**Email Classification**

**Group 1:**

**Team Members: Shantanu Deshmukh, Amit Divekar and Bhushan Mahajan**

**Mentors: Karthik and Parth**

Deployment Link:<https://emailclassification.herokuapp.com/>

* Objective:- -The goal of the project is to identify such emails in the given day based on the above inappropriate content.
* Steps to achieve objective:

1. Reading the data and understanding the data set.

Finding duplicates using “df.drop\_duplicates”

Finding Null values using “df.isnull().sum”

1. Data Preprocessing.

Textual data is usually contains lots of numbers, punctuations, special characters.

These are not useful or increases complexity of data for EDA. So data is cleaned to simplify it.

Library used “re”

Also tokenization and lemmatization of mails is completed using “TextBlob”.

After this stopwords are imported from nltk corpus & removed from preprocessed data. Also some more stopwords were added after trials.

1. Exploratory data Analysis(EDA):-

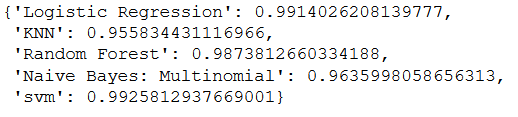
Library used matplotlib.pyplot

|  |  |  |  |
| --- | --- | --- | --- |
| EDA Method | Function used | EDA | Remark |
| Pie Chart | ax1.pie |  | Class distribution is uneven.  93.3%-Non Abusive  6.7%-Abusive  So data provided is imbalanced. |
| Bar Plot | .from\_records(counter.most\_common()  freq.plot(kind='bar',x='word') |  | This analysis is cone for Abusive, Non abusive class & for whole data.  Abusive class contains words like shit, fuck etc.  Non abusive class contains words like meet, information,  Message etc.  Whole data contains words like trade, corp, market etc. |
| Word  cloud | Wordcloud  (width, height, color) | Negative WordCloud    Positive Wordcloud | Wordcloud is drawn for Abusive, non-abusive class.    Also Positive and negative wordcloud is drawn |

1. Data Preparation

* Splitting the data into train and test
* Tokenization of words into a bag of words using CountVectorizer
* Normalizing the bag of words using TFIDF Vecotrizer
* Preparing Train and Test sparse matrix by transforming test and train data
* Implementing SMOT for oversampling due to unbalanced dataset

1. Model Selection



From the above results SVM model was base lined for further evaluation

1. Model building:

* SVM model is fitting
* Model evaluated using Confusion Matrix, Classification Report as evaluation metrics.
* Hyper-parameter tuning using Grid Search CV
* Evaluation of the tuned model
* Finalizing the tuned model by exporting the model into a pickle format

1. Model Deployment

Model is deployed using Heroku which PaaS (Platform as a Service). Following things are used to deploy the ML model.

* *model.py* —Code for the machine learning model to classify mails.
* *app.py* — Flask APIs that receives mail through GUI or API calls,

provide class of mail based on our model and returns it.

* *request.py* — Uses requests module to call APIs defined in app.py

and displays the returned value.

* *HTML/CSS* —HTML template and CSS styling to allow user to

enter mail and displays the class.

Link:<https://emailclassification.herokuapp.com/>