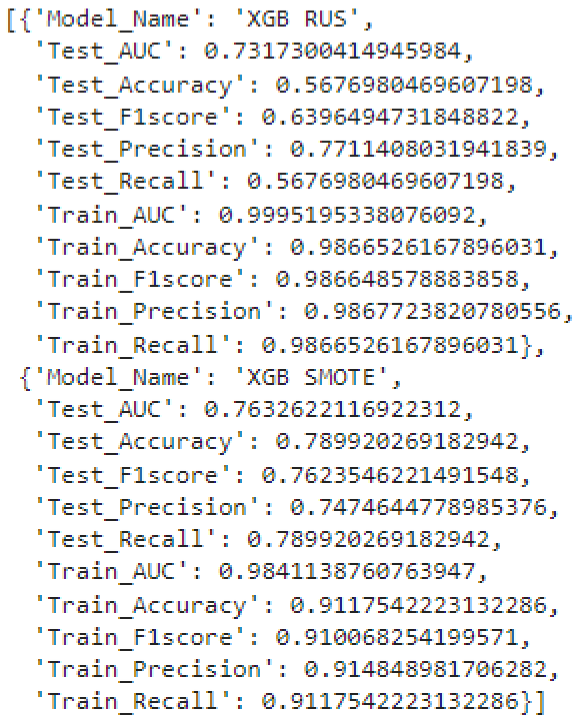
```
[{'Model_Name': 'KNN Tuned RUS',
  'Test_AUC': 0.7120568798168657,
  'Test_Accuracy': 0.5780118499012509,
  'Test_F1score': 0.645603496972498,
  'Test_Precision': 0.7695832996291636,
  'Test_Recall': 0.5780118499012509,
  'Train_AUC': 0.8181196963657232,
  'Train_Accuracy': 0.6243414120126449,
  'Train_F1score': 0.623237047408161,
  'Train_Precision': 0.6234362654846187,
  'Train_Recall': 0.6243414120126449},
 {'Model_Name': 'KNN Tuned SMOTE',
  'Test_AUC': 0.6820543345467238,
  'Test_Accuracy': 0.627313290907761,
  'Test_F1score': 0.6769157196720779,
  'Test_Precision': 0.754417249103387,
  'Test_Recall': 0.627313290907761,
  'Train_AUC': 0.9791208368560474,
  'Train_Accuracy': 0.880819663428359,
  'Train_F1score': 0.8791915206050119,
  'Train_Precision': 0.8846573457260328,
  'Train_Recall': 0.880819663428359}]
```

XG Boost: XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The two reasons to use XGBoost are also the two goals of the project:

- Execution Speed.
- Model Performance.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

XGB SMOTE gave the best results till now, with good Test Recall, F1 score and AUC ROC.



```
[{'Model_Name': 'XGB RUS',
  'Test_AUC': 0.7317300414945984,
  'Test_Accuracy': 0.5676980469607198,
  'Test_F1score': 0.6396494731848822,
  'Test_Precision': 0.7711408031941839,
  'Test_Recall': 0.5676980469607198,
  'Train_AUC': 0.9995195338076092,
  'Train_Accuracy': 0.9866526167896031,
  'Train_F1score': 0.986648578883858,
  'Train_Precision': 0.9867723820780556,
  'Train_Recall': 0.9866526167896031},
 {'Model_Name': 'XGB SMOTE',
  'Test_AUC': 0.763262211692312,
  'Test_Accuracy': 0.789920269182942,
  'Test_F1score': 0.7623546221491548,
  'Test_Precision': 0.7474644778985376,
  'Test_Recall': 0.789920269182942,
  'Train_AUC': 0.9841138760763947,
  'Train_Accuracy': 0.9117542223132286,
  'Train_F1score': 0.910068254199571,
  'Train_Precision': 0.914848981706282,
  'Train_Recall': 0.9117542223132286}]
```

Conclusion:

Challenges:

- We had a highly imbalanced target variable with 80% data of the majority class, 16% and 4% data of the minority classes respectively.
- The second challenge we faced was of outliers. Almost all the continuous variables had a good number of outliers. Upon calculating we came to know of the fact that more than 5% of outliers were in minority classes itself.
- This made the classifiers a bit confused, when there was a low number of data observations related to minority class and with outliers.
- The last challenge was to resample the unbalanced data. Many a times we see resampling on the whole dataset and then splitting it into train test set so that we don’t have an unbalanced validation set too but it may create a bias in the models and they might cheat towards synthetically created data in the validation set, so in this project resampling was done only on the training set and validation set was kept unknown to the models to predict on, which obviously gave lower results than the other way. Both ways were tried.

Evaluation Metrics:

There are a number of model evaluation metrics to choose from but since our dataset was highly imbalanced, it is critical to understand which metric should be evaluated to understand the model performance.

Accuracy- Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions. Accuracy is useful when the target class is well balanced but is not a good choice for the unbalanced classes, because if the model poorly predicts every observation as of the majority class, we are going to get a pretty high accuracy.

Confusion Matrix - It is a performance measurement criteria for the machine learning classification problems where we get a table with a combination of predicted and actual values.

Precision - Precision for a label is defined as the number of true positives divided by the number of predicted positives.

Recall - Recall for a label is defined as the number of true positives divided by the total number of actual positives. Recall explains how many of the actual positive cases we were able to predict correctly with our model.

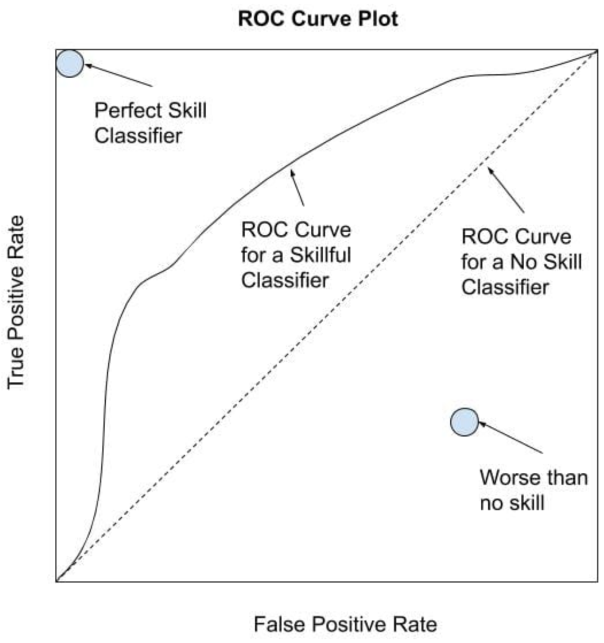
F1 Score - It's actually the harmonic mean of Precision and Recall. It is maximum when Precision is equal to Recall.

AUC ROC - The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes. When AUC is 0.5, the classifier is not able to distinguish between the classes and when it's closer to 1,the better it becomes at distinguishing them.

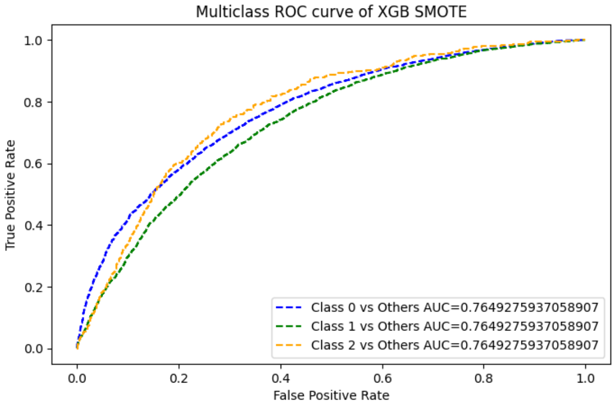
So among all the above metrics, which metric should be prioritized in comparing the performance of our various models? That's the major question here as we have a multiclass classification problem, where the problem statement just asks us to track and classify between ignored, read and acknowledged classes, we can not decide here what we want to prioritize in terms of classification, we just want to correctly classify and characterize accordingly.
When we have a high class imbalance, we'll choose the F1 score because a high F1 score considers both precision and recall. To get a high F1, both false positives and false negatives must be low. The F1 score depends on how highly imbalanced our dataset is!

Upon having this discussion, it’s clear that XGB SMOTE did the best classification, followed by Random Forest tuned SMOTE model.

.



Our Results:



Recommendations:

- Email Campaign Type 1 and 3 are doing better than 2. So, focusing on improving 2, can do the trick.
- The word count should be reasonable. The content should be crisp and to the point with a few marketing gimmicks.
- The number of images and links should be kept in check.

- Total past communications had a positive influence, hence having a healthy relationship with customers is a big yes.

References:

- Machine Learning Mastery
- GeeksforGeeks
- Analytics Vidhya Blogs
- Towards Data Science Blogs
- Built in Data Science Blogs
- Statistics by Jim