Capstone Project 3

Email Campaign Effectiveness Prediction

Individual Project:

Name: Vithika Karan

Email: vithika16k@gmail.com

Content

- Problem Statement
- Email Campaign Effectiveness Prediction
- Data Summary
- Approach
- Exploratory Data Analysis
- Correlation Matrix
- Data Manipulation
- Modeling:
  - Handling Class Imbalance
  - Logistic Regression
  - Decision Trees
  - Random Forest
  - KNN
  - XGBoost Algorithm
- Model Performance and Evaluation Metrics
- Conclusion and Recommendations

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.

Email Campaign Effectiveness Prediction

Email Marketing can be defined as a marketing technique in which businesses stay connected with their customers through emails, making them aware about their new products, updates, important notices related to the products they are using.

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.

But many of times as customers we do not tend to read an email 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.

Data Summary

- **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 email regarding the businesses.
- **Subject Hotness Score** - It is the email's subject 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

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

- **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.

Exploratory Data Analysis

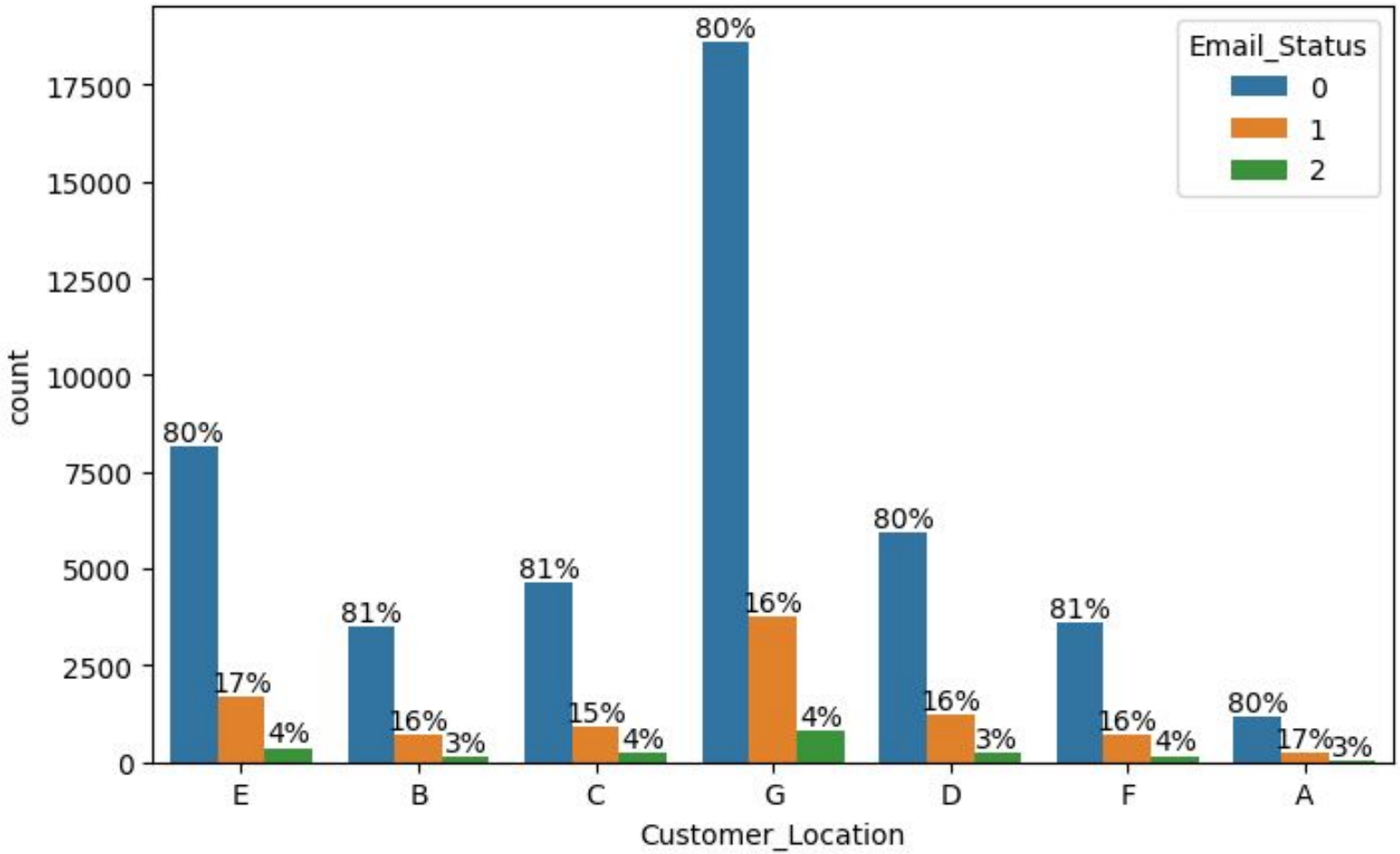
We did a detailed exploratory data analysis of the categorical and continuous variables against their target variable and these are few important insights we got:

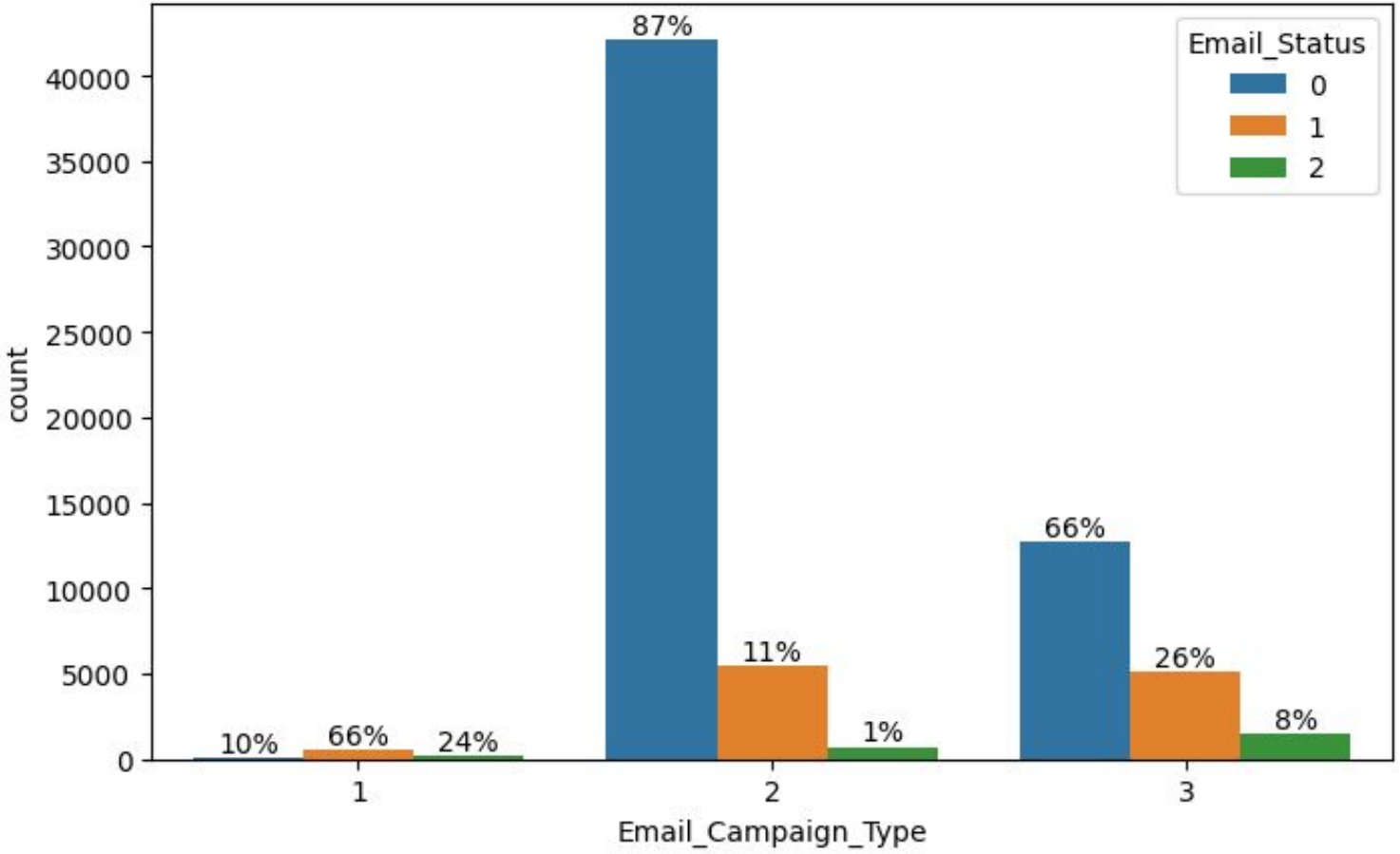
Categorical Data:

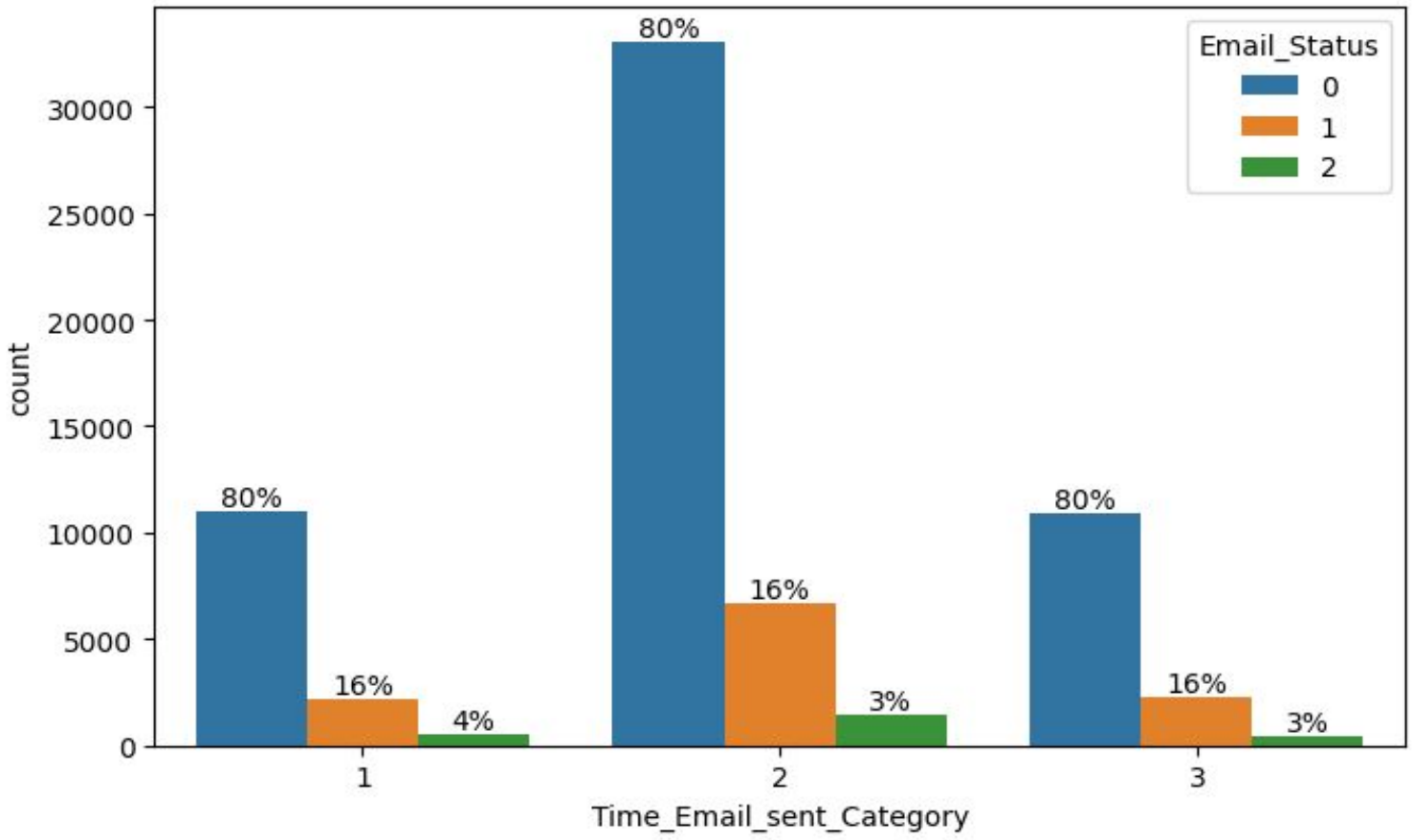
- In the customer location feature we can find that irrespective of the location, the percentage ratio of emails being ignored, read and acknowledge are kind of similar. It does not exclusively influence our target variable. It would be better to not consider location as factor in people ignoring, reading or acknowledging our emails.

- In the Email Campaign Type feature, it seems like in campaign type 1 very few emails were sent but has 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.

- Time email sent category cannot be considered as a relevant factor in classifying the emails. Both the feature importance showed this particular thing. If we consider Time email sent category 2 as middle of the of course they are going to be read and acknowledged more than morning and night.


Email_Status
0
1
2
count
17500
15000
12500
10000
7500
5000
2500
0
80%
17%
4%
81%
16%
3%
81%
15%
4%
80%
16%
4%
80%
16%
3%
81%
16%
4%
80%
17%
3%
E
B
C
G
D
F
A
Customer_Location

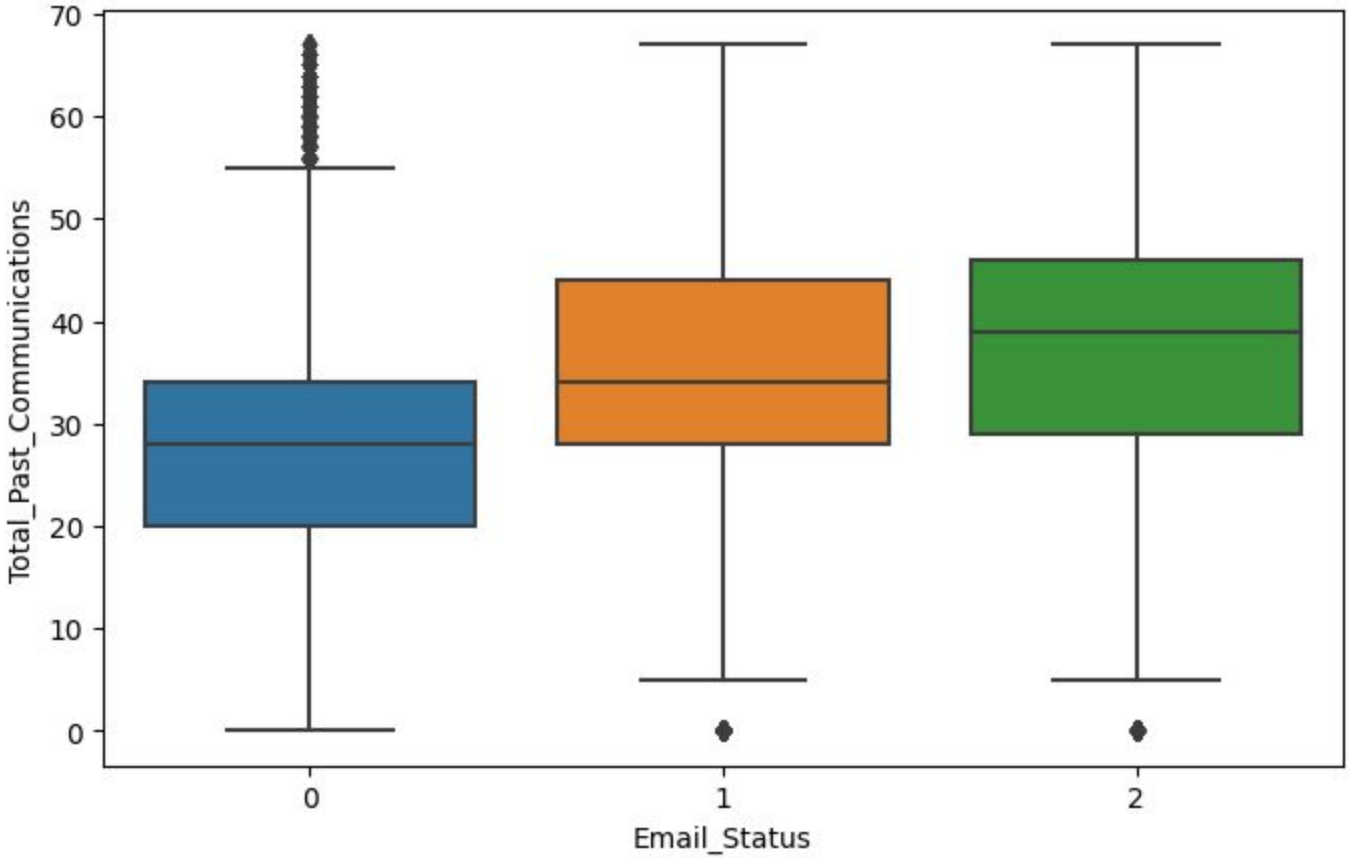

Email_Status
0
1
2
count
40000
35000
30000
25000
20000
15000
10000
5000
0
87%
66%
11%
1%
66%
26%
8%
10%
66%
24%
1
2
3
Email_Campaign_Type

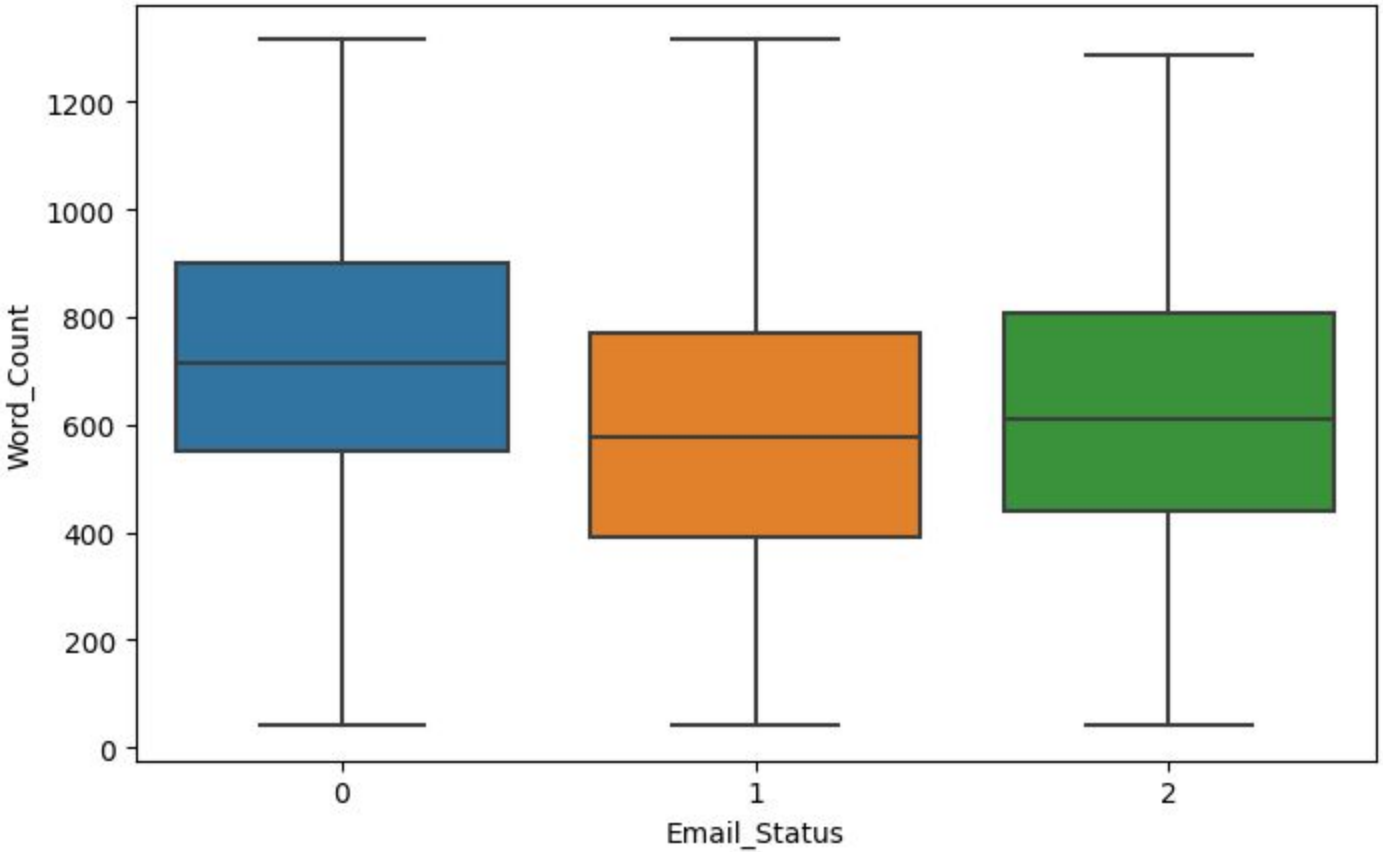

Email_Status
0
1
2
count
30000
25000
20000
15000
10000
5000
0
80%
16%
4%
80%
16%
3%
80%
16%
3%
1
2
3
Time_Email_sent_Category

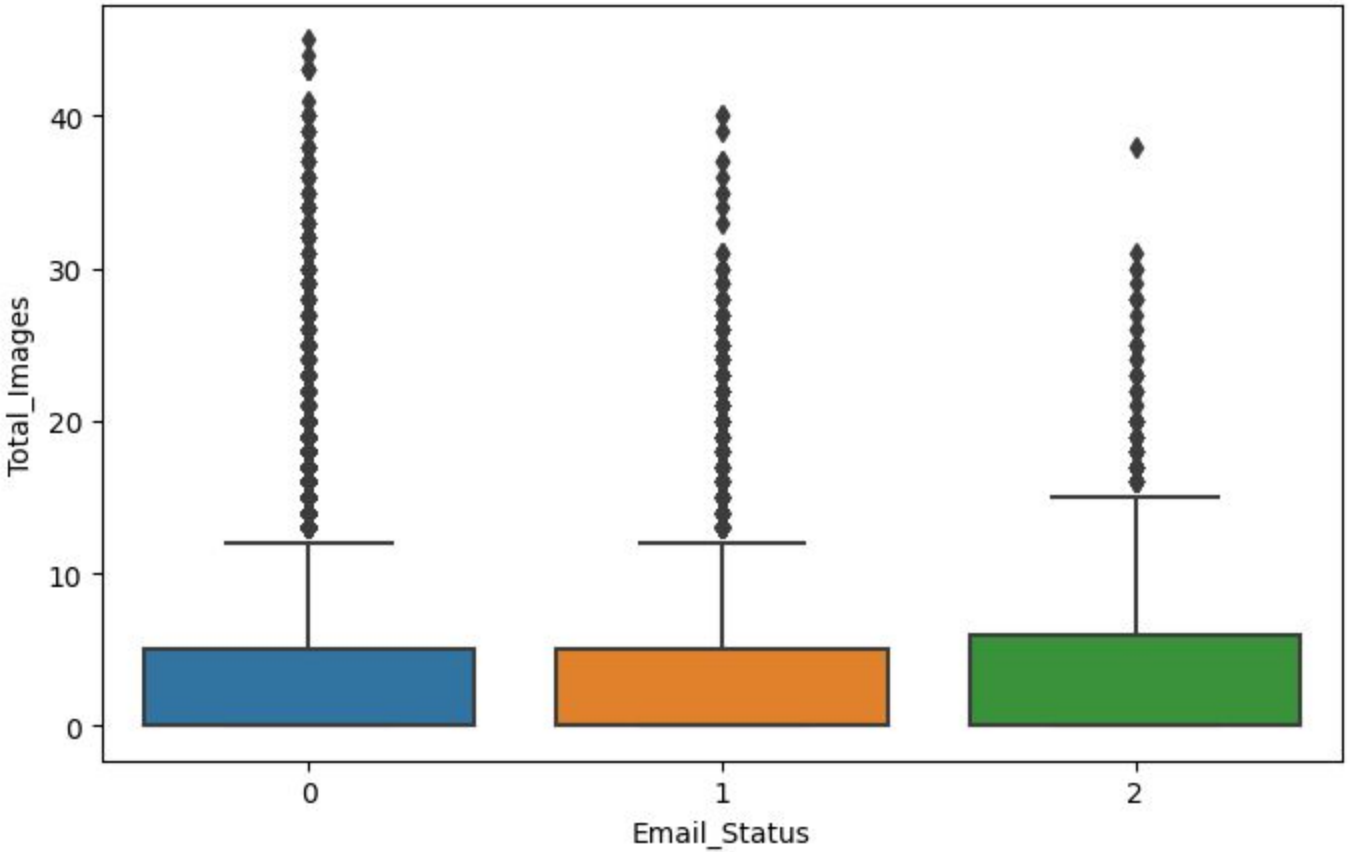
Exploratory Data Analysis

Continuous Data:

- 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 connection with your customers.
- The more the words in an email, the more it has a tendency it has to get ignored. Too lengthy emails are getting ignored.
- More images were there in ignored emails.
- There are outliers in almost every continuous variable except Word Count and upon analyzing, it was found that outliers make up for more than 5% of the minority data and will influence the results either way, so it was better not to get rid of them.


70
60
50
40
30
20
10
0
Total_Past_Communications
0
1
2
Email_Status


1200
1000
800
600
400
200
0
Word_Count
0
1
2
Email_Status


Total_Images
40
30
20
10
0
0
1
2
Email_Status

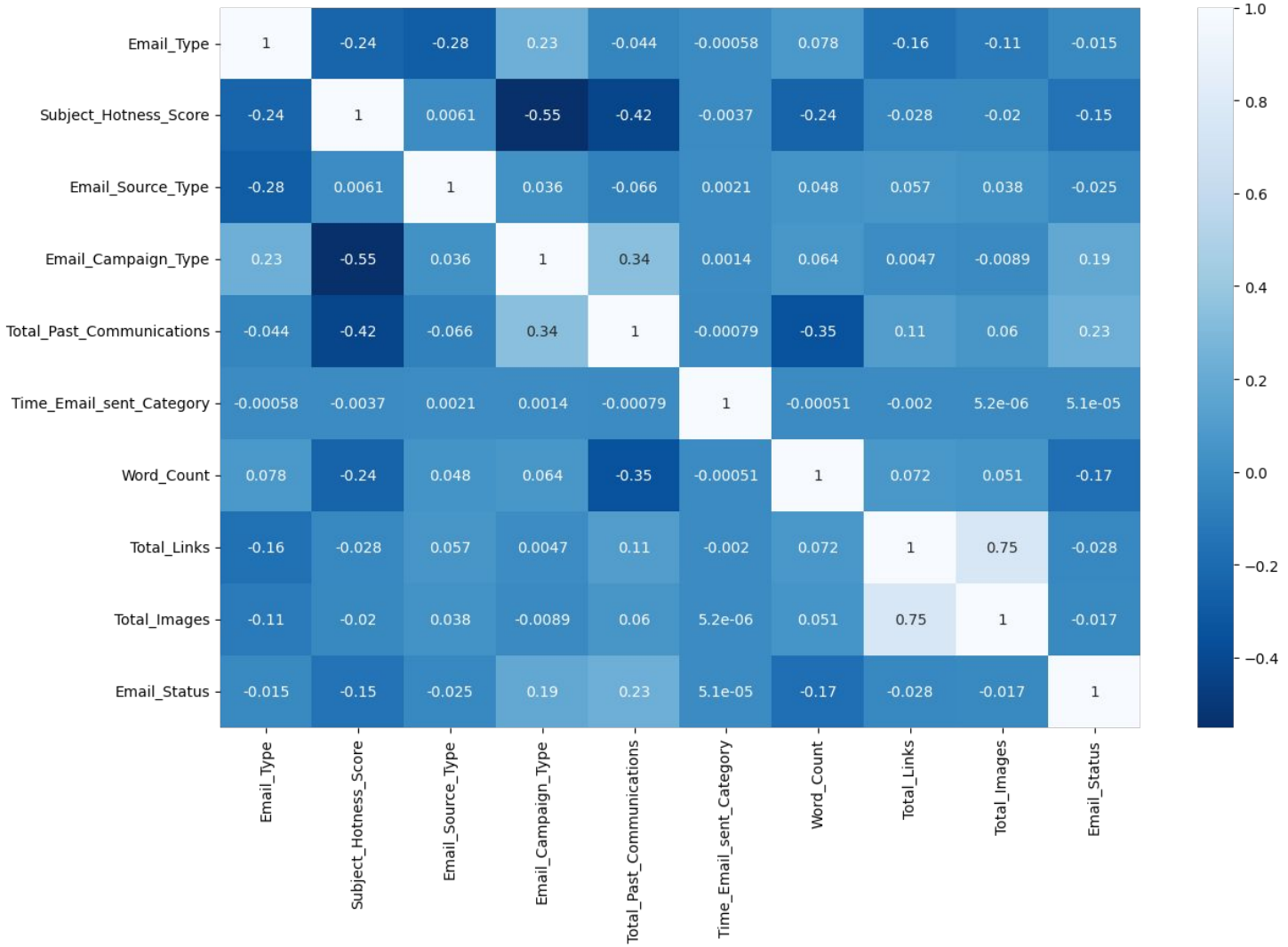
Correlation Matrix

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.

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 negatives amongst other. We can see multicollinearity involved in Email Campaign Type, Total past communication and Total links, Total Images among others and we will have to deal with it.


Email_Type 1 -0.24 -0.28 0.23 -0.044 -0.00058 0.078 -0.16 -0.11 -0.015
Subject_Hotness_Score -0.24 1 0.0061 -0.55 -0.42 -0.0037 -0.24 -0.028 -0.02 -0.15
Email_Source_Type -0.28 0.0061 1 0.036 -0.066 0.0021 0.048 0.057 0.038 -0.025
Email_Campaign_Type 0.23 -0.55 0.036 1 0.34 0.0014 0.064 0.0047 -0.0089 0.19
Total_Past_Communications -0.044 -0.42 -0.066 0.34 1 -0.00079 -0.35 0.11 0.06 0.23
Time_Email_sent_Category -0.00058 -0.0037 0.0021 0.0014 -0.00079 1 -0.00051 -0.002 5.2e-06 5.1e-05
Word_Count 0.078 -0.24 0.048 0.064 -0.35 -0.00051 1 0.072 0.051 -0.17
Total_Links -0.16 -0.028 0.057 0.0047 0.11 -0.002 0.072 1 0.75 -0.028
Total_Images -0.11 -0.02 0.038 -0.0089 0.06 5.2e-06 0.051 0.75 1 -0.017
Email_Status -0.015 -0.15 -0.025 0.19 0.23 5.1e-05 -0.17 -0.028 -0.017 1
Email_Type
Subject_Hotness_Score
Email_Source_Type
Email_Campaign_Type
Total_Past_Communications
Time_Email_sent_Category
Word_Count
Total_Links
Total_Images
Email_Status
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Data Manipulation

Multicollinearity: Multicollinearity occurs when two or more independent continuous features in the dataset are highly correlated and can help predict each other and the dependent variable. This makes it difficult to individually analyse the effect of these individual independent variables on the target or dependent variable.
We can quantify multicollinearity using Variance Inflation Factors (VIF).

VIF = 1/(1-R^2) 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 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

	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

Data Manipulation

Outlier Detection:

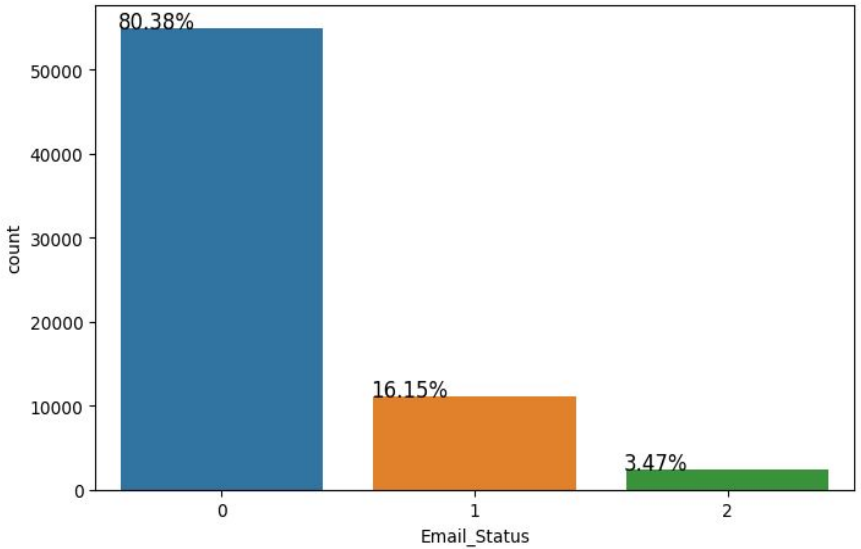
We calculated the number of outliers with respect to the individual classes of our target variable. The minority class in our target variable is 1 and 2. But upon calculating we found that the percentage of outliers in minority classes is 5.256486728303012. It is more than 5% information of our minority class and hence we decided against deleting them because the model will not be able to classify our minority classes correctly having lack of information. They are going to affect the models either way.

We did the feature scaling and one hot encoding of other categorical variables as well before feeding the data to different classification models.

Modeling:

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. We can handle it with Oversampling with SMOTE and Random Undersampling.



We did the train test split before applying any resampling technique so that the test set remains unknown to the models.
We resampled only the train set first by undersampling and then by SMOTE.

Before balancing, we made sure the train split has class distribution as same as the main dataset by using stratify while splitting.

Our 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.

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 predict probability distribution to a multinomial probability distribution to natively support multi-class classification problems.

After evaluating, we got these results. As our test set was highly imbalanced, we can not trust accuracy with this but seeing F1 score and AUC ROC we can see the results weren't thrilling.

	Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
0	LogisticReg RUS	0.541974	0.541974	0.532582	0.519821	0.722834	0.622120	0.622120	0.773710	0.679839	0.762724
1	LogisticReg SMOTE	0.535132	0.535132	0.519679	0.509052	0.720826	0.625192	0.625192	0.771306	0.681115	0.765244

Decision Tree Model:

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. Decision tree uses 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	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
2	Decision Tree RUS	0.999122	0.999122	0.999123	0.999122	0.999999	0.484968	0.484968	0.741790	0.566684	0.601776
3	Decision Tree SMOTE	0.999416	0.999416	0.999417	0.999416	1.000000	0.698925	0.698925	0.729655	0.713251	0.605552

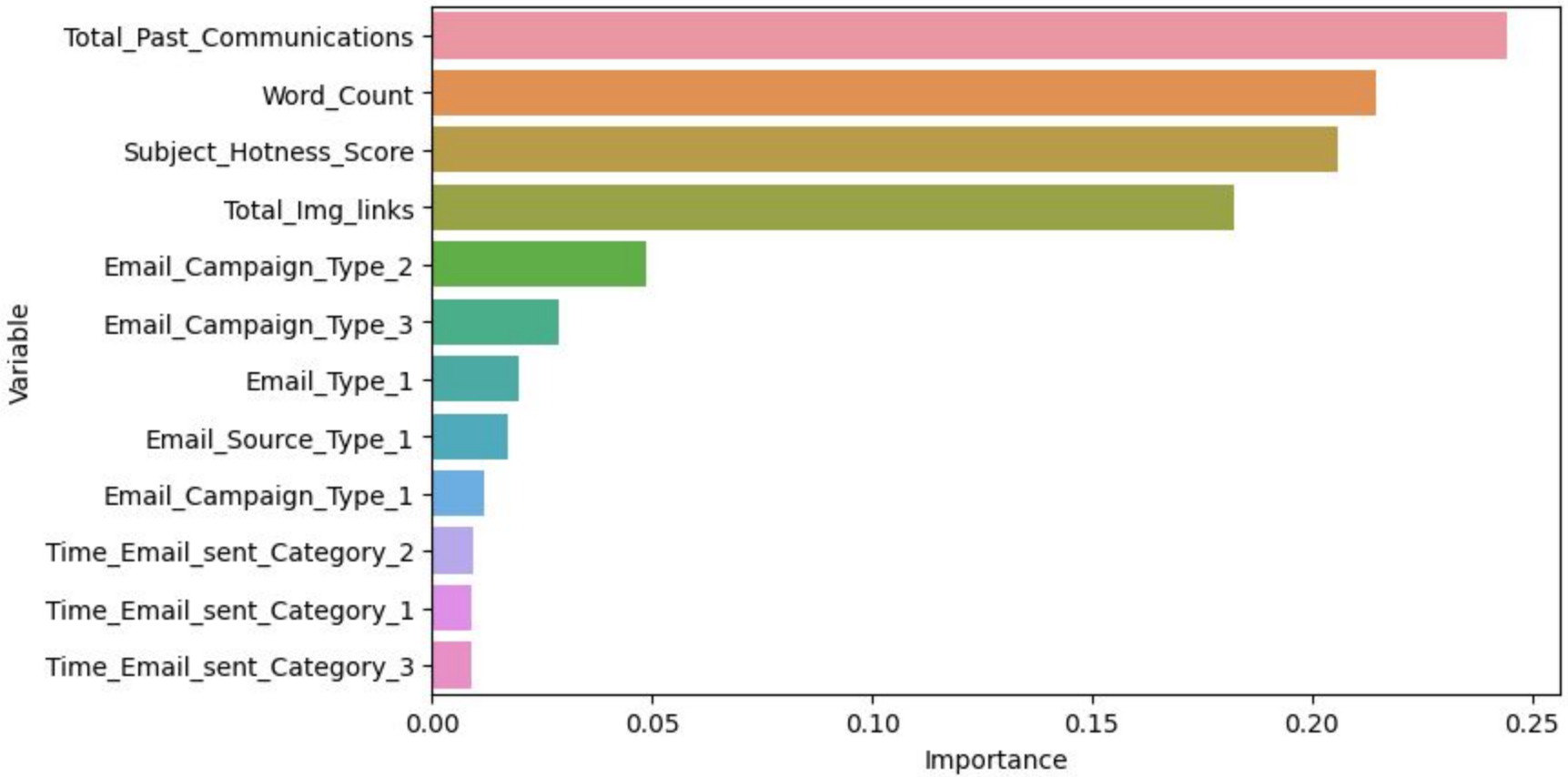
Random Forest:

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, we built random forest model. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. The ensemble models with only one tree will overfit to data as well because it is the same as a single decision tree. When we add trees to the Random Forest then the tendency to overfitting decreases as it combines the results of other trees as well.

We got better results with SMOTE and decided to get a hyperparameter tuned model as well and then tuned SMOTE version gave the best results till now with good F1 score and AUC ROC as well and hence we decided to get feature importance to get an understanding of how the model works.

	Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
4	Random Forest RUS	0.565332	0.565332	0.559548	0.542742	0.753785	0.628045	0.628045	0.775423	0.684171	0.763473
5	Random Forest SMOTE	0.558005	0.558005	0.539292	0.529095	0.757003	0.678297	0.678297	0.770111	0.714232	0.764023
6	RandomF Tuned RUS	0.744819	0.744819	0.748197	0.743932	0.913410	0.614805	0.614805	0.779785	0.676210	0.757659
7	RandomF Tuned SMOTE	0.906157	0.906157	0.906016	0.905595	0.983403	0.740546	0.740546	0.759502	0.749551	0.756819


Total_Past_Communications
Word_Count
Subject_Hotness_Score
Total_Img_links
Email_Campaign_Type_2
Email_Campaign_Type_3
Email_Type_1
Email_Source_Type_1
Email_Campaign_Type_1
Time_Email_sent_Category_2
Time_Email_sent_Category_1
Time_Email_sent_Category_3
Variable
0.00
0.05
0.10
0.15
0.20
0.25
Importance

KNN Classification Model:

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 classifier 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	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
8	KNN RUS	0.651914	0.651914	0.654008	0.650015	0.841923	0.585839	0.585839	0.759384	0.648853	0.696987
9	KNN SMOTE	0.881419	0.881419	0.889769	0.878804	0.983693	0.595128	0.595128	0.750594	0.652169	0.673223
10	KNN Tuned RUS	0.624341	0.624341	0.623436	0.623237	0.818120	0.578012	0.578012	0.769583	0.645603	0.712057
11	KNN Tuned SMOTE	0.880205	0.880205	0.883965	0.878613	0.979051	0.624314	0.624314	0.752105	0.674542	0.679967

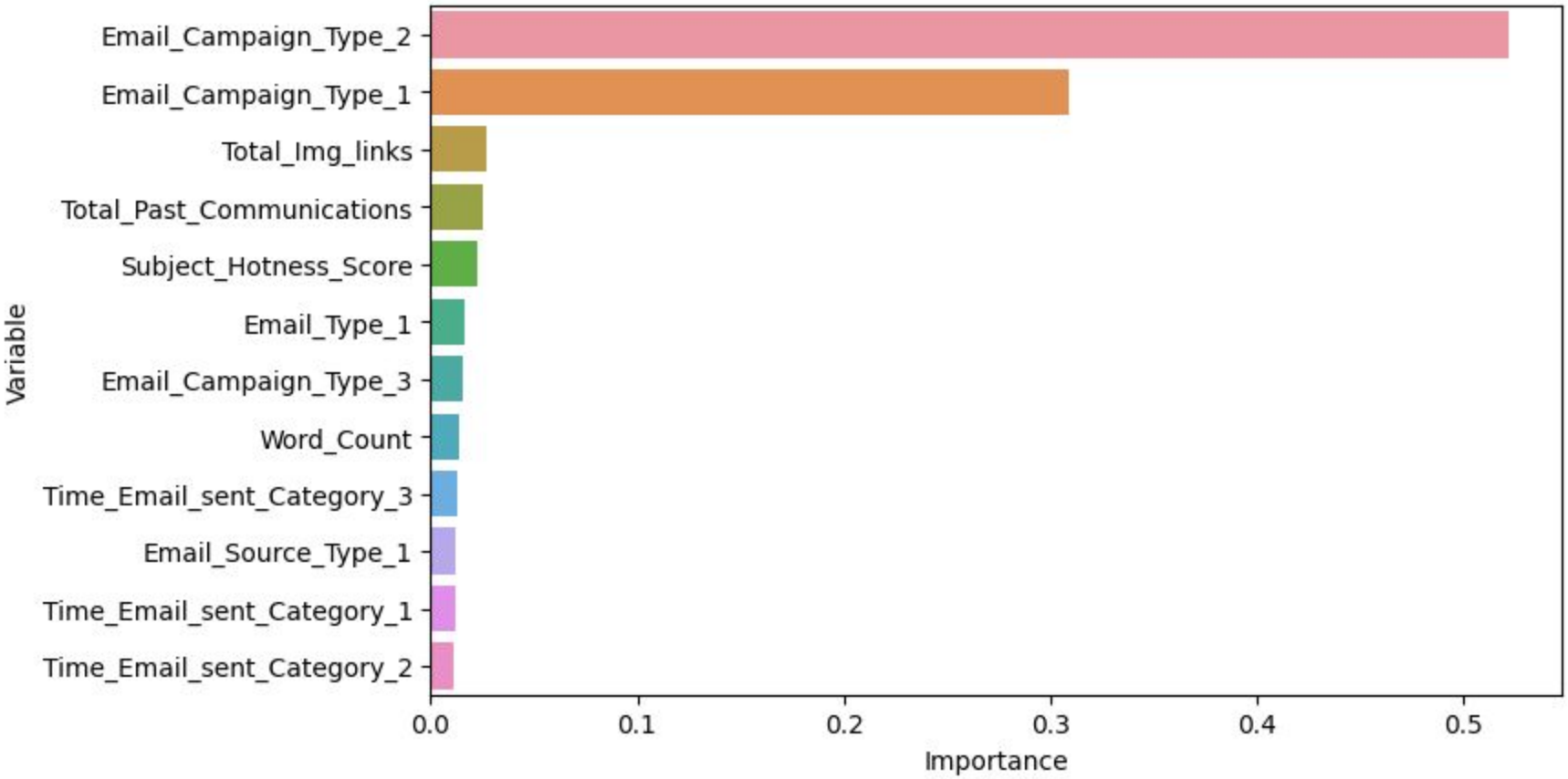
XGBoost Model:

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	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
12	XGB RUS	0.986653	0.986653	0.986772	0.986649	0.999520	0.567698	0.567698	0.771141	0.639649	0.731730
13	XGB SMOTE	0.910162	0.910162	0.913100	0.908415	0.983590	0.789189	0.789189	0.747196	0.762021	0.764928


Email_Campaign_Type_2
Email_Campaign_Type_1
Total_Img_links
Total_Past_Communications
Subject_Hotness_Score
Email_Type_1
Email_Campaign_Type_3
Word_Count
Time_Email_sent_Category_3
Email_Source_Type_1
Time_Email_sent_Category_1
Time_Email_sent_Category_2
Variable
0.0
0.1
0.2
0.3
0.4
0.5
Importance

Model Performance and Evaluation Metrics

Challenges:

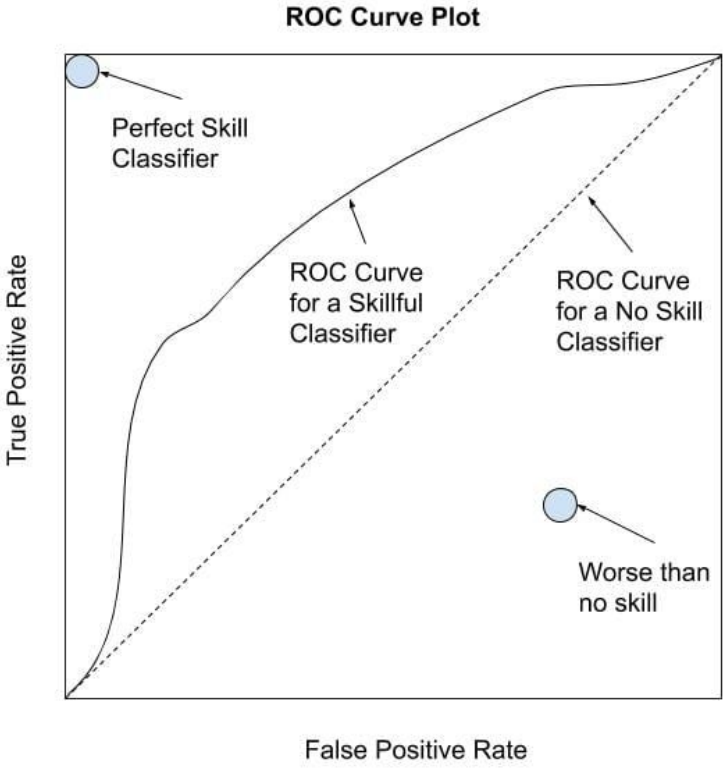
- We had a highly imbalanced target variable with 80% data of the majority class, 16% and 4% data of the minority classes respectively.
- The second challenge we faced was of outliers. Almost all the continuous variables had a good number of outliers. Upon calculating we came to know of the fact that 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.

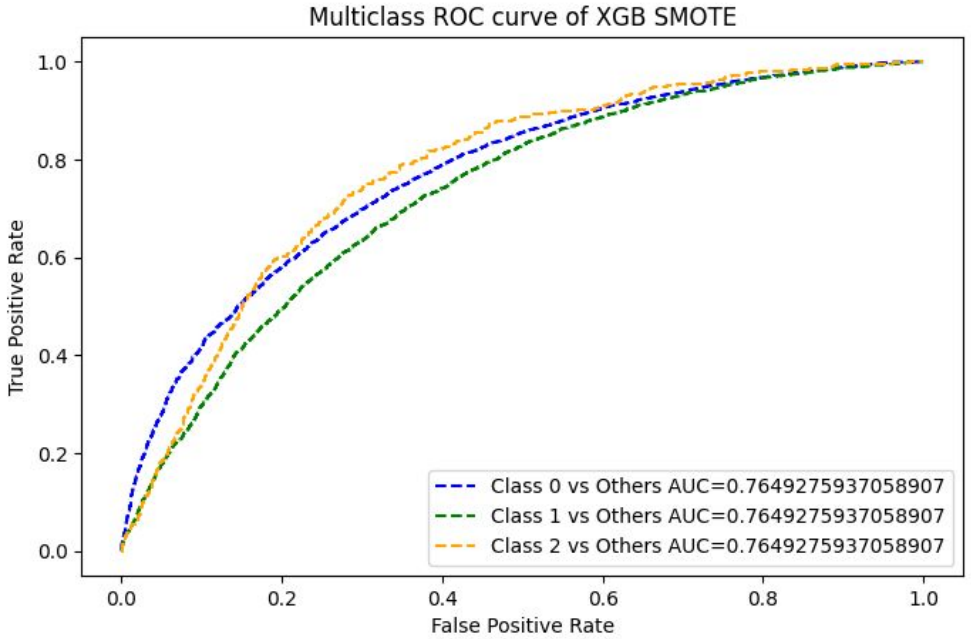
Model Performance and Evaluation Metrics

Evaluation Metrics:

- 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 prioritise in terms of classification, we just want to correctly classify as much as possible. On top of that our data is highly imbalanced, which we tried to encounter in the ways possible.
- F1 Score - It's actually the harmonic mean of Precision and Recall. It is maximum when Precision is equal to Recall. 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!
- 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 more good it becomes in distinguishing them.

Model Performance and Evaluation Metrics





Reference: Machine Learning Mastery

Conclusion and Recommendations:

Upon having this discussion, it's clear that XGB SMOTE did the best classification, followed by Random Forest tuned SMOTE model. Here's an image of our top 5 sorted models.

```
#sorting values
comparison_df.sort_values(by=["Test_F1score",'Test_AUC'], ascending=(False,False), inplace = True, ignore_index = True)
comparison_df
```

	Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
0	XGB SMOTE	0.910162	0.910162	0.913100	0.908415	0.983590	0.789189	0.789189	0.747196	0.762021	0.764928
1	RandomF Tuned SMOTE	0.906157	0.906157	0.906016	0.905595	0.983403	0.740546	0.740546	0.759502	0.749551	0.756819
2	Random Forest SMOTE	0.558005	0.558005	0.539292	0.529095	0.757003	0.678297	0.678297	0.770111	0.714232	0.764023
3	Decision Tree SMOTE	0.999416	0.999416	0.999417	0.999416	1.000000	0.698925	0.698925	0.729655	0.713251	0.605552
4	Random Forest RUS	0.565332	0.565332	0.559548	0.542742	0.753785	0.628045	0.628045	0.775423	0.684171	0.763473

- 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