**EMAIL CAMPAIGN EFFECTIVENESS PREDICTION**

**Vishal Sahu**

**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.

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, Handling Imbalance***

**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:**

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.

Most 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 or 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 all cases.

**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.

**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.

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

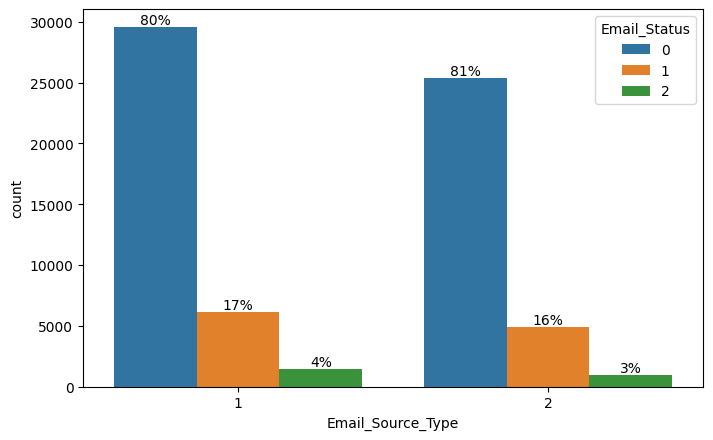
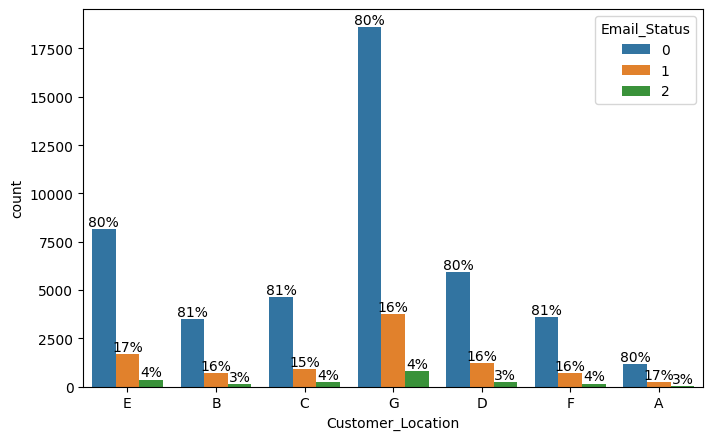
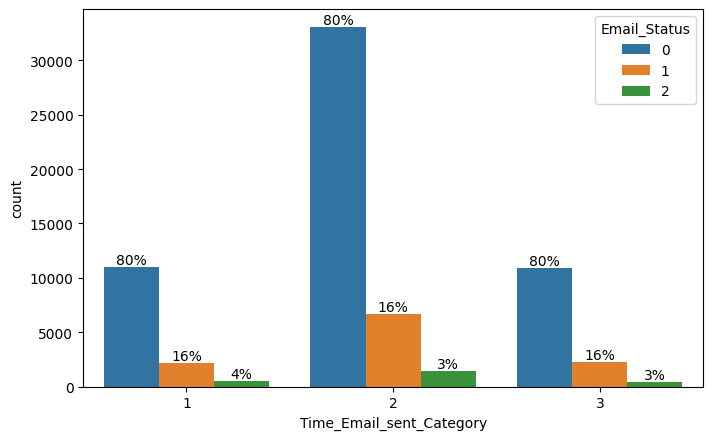
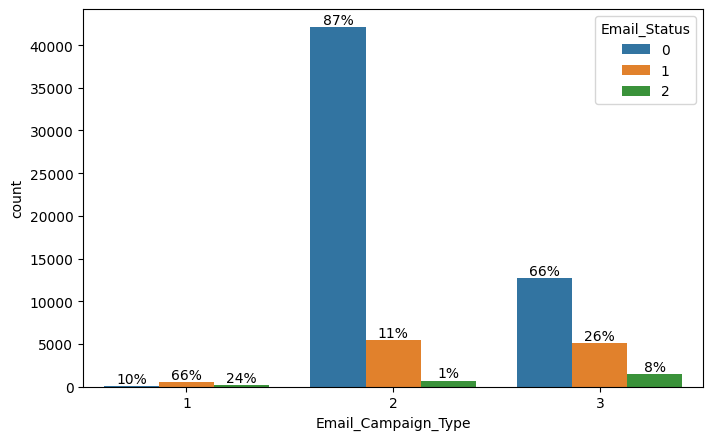
**Categorical**- A categorical variable is a variable that can take on one of a limited, and usually fixed, number of possible values putting a a particular category to the observation.

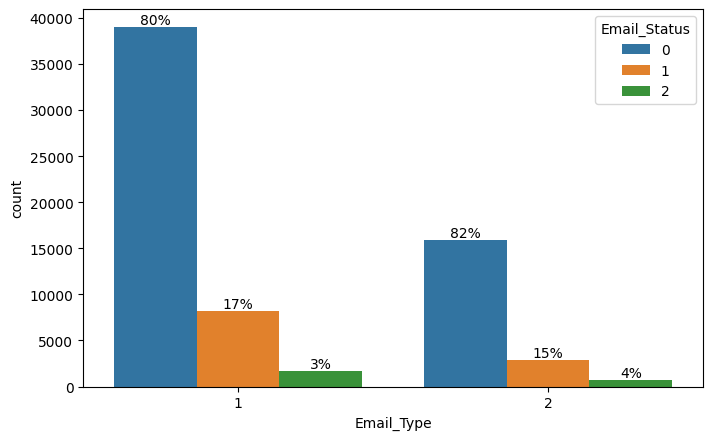
**Continuous**- A non categorical or continuous variable is a variable whose value is obtained by measuring, i.e., one which can take on an uncountable set of values.

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.

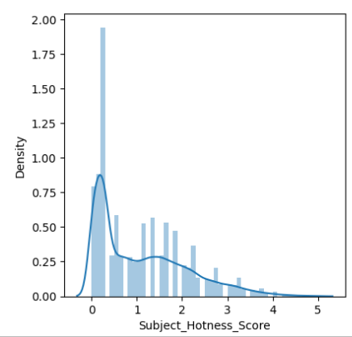
**Categorical Insights:**

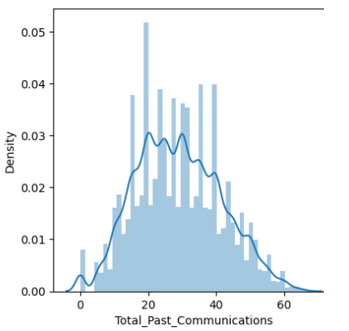
  

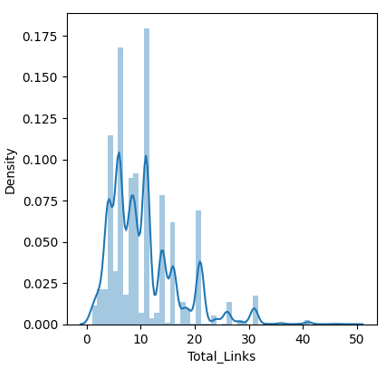


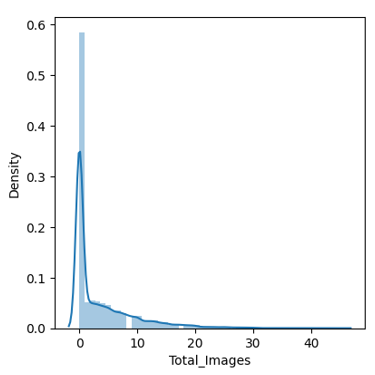
As it can observed that the distribution of Email\_Status is almost similar in all the categories except in Email\_Campaign\_Type, it shows a totally different trend. For Email\_Campaign\_Type = 1 it's only 10% of the customers who are ignoring the email and for 2 around 87% customer ignore the emails.

**Continuous Insights:**

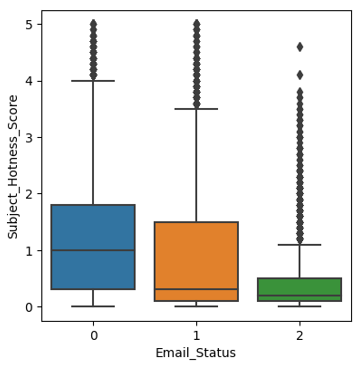


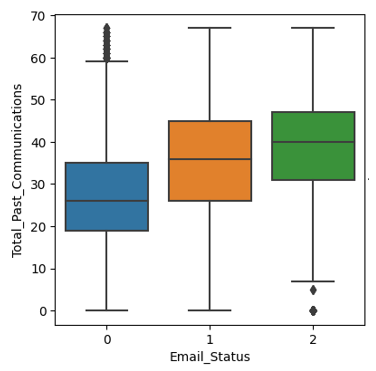


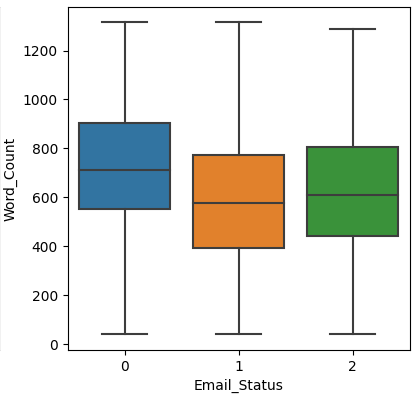


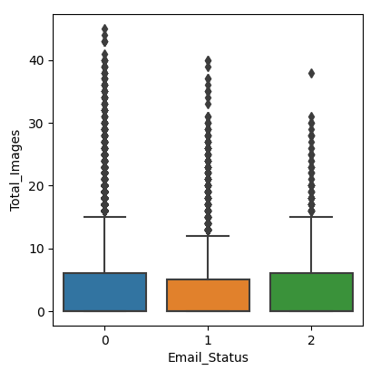


It's evident that Word Count and Total\_Past Communications follow almost a normal distribution. The rest of the features were highly skewed to the left.





****

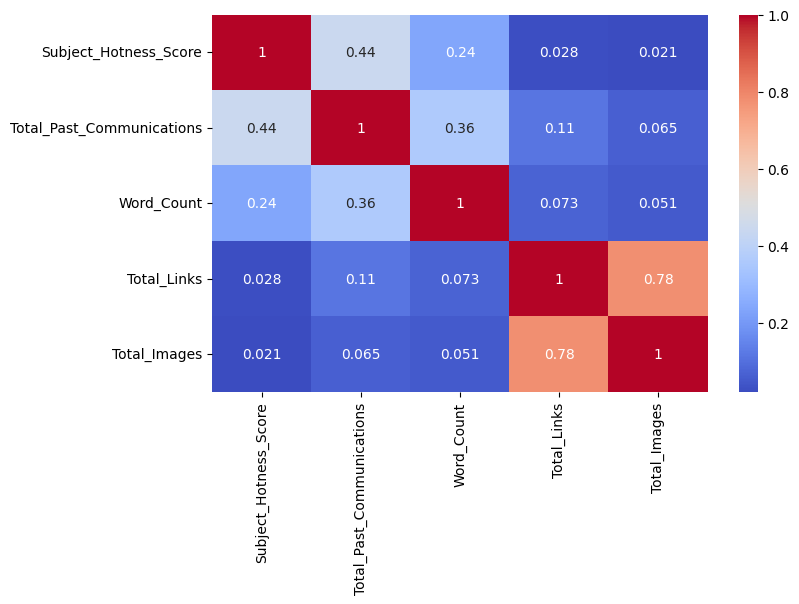


From the above boxplots, following observations can be made:

* For high Subject\_Hotness\_Score the chances of mail getting ignored is also high.
* 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.

**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.



Here it can observed that the correlation score is 0.78 for Total\_Images and Total\_Links which is on a scale of (-1 ,1) so it can be understood as a high positive correlation.

**Data Cleaning:**

**Handling Missing Data:**

It's been already seen in our missing values analysis that the Customer\_Location feature has the most number of missing values (16.963411 % 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 Customer\_Location.

* The Customer\_Location feature does not affect Email\_Status and it can be dropped.
* From the continuous data analysis part it's known get that the graph of Total\_past\_Communications follows approximate Normal Distribution. So, let's impute the missing values by the mean of the values.
* From the continuous data analysis part it's known that the graph of Total\_Links & Total\_Images is left skewed. So, imputing the missing values by the mode of the values is most appropriate.

**Feature Engineering:**

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. As you may know, a “feature” is any measurable input that can be used in a predictive model — it could be the color of an object or the sound of someone’s voice. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches.

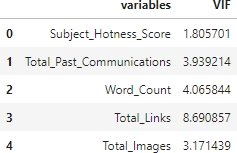
**Multicollinearity:**

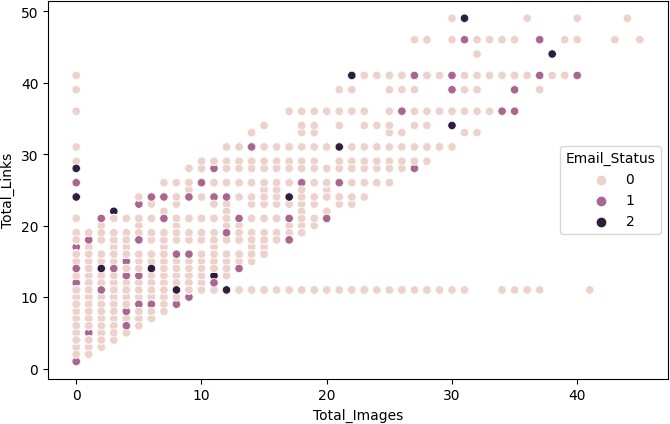
Multicollinearity occurs when two or more independent continuous variables are highly correlated with one another in classification or regression models. This means that an independent variable can be predicted by another independent variable and they both also predict the dependent variable. Multicollinearity 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).

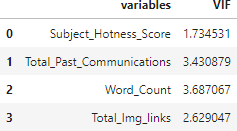
VIF determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable. VIF score of an independent variable represents how well the variable is explained by other variables.

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

R-squared is a statistical measure of how close the data are to the fitted regression line.

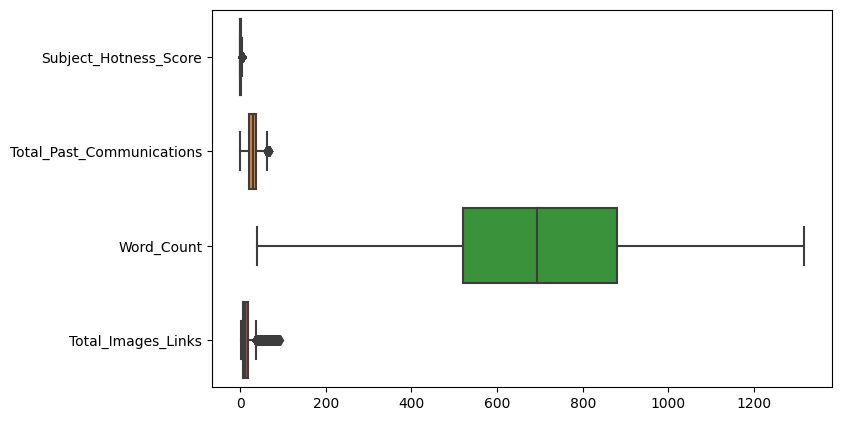






**Outliers:**

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.





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]



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]

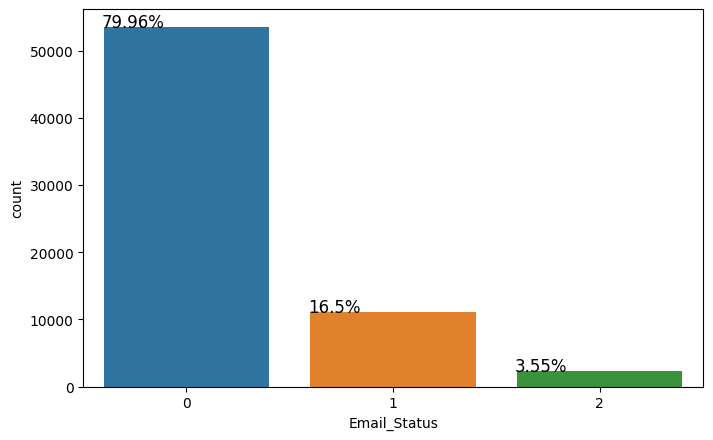
Normalization of the continuous variables was done in this step.

**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 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 3.5% 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 evaluates and predicts results and visualizes model evaluation metrics for both of them.

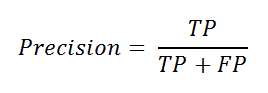
**Model Implementation and Evaluation:**

**Model Evaluation Metrics:**

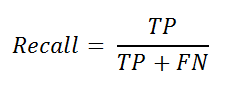
Evaluation metrics are tied to machine learning tasks. There are different metrics for the tasks of classification and regression. Some metrics, like precision-recall, are useful for multiple tasks. Classification and regression are examples of supervised learning, which constitutes a majority of machine learning applications. Using different metrics for performance evaluation, we should be able to improve our model’s overall predictive power before we roll it out for production on unseen data. Without doing a proper evaluation of the Machine Learning model by using different evaluation metrics, and only depending on accuracy, can lead to a problem when the respective model is deployed on unseen data and may end in poor predictions.

**Accuracy:** It is simply the ratio of the number of correct predictions to the number of all predictions.

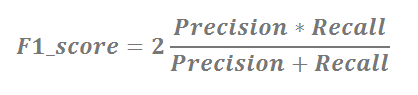
**Precision:** It is mainly used in binary classification tasks. It focuses on the positive predictions.



**Recall:** It is also used in binary classification tasks. It focuses on the positive class.



**F1 Score:** It's actually the harmonic mean of Precision and Recall.

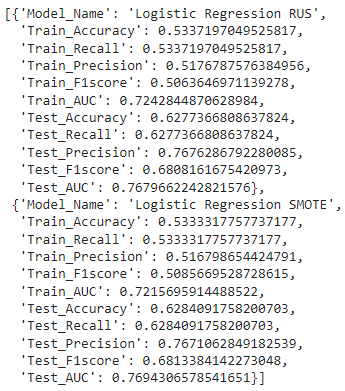


F1 score is a more useful measure than accuracy for problems with uneven class distribution because it takes into account both false positive and false negatives.

**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.

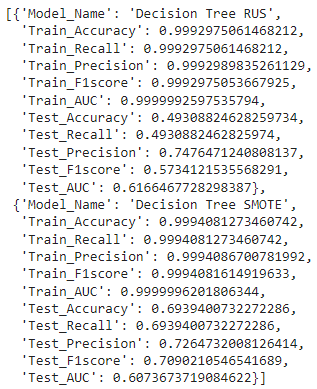


**Decision Trees:**

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.

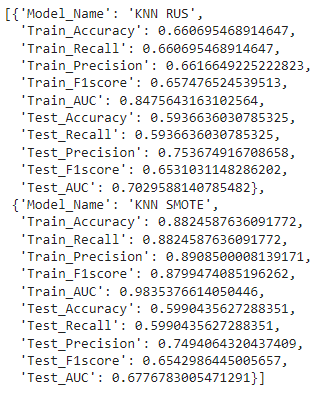
Both the datasets, whether undersampled or oversampled with SMOTE worked really well on train data but not on test data.



**KNN:**

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.

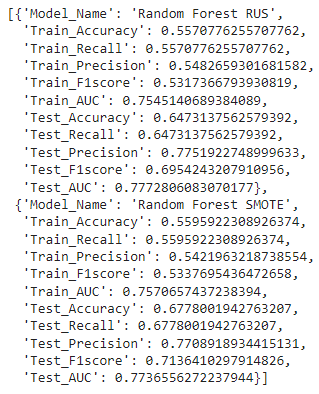
We modeled through KNN classifiers and the results were worse, test recall with 0.59 indicated that there was a high number of false negatives involved and it made sense. Earlier we did not get rid of the outliers because more than 5% of minority data were outliers and this model evaluates on the basis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



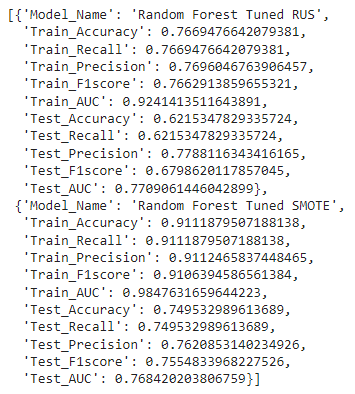
**Random Forest:**

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.



### **Random Forest Hyperparameter Tuning:**



**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:

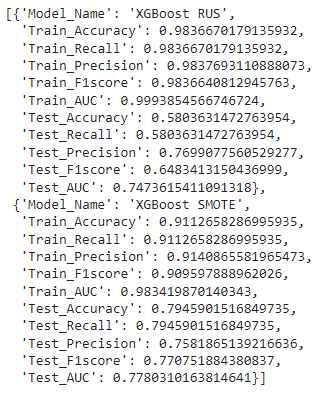
● Execution Speed.

● 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.

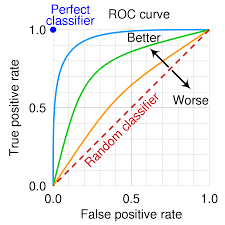
Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

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

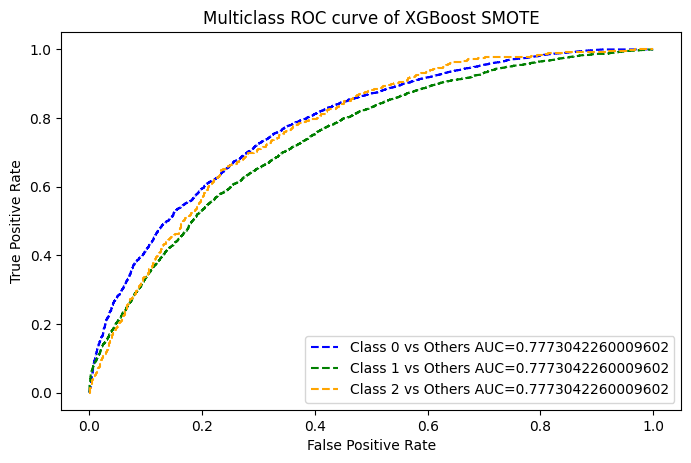


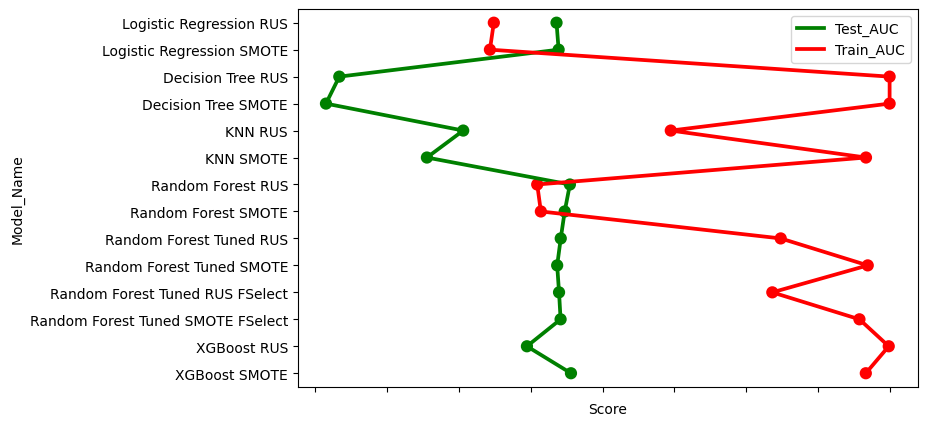
**Comparison of all the models:**

**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.



Our best model’s ROC curve:





**Conclusions:**

* It could be observed from the EDA that Email\_Campaign\_Type was the most important feature. If the Email\_Campaign\_Type was 1, there is a 90% likelihood of your Email to be acknowledged.
* As the word\_count increases beyond the 600 mark we can see that there is a high possibility of that email being ignored. The ideal mark was 400-600.
* Decision Tree Model was overfitting as it was working really good on train data but bad on test data.
* Hyperparameter tuning wasn't able to improve the results to a better extent and casused a lot computaional time.
* XGBoost Algorithm worked in the best way possible with such an imbalanced data having outliers, followed by Random Forest Hyperparameter Tuned model after feature selection with F1 Score of 0.75 on the test set.

**References:**

* GeeksforGeeks
* AnalyticsVidhya
* Medium
* Scikitlearn.org