Capstone Project-3

EMAIL CAMPAIGN EFFECTIVENESS PREDICTION

Individual Project

VISHAL SAHU

Contents

- Defining the problem statement
- Data summary
- Exploratory Data Analysis
- Feature Engineering
- Model Implementation and Evaluation
- Conclusions
- Challenges

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.

Business Context

Email marketing is the act of sending a commercial message, typically to a group of people, using email. In its broadest sense, every email sent to a potential or current customer could be considered email marketing. It involves using email to send advertisements, request business, or solicit sales or donations.

Email marketing strategies commonly seek to achieve one or more of three primary objectives, to build loyalty, trust, or brand awareness. The term usually refers to sending email messages with the purpose of enhancing a merchant's relationship with current or previous customers, encouraging customer loyalty and repeat business, acquiring new customers or convincing current customers to purchase something immediately, and sharing third-party ads.

Data Description

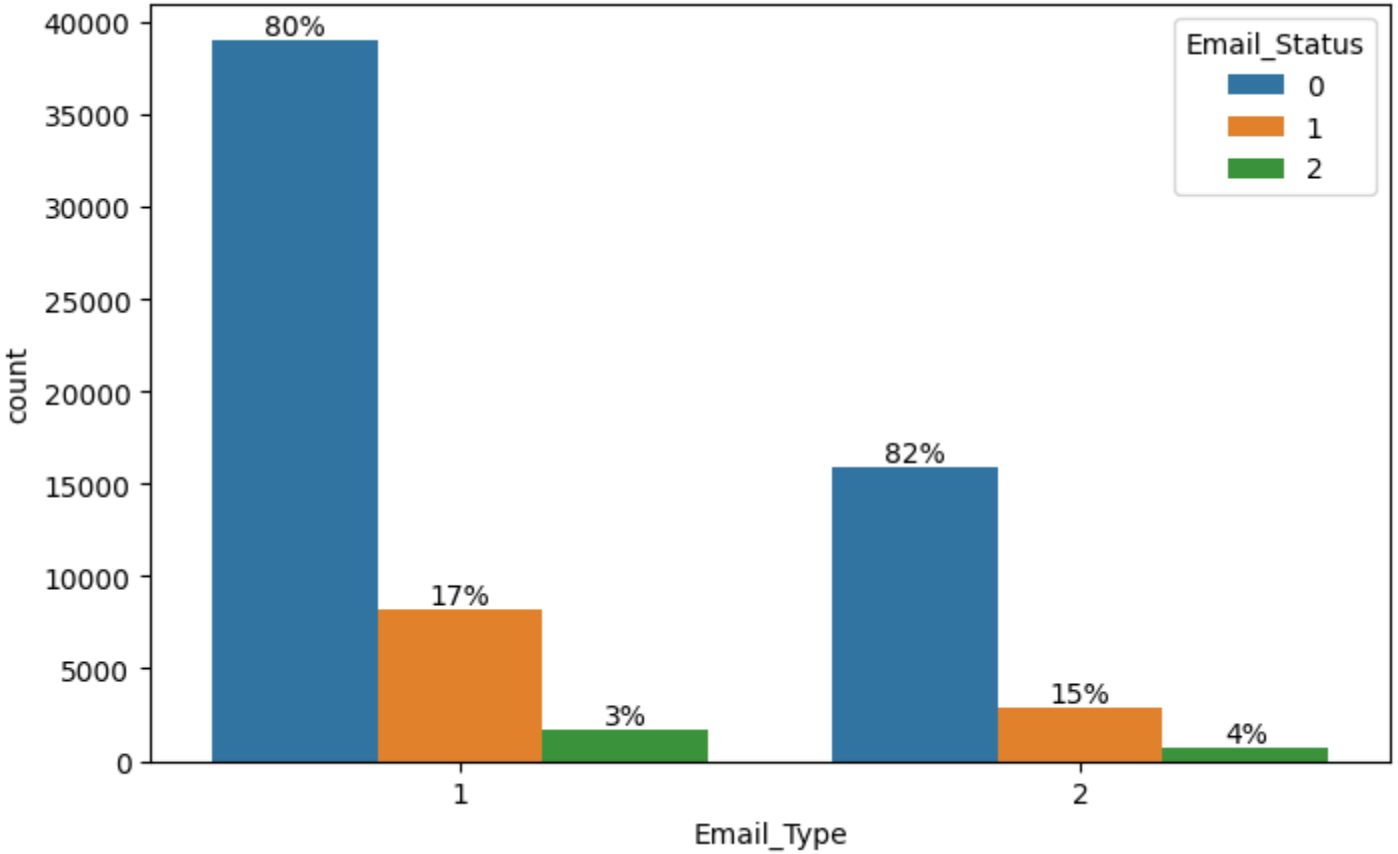
- **Email Id** - It contains the email id's of the customers/individuals
- **Email Type** - There are two categories 1 and 2. We can think of them as marketing emails or important updates, notices like emails regarding the business.
- **Subject Hotness Score** - It is the email's subject's score on the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

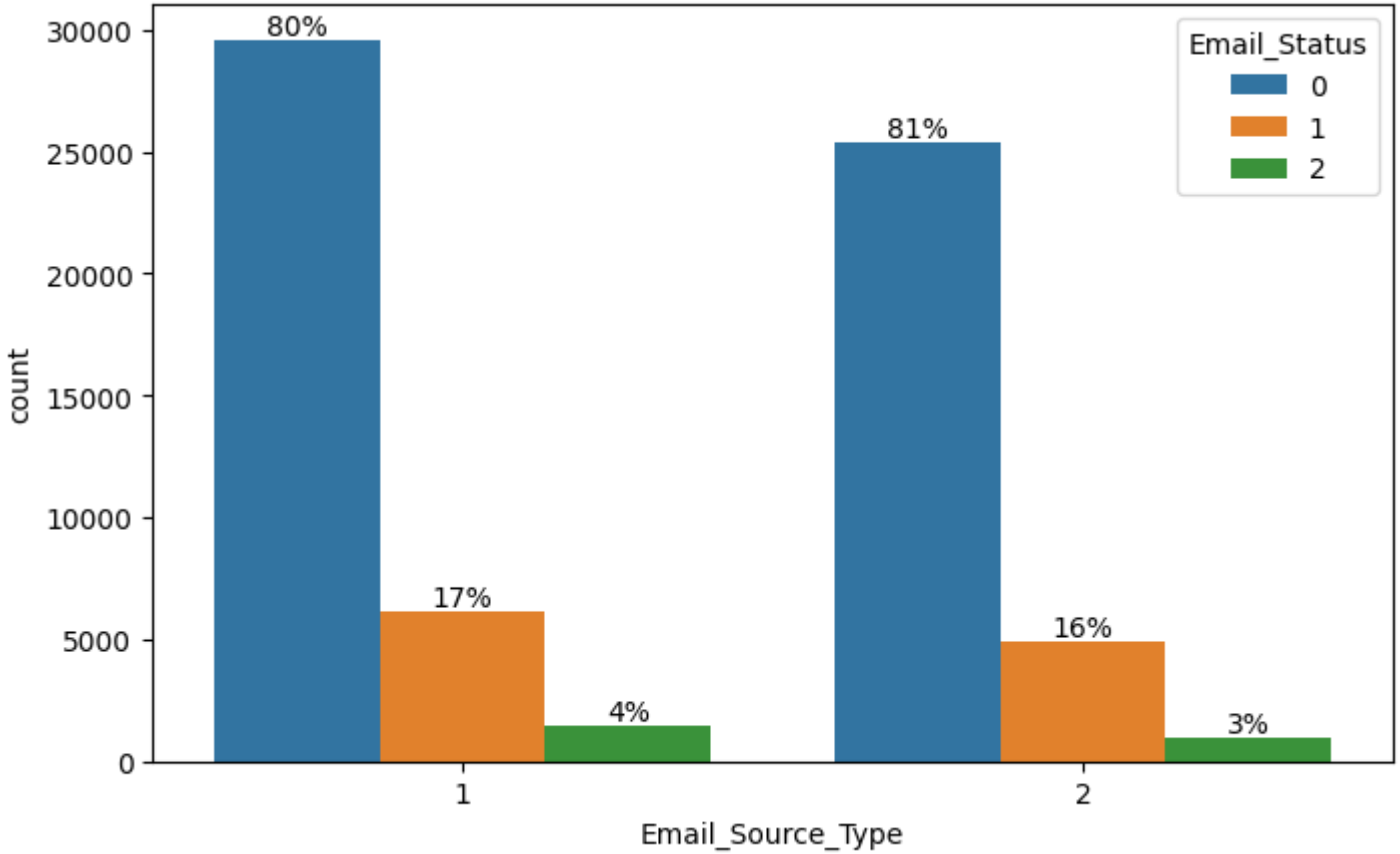
- **Time Email sent Category** - It has three categories 1,2 and 3; the time of the day when the email was sent, we can think of it as morning, evening and night time slots.
- **Word Count** - The number of words contained in the email.
- **Total links** - Number of links in the email.
- **Total Images** - Number of images in the email.
- **Email Status** - Our target variable which contains whether the mail was ignored, read, acknowledged by the reader.

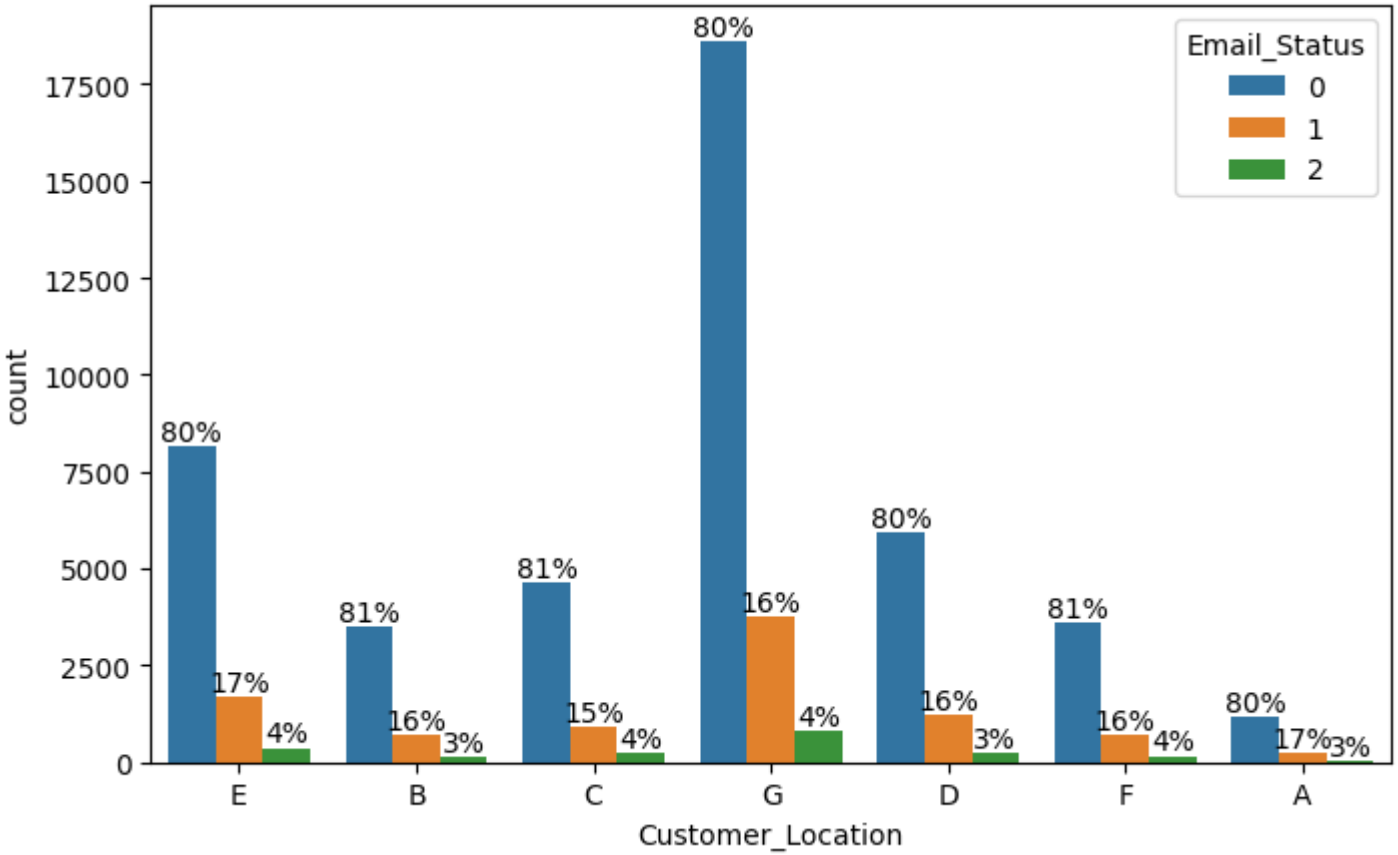
Exploratory Data Analysis

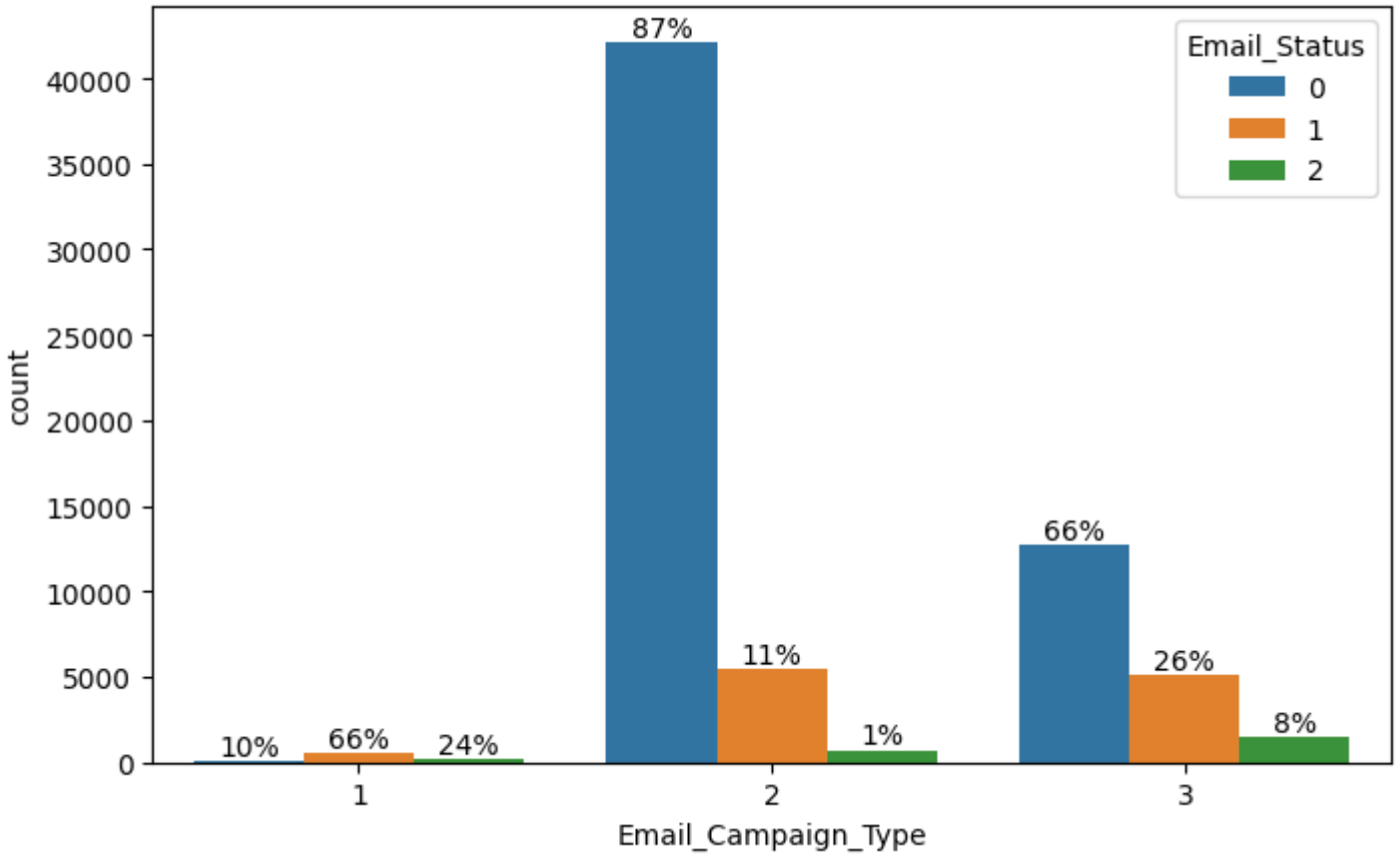
- A detailed exploratory data analysis of the
- categorical and continuous variables against
- their target variable has been done and some
- important insights has also been drawn.

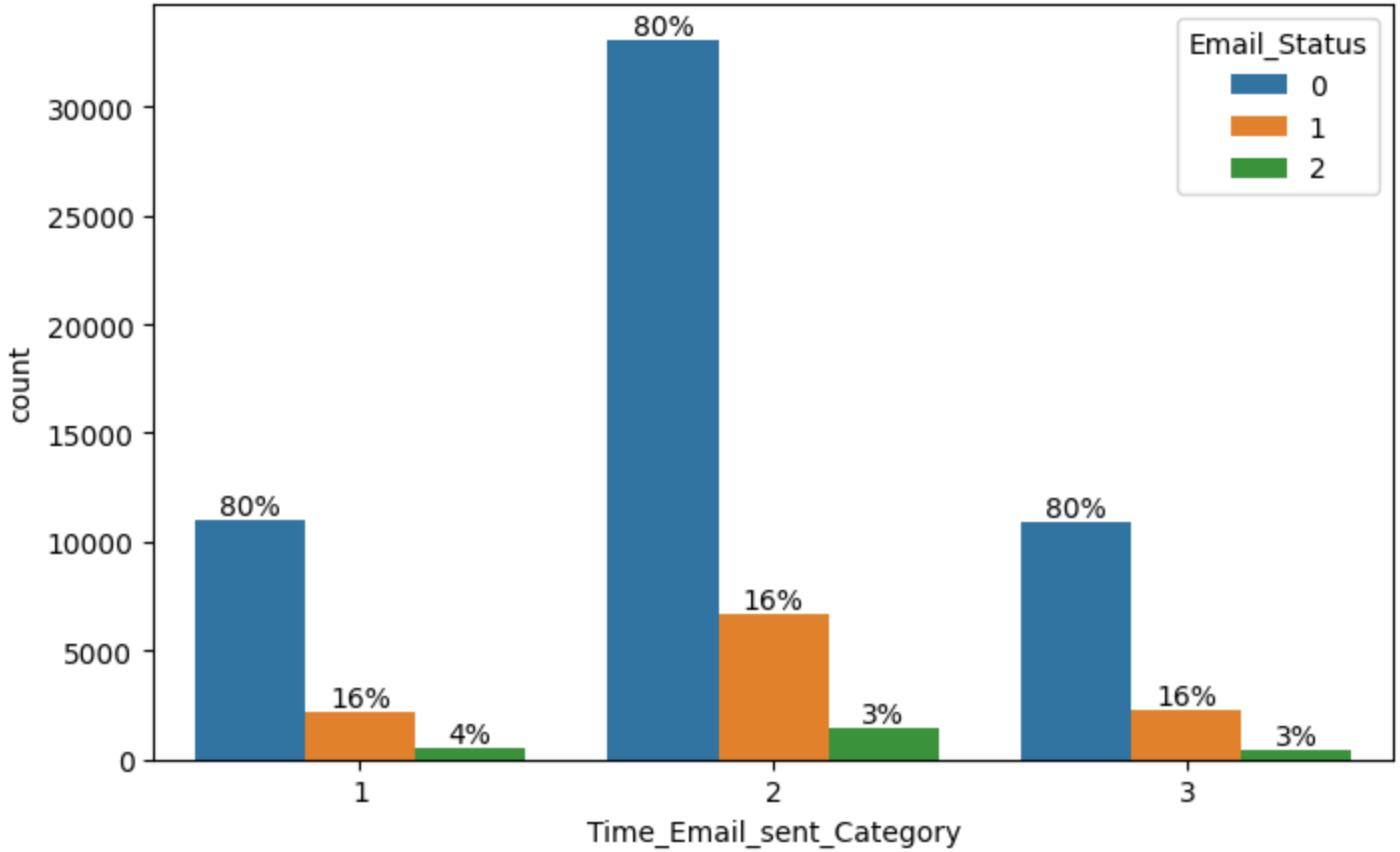
Categorical Data




30000
25000
20000
15000
10000
5000
0
count
80%
17%
4%
81%
16%
3%
1
2
Email_Source_Type
Email_Status
0
1
2

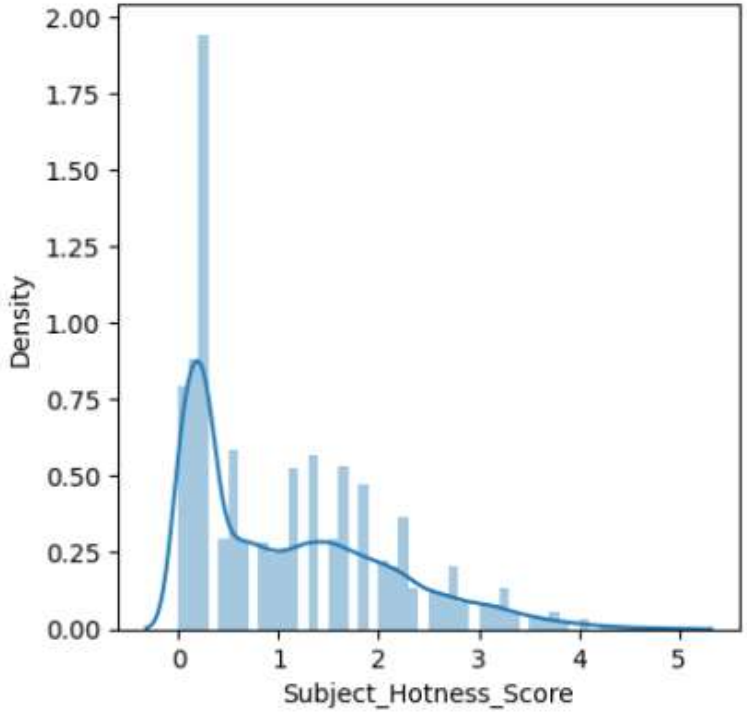

Email_Status
0
1
2
count
17500
15000
12500
10000
7500
5000
2500
0
80%
17%
4%
81%
16%
3%
81%
15%
4%
80%
16%
4%
80%
16%
3%
81%
16%
4%
80%
17%
3%
E
B
C
G
D
F
A
Customer_Location

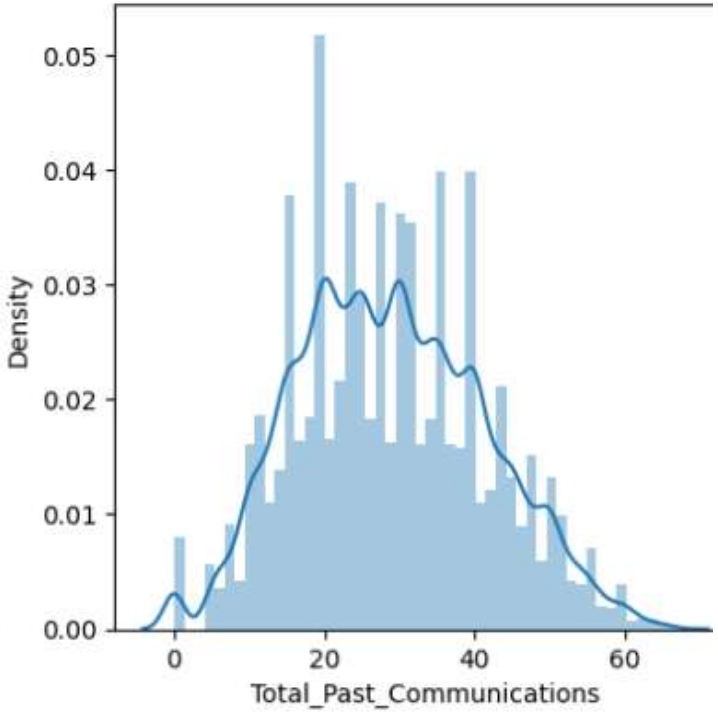

Email_Status
0
1
2
count
40000
35000
30000
25000
20000
15000
10000
5000
0
87%
66%
11%
1%
66%
26%
8%
10%
66%
24%
1
2
3
Email_Campaign_Type

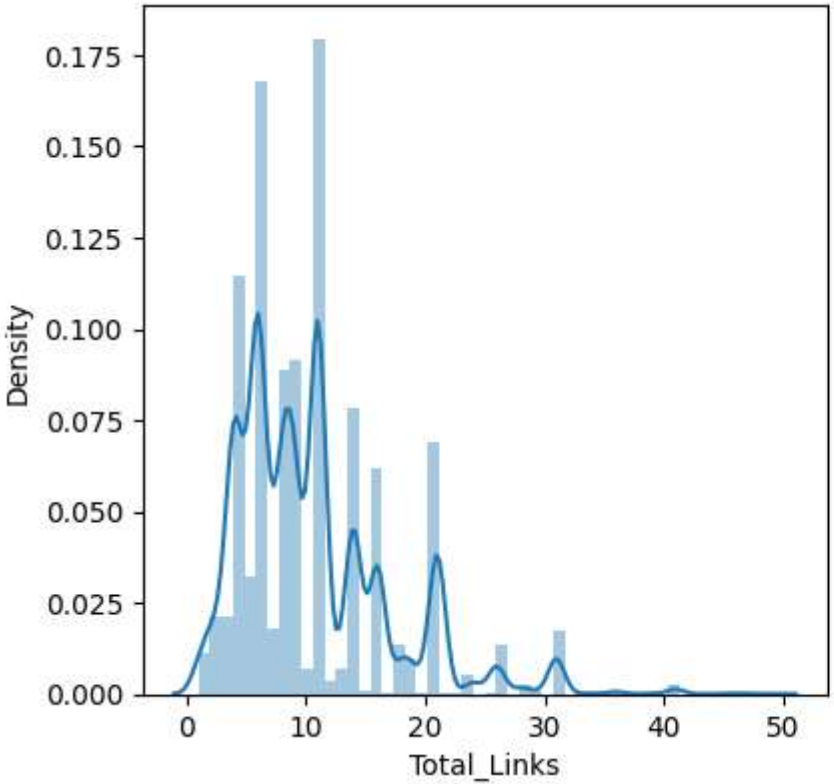

Email_Status
0
1
2
count
30000
25000
20000
15000
10000
5000
0
80%
16%
4%
80%
16%
3%
80%
16%
3%
1
2
3
Time_Email_sent_Category

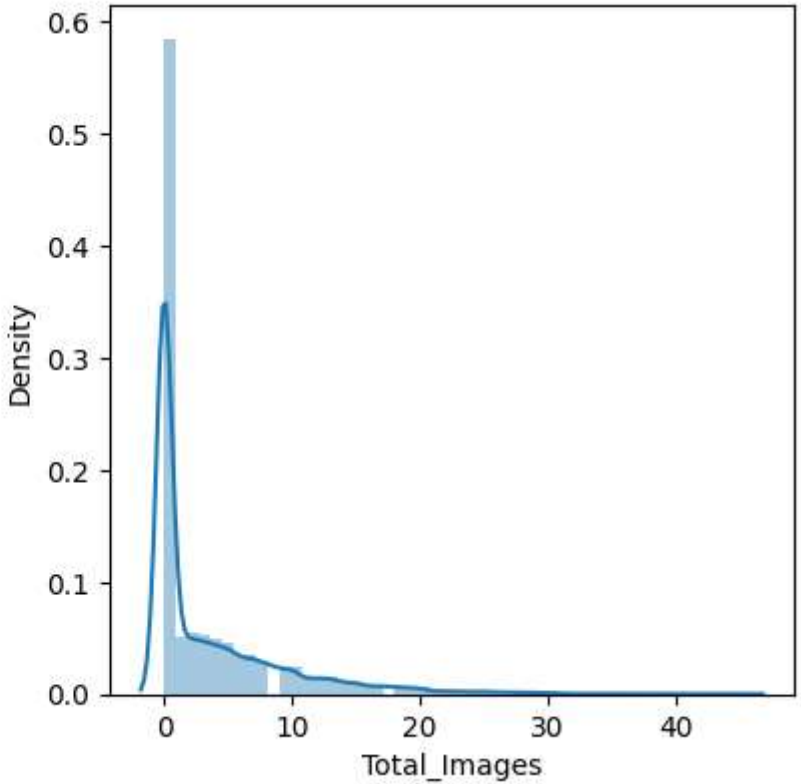
Continuous Data

Univariate

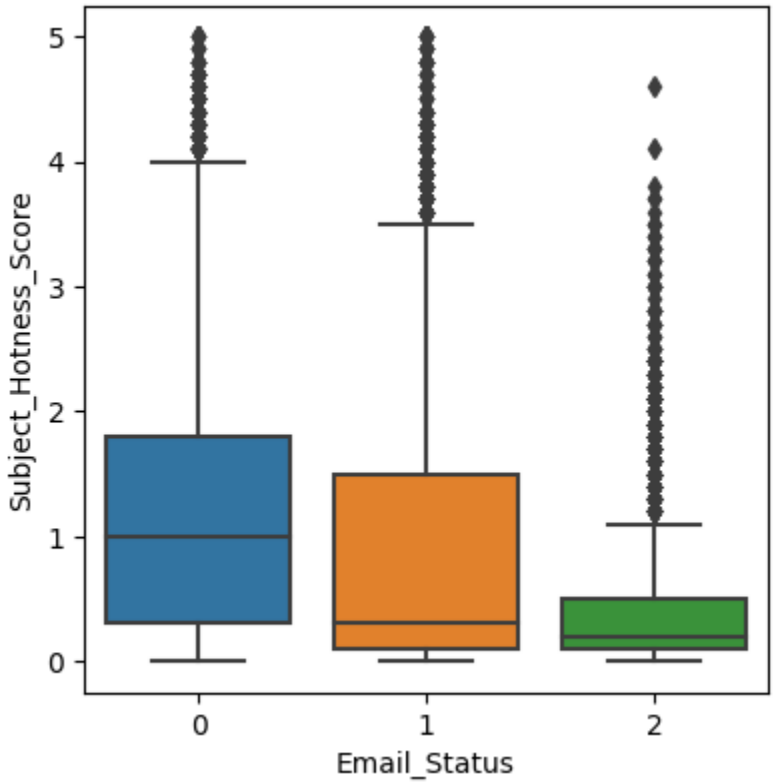


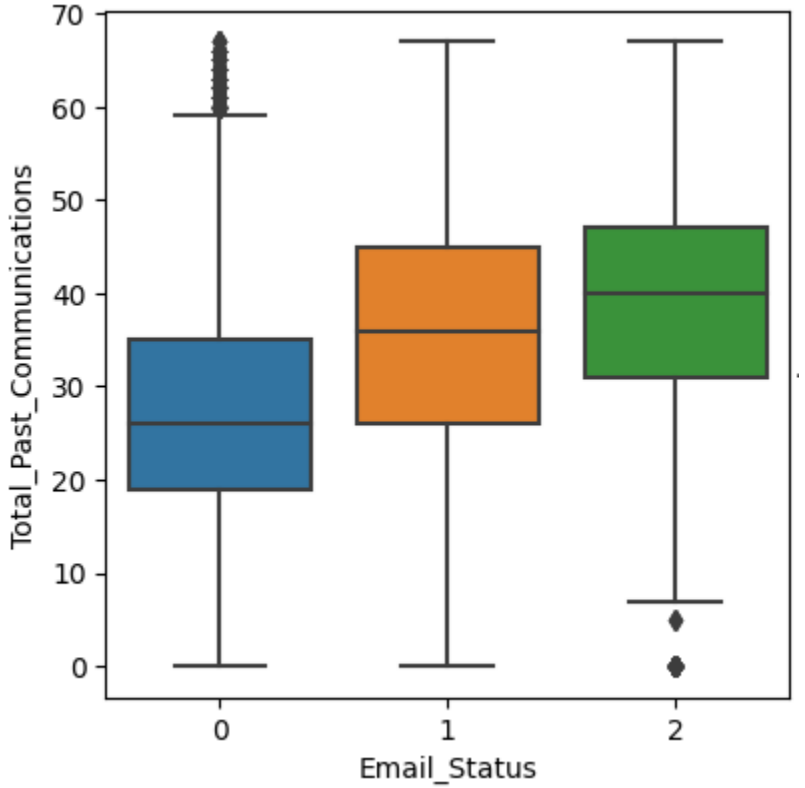


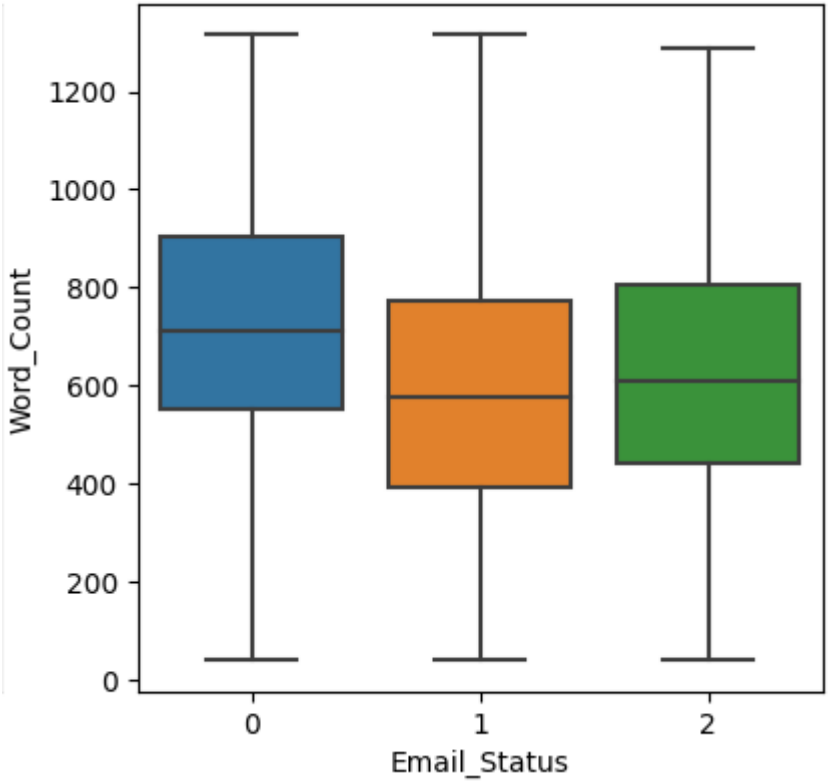

0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Density
0
10
20
30
40
50
Total_Links

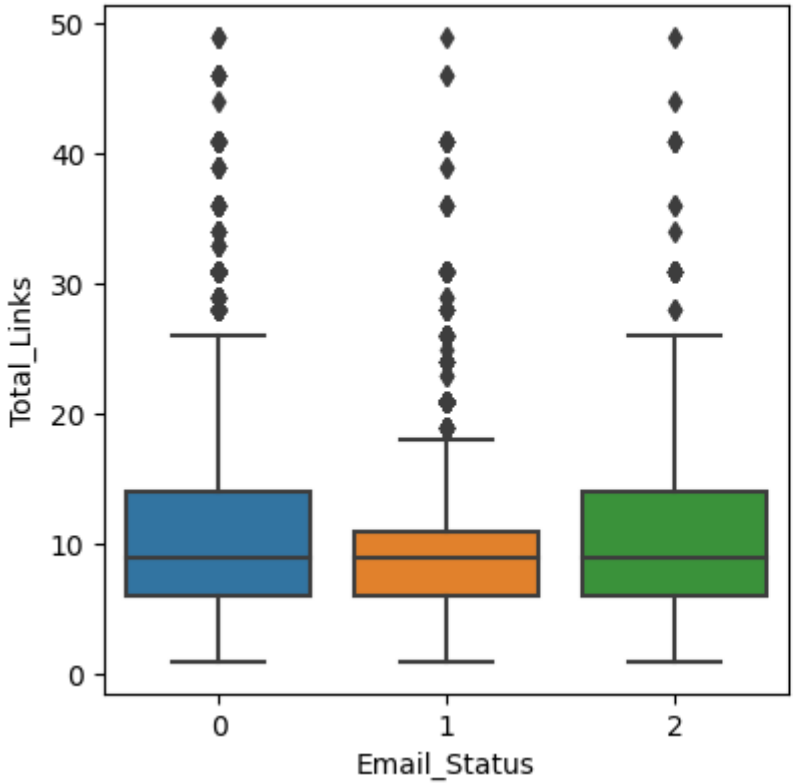

0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
0
10
20
30
40
Total_Images

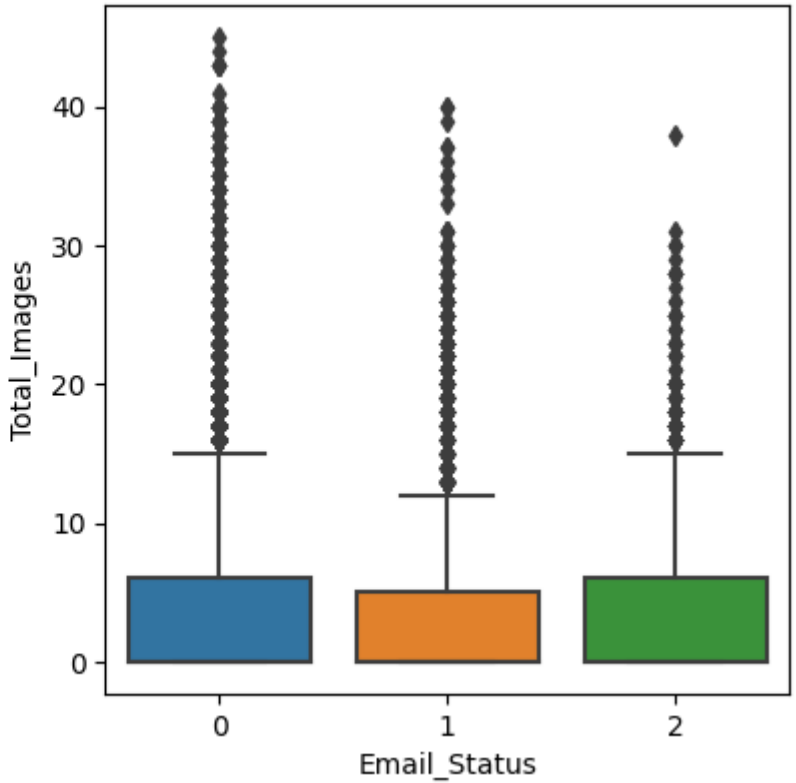
Bivariate


Subject_Hotness_Score
0
1
2
3
4
5
0
1
2
Email_Status

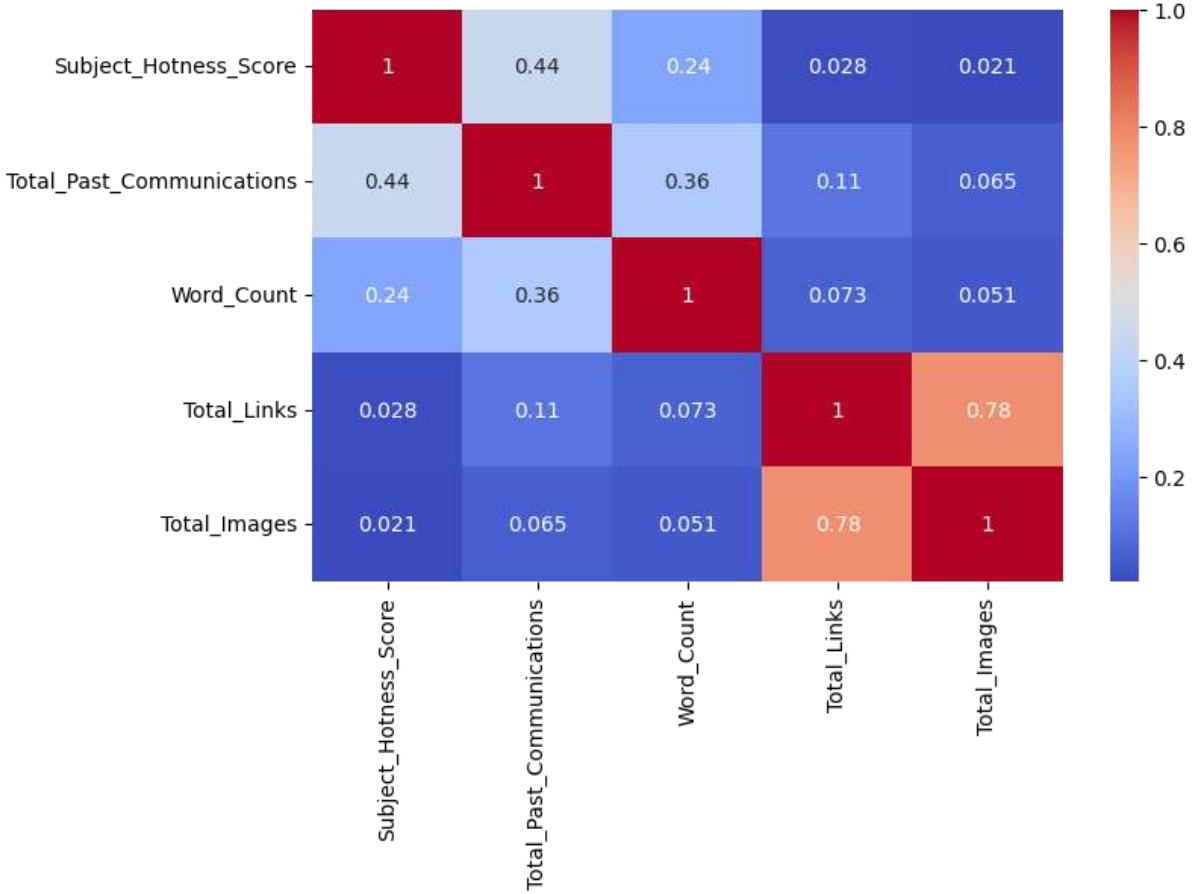

Total_Past_Communications
0
10
20
30
40
50
60
70
0
1
2
Email_Status


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


Total_Links
50
40
30
20
10
0
0
1
2
Email_Status


Total_Images
40
30
20
10
0
0
1
2
Email_Status

Correlation



Feature Engineering

Multicollinearity: It 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 analyze 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

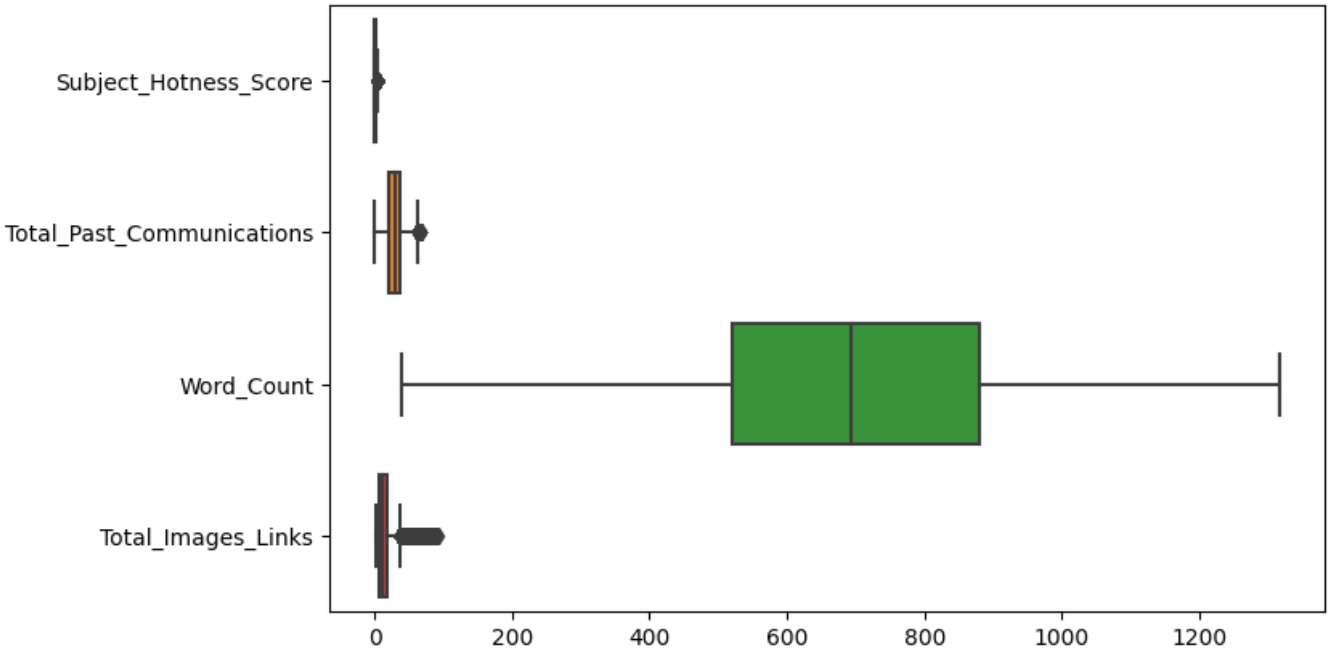
	variables	VIF
0	Subject_Hotness_Score	1.805701
1	Total_Past_Communications	3.939214
2	Word_Count	4.065844
3	Total_Links	8.690857
4	Total_Images	3.171439

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

Outliers Treatment

An outlier is **an individual point of data that is distant from other points in the dataset**. It is an anomaly in the dataset that may be caused by a range of errors in capturing, processing or manipulating data.

Since the dependent variable was highly imbalanced so before dropping outliers it must be checked that it will not delete more than 5% of the minority class which is Email_Status =1,2.



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

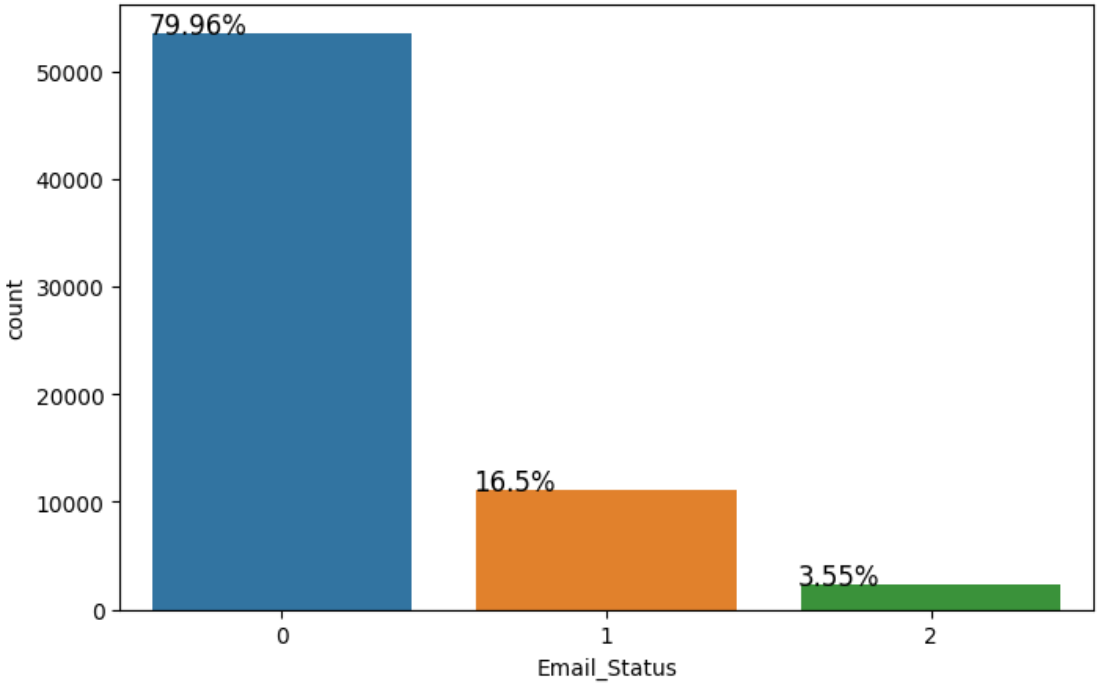
feature scaling and **one hot encoding** of categorical variables were peformed before feeding the data to different classification models.

Handling Imbalance

Only around 3.5% of observations are classified as acknowledged emails and 80% are ignored emails. This will create a bias in favor of ignored emails in the model.

We handled it with Oversampling with SMOTE and Random Under Sampling.

We did the train test split before applying any resampling technique so that the test set remains unknown to the models.



Model Implementation and Evaluation

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.

$$Precision = \frac{TP}{TP + FP}$$

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

$$Recall = \frac{TP}{TP + FN}$$

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

$$F1\_score = 2\frac{Precision * Recall}{Precision + Recall}$$

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.

Logistic Regression: It 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.

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, these results were obtained. As the test set was highly imbalanced, accuracy cannot be trusted but seeing the F1 score and AUC ROC it can be stated that the results weren't satisfactory.

	Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
0	Logistic Regression RUS	0.533720	0.533720	0.517679	0.506365	0.724284	0.627737	0.627737	0.767629	0.680816	0.767966
1	Logistic Regression SMOTE	0.535886	0.535886	0.519509	0.509486	0.723124	0.629082	0.629082	0.767260	0.681743	0.769292

Decision Tree:

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 was 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
Decision Tree RUS	0.999298	0.999298	0.999299	0.999298	0.999999	0.491893	0.491893	0.746085	0.572358	0.614495
Decision Tree SMOTE	0.999408	0.999408	0.999409	0.999408	1.000000	0.698573	0.698573	0.727651	0.712184	0.610119

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 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.

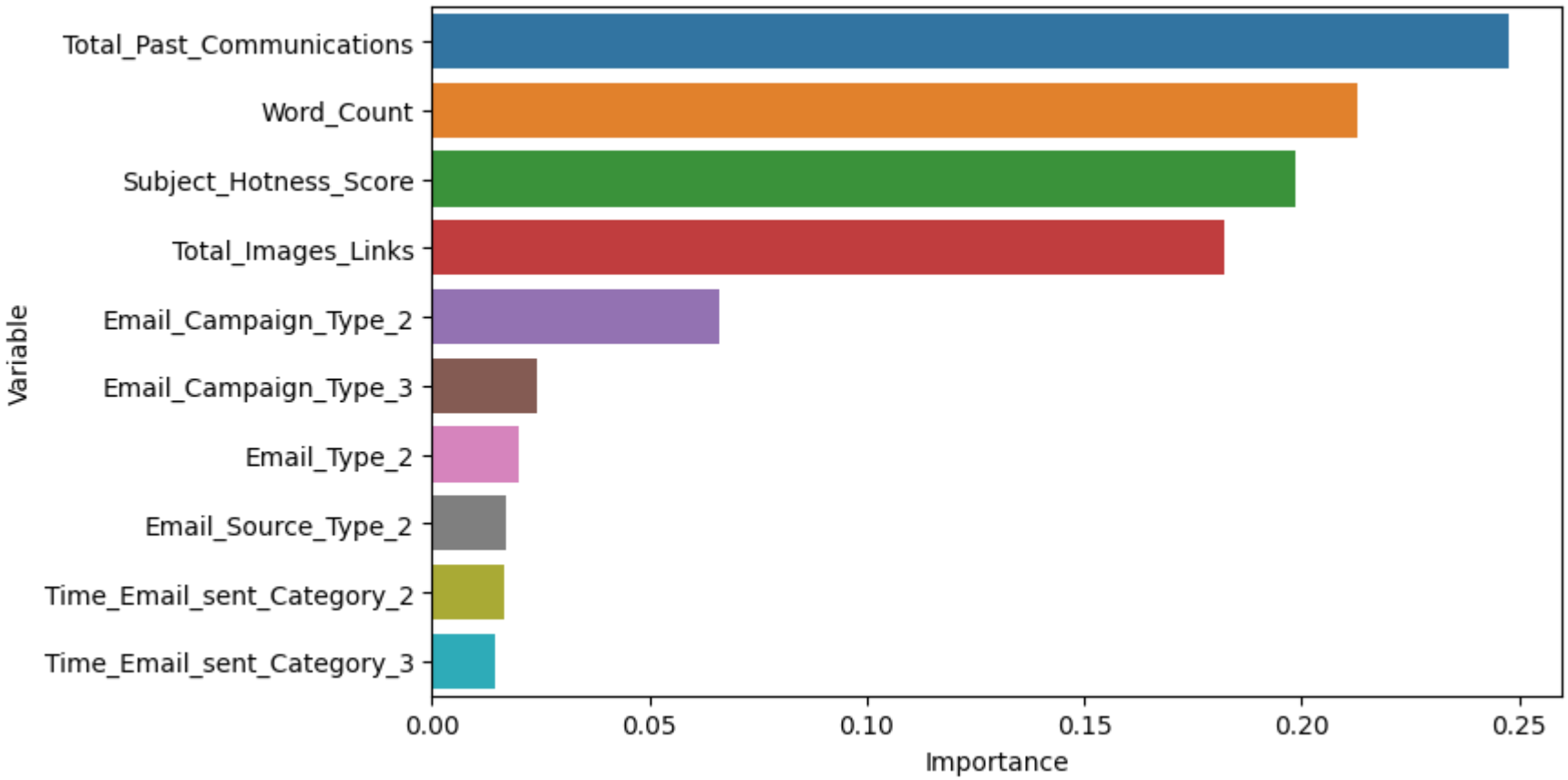
Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
KNN RUS	0.651914	0.651914	0.654008	0.650015	0.841923	0.585839	0.585839	0.759384	0.648853	0.696987
KNN SMOTE	0.881419	0.881419	0.889769	0.878804	0.983693	0.595128	0.595128	0.750594	0.652169	0.673223

Random Forest:

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 that 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 worked.

Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
Random Forest RUS	0.557078	0.557078	0.548266	0.531737	0.754514	0.647314	0.647314	0.775192	0.695424	0.777281
Random Forest SMOTE	0.564911	0.564911	0.549048	0.542821	0.758856	0.674512	0.674512	0.772719	0.712767	0.773859
Random Forest Tuned RUS	0.766948	0.766948	0.769605	0.766291	0.924141	0.621535	0.621535	0.778812	0.679862	0.770906
Random Forest Tuned SMOTE	0.909366	0.909366	0.909476	0.908832	0.984461	0.747740	0.747740	0.761828	0.754455	0.768217


Total_Past_Communications
Word_Count
Subject_Hotness_Score
Total_Images_Links
Email_Campaign_Type_2
Email_Campaign_Type_3
Email_Type_2
Email_Source_Type_2
Time_Email_sent_Category_2
Time_Email_sent_Category_3
Variable
0.00
0.05
0.10
0.15
0.20
0.25
Importance

XGBoost:

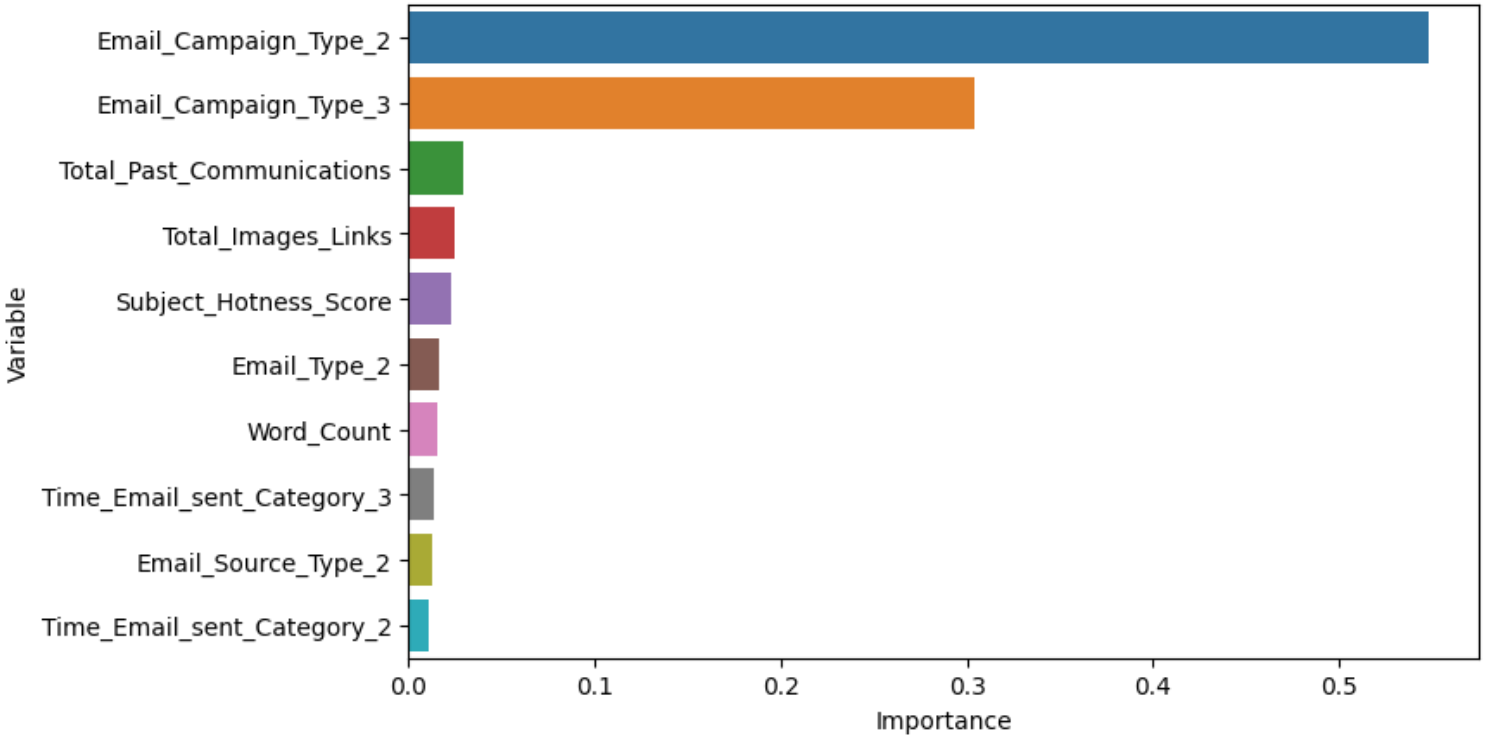
XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The two reasons to use XGBoost are also the two goals of the project first execution Speed and second model performance.

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.

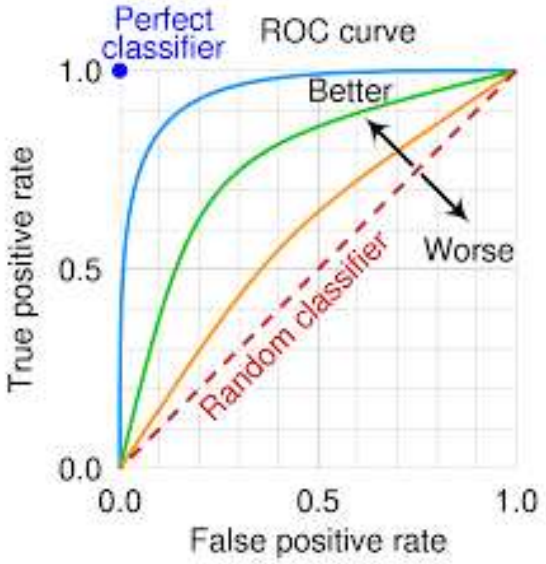
Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

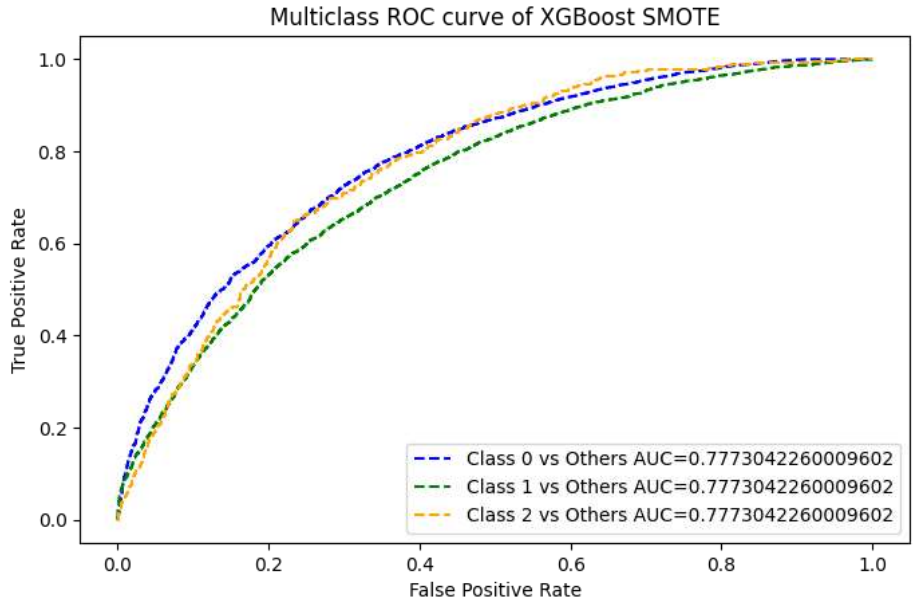
XGBoost SMOTE gave the best results till now, with good Test Recall, F1 score and AUC ROC.

Model_Name	Train_Accuracy	Train_Recall	Train_Precision	Train_F1score	Train_AUC	Test_Accuracy	Test_Recall	Test_Precision	Test_F1score	Test_AUC
XGBoost RUS	0.983667	0.983667	0.983769	0.983664	0.999385	0.580363	0.580363	0.769908	0.648341	0.747362
XGBoost SMOTE	0.910822	0.910822	0.913632	0.909139	0.983722	0.795860	0.795860	0.758843	0.771842	0.777304

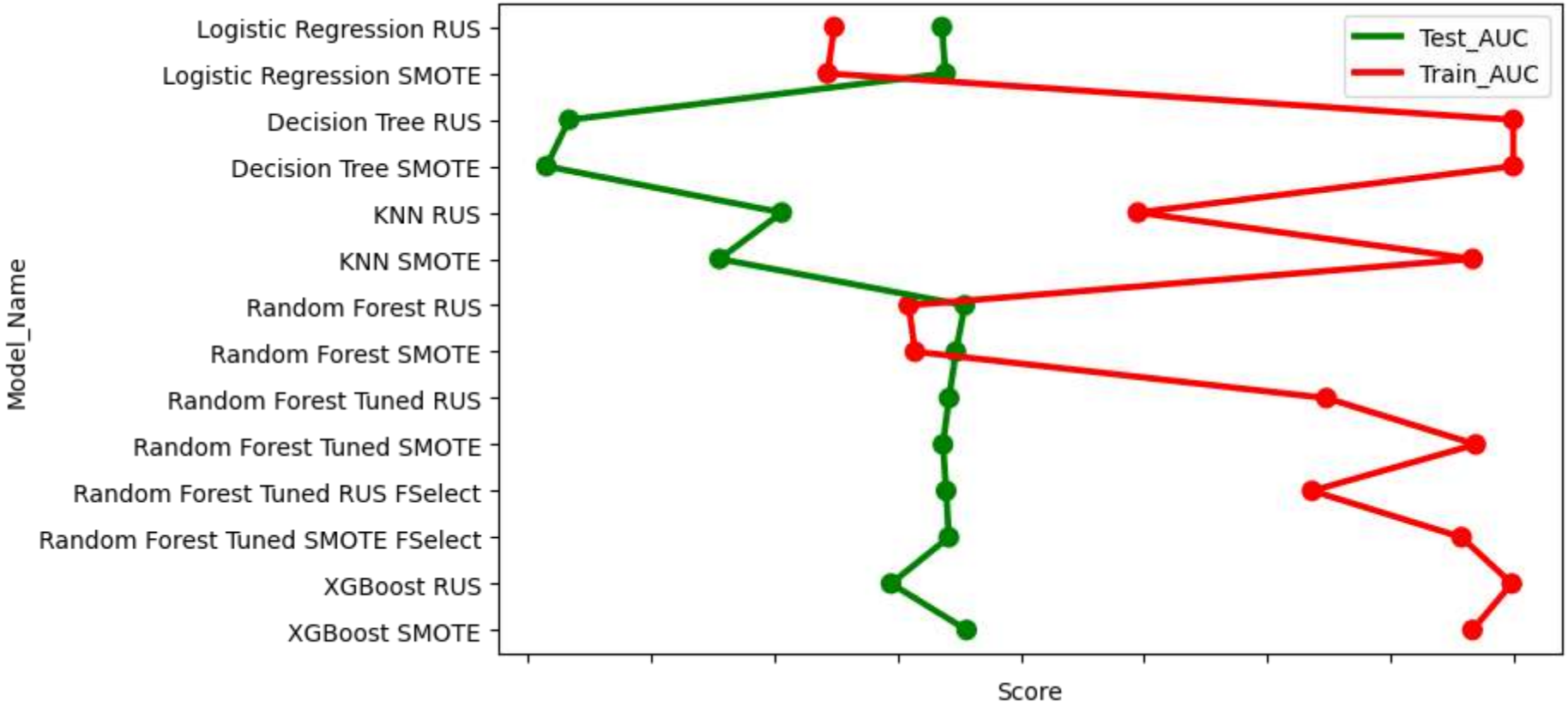


AUC ROC: The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes. When AUC is 0.5, the classifier is not able to distinguish between the classes and when it's closer to 1,the more good it becomes in distinguishing them.





Comparison of all the models



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.

Challenges

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

Thank You

Thanks for your attention