AI Based On Email Campaign Effectiveness Prediction Model

Priyanka Pal

12-05-2023

Step 1: Prototype Selection

ABSTRACT:

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. Email Campaign Effectiveness Prediction is also a product idea that uses machine learning. It can be a tool used by marketing companies or organizations that send out mass email campaigns. The tool can analyze data from past email campaigns to predict the success of future campaigns and help organizations make data-driven decisions.

INTRODUCTION:

Email marketing has become a popular and effective way for businesses and organizations to reach their target audience and promote their products or services. However, designing an effective email campaign can be challenging, and it's not always easy to predict how well a campaign will perform. 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. The Email Campaign Effectiveness Prediction project aims to address this challenge by using machine learning to predict the effectiveness of email campaigns. By analyzing historical data on past email campaigns, the project will build and train a predictive model that can predict the success of future email campaigns. This project has the potential to provide valuable insights for marketing companies and organizations that rely on email marketing to reach their target audience. By predicting the success of email campaigns,

organizations can optimize their marketing strategies and improve the effectiveness of their email marketing campaigns.

PROBLEM STATEMENT:

Most small to medium-sized business owners use Gmail-based Email marketing Strategies for offline targeting and converting prospective customers into leads so that they stay with them in business. The primary goal is to develop a machine learning model to characterise mail and track mail that is ignored, read, and acknowledged by the reader. The columns of data are self-explanatory. Overall , the problem statement for Email Campaign Effectiveness Prediction is to provide a solution for predicting the success of email campaigns and helping marketing professionals make data-driven decisions to improve the effectiveness of their email marketing strategies.

Market/Customer/Business Need Assessment:

1. Market demand:

There is a growing demand for email marketing services that can provide valuable insights and increase the effectiveness of email campaigns. Marketing companies and organizations need to be able to predict the success of their email campaigns in order to optimize their marketing strategies and increase the ROI of their marketing budgets. Email Campaign Effectiveness Prediction provides reliable predictive models to help organizations make data-driven decisions and improve the effectiveness of their email campaigns.

2. client needs:

The client needed a solution that would help them design effective email campaigns and reach their target audience. Email Campaign Effectiveness Prediction provides clients with a data-driven approach to predict the success of their email campaigns, which can help them optimize their marketing strategies and increase the ROI of their marketing budgets. Email campaign performance forecasting helps clients improve their email marketing campaigns and increase revenue by providing them with valuable insights and forecasts.

3. Business needs:

Businesses need a reliable and effective way to reach their target audience and promote their products or services. Email marketing is a cost-effective way to do this, but designing effective email campaigns can be challenging. Email Campaign Effectiveness Prediction provides businesses with a predictive model that helps them optimize their email marketing strategies and increase the effectiveness of their email campaigns. By increasing the ROI of marketing budgets, businesses can increase revenue and remain competitive in their respective markets.

Target Specifications and Characterization :

1. Accuracy: Email Campaign Effectiveness Prediction's predictive models should have high accuracy in predicting the success of email campaigns. The target accuracy level should be at least 80% or higher.
2. Speed: A predictive model should be able to generate predictions quickly, ideally in real time or near real time. The target velocity should be less than a few seconds to generate predictions.
3. Scalability: The system should be able to handle large amounts of data and scale to meet the needs of growing businesses and organizations.
4. Security: The system should be secure to protect the privacy and confidentiality of sensitive data.

External Search (online information sources/references/links):

Dataset:

https://drive.google.com/drive/folders/1zT3MIxlgRSUMwWkEguazPcsdTQVbnBow

Online Research:

https://shwetasingh8597.medium.com/email-campaign-effectiveness-prediction-a5bbdc2ff781

https://noaconnect.com/article/how-to-predict-the-future-of-your-email-marketing-with-mathematics/

https://fulcrumtech.net/resources/anatomy-of-predictive-analytics-in-email-marketing/

Applicable Constraints:

these applicable constraints of Email Campaign Effectiveness Prediction suggest that organizations must carefully consider the availability and quality of data, data privacy regulations, technical expertise, cost, integration challenges, model overfitting, and limitations of machine learning when implementing and using the system. By addressing these constraints, organizations can maximize the effectiveness and value of Email Campaign Effectiveness Prediction for their email marketing campaigns:

1. **Data Availability: The effectiveness of Email Campaign Effectiveness Prediction depends on the availability and quality of historical data. If data is limited or of poor quality, the accuracy of predictive models may suffer.**
2. **Data Privacy: The system should comply with data privacy regulations to protect the privacy and confidentiality of sensitive data.**
3. **Technical expertise: Systems may require technical expertise to build and maintain, which may limit some organizations that lack the necessary resources or expertise.**
4. **Cost: The cost of implementing and maintaining a system can be a constraint for some organizations, especially small businesses or startups.**
5. **Integration: The system may face integration challenges with existing email marketing platforms and tools, which may affect its effectiveness and usability.**
6. **Model Overfitting: Models run the risk of overfitting historical data, which can lead to inaccurate predictions of future email activity.**
7. **Limitations of machine learning: Machine learning models have limitations in their ability to predict human behavior and may not always provide a complete understanding of factors that contribute to the success of an email campaign.**

Business Model of Email Campaign Effectiveness Prediction:

The business model of Email Campaign Effectiveness Prediction could be based on a combination of subscription, commission, customization, freemium, and consulting models. By offering a range of pricing tiers and services, the system can attract a diverse range of clients and generate revenue from multiple sources:

1.Subscription model: The system can be offered as a subscription-based service with different pricing tiers depending on the size of the organization and the volume of email campaigns being analyzed.

2.Commission Model: Another business model can be based on commissions on the ROI generated by email campaigns. In this model, the system takes a percentage of revenue generated from email campaigns optimized using its forecasts.

3.Custom Models: The system can provide customized services to its clients, such as creating custom predictive models based on the unique characteristics of the target audience and email campaign.

4.Freemium model: A freemium model can be offered, where basic predictive analytics will be provided for free, but more advanced features and analytics will require a paid subscription.

5.Consulting Models: In addition to providing predictive models, the system can also provide consulting services to help clients optimize their email marketing campaigns and generate better results.

Understanding the Data:

The 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 integer, float, and object in nature. Let's define the features involved:

- **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 emails from the same source, and 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 the 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 or 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 in our approach:

- Understanding the Data
- Data cleaning and preprocessing- finding null values and imputing them

with appropriate values.

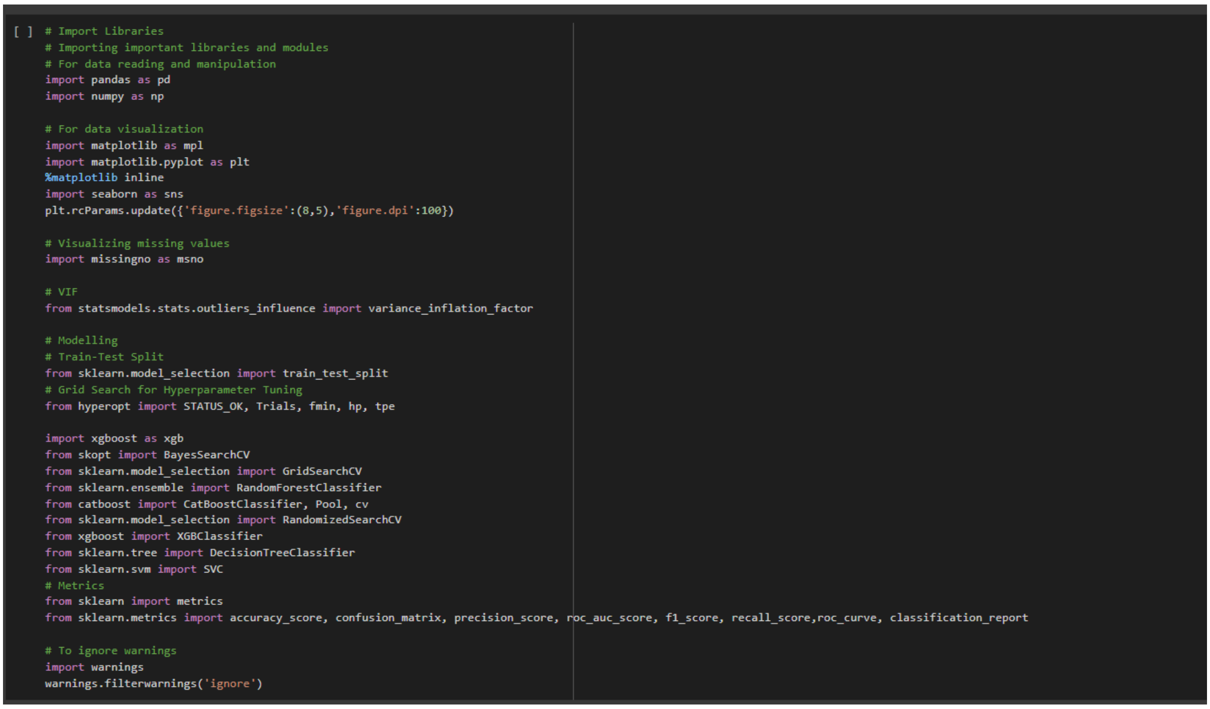
- Exploratory data analysis- of categorical and continuous variables against our target variable.
- Data manipulation- feature selection and engineering, handling multicollinearity with the help of VIF scores, feature scaling, and encoding.
- Handling Class Imbalance- our dataset was highly imbalance with an 80% majority, strategy was to undersampling and oversampling with SMOTE on the train sets only so that our test set remains unknown to

the models, we also applied SMTETomek which is hybridisation of oversampling and undersampling ,so that we can compare our results

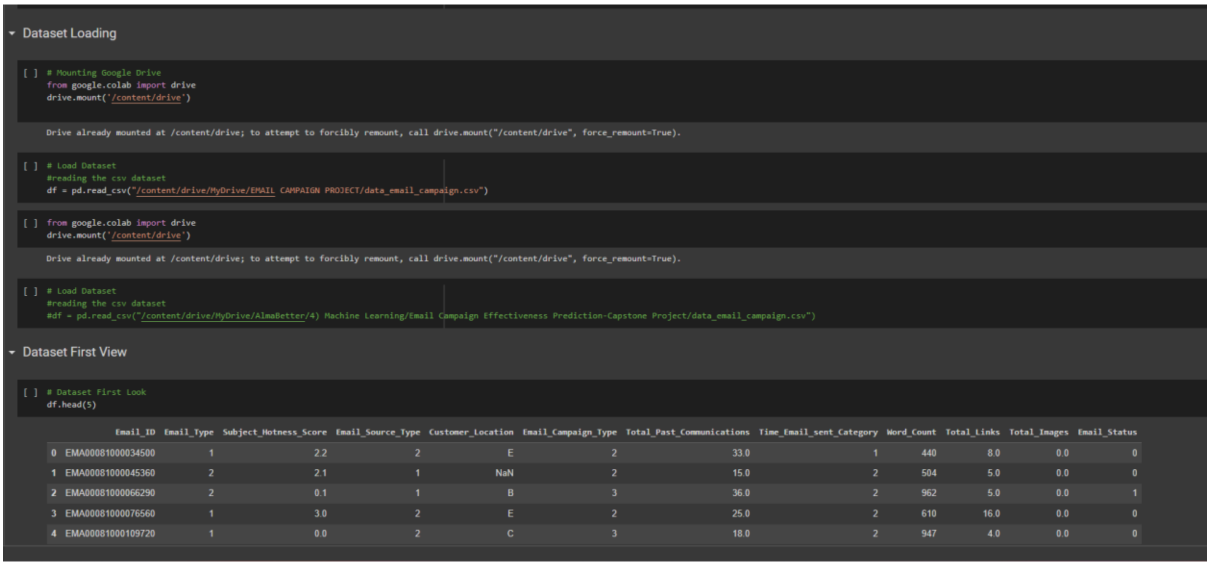
- Modeling- worked on an evaluation code which was frequently used to evaluate the same models on undersampled , oversampled SMOTE and SMOTETomek data in one go, decision trees, random forest, KNN, and XGB, Gradient Boosting, AdaBoost and SVM were run to evaluate the results and then concluded on the basis of model performance and enhancedits performance using Hypertuning the model parameters.We used RandomisedSearchCV technique.

Step 2: Prototype Development

Code Implementation:



```
[ ] # Import Libraries
    # Importing important libraries and modules
    # For data reading and manipulation
    import pandas as pd
    import numpy as np

    # For data visualization
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    plt.rcParams.update({'figure.figsize':(8,5),'figure.dpi':100})

    # Visualizing missing values
    import missingno as msno

    # VIF
    from statsmodels.stats.outliers_influence import variance_inflation_factor

    # Modelling
    # Train-Test Split
    from sklearn.model_selection import train_test_split
    # Grid Search for Hyperparameter Tuning
    from hyperopt import STATUS_OK, Trials, fmin, hp, tpe

    import xgboost as xgb
    from skopt import BayesSearchCV
    from sklearn.model_selection import GridSearchCV
    from sklearn.ensemble import RandomForestClassifier
    from catboost import CatBoostClassifier, Pool, cv
    from sklearn.model_selection import RandomizedSearchCV
    from xgboost import XGBClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.svm import SVC
    # Metrics
    from sklearn import metrics
    from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, roc_auc_score, f1_score, recall_score,roc_curve, classification_report

    # To ignore warnings
    import warnings
    warnings.filterwarnings('ignore')
```



Dataset Loading

```
[ ] # Mounting Google Drive
    from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] # Load Dataset
    #reading the csv dataset
    df = pd.read_csv("/content/drive/MyDrive/EMAIL CAMPAIGN PROJECT/data_email_campaign.csv")
```

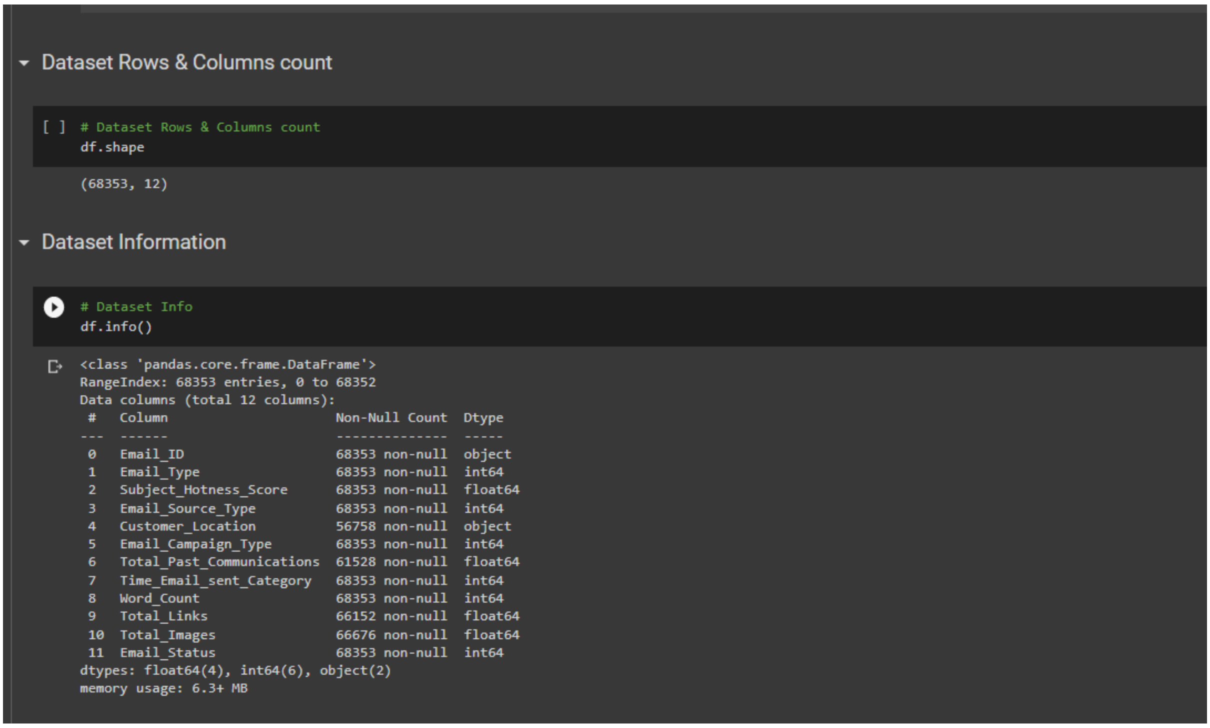
```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] # Load Dataset
    #reading the csv dataset
    #df = pd.read_csv("/content/drive/MyDrive/AlmaBetter/4) Machine Learning/Email Campaign Effectiveness Prediction-Capstone Project/data_email_campaign.csv")
```

Dataset First View

```
[ ] # Dataset First Look
    df.head(5)
```

	Email_ID	Email_Type	Subject_Hotness_Score	Email_Source_Type	Customer_Location	Email_Campaign_Type	Total_Past_Communications	Time_Email_sent_Category	Word_Count	Total_Links	Total_Images	Email_Status
0	EMA00081000034500	1	2.2	2	E	2	33.0	1	440	8.0	0.0	0
1	EMA00081000045360	2	2.1	1	NaN	2	15.0	2	504	5.0	0.0	0
2	EMA00081000066290	2	0.1	1	B	3	36.0	2	962	5.0	0.0	1
3	EMA00081000076560	1	3.0	2	E	2	25.0	2	610	16.0	0.0	0
4	EMA00081000109720	1	0.0	2	C	3	18.0	2	947	4.0	0.0	0

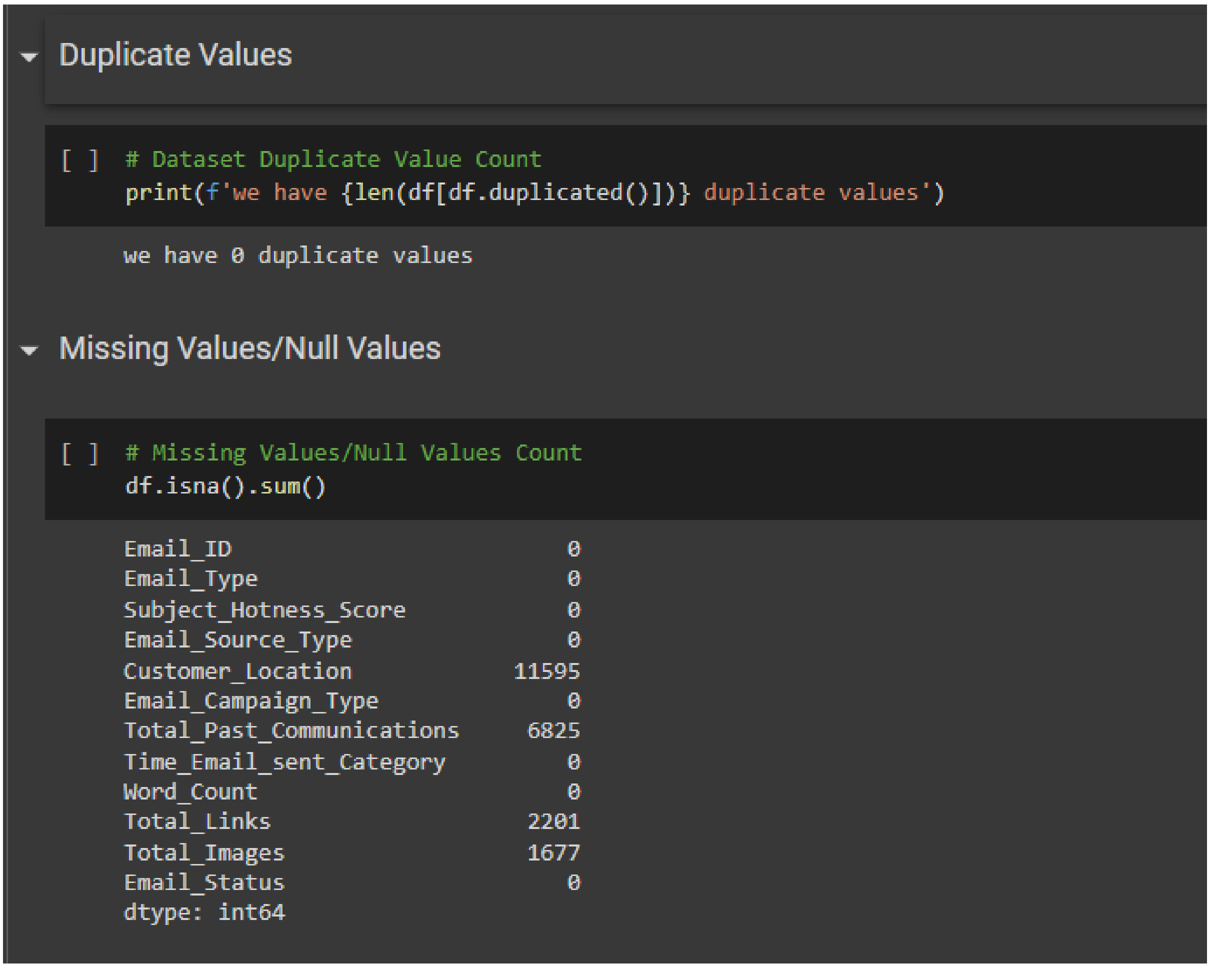


Dataset Rows & Columns count

```
[ ] # Dataset Rows & Columns count
    df.shape

    (68353, 12)
```

Dataset Information

```
# Dataset Info
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68353 entries, 0 to 68352
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Email_ID                   68353 non-null  object 
 1   Email_Type                 68353 non-null  int64  
 2   Subject_Hotness_Score      68353 non-null  float64
 3   Email_Source_Type          68353 non-null  int64  
 4   Customer_Location          56758 non-null  object 
 5   Email_Campaign_Type        68353 non-null  int64  
 6   Total_Past_Communications  61528 non-null  float64
 7   Time_Email_sent_Category   68353 non-null  int64  
 8   Word_Count                 68353 non-null  int64  
 9   Total_Links                66152 non-null  float64
 10  Total_Images               66676 non-null  float64
 11  Email_Status               68353 non-null  int64  
dtypes: float64(4), int64(6), object(2)
memory usage: 6.3+ MB
```



Duplicate Values

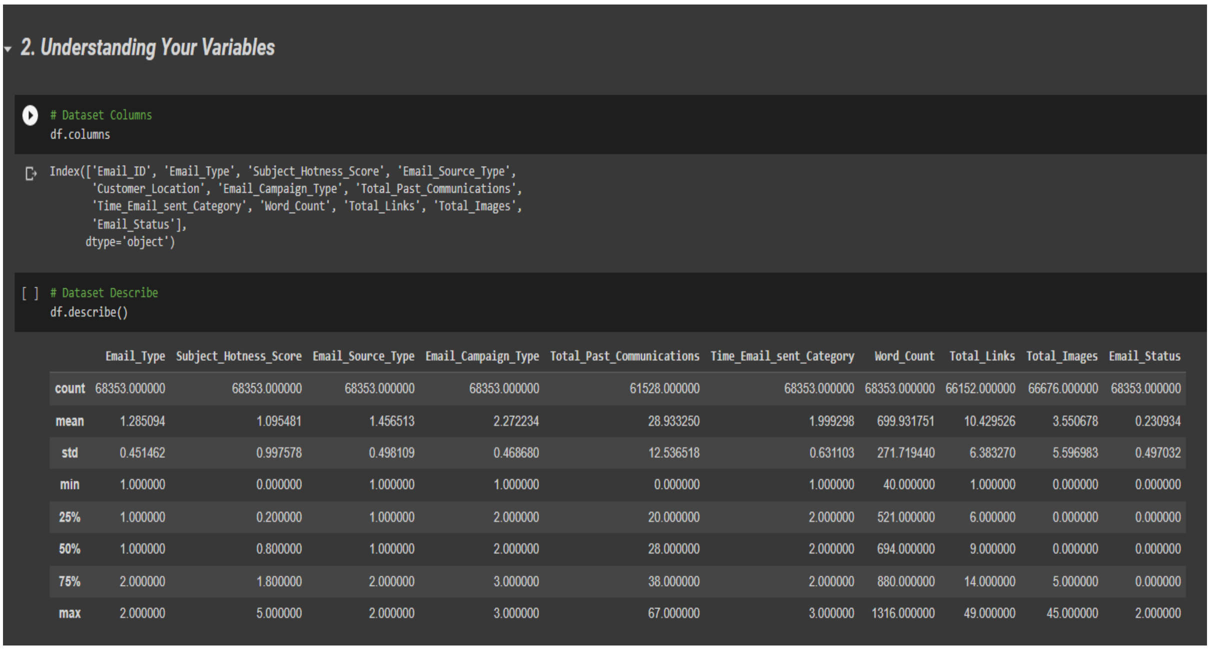
```
[ ] # Dataset Duplicate Value Count
    print(f'we have {len(df[df.duplicated()])} duplicate values')
```

```
we have 0 duplicate values
```

Missing Values/Null Values

```
[ ] # Missing Values/Null Values Count
    df.isna().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
```



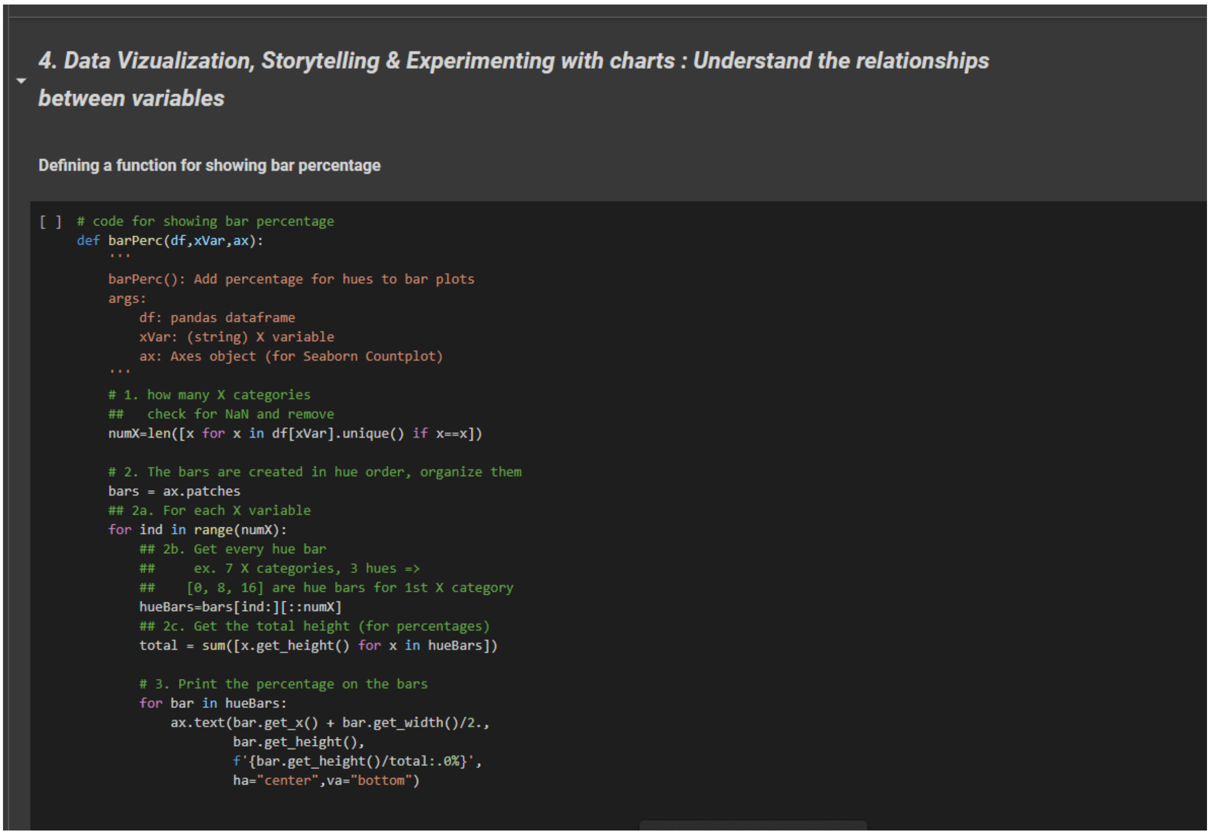
2. Understanding Your Variables

```
# Dataset Columns
df.columns
```

```
Index(['Email_ID', 'Email_Type', 'Subject_Hotness_Score', 'Email_Source_Type',
       'Customer_Location', 'Email_Campaign_Type', 'Total_Past_Communications',
       'Time_Email_sent_Category', 'Word_Count', 'Total_Links', 'Total_Images',
       'Email_Status'],
      dtype='object')
```

```
[ ] # Dataset Describe
    df.describe()
```

	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
count	68353.000000	68353.000000	68353.000000	68353.000000	61528.000000	68353.000000	68353.000000	66152.000000	66676.000000	68353.000000
mean	1.285094	1.095481	1.456513	2.272234	28.933250	1.999298	699.931751	10.429526	3.550678	0.230934
std	0.451462	0.997578	0.498109	0.468680	12.536518	0.631103	271.719440	6.383270	5.596983	0.497032
min	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	40.000000	1.000000	0.000000	0.000000
25%	1.000000	0.200000	1.000000	2.000000	20.000000	2.000000	521.000000	6.000000	0.000000	0.000000
50%	1.000000	0.800000	1.000000	2.000000	28.000000	2.000000	694.000000	9.000000	0.000000	0.000000
75%	2.000000	1.800000	2.000000	3.000000	38.000000	2.000000	880.000000	14.000000	5.000000	0.000000
max	2.000000	5.000000	2.000000	3.000000	67.000000	3.000000	1316.000000	49.000000	45.000000	2.000000



4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Defining a function for showing bar percentage

```
[ ] # code for showing bar percentage
    def barPerc(df,xVar,ax):
        '''
        barPerc(): Add percentage for hues to bar plots
        args:
            df: pandas dataframe
            xVar: (string) X variable
            ax: Axes object (for Seaborn Countplot)
        '''
        # 1. how many X categories
        ##   check for NaN and remove
        numX=len([x for x in df[xVar].unique() if x==x])

        # 2. The bars are created in hue order, organize them
        bars = ax.patches
        ## 2a. For each X variable
        for ind in range(numX):
            ## 2b. Get every hue bar
            ##     ex. 7 X categories, 3 hues =>
            ##    [0, 8, 16] are hue bars for 1st X category
            hueBars=bars[ind:][::numX]
            ## 2c. Get the total height (for percentages)
            total = sum([x.get_height() for x in hueBars])

            # 3. Print the percentage on the bars
            for bar in hueBars:
                ax.text(bar.get_x() + bar.get_width()/2.,
                        bar.get_height(),
                        f'{bar.get_height()/total:.0%}',
                        ha="center",va="bottom")
```

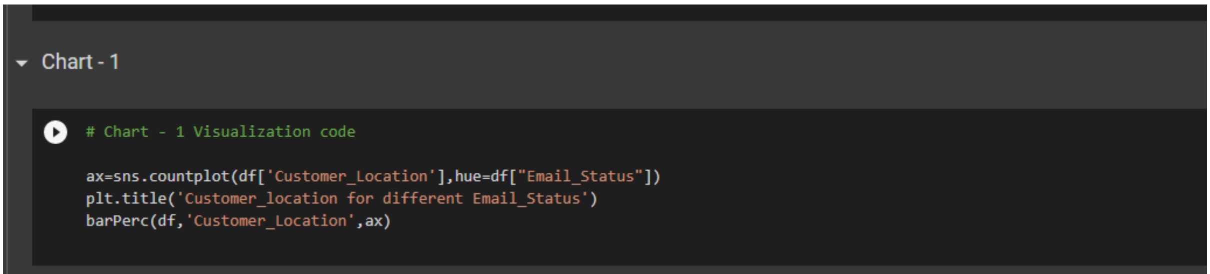
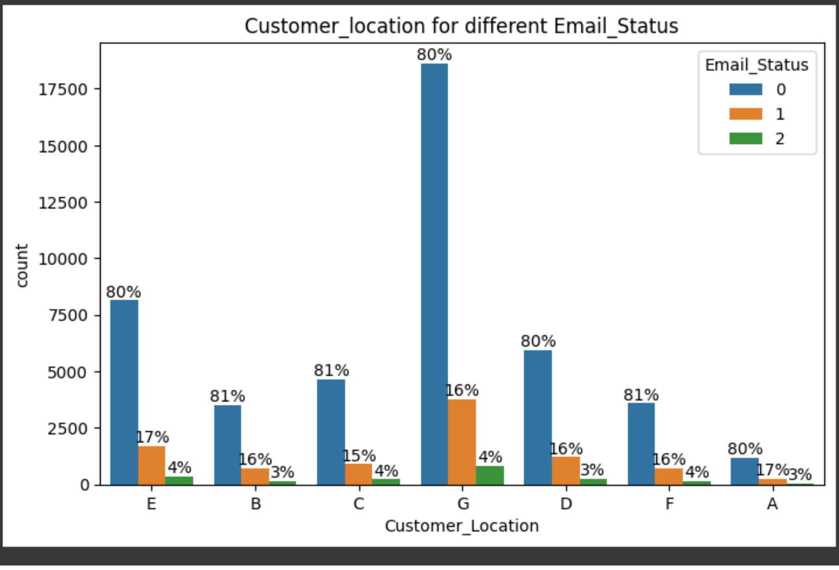


Chart - 1

```
# Chart - 1 Visualization code

ax=sns.countplot(df['Customer_Location'],hue=df["Email_Status"])
plt.title('Customer_location for different Email_Status')
barPerc(df,'Customer_Location',ax)
```



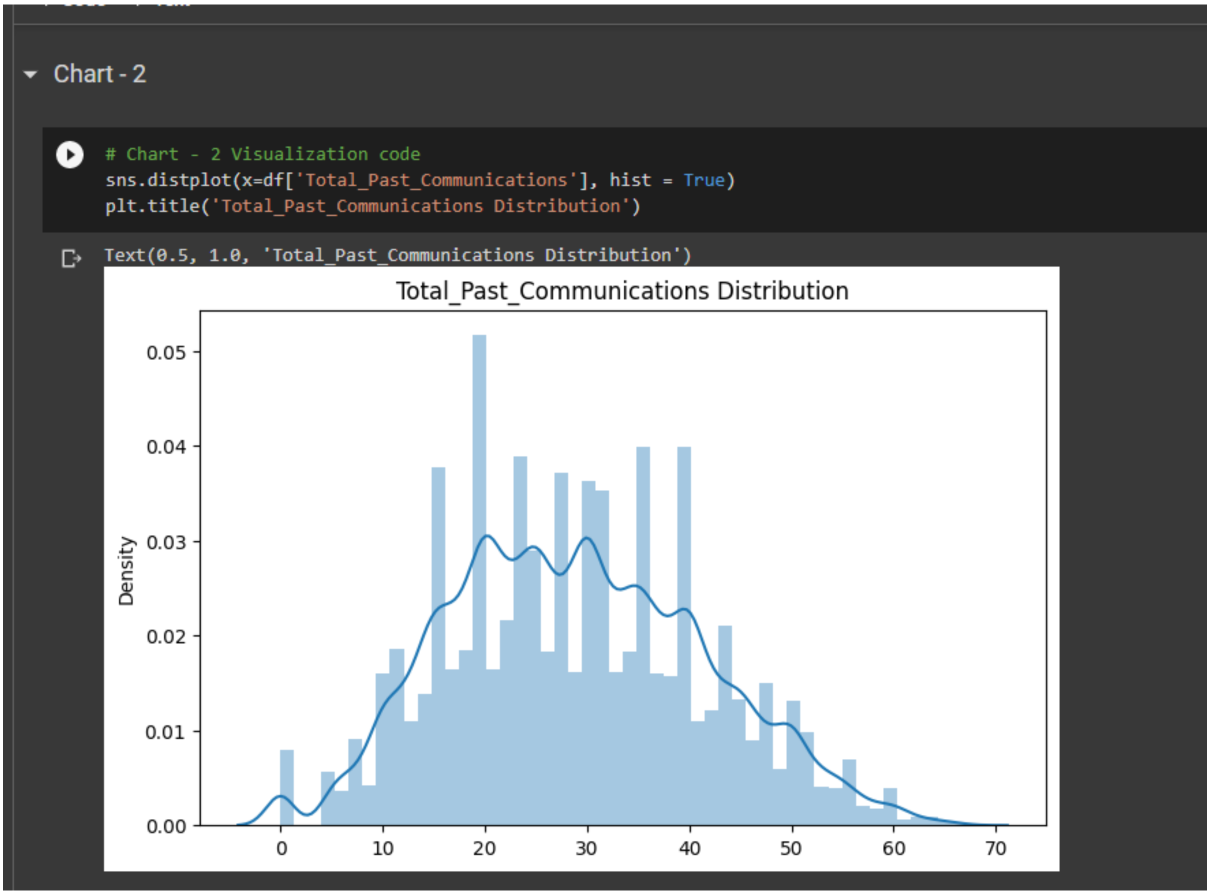


Chart - 2

```
# Chart - 2 Visualization code
sns.distplot(x=df['Total_Past_Communications'], hist = True)
plt.title('Total_Past_Communications Distribution')
```

```
Text(0.5, 1.0, 'Total_Past_Communications Distribution')
```

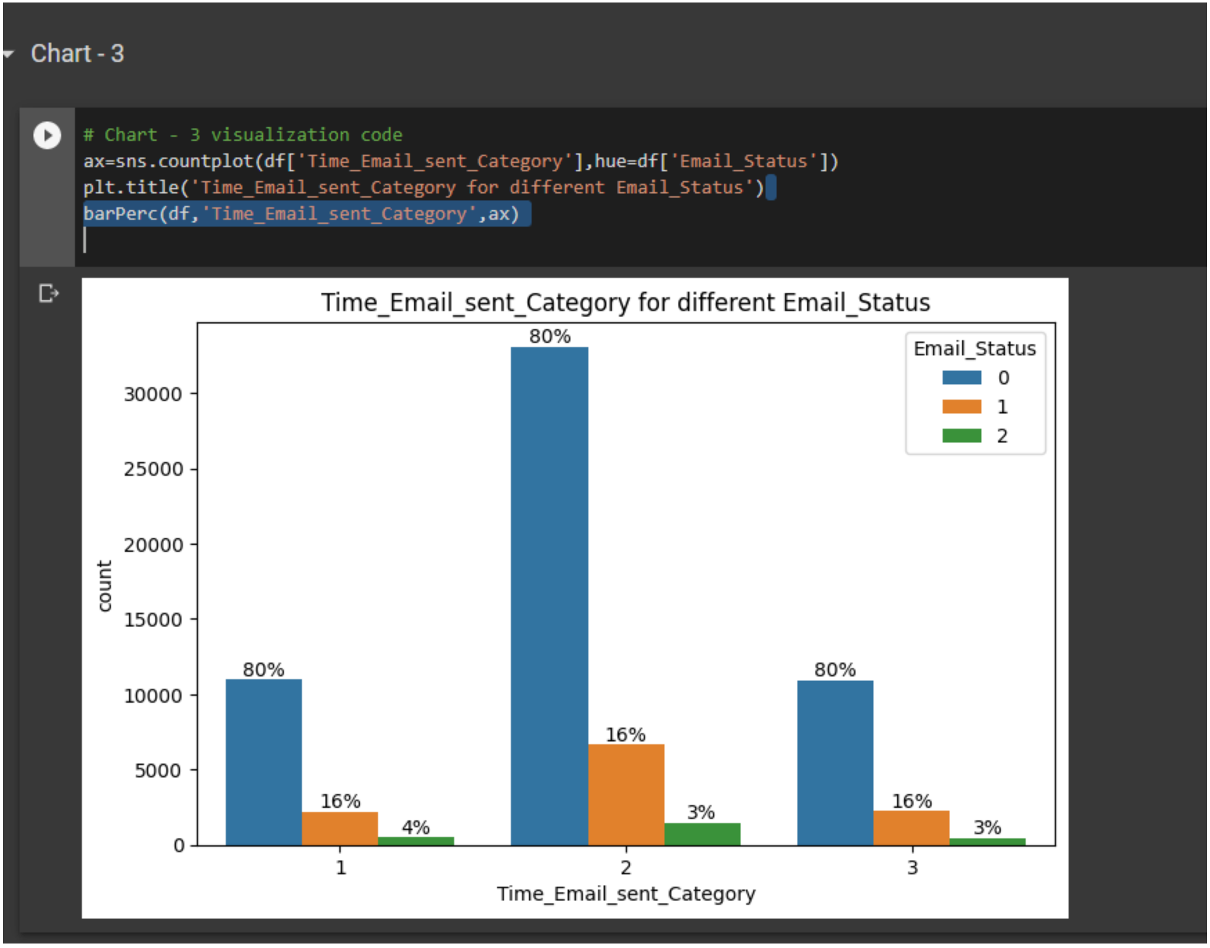
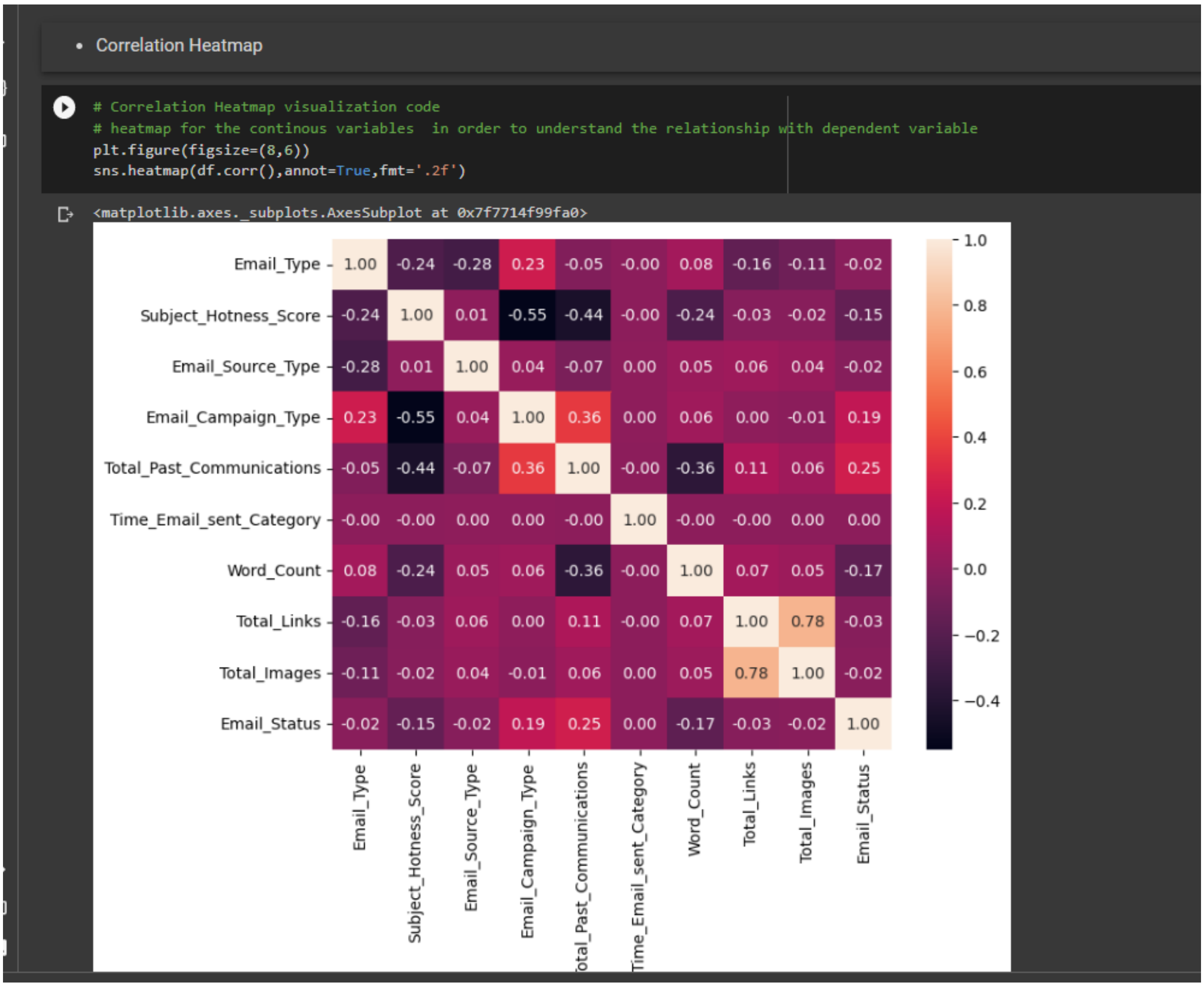


Chart - 3

```
# Chart - 3 visualization code
ax=sns.countplot(df['Time_Email_sent_Category'],hue=df['Email_Status'])
plt.title('Time_Email_sent_Category for different Email_Status')
barPerc(df,'Time_Email_sent_Category',ax)
```



- Correlation Heatmap

```
# Correlation Heatmap visualization code
# heatmap for the continous variables  in order to understand the relationship with dependent variable
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(),annot=True,fmt='.2f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7714f99fa0>
```

	Email_Type	Subject_Hotness_Score	Email_Source_Type	Email_Campaign_Type	otal_Past_Communications	Time_Email_sent_Category	Word_Count	Total_Links	Total_Images	Email_Status
Email_Type	1.00	-0.24	-0.28	0.23	-0.05	-0.00	0.08	-0.16	-0.11	-0.02
Subject_Hotness_Score	-0.24	1.00	0.01	-0.55	-0.44	-0.00	-0.24	-0.03	-0.02	-0.15
Email_Source_Type	-0.28	0.01	1.00	0.04	-0.07	0.00	0.05	0.06	0.04	-0.02
Email_Campaign_Type	0.23	-0.55	0.04	1.00	0.36	0.00	0.06	0.00	-0.01	0.19
Total_Past_Communications	-0.05	-0.44	-0.07	0.36	1.00	-0.00	-0.36	0.11	0.06	0.25
Time_Email_sent_Category	-0.00	-0.00	0.00	0.00	-0.00	1.00	-0.00	-0.00	0.00	0.00
Word_Count	0.08	-0.24	0.05	0.06	-0.36	-0.00	1.00	0.07	0.05	-0.17
Total_Links	-0.16	-0.03	0.06	0.00	0.11	-0.00	0.07	1.00	0.78	-0.03
Total_Images	-0.11	-0.02	0.04	-0.01	0.06	0.00	0.05	0.78	1.00	-0.02
Email_Status	-0.02	-0.15	-0.02	0.19	0.25	0.00	-0.17	-0.03	-0.02	1.00

Techniques that are used to handle the imbalance dataset are

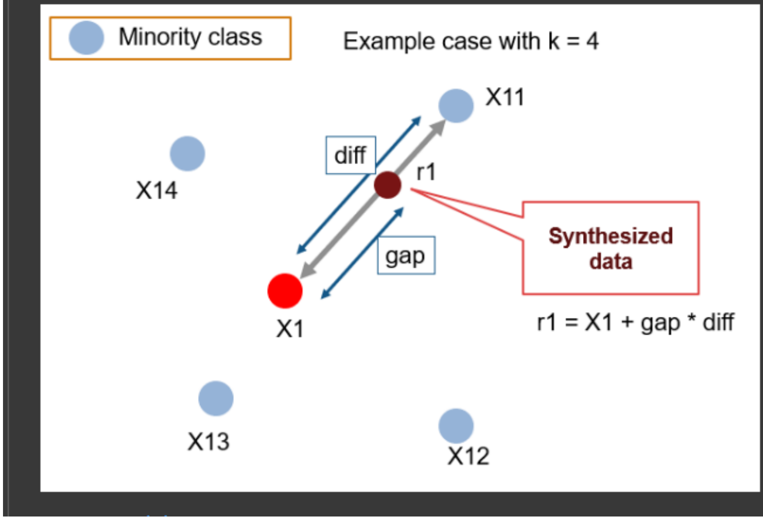
- **RANDOM UNDER SAMPLING**

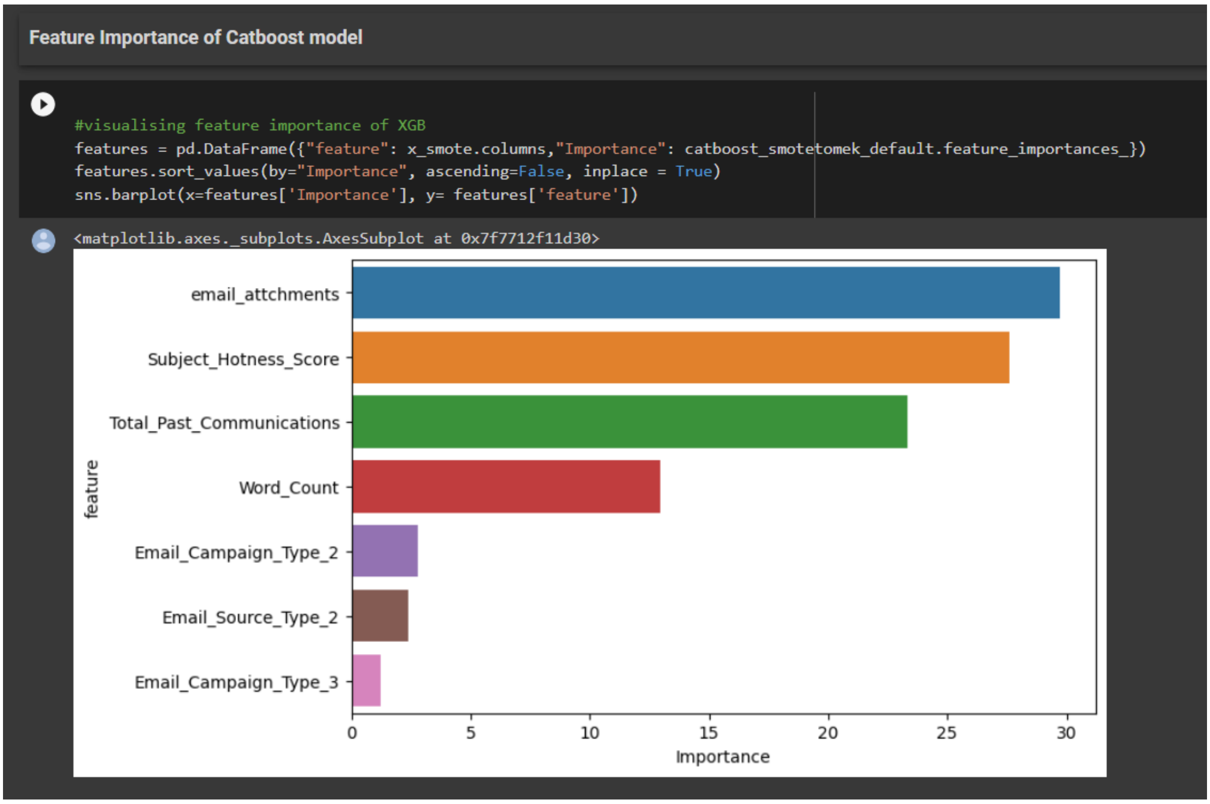
This is a Undersampling Technique.Here the majority class is reduced to the minimum number of members for minority class , and this process of deleting instances is done randomly .Therefore this process is known as Random Under Sampling. This process can be repeated until the desired class distribution is achieved , like for getting upto equal number of classes

- **SMOTE** (Synthetic Minority Oversampling Technique)

It is smart way of increasing the number of samples for the minority classes , its better than duplicating the instances from the minority class it uses synthetic generation of instances from the minority classes. The instances formed newly are samples of the feature space for each target class and its nearest neighbour .

here is the the visualization of how the synthetic data is generated





Feature Importance of Catboost model

```
#visualising feature importance of XGB
features = pd.DataFrame({"feature": x_smote.columns,"Importance": catboost_smotetomek_default.feature_importances_})
features.sort_values(by="Importance", ascending=False, inplace = True)
sns.barplot(x=features['Importance'], y= features['feature'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7712f11d30>
```

Model Interpretations

- For the modeling part we found that SMOTETomek data was good for most of the models.
- RandomizedSerchCV hypertunning technique was used and it was useful and time saving.
- Overall Catboost with SMOTE was giving best results with 77.1% F1 score on test data and 80.1% F1 score on Training data.
- According to catboost model Total_past_communications,subject_hotness_score word_count and email_attachments which was combination of total_links and total_images are most important features.

CONCLUSION:

- We found that our customer location feature to be not related with dependent variable email status.

- **We found total_links and total_images are multicorrelated, so we created another feature email_attachments which combines the total_links and total_images.**
- **We found through feature selection that email_type and time_email_sent_category to be not related to the dependent variable.**
- **We found that removing outliers would impact the minority class with more than 5% loss of data, so we decided not to drop the outliers.**
- **We found that 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.**