**Email Campaign Effectiveness Prediction**

By – Ranajay Biswas

**Data Science Trainee,**

**Almabetter, Bangalore**

**Abstract:**

Email campaign is a sequence of marketing efforts that contacts multiple recipients at once. Email campaigns are designed to reach out to subscribers at the best time and provide valuable content and relevant offers. Using email campaigns allows businesses to build deep and trusting relationships with their customers.

Marketing through Email can make communication with clients easier and more effective.

Email campaigns are a very powerful medium between a business company and it’s audience. It helps not only to increase sales but build brand image.

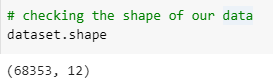
**Problem Statement:**

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 Summary & Attributes:**

The Email campaign data contains various types of information regarding the emails that were sent, it contains info about their customers and their responses. Checking the shape of the data, we found that it has 68353 observations and 12 columns.



**Attributes:**

* **Email Id** - It contains the email ids 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.

**Approach:**

Performing Exploratory data analysis will help us understand the features and relationships that they have and their impact on the target or the client's response. We will try to find out important features and we will do feature engineering.

Data is labeled and the target column being categorical, we shall implement classification based machine learning algorithms to complete the prediction task.

**Steps Involved:**

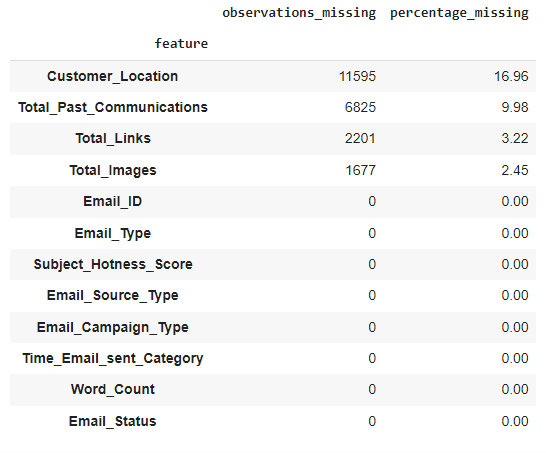
* **Loading & Understanding the Data**

The csv file containing the Email Data was loaded in our work environment as a data frame using pandas. We checked top 5 rows and bottom 5 rows to get an initial idea about the data.

* **Exploratory Data Analysis**

Exploratory Data Analysis is unavoidable and one of the major step to fine-tune the given data set(s) in a different form of analysis to understand the insights of the key characteristics of various entities of the data set like column(s), row(s) by applying Pandas, NumPy, Statistical Methods, and Data visualization packages.

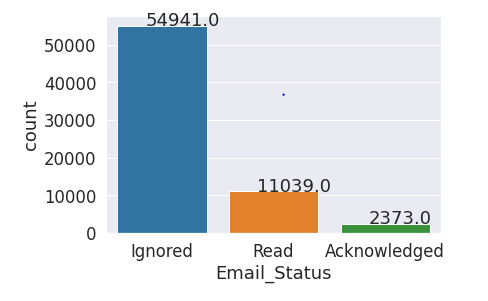
* **Checking Null Values:** Customer Location, Total Past Communications, Total Links & Total Images columns were the numerical columns that have null values.



* **Class Label Distribution of Dependent Variable:**

Dependent variable or target is the column that is going to be predicted by our machine learning models. In our case, that is the *Email Status* column.

Class labels for the target column is highly imbalanced. Only 3.5% of total sent emails were acknowledged. 16% were read and more than 80% were ignored.



* **Analysis of Categorical features:**

We have 5 categorical columns in the data, which are – ‘Email Type’, ‘Email Source Type’, ‘Customer Location’, ‘Email Campaign Type’, ‘Time Email sent Category’.

We used bar charts and count plots to see the differences in terms of numbers for different categories of each categorical column.

We checked the relationship of features for different class labels as well to find out how different categories for a feature impacted customer responses. The visualizations are available in the notebook and the presentation.

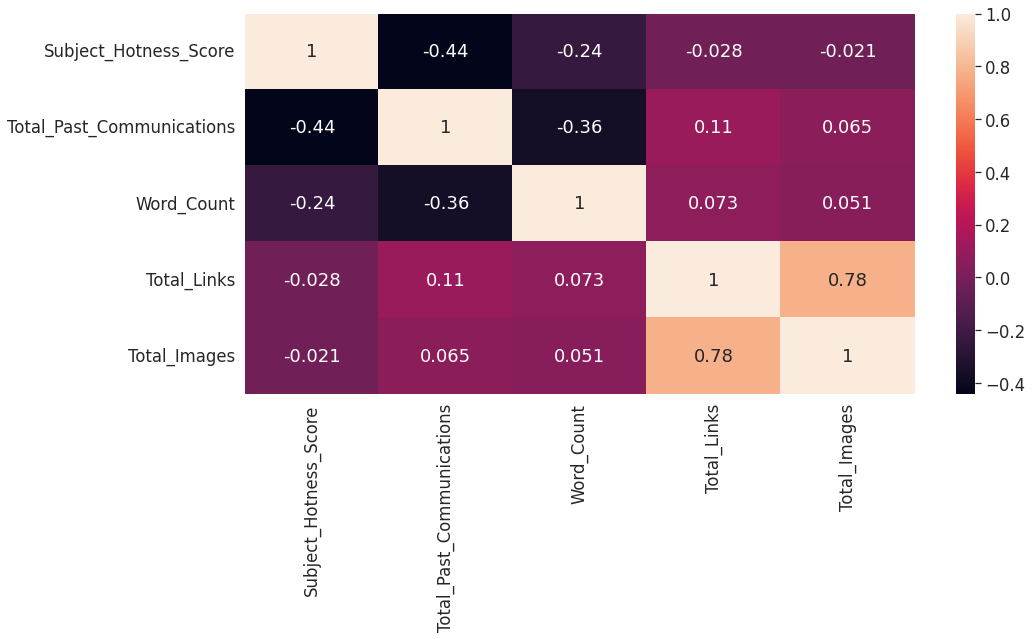
* **Analysis of Numerical columns :**

The numerical columns in the data are : ‘Subject Hotness Score’, ‘Total Past Communications’, ‘Word Count’, ‘Total Links’, ‘Total Images’.

Histograms were used to check the distribution of these features. Then we plotted the boxplots with respect to the target variable to see how difference in the values for a feature impact on the response of the emails.

* **Checking for Multicollinearity :**

Multicollinearity occurs when a change in an independent variable impacts one or more independent variables. It is a problem because it undermines the statistical significance of an independent variable.



From the Heat-map, we found that –

* **Total Images** & **Total Links** are the most correlated features in the data. The correlation strength is 0.78
* **Total Past Communications** & **Subject Hotness Score** share a negative correlation value of 0.44
* **Word Count** &  **Total Past Communications** have a moderate negative correlation of 0.36
* **Data Cleaning :**

**Unimportant Features:**

* **'Email ID'** column is just a unique identifier for each observation. It is not helpful for any kind of modelling. So, we drop this column.
* **'Customer Location'** column has the most number of missing values. And it being a categorical column, it is not easy to impute and we cannot simply use mode imputation for this many large number of missing values. It was previously seen that this column is not really helpful for class differentiation & prediction. So, instead of imputing this large number of null values, we dropped this column.

**Null Value Imputation:**

There were now 3 columns in the data with null values. We used **MICE** algorithm or the Iterative Imputation method for the null value imputation in these columns.

It is better than the general methods of mean or median imputation.

**Outlier Detection:**

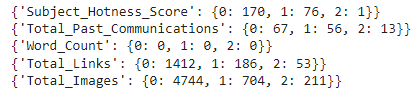
An outlier is usually a data point that differs significantly from other observations. In this project, we found the outliers using the IQR method for outlier detection.

**Outlier Handling:**

The target column being highly imbalanced, we needed a better understanding of how the outliers were distributed for different classes.

We found –

* All the numeric columns except **Word Count** have outliers.
* Minority classes also contain outliers, though in smaller numbers compared to the majority class.
* **'Total Images**' have 8.28 percent and **'Total Links**' have 2.42 percent outliers. **'Subject Hotness Score**' and **'Total Past Communications**' have 0.36 and 0.20 percent outliers respectively.



Since, we had more than enough data for majority class, we decided to drop the outliers only for those observations where the target is of majority class. Our data is imbalanced, so we do not want to lose any information about the minority classes.

* **Pre-Processing & Feature Engineering:**

**Feature Creation:**

To avoid multicollinearity, we combined **'Total Links**' and **'Total Images’** features and created a new feature and dropped the original features. The new feature is called ‘**Total link Images’**.

**Encoding:**

On the categorical features, we performed one hot encoding so that they can be passed through the models.

**Train-Test Split:** Train test split is the process of splitting dataset into 2 different parts. One part is used for training the model and other part is for testing the models performance. We performed train-test split with stratified sampling since dependent variable was imbalanced. 80 percent of data was used for training and 20 percent for testing.

**Feature Scaling:** We used mean max scaler for normalizing the data. This step might not be that necessary because we will be mostly using tree based and ensemble models. But performing this step doesn’t hurt.

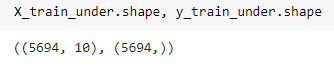
**Handling Imbalance:** There are many techniques of handling class imbalance. The process that we followed were –

1. **Undersampling :**

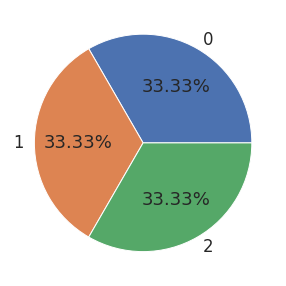
Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class.

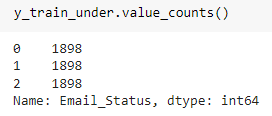
Though this process helps to bring balance, but we also lose a lot of information. This loss of information might affect models’ performances.

**Shape of the training data X and Y after undersampling:**

****

**Distribution of class labels for training data after undersampling :**





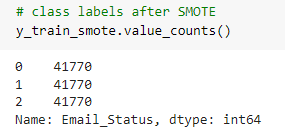
1. **Oversampling using SMOTE:**

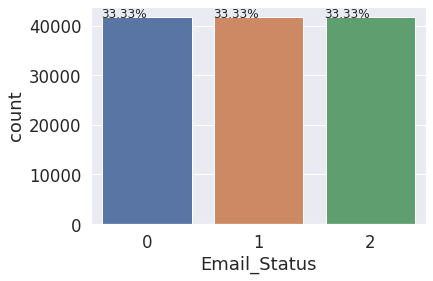
Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset.

Though we do not lose any information, the shortcoming of this method is that the models train on the duplicate data for the minority class. Even though it gives better results than just training with imbalanced data, the better option is to use SMOTE.

Using KNN algorithm, **SMOTE** creates observations that mimic the features and behaviors of the minority classes.

**Distribution of class labels for training data after oversampling:**

****

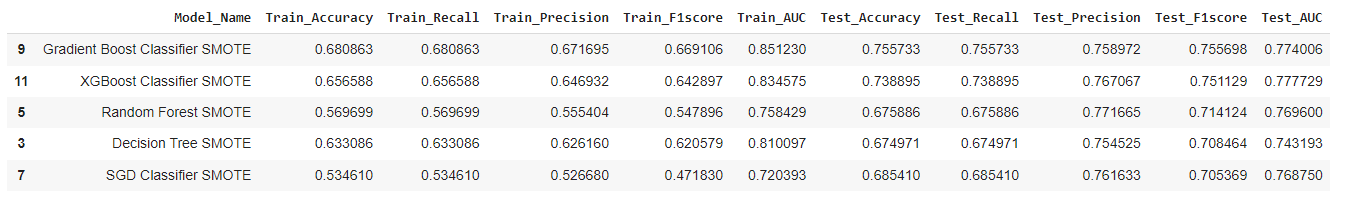
****

* **Modelling:**

We trained **logistic Regression,** **Decision Tree,** **SGD Classifier, Random Forest, Gradient Boost** and **Xgboost Classifier** on the transformed training data.

We trained these models for both undersampled and oversampled data. Ensemble learning methods generally give better performances. That is what we mostly here as well.

Here are the top 5 models that had the best performances –



We can also see that all the top performing models were trained on the data that was oversampled using the **SMOTE** technique.

So, we can definitely say that SMOTE is doing better job compared to undersampling.

* **Feature Importance:**

We checked feature importance for 3 ensemble models (Random Forest, Gradient Boost & XGBoost). The top most important features that we found were –

* **Email Campaign Type**
* **Total Past Communications**
* **Subject Hotness Score**
* **Total Link Images**
* **Word Count**
* **Feature Selection:**

**Time Email Sent Category** & **Email Source Type** were two of the columns that came last in terms of feature importance.

The reasonable explanation can be that for the kind of data that we had, it did not really matter when the emails were sent. This did not have much of an impact on the customers’ responses. So, we dropped these two columns from the original set of features.

* **Hyperparameter Tuning:**

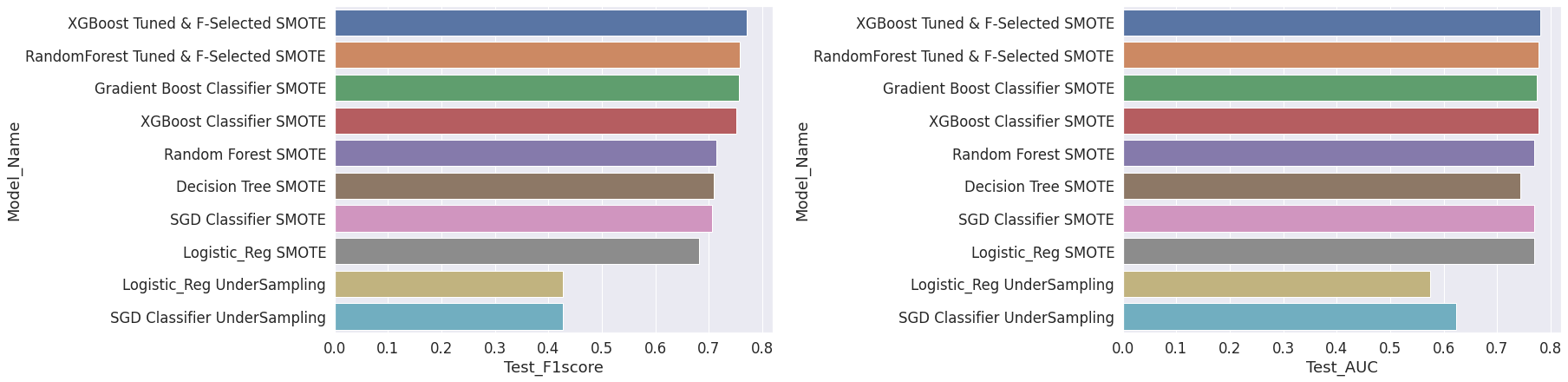
The process of selecting the best hyper-parameters to use is known as hyper-parameter tuning.

To perform this task, we used **Grid** **Search** with 3 fold **cross validation**. We tried out combinations of different parameters in iterations to figure out which ones give the best results. We tuned the hyper parameters for **Random Forest** and **Xgboost** model.

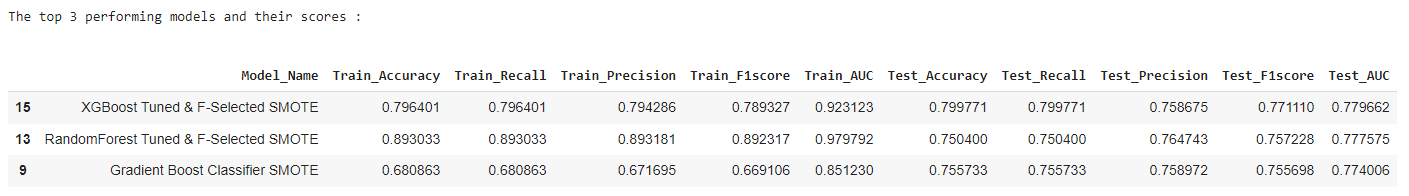
**Evaluation Metric:** Our problem statement clearly tells us to characterize the mail and track the mail that is ignored; read; acknowledged by the reader. So, we need to make sure that false predictions are as low as possible.

For this reason, we chose **F1 Score** and **AUC-ROC** (Area under the Curve-Receiver Operating Characteristics)Score. Both F1 and AUC-Roc score becomes low if our model keeps making wrong predictions. As these scores get closer to 1, we can be confident that our model is being able to distinguish different classes better.

**Model Performance after Hyperparameter Optimization:**



**Top 3 Best Performing Algorithms:**

****

* **Conclusion:**

Email campaign is a very popular and effective way of marketing. It can help converting potential clients, sustaining old clients and most importantly it leaves room for experimentations. Being able to understand a customer's preference and consumer habits can help build a long-term relationship and trust between a brand and the customer. So, it becomes crucial for a business to figure out the Do's and Don'ts to provide a decent experience.

In this project, we tried to find out exactly those factors that influence the effectiveness of marketing through Email campaigns. Here's the summary of what we found -

Observations:

* **Exploratory Data Analysis :** During the EDA, we found-
* The demographic of potential clients does not have an influence on that person ignoring, reading or acknowledging to email campaigns. We noticed that the distributions for the client's response remains more or less similar regardless of where the person lives. For the data that we have, it is wise to not consider **Customer Location** as an influencing factor.
* **Email Campaign Type 1** was sent the least number of times which had the best probability of being read and acknowledged. **Email Campaign Type 2** was sent most and it was ignored the most.
* The time of day when the email was sent does not have much of an impact on the response that was received. We found that during the feature importance evaluation as well.
* There were outliers for all the numeric columns except for **Word count** column.
* **Total communications** made prior is a very important factor to consider if we want to avoid emails being ignored. It was observed that, more the businesses kept in touch with the potential customers, the chances of the next emails being read and acknowledged increased.
* The emails with lower **Subject Hotness Score** were read and acknowledged more.
* Emails with high number of links and images were ignored more.
* Emails that were too lengthy, have a higher chance of being ignored.
* Among all the numeric features, **Total Images** and **Total Links** were highly correlated. So, during the feature engineering, we merged these two columns to create a new column that would have the essence of both these columns.
* **Modelling Approach :**
* To handle the class imbalance problem, we used under sampling and oversampling both.
* We used SMOTE (synthetic Minority Oversampling Technique) for oversampling as it creates synthetic data for the minority classes instead of simply just copying the existing observations.
* We noticed that in every case, Machine Learning models with SMOTE outperformed the under-sampling methods.
* Since we were having some outliers for the minority class, we decided to use more tree based and ensemble learning algorithms to mitigate the impact of outliers as much as possible.
* We checked and compared the feature importance for 3 different ensemble models and discarded the two of least contributing features. These two features were **'Time Email Sent Category**' and **'Email Source Type**'.
* We boosted the model performance for **Random** **Forest** and **XGBoost** model, performing hyper-parameter tuning with grid search and cross validation.
* **Model Selection :**
* Hyper-parameters tuned XGBoost model with the most relevant features selected performs the best. It has the F1 Score of 0.77. The Accuracy, precision and Recall scores are also overall better for this model.
* At 2nd & 3rd, we have Gradient Boost and Random Forest Model with feature selection.

**References-**

1. AlmaBetter
2. Kaggle
3. MachineLearningMastery
4. GeeksforGeeks
5. Analytics Vidhya