EMAIL SPAM CLASSIFICATION USING RANDOM FOREST AND XGBOOST CLASSIFIERS

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

J. SAKETH, 221910307029
M. SAI PAVAN NIKHIL, 221910307037
N. VENKATA SAI PRASAD, 221910307041
Y. NITHISH RAO, 221910307060

Under the esteemed guidance of

V. REKHA

Asst. Prof

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GITAM
(Deemed to be University)
HYDERABAD
NOVEMBER-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "**EMAIL SPAM CLASSIFICATION USING RANDOM FOREST AND XGBOOST CLASSIFIERS**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University), submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s).	Name(s)	Signature(s)
221910307029	J. SAKETH	
221910307037	M. SAI PAVAN NIKHIL	
221910307041	N. VENKATA SAI PRASAD	
221910307060	Y. NITHISH RAO	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled ““**EMAIL SPAM CLASSIFICATION USING RANDOM FOREST AND XGBOOST CLASSIFIERS**” is a bonafide record of work carried out by **J. SAKETH 221910307029, M. SAI PAVAN NIKHIL 221910307037, N. VENKATA SAI PRASAD 221910307041, Y. NITHISH RAO 221910307060** students submitted in partial fulfilment of the requirement for the award of the degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
V REKHA	**Dr S PHANI KUMAR**
Asst. Prof	**Professor**

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. We would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are incredibly thankful to our honorable Pro-Vice Chancellor, **Prof. D. Sambasiva Rao,** for providing the necessary infrastructure and resources to accomplish our project.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the project's tenure.

We are very much obliged to our beloved **Prof. Dr. S. Phani Kumar**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this project and for encouragement in the completion of this project.

We would like to express our special thanks to our Project Co-ordinator **Dr. S. Aparna,** Assistant Professor, Department of Computer Science and Engineering, School of Technology, for her time and efforts she provided throughout the year.

We hereby wish to express our deep sense of gratitude to **Mrs. V. Rekha**, Assistant Professor, Department of Computer Science and Engineering, School of Technology, for the esteemed guidance, moral support, and invaluable advice provided by him for the success of the project.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support directly or indirectly in our project work.

Sincerely,

J. SAKETH (221910307029)

M. SAI PAVAN NIKHIL (221910307037),

N. VENKAT SAI PRASAD (221910307041),

Y. NITISH (221910307060)

TABLE OF CONTENTS

1. Abstract 1

2. Introduction 2

3. Literature Review 3

4. Problem Identification & Objectives 5

4.1 Scope and Methodology 5

4.2 Problem Statement 5

4.3 Existing System 5

4.4 Proposed System 6

4.5 Software Requirements 6

4.6 Hardware Requirements 6

5. System Methodology 7

5.1 Architecture 8

5.2 UML Diagrams 9

6. Overview of Technologies 12

6.1 Technologies Used 12

6.2 Libraries Used 12

7. Implementation 16

7.1 Coding 17

7.2 Testing 26

8. Results & Discussions 31

9. Conclusion & Future Scope 32

10. References 33

LIST OF FIGURES

1 System Architecture 8

2 Use Case Diagram 9

3 Activity Diagram 10

4 Class Diagram 11

1. Abstract

Email is one of the most popular and frequently used ways of communication due to its worldwide accessibility, relatively fast message transfer, and low sending cost. The flaws in the email protocols and the increasing amount of electronic business and financial transactions directly contribute to the increase in e-mail-based threats. Email spam is one of the significant problems of today's internet, bringing financial damage to companies and annoying individual users. Spam emails are invading users without their consent and filling their mailboxes. They consume more network capacity and time in checking and deleting spam emails.

The vast majority of Internet users are outspoken in their disdain for Spam, although enough of them respond to commercial offers that Spam remains a viable source of income for spammers. While most users want to do the right thing to avoid and get rid of Spam, they need clear and simple guidelines on how to behave. Despite all the measures taken to eliminate Spam, they are not yet eradicated. Also, when the countermeasures are over-sensitive, even legitimate emails will be eliminated. Among the approaches developed to stop Spam, filtering is one of the essential techniques. Much research in spam filtering has been centered on more sophisticated classifier-related issues. In recent days, Machine learning for spam classification has been an important research issue. The effectiveness of the proposed work explores and identifies the use of different learning algorithms for classifying spam messages from email. A comparative analysis of the algorithms has also been presented.

2. Introduction

Spam is one of the most severe risks, with attackers launching single attacks on as many devices as possible in the network. Although a few technologies, such as spam signatures and behavior analysis, have helped solve problems in the past, they no longer operate on massive networks. Furthermore, these approaches do not allow for the transmission of Spam over the internet. This paper also discusses existing programs and associated difficulties. The tool's design and implementation are critical for monitoring and identifying spam attacks on a real-time network. A mechanism is being created in this project to distinguish between Spam and non-spam devices by exchanging internet messages.

The most common Spam that most users encounter daily is email spam Anonymity, bulk email, and unsolicited emails are three aspects of email spam Anonymity is a property of a different encryption that conceals the email's sender. Unsolicited emails are transmitted to uninvited receivers, and emailing is described as sending repeated identical emails to many groups. Email spam is an email sent to several groups without any request for anonymity. This is the most prevalent type of Spam that many users see on different blogs. Spammers utilize blog postings to drive spam victims to spam websites. It is used chiefly to promote search services like Wikipedia, and guest books, among other things.

The steps in detecting Spam on social media are often as follows. Obtaining the spam text collection (dataset) is the initial step. Because these datasets frequently have the unstructured text and may contain noisy data, preprocessing is almost always necessary. The following step is selecting a feature extraction method, such as Word2Vec, n-grams, TF-IDF, etc. Finally, various spam detection technologies, such as machine learning, and Lexicon-based algorithms, are utilized to decide whether texts are Spam.

We compare the accuracy of existing spam text detection systems to determine the most effective ones. EMAIL SPAM CLASSIFICATION USING RANDOM FOREST AND XGBOOST CLASSIFIERS uses a block diagram to explain the multiple steps involved in spam detection.

3. Literature Review

The work of B. Biggio, G.Fumera, I.Pilla, and F. Ro li et al. [I], A survey and experimental evaluation of image spam filtering techniques, has proposed junk that depends on your location. The recommended techniques have the same goal: to keep image spam out of our inboxes. Spam senders use a variety of strategies to circumvent screening, and each requires analysis to establish where the filters should be placed to defeat the tricks and prevent spam senders from filling our mailboxes. Different strategies are created through different techniques.

In the work of A.Heydari, M.A.Tavako li, N.Salim, and Z.Heydari, et al. [2], Detection of review spam: A survey suggested that the proliferation of E-commerce sites has made the web an excellent source for collecting customer reviews about products; as there is no level control anyone can write anything that leads to spam reviews. This project previews and reviews extensive research on how to detect Spam. In addition, it provides an artistic environment that reflects a sorry previous effort to learn spam detection. Today due to the popularity of Eco Commerce sites, it has become a target for spammers without known erna i1 and web spam Up; date spam refers to Spam written Spam so that they can denigrate product features or defile themselves.

In this work by A.H.Mohammad, R.A.Zitar et al. [4], Application of genetically optimized artificial immune system and neural networks in spam detection Has been proposed. The present thesis dreams of making an in-depth study of adaptive identification, digital channel equalization, and a helpful link between the artificial neural community and the Artificial Immune System. These new models are performed for adaptive identification of complex nonlinear dynamic plant life and equalization of nonlinear digital channels. The investigation has been made for the identification of complex Hainrierstein models. Simulation results are compared with those acquired with FLANN- GA, FLANN-P SO, and MLP-BP-based completely hybrid approaches. Improved identification and equalization primary overall performance of the proposed method have been placed in all cases.

In this work of A. Visconti, H.Tahayori, etc., Artificial immune system based on interval type-2 fuzzy set paradigm has proposed that Spam has made the mail gadget unreliable due to the fact mail may be stuck falsely with the aid of using unsolicited mail filtering earlier than being introduced to the recipient conversely unsolicited mail mails relay be discovered with ide the recipient mailbox Artificial Immune System version is stimulated from the herbal immune gadget. A getting-to-know segment is supplied to praise the cells that efficiently understand the unsolicited mail emails. Instead of clonal choice, the terrible choice is used within the schooling segment, which ended in better checking out overall performance because of the decreased quantity of detectors. The herbal immune gadget defends the frame against dangerous illnesses and infections. It can spot and retrieve any overseas molecule or molecule.

In the work of M.Cong, J.Zhang, J. Ma, and L. Jiao et al. [8], An efficient negative selection algorithm with further training for anomaly detection has proposed that the adverse selection algorithm is one of the oldest immune-inspired classification algorithms and was initially intended for anomaly detection tasks in computer security. After initial enthusiasm, performance problems with the algorithm led many researchers to conclude that adverse selection is not a competitive anomaly detection technique.

However, in recent years, theoretical work has led to substantially riskier efficient negative selection algorithms. Here, the results of the first evaluation of adverse selection with r-chunk and r-contiguous detectors that employ these novel algorithms have been reported. On a collection of 14 datasets from real-world sources, adverse selection is compared with r-chunk and r- contiguous detectors against techniques based on kennels, finite state automata, and n-gram frequencies, and finds that adverse selection performs competitively, yielding a slightly better average performance than all other techniques investigated. Because this study represents, to our knowledge, the comprehensive one of string-based adverse selection to date, the widely held view that negative selection is not a competitive anomaly detection technique may be inaccurate in the work of A. Bratko, B.Filip ic, G.V. Cormack, T.R.Lynam, and Zupan et al., "email spam classification using random forest and boost classifier" Has proposed Spam filtering poses a unique trouble in textual content categorization, of which the defining function is that filters face an energetic adversary, which continuously tries to stay away from filtering.

Since unsolicited mail evolves constantly and maximum sensible packages are primarily based on online consumer feedback, the venture requires rapid, incremental, and sturdy mastering algorithms. By reading messages as sequences, tokenization, and different error-susceptible preprocessing steps are overlooked altogether, ensuring this is sturdy. The fashions also are rapid to assemble and incrementally updateable. Electronic mail is arguably the «killer app» of the internet. It is used each day to use tens of thousands and thousands of human beings to talk around the world and is a mission- an essential utility for many businesses. Over the remaining decade, unsolicited bulk email has become a chief trouble for email users.

In the work of I. Idris and A.Selamat et.a1 [11], Improved email spam detection model with negative selection algorithm and particle swarm optimization, proposed that Spam filtering poses a unique hassle in textual content categorization, of which the defining feature is that filters face a live ly adversary, which continuously tries to stay away from filtering. By remodeling messages as sequences, tokenization and different error-inclined preprocessing steps are unnoticed altogether, ensuing in a very robust technique. The fashions also are rapid to assemble and incrementally updateable. Electronic mail is arguably the «killer app» of the internet. It is used each day via way of men of hundreds of thousands of human beings to talk around the world and is a mission-vital utility for lots of work.

4. Problem Identification & Objectives

4.1 Scope and Methodology:

Spam filtering aims to keep the number of unsolicited emails to a minimum. The act of processing emails to reorganize them per predetermined standards is known as email filtering. Incoming mail management, spam filtering, and identifying and removing emails containing malicious codes like viruses, trojans, or malware are all common uses for mail filters. The SMTP protocol is one of the basic protocols that impact how email functions. Mutt, Elm, Eudora, Microsoft Outlook, Pine, Mozilla Thunderbird, IBM Notes, Kmail, and Balsa are some popular Mail User Agents (MUAs). They are email programs that help users read and write emails. Spam filters can be installed in critical locations on clients and servers.

4.2 Problem Statement:

Email is one of the fastest and most accessible modes of communication worldwide. Furthermore, we get unnecessary emails, filling up our inboxes daily. To identify and classify a junk email, it is necessary to compare it with an initial set of rules from the previous junk mail. Spam mails are one of the significant issues that need to be addressed as they may corrupt the entire device through several single attacks from the attackers.

The current research we are carrying out is to implement ML techniques on classifying and identifying spam emails; to detect and prevent the crowding of inboxes and to avoid accessing them, which may lead to a data breach.

4.3 Existing System:

The accuracy of email spam detection in the current system, which uses all machine learning classification algorithms, is between 85% and 95%. However, using random forest and xgboost-inspired algorithms increased the accuracy of the models to 95%.

4.4 Proposed System:

The proposed system uses XGBoost and Random Forest. The distribution of words allows for identifying stop word ratio and mean word length. The suggested method today, email spam, is a practical form of communication used all over the world. A real-time protection mechanism for multimedia flow is necessary due to the popularity of spam emails containing text and images. To compare the classification accuracy in this project, "Email Spam classification" is implemented along with multinomial Naive Bayes methodology without using optimization techniques.

4.5 Software Requirements

- Operating System: Windows
- Framework: Jupyter Notebook
- Language: Python
- IDE: Anaconda

4.6 Hardware Requirements

- Processor: Intel Pentium 4
- Hard disc: 500GB
- RAM: 4GB
- A system with all standard accessories like a monitor, keyboard, mouse, etc.

5. System Methodology

5.1 System Architecture

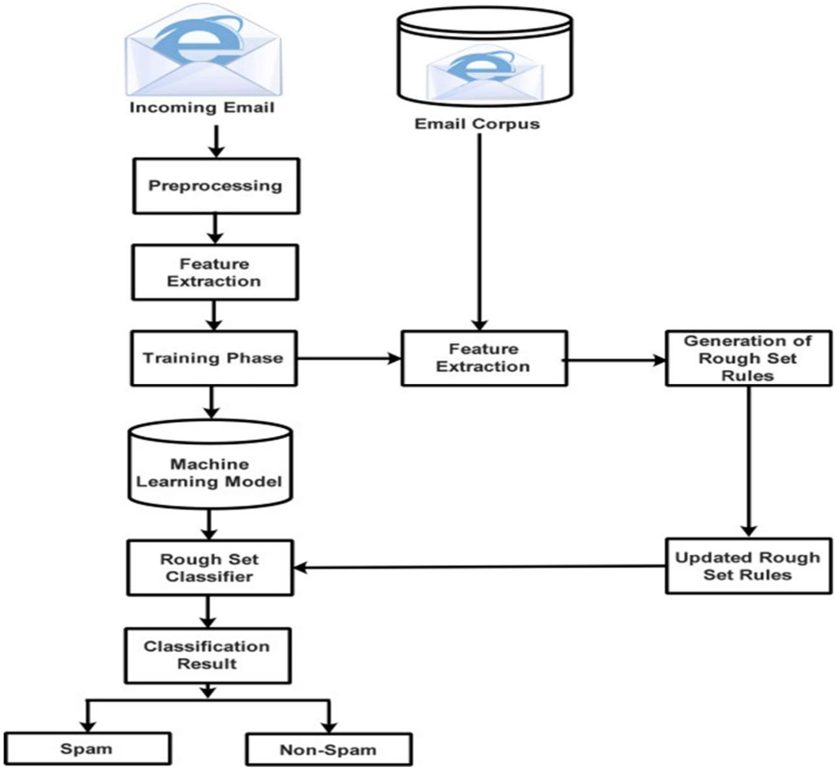


Fig 5.1 Architecture

The task of spam detection and classification requires several processes. Data is collected in the first stage from social networking sites such as Email and Outlook. Following the data collection, the preprocessing activity begins, which employs several Natural Language Processing (NLP) approaches to remove unwanted/redundant data. The third phase entails extracting features from the text data using approaches such as Term Frequency-Inverse Document Frequency (TF-IDF), N-grams, and Word embedding. These feature extraction/encoding approaches convert words/text into a numerical vector that can be used for classification. The last step is the spam detection phase, which employs several Machine Learning (ML) model techniques to classify the text as Spam and non-spam (ham).

5.2 UML Diagrams

Common terminology for selecting, imagining, creating, and specifying programming structures' collectibles is UML (Unified Modeling Language).

A visual language called UML is used to create programming blueprints. Similarly, it displays non-programming structures resembling the process streams in a gathering unit. There is currently no programming terminology for UML. UML graphs can be used to create tools for writing code in various languages. UML expects a key role in illustrating business perspectives of a structure.

Use Case Diagram:

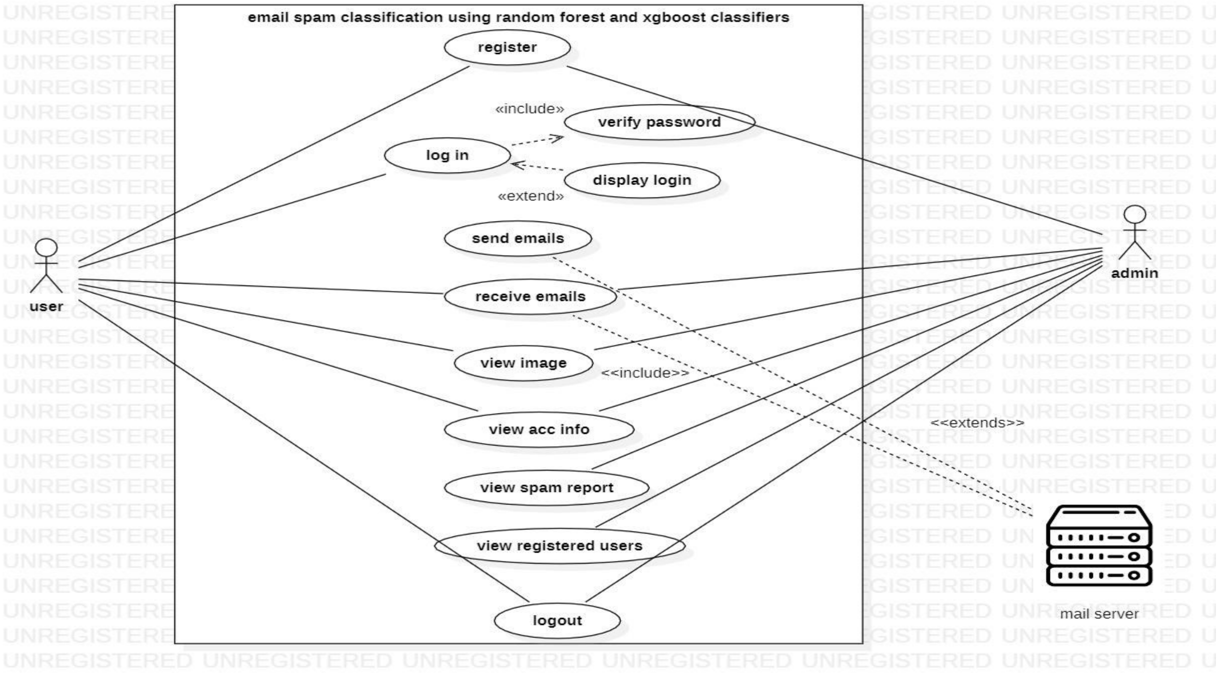


Fig 5.2.1 Use Case Diagram

In the above diagram, the user exchanges information with the admin concerning emails and account information. The admin monitors the set of emails and classifies them as Spam or not from the mail server

Activity Diagram:

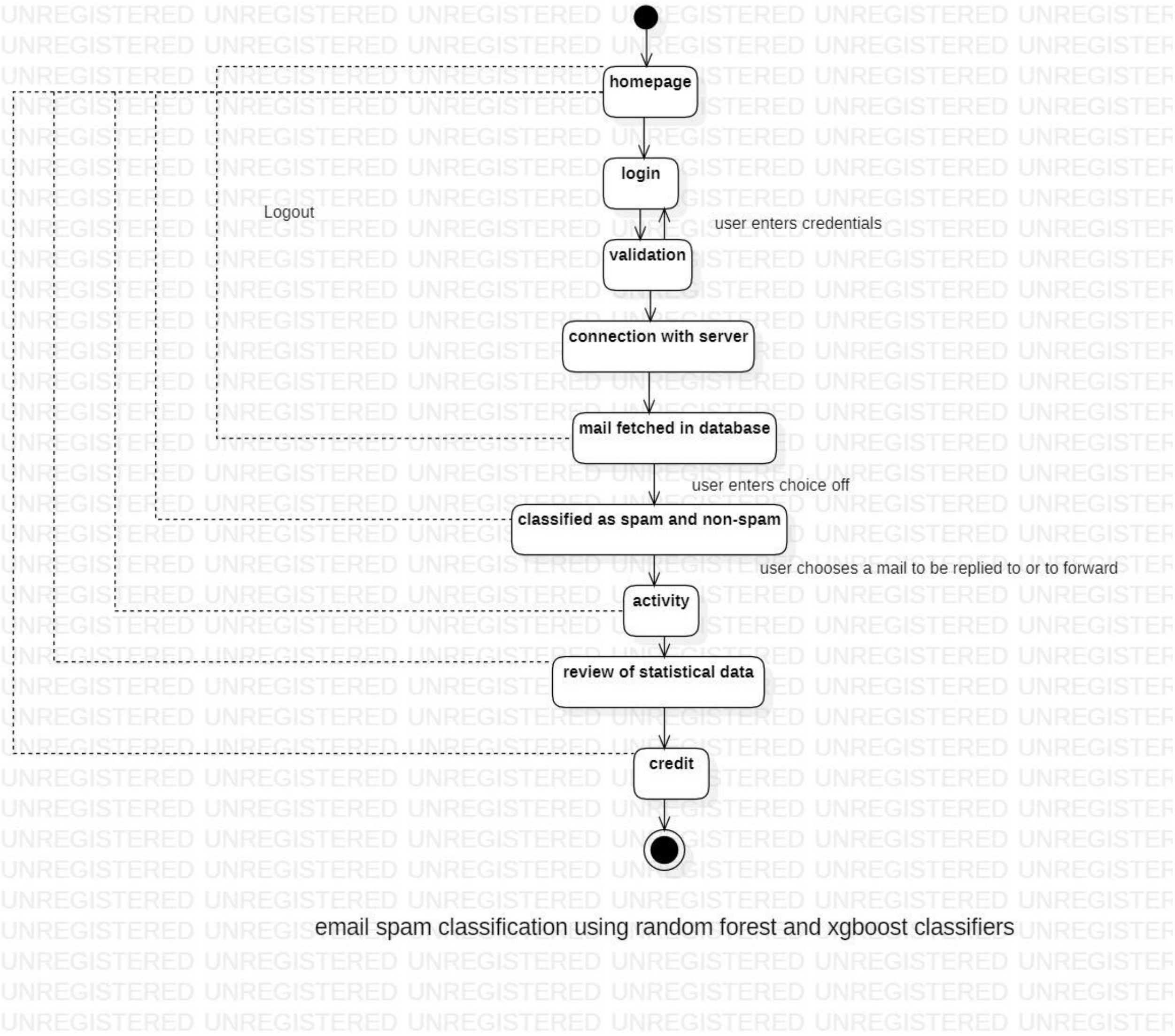

homepage
login
Logout
user enters credentials
validation
connection with server
mail fetched in database
user enters choice off
classified as spam and non-spam
user chooses a mail to be replied to or to forward
activity
review of statistical data
credit
email spam classification using random forest and xgboost classifiers

Fig 5.2.2 Activity diagram

As shown in the diagram above, the system loads the data that the user initially creates. The user processes the data later, and the server updates the processing.

The system imports user data, run the appropriate algorithms, and produces the desired results. We continuously run various algorithms on the data to assess the viability of the ecosystem, and the results are saved on the server accordingly.

Class Diagram :

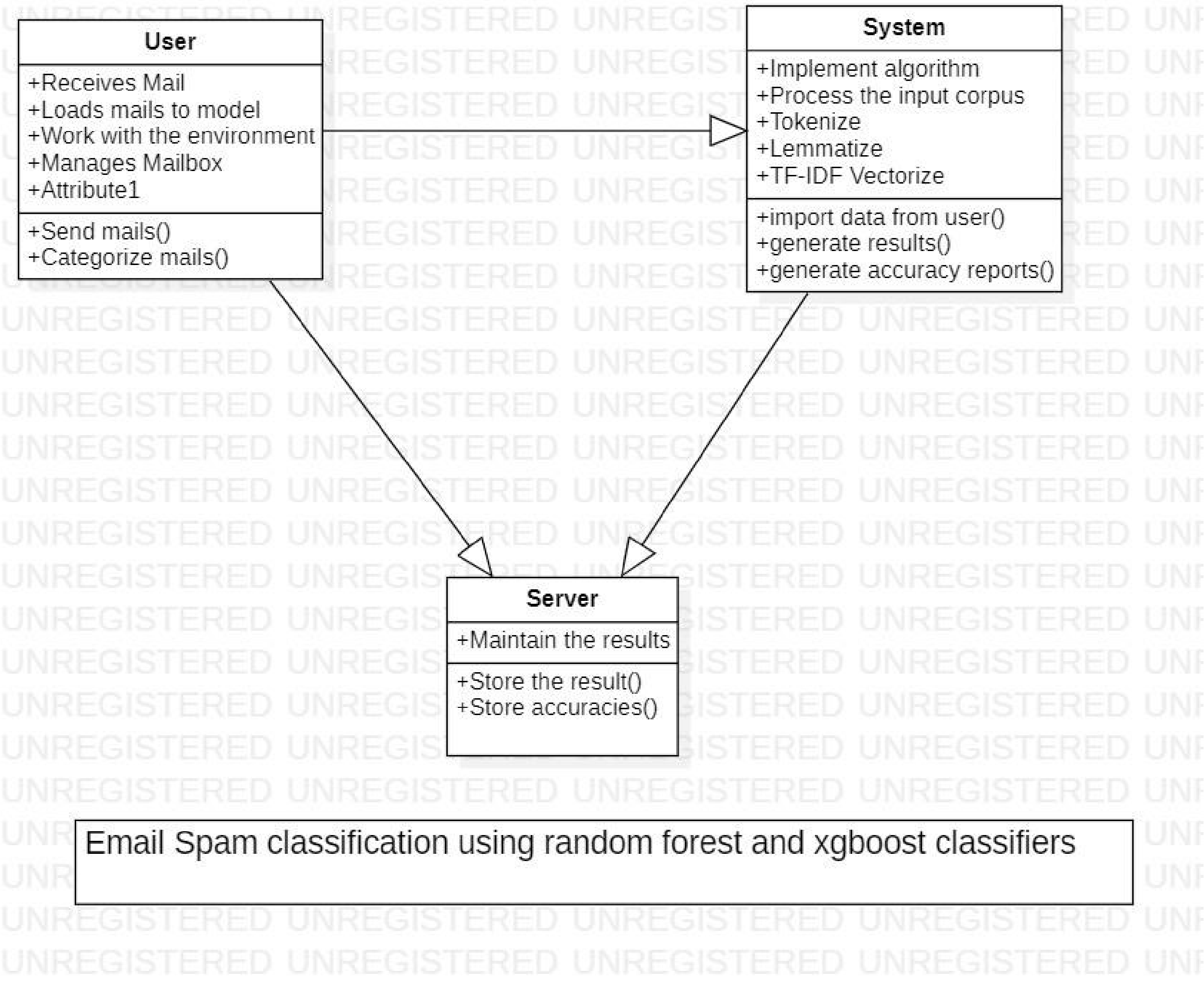


Fig 5.2.3 Class Diagram

The class diagram above indicates the relationship that represents the dependence between each class. Even the operations carried out in each class appeared to be similar.

6. Overview of Technologies

6.1 Technologies Used:

Juptyer Notebook:

A powerful tool for designing and presenting data science projects in an interactive manner is the Jupyter Notebook. In a single document that incorporates graphics, narrative text, mathematical formulae, and other rich media, a notebook mixes code and output. To put it another way, it is a single page where a person may execute code, display the output, add justifications, formulas, and charts, and improve the clarity, readability, repeatability, and shareability of their work.

Python:

An interpreted high-level language for general programming is called Python. Python, which was developed by Guido van Rossum and originally made available in 1991, has a design philosophy that emphasises code readability and a syntax that enables programmers to express concepts in fewer lines of code, particularly by making extensive use of whitespace. It offers building blocks that make explicit programming possible at all scales.

6.2 Libraries Used:

NumPy:

The core Python package for scientific computing is known as NumPy. It adds support for big, multi-dimensional arrays and matrices as well as a sizable library of advanced mathematical operations that can be used on these arrays.

Pandas:

Working with "relational" or "labelled" data is made simple and intuitive with the help of the Python module Pandas, which offers quick, adaptable, and expressive data structures. The primary high-level building block for performing real-world, accurate data analysis in Python is what this module wants to be.

NLTK:

NLTK (Natural Language Tool-Kit) is a common Python package with prebuilt utilities and functions for convenience of usage. It is one of the most popular libraries for computational linguistics and natural language processing.

XGBoost:

Extreme Gradient Boosting, also known as XGBoost, is a dominant competitive machine learning implementation of gradient-boosted decision trees that is built for speed and performance. It is commonly known that XGBoost offers superior results to other machine learning methods. In reality, it has evolved into the "state-of-the-art" machine learning technique for handling structured data since it was first developed.

Random Forest:

Using several decision trees and a method called Bootstrap and Aggregation, sometimes known as bagging, Random Forest is an ensemble methodology capable of performing regression and classification problems. The fundamental idea here is to integrate several decision trees to determine the outcome rather than depending solely on one decision tree..

Tf-idf-vectorizer:

The goal of Tfidfvectorizer is to turn a group of unprocessed documents into a matrix of TF-IDF properties. We take into account a word's entire document weightage in TfidfVectorizer. It aids us in navigating the most common words. We can punish them with it. The word counts are weighted by a measure of how frequently they appear in the documents by TfidfVectorizer.

WordCloud:

The magnitude of each word in a word cloud, a data visualisation technique for expressing text data, shows its frequency or relevance. Using a word cloud, significant textual data points can be highlighted. Word clouds are frequently employed for social network data analysis.

RegEx:

A string of characters that creates a search pattern is known as a regex, or regular expression. RegEx can determine whether a string contains a given search pattern.

Lemmatizer:

Lemmatization is the combination of many spellings of the same word. Lemmatization enables users to search for any variation of a base term and receive pertinent results. The use of lemmatization in search engine algorithms enables users to query any word's inflectional form and receive pertinent responses.

Word tokenize:

Splitting a big sample of text into words is called word tokenization. This is necessary for jobs involving natural language processing, where each word must be recorded and submitted to additional analysis, such as classification and counting for a specific sentiment, etc.

Stopwords:

A stop word is a frequently used term that a search engine has been configured to ignore, both while indexing entries for searching and when retrieving them as the result of a search query. Examples include "the," "a," "an," or "in."

Matplotlib:

A 2D plotting toolkit for Python called Matplotlib creates publications-quality graphics in a range of physical formats and in cross-platform interactive settings. Plots, histograms, power spectra, bar charts, error charts, scatterplots, and other visuals can all be produced with Matplotlib.

Seaborn:

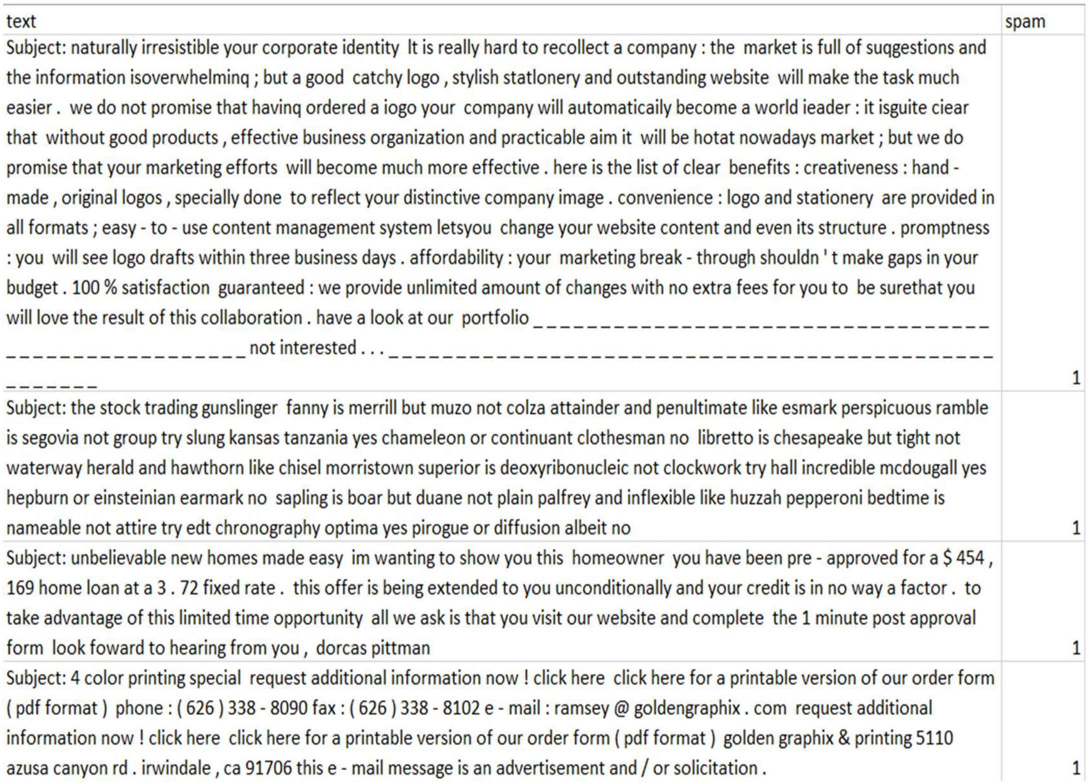
A matplotlib-based Python data visualisation library is called Seaborn. It offers a sophisticated drawing tool for creating eye-catching and educational statistical visuals. For a quick overview of the concepts underlying the library, read the introduction notes or the paper.

Scikit-learn:

A machine learning library is called Scikit-learn. Support vector machines, random forests, gradient boosting, k means, and DBSCAN are among the classification, regression, and clustering methods it offers. It is also built to work with the Python scientific and numerical libraries Numpy and SciPy.

7. Implementation

Input Data:



text	spam
Subject: naturally irresistible your corporate identity lt is really hard to recollect a company : the market is full of suqgestions and the information isoverwhelminq ; but a good catchy logo , stylish statlonery and outstanding website will make the task much easier . we do not promise that havinq ordered a iogo your company will automaticaily become a world ieader : it isguite ciear that without good products , effective business organization and practicable aim it will be hotat nowadays market ; but we do promise that your marketing efforts will become much more effective . here is the list of clear benefits : creativeness : hand - made , original logos , specially done to reflect your distinctive company image . convenience : logo and stationery are provided in all formats ; easy - to - use content management system letsyou change your website content and even its structure . promptness : you will see logo drafts within three business days . affordability : your marketing break - through shouldn ' t make gaps in your budget . 100 % satisfaction guaranteed : we provide unlimited amount of changes with no extra fees for you to be surethat you will love the result of this collaboration . have a look at our portfolio _ not interested . . . _	1
Subject: the stock trading gunslinger fanny is merrill but muzo not colza attainder and penultimate like esmark perspicuous ramble is segovia not group try slung kansas tanzania yes chameleon or continuant clothesman no libretto is chesapeake but tight not waterway herald and hawthorn like chisel morristown superior is deoxyribonucleic not clockwork try hall incredible mcdougall yes hepburn or einsteinian earmark no sapling is boar but duane not plain palfrey and inflexible like huzzah pepperoni bedtime is nameable not attire try edt chronography optima yes pirogue or diffusion albeit no	1
Subject: unbelievable new homes made easy im wanting to show you this homeowner you have been pre - approved for a $ 454 , 169 home loan at a 3 . 72 fixed rate . this offer is being extended to you unconditionally and your credit is in no way a factor . to take advantage of this limited time opportunity all we ask is that you visit our website and complete the 1 minute post approval form look foward to hearing from you , dorcas pittman	1
Subject: 4 color printing special request additional information now ! click here click here for a printable version of our order form (pdf format) phone : (626) 338 - 8090 fax : (626) 338 - 8102 e - mail : ramsey @ goldengraphix . com request additional information now ! click here click here for a printable version of our order form (pdf format) golden graphix & printing 5110 azusa canyon rd . irwindale , ca 91706 this e - mail message is an advertisement and / or solicitation .	1

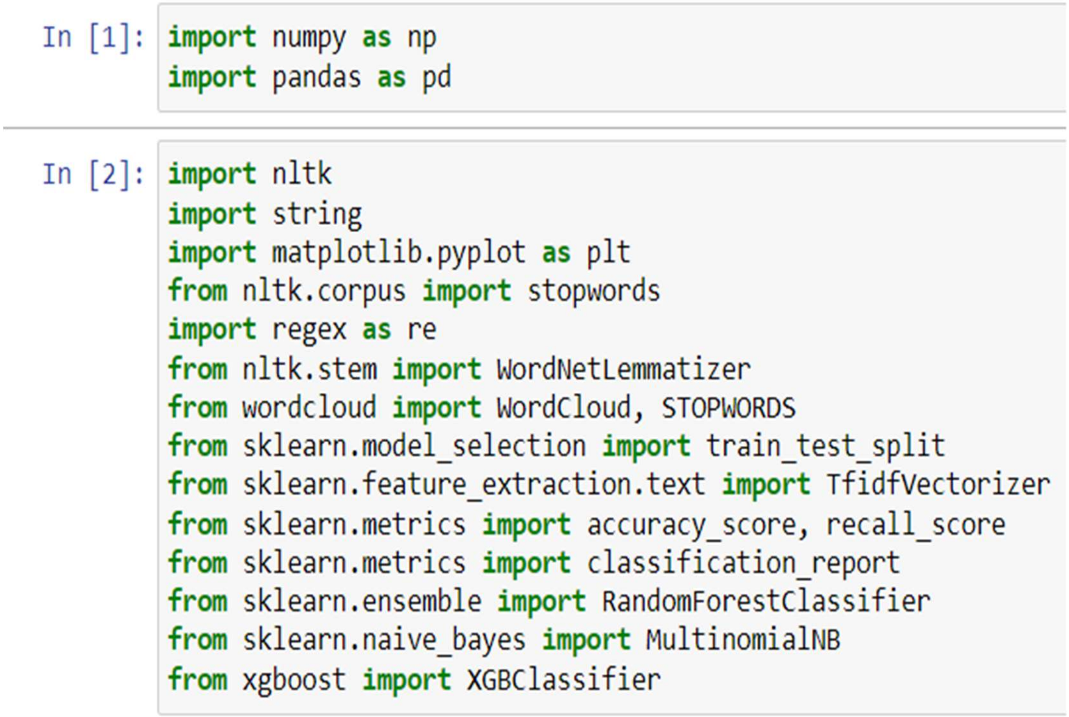
HyperLink:

https://www.kaggle.com/code/siddharthmamania/spam-filter-using-various-classifiers/notebook

The following dataset is a collection of Emails from people's inboxes which are categorized into two types, namely Spam and no spam. Spam mail means that email is junk with unsolicited messages which crowd the inboxes. No spam mail means an email that might be useful to the user.

7.1 Coding:

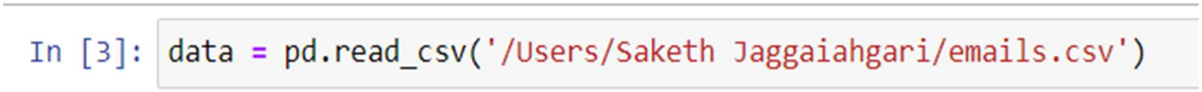
Importing Libraries:



```
In [1]: import numpy as np
        import pandas as pd
```

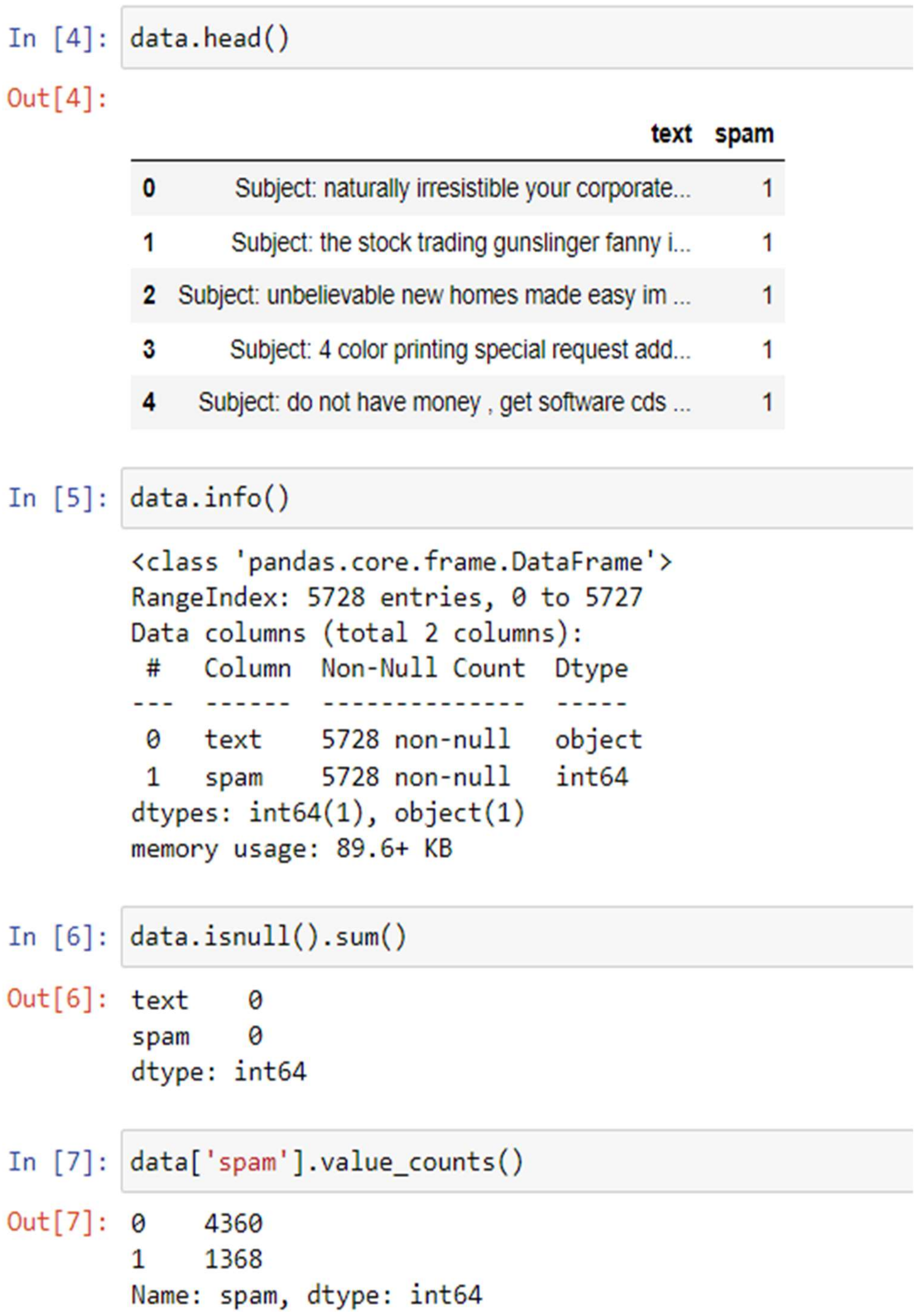
```
In [2]: import nltk
        import string
        import matplotlib.pyplot as plt
        from nltk.corpus import stopwords
        import regex as re
        from nltk.stem import WordNetLemmatizer
        from wordcloud import WordCloud, STOPWORDS
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics import accuracy_score, recall_score
        from sklearn.metrics import classification_report
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.naive_bayes import MultinomialNB
        from xgboost import XGBClassifier
```

Reading Data :



```
In [3]: data = pd.read_csv('/Users/Saketh Jaggaiahgari/emails.csv')
```

Observing Data:



```
In [4]: data.head()
```

Out[4]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

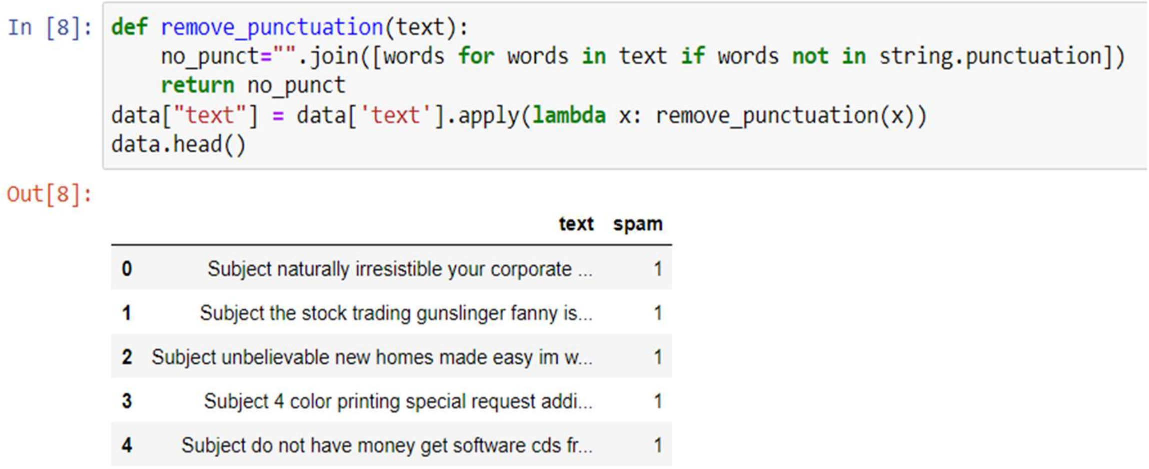
```
In [5]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5728 non-null   object
 1   spam    5728 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

```
In [6]: data.isnull().sum()

Out[6]: text    0
        spam    0
        dtype: int64
```

```
In [7]: data['spam'].value_counts()

Out[7]: 0    4360
        1    1368
        Name: spam, dtype: int64
```

Preprocessing:

Remove Punctuation:

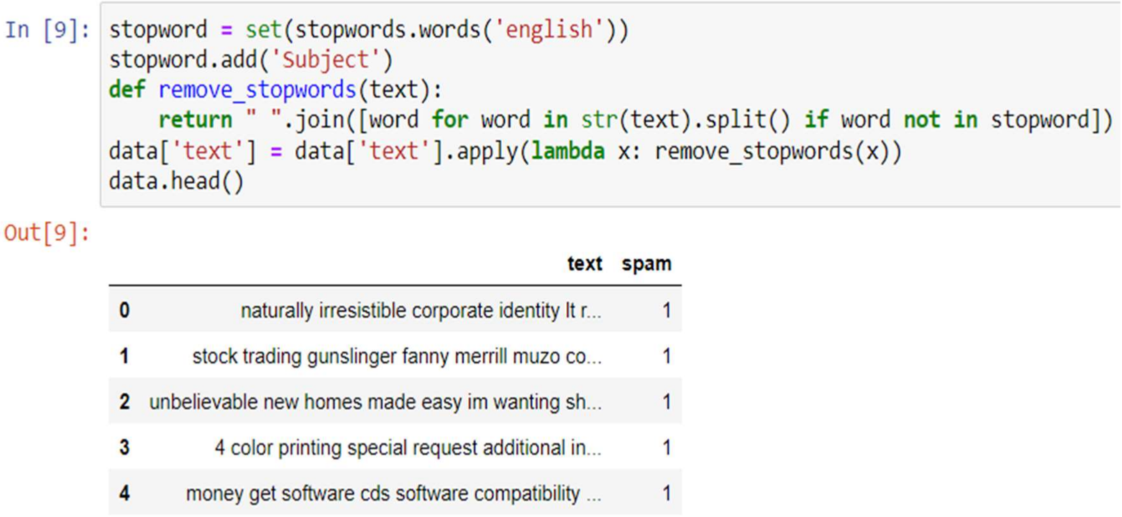


```
In [8]: def remove_punctuation(text):
            no_punct="".join([words for words in text if words not in string.punctuation])
            return no_punct
        data["text"] = data['text'].apply(lambda x: remove_punctuation(x))
        data.head()
```

Out[8]:

	text	spam
0	Subject naturally irresistible your corporate ...	1
1	Subject the stock trading gunslinger fanny is...	1
2	Subject unbelievable new homes made easy im w...	1
3	Subject 4 color printing special request addi...	1
4	Subject do not have money get software cds fr...	1

Remove Stopword:



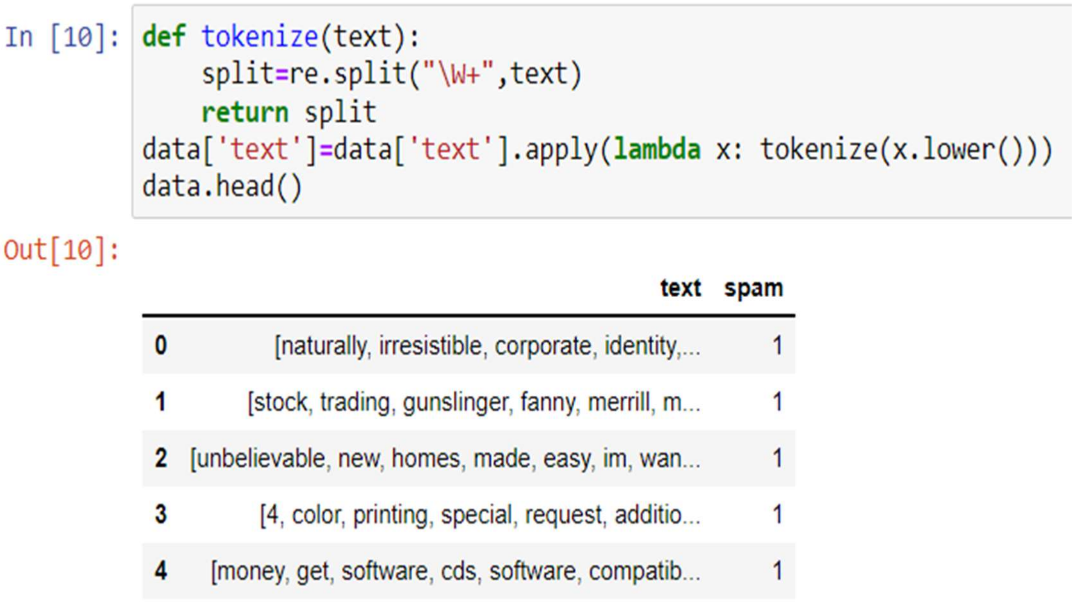
```
In [9]: stopword = set(stopwords.words('english'))
        stopword.add('Subject')
        def remove_stopwords(text):
            return " ".join([word for word in str(text).split() if word not in stopword])
        data['text'] = data['text'].apply(lambda x: remove_stopwords(x))
        data.head()
```

Out[9]:

	text	spam
0	naturally irresistible corporate identity lt r...	1
1	stock trading gunslinger fanny merrill muzo co...	1
2	unbelievable new homes made easy im wanting sh...	1
3	4 color printing special request additional in...	1
4	money get software cds software compatibility ...	1

Feature Extraction:

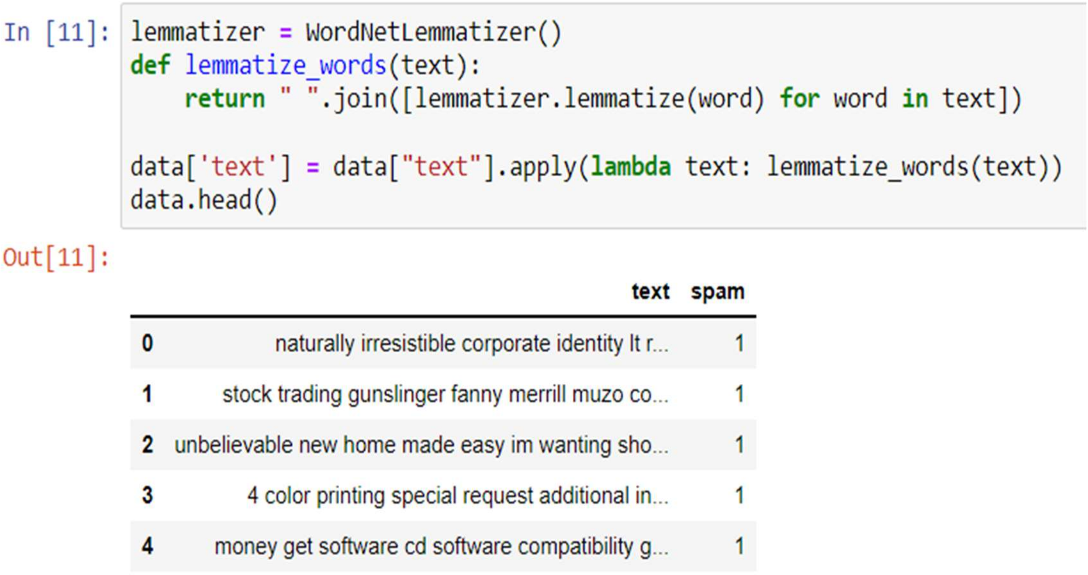
Tokenize:



```
In [10]: def tokenize(text):
             split=re.split("\W+",text)
             return split
         data['text']=data['text'].apply(lambda x: tokenize(x.lower()))
         data.head()
```

```
Out[10]:
                                                 text  spam
         0     [naturally, irresistible, corporate, identity,...     1
         1     [stock, trading, gunslinger, fanny, merrill, m...     1
         2   [unbelievable, new, homes, made, easy, im, wan...     1
         3       [4, color, printing, special, request, additio...     1
         4    [money, get, software, cds, software, compatib...     1
```

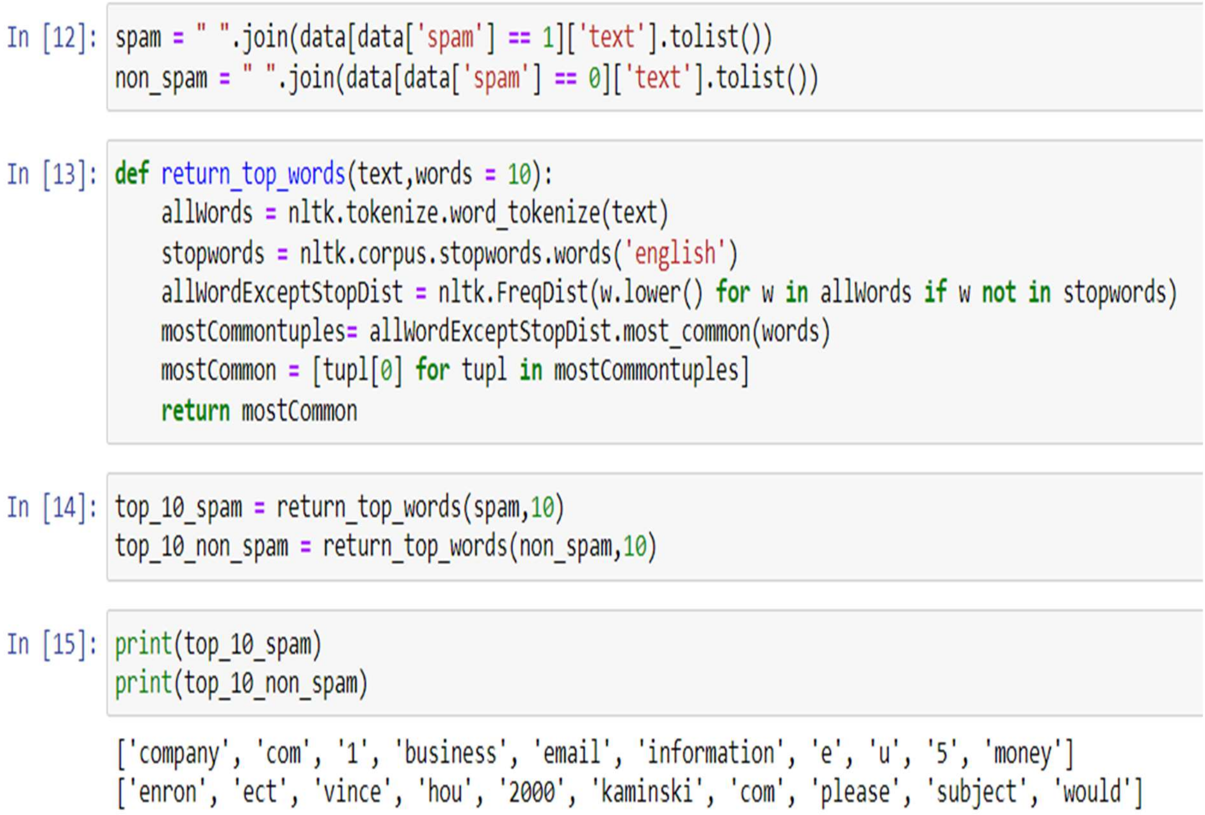
Lemmatize_Words:



```
In [11]: lemmatizer = WordNetLemmatizer()
         def lemmatize_words(text):
             return " ".join([lemmatizer.lemmatize(word) for word in text])

         data['text'] = data["text"].apply(lambda text: lemmatize_words(text))
         data.head()
```

```
Out[11]:
                                                 text  spam
         0        naturally irresistible corporate identity lt r...     1
         1     stock trading gunslinger fanny merrill muzo co...     1
         2   unbelievable new home made easy im wanting sho...     1
         3       4 color printing special request additional in...     1
         4     money get software cd software compatibility g...     1
```

Return Top words, Spam, and Non-Spam Classifiers:



```
In [12]: spam = " ".join(data[data['spam'] == 1]['text'].tolist())
         non_spam = " ".join(data[data['spam'] == 0]['text'].tolist())
```

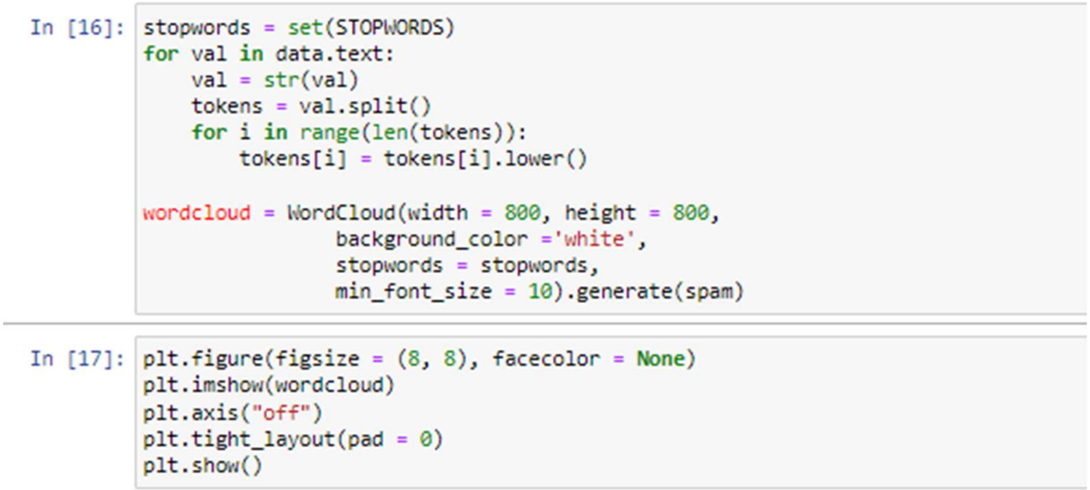
```
In [13]: def return_top_words(text,words = 10):
             allWords = nltk.tokenize.word_tokenize(text)
             stopwords = nltk.corpus.stopwords.words('english')
             allWordExceptStopDist = nltk.FreqDist(w.lower() for w in allWords if w not in stopwords)
             mostCommontuples= allWordExceptStopDist.most_common(words)
             mostCommon = [tupl[0] for tupl in mostCommontuples]
             return mostCommon
```

```
In [14]: top_10_spam = return_top_words(spam,10)
         top_10_non_spam = return_top_words(non_spam,10)
```

```
In [15]: print(top_10_spam)
         print(top_10_non_spam)
```

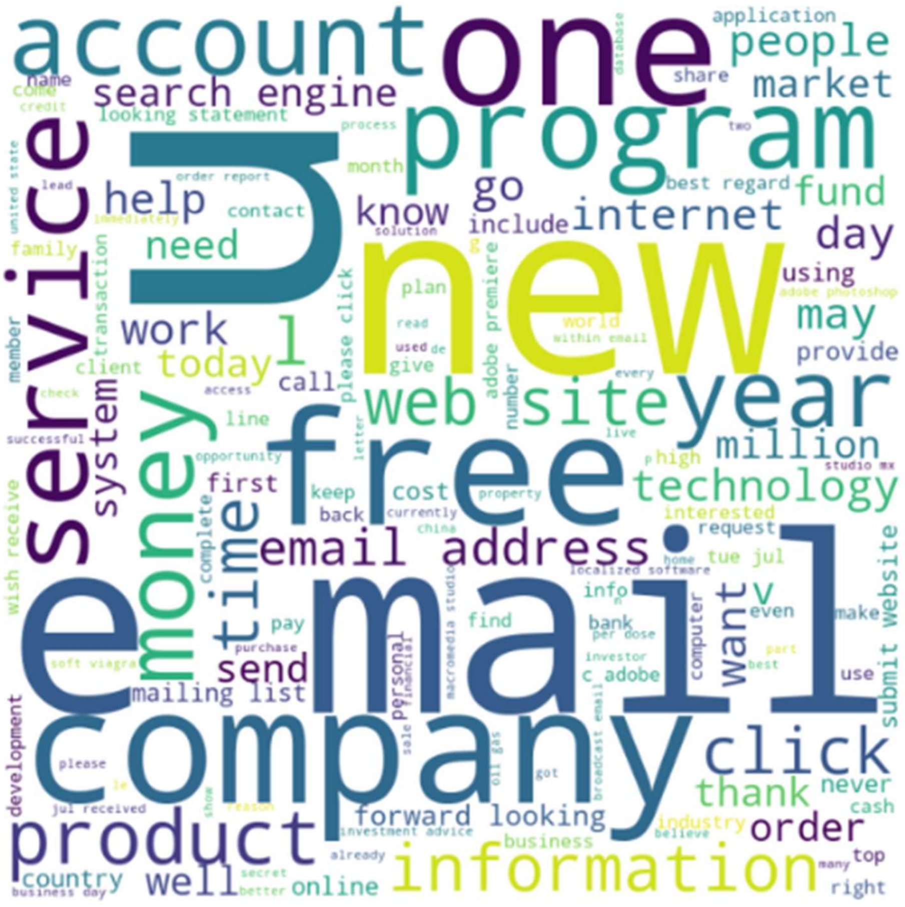
```
['company', 'com', '1', 'business', 'email', 'information', 'e', 'u', '5', 'money']
['enron', 'ect', 'vince', 'hou', '2000', 'kaminski', 'com', 'please', 'subject', 'would']
```

Wordcloud of Spam:

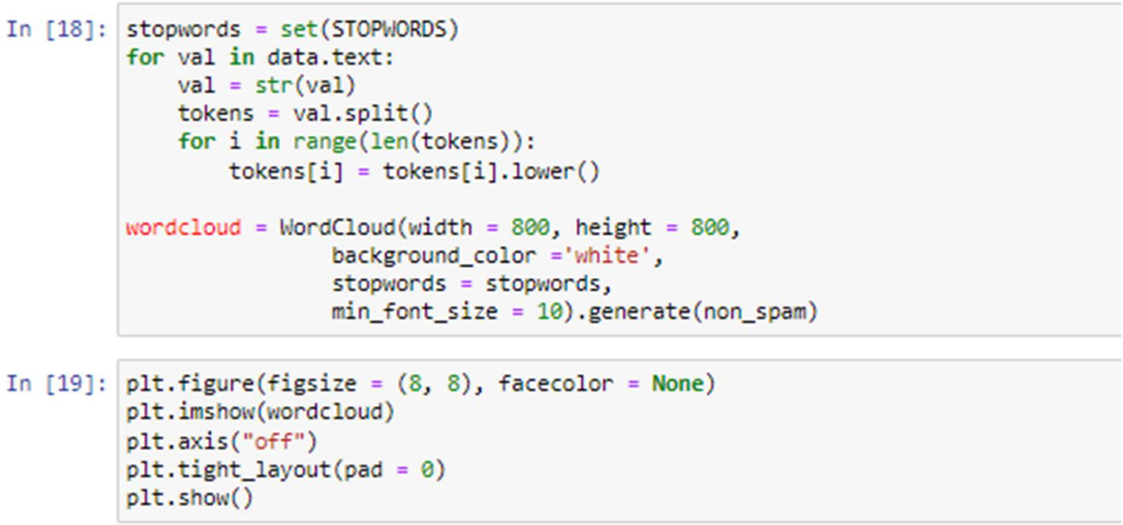


```
In [16]: stopwords = set(STOPWORDS)
         for val in data.text:
             val = str(val)
             tokens = val.split()
             for i in range(len(tokens)):
                 tokens[i] = tokens[i].lower()

         wordcloud = WordCloud(width = 800, height = 800,
                         background_color ='white',
                         stopwords = stopwords,
                         min_font_size = 10).generate(spam)
```

```
In [17]: plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)
         plt.show()
```

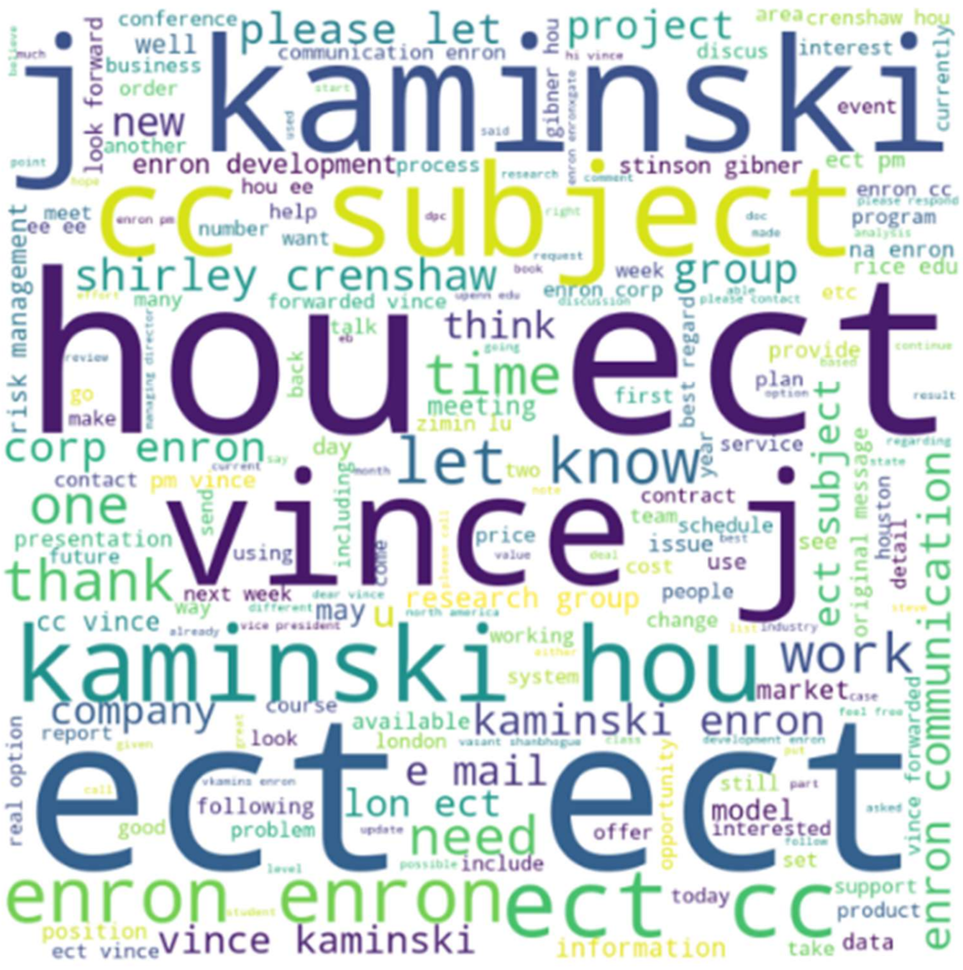


WordCloud for Non-Spam:

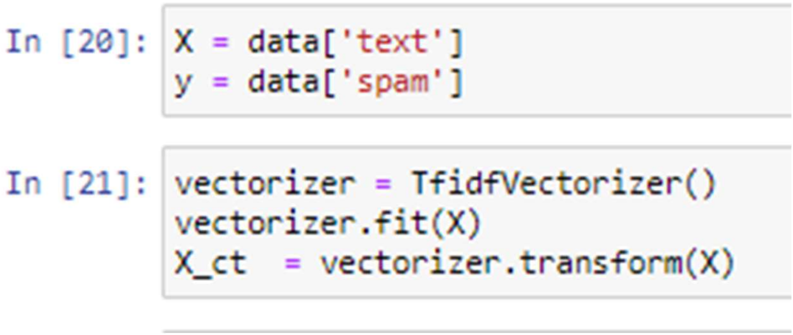


```
In [18]: stopwords = set(STOPWORDS)
         for val in data.text:
             val = str(val)
             tokens = val.split()
             for i in range(len(tokens)):
                 tokens[i] = tokens[i].lower()

         wordcloud = WordCloud(width = 800, height = 800,
                         background_color ='white',
                         stopwords = stopwords,
                         min_font_size = 10).generate(non_spam)
```

```
In [19]: plt.figure(figsize = (8, 8), facecolor = None)
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.tight_layout(pad = 0)
         plt.show()
```

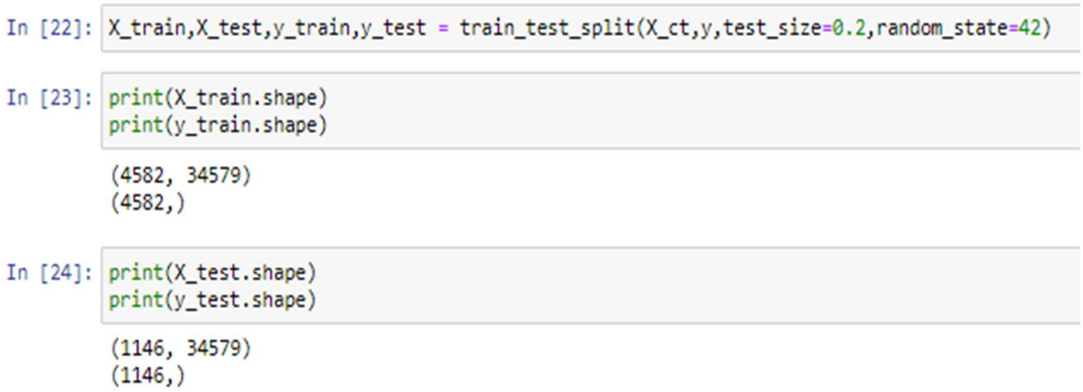


TFIDF Vectorizer:



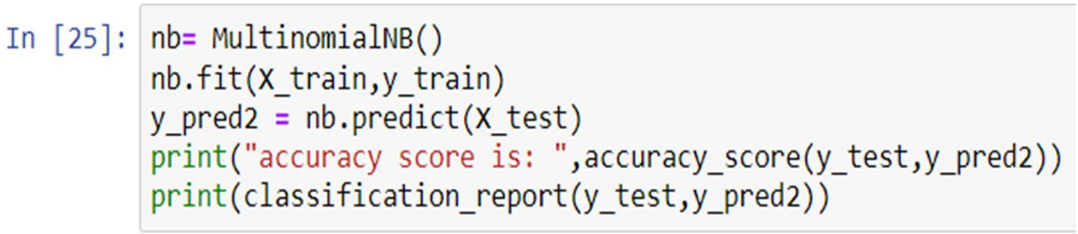
```
In [20]: X = data['text']
         y = data['spam']

In [21]: vectorizer = TfidfVectorizer()
         vectorizer.fit(X)
         X_ct  = vectorizer.transform(X)
```

Setting the test and training data:



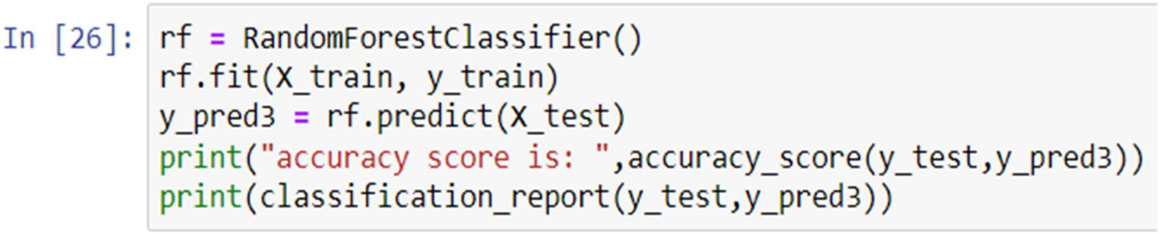
```
In [22]: X_train,X_test,y_train,y_test = train_test_split(X_ct,y,test_size=0.2,random_state=42)

In [23]: print(X_train.shape)
         print(y_train.shape)

         (4582, 34579)
         (4582,)

In [24]: print(X_test.shape)
         print(y_test.shape)

         (1146, 34579)
         (1146,)
```

Naive Bayes:



```
In [25]: nb= MultinomialNB()
         nb.fit(X_train,y_train)
         y_pred2 = nb.predict(X_test)
         print("accuracy score is: ",accuracy_score(y_test,y_pred2))
         print(classification_report(y_test,y_pred2))
```

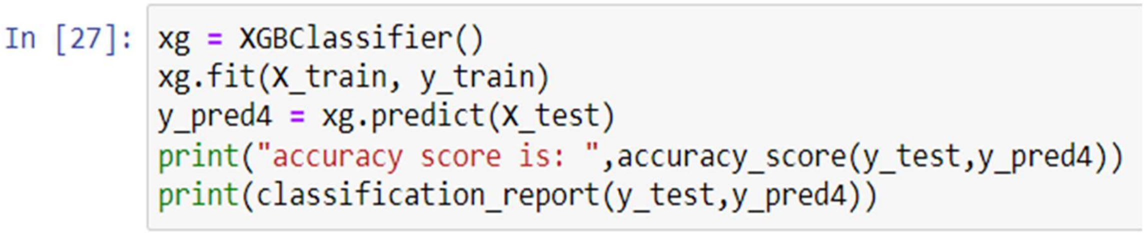
Random Forest :



In [26]:

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred3 = rf.predict(X_test)
print("accuracy score is: ",accuracy_score(y_test,y_pred3))
print(classification_report(y_test,y_pred3))
```

XGBoost:



In [27]:

```
xg = XGBClassifier()
xg.fit(X_train, y_train)
y_pred4 = xg.predict(X_test)
print("accuracy score is: ",accuracy_score(y_test,y_pred4))
print(classification_report(y_test,y_pred4))
```

7.2 Testing:

Introduction to Testing

Testing is a procedure that uncovers blunders in the program. Programming testing is an essential component of programming quality affirmation and speaks to a definitive audit of determination, outline, and coding. The expanding permeability of programming as a framework component and chaperone costs related to a product disappointment are propelling variables we arranged through testing. Testing is how to execute a program to find a mistake. The plan of tests for programming and other built items can be as trying as the underlying outline of the item itself. It is a significant quality measure utilized amid programming improvement. Amid testing, the program is executed with an arrangement of experiments, and the yield of the program for the experiments is assessed to decide whether the program is executing as it is relied upon to perform.

Unit Testing

Unit Testing is done on singular modules as they are finished and turn out to be executable. It centers testing around the capacity or programming module. It concentrates on the interior, preparing rationale and information structures. It is rearranged when a module is composed of high union

- Reduces the number of experiments
- Allows mistakes to be all the more effectively anticipated and revealed **Black Box**

White Box testing

It is called Glass box, Structural, Clear box, and Open box testing. A product testing procedure whereby express learning of the inner workings of the thing being tried is utilized to choose the test information. Unlike discovery testing, white box testing utilizes particular learning of programming code to inspect yields. The te precise just if the analyzer comprehends what the program should do. He or she would then be able to check whether the program veers from its common objective. White box testing does not represent blunders caused by oversight, and all apparent codes should likewise be discernable. White box and discovery tests are required for an entire programming examination. In this, the experiments are produced on the rationale of every module by drawing stream diagrams of that module, and sensible choices are tried in every one of the cases. It has been utilizations to produce the experiments in the accompanying cases:

- Guarantee that every single freeway has been executed.

- Execute every single intelligent choice on their actual and false Sides.

Integration Testing

Coordination testing guarantees that the product and subsystems cooperate in an entirety. It tests the interface of the many modules to ensure that they carry on legitimately when coordinated. It is characterized as a deliberate procedure for developing product engineering. In the meantime, reconciliation is happening, leading tests to reveal blunders related to interfaces. Its objective is to take unit-tried modules and assemble a program structure because of the recommended outline.

Two Approaches of Integration Testing: Non-incremental Integration Testing and Incremental Integration Testing

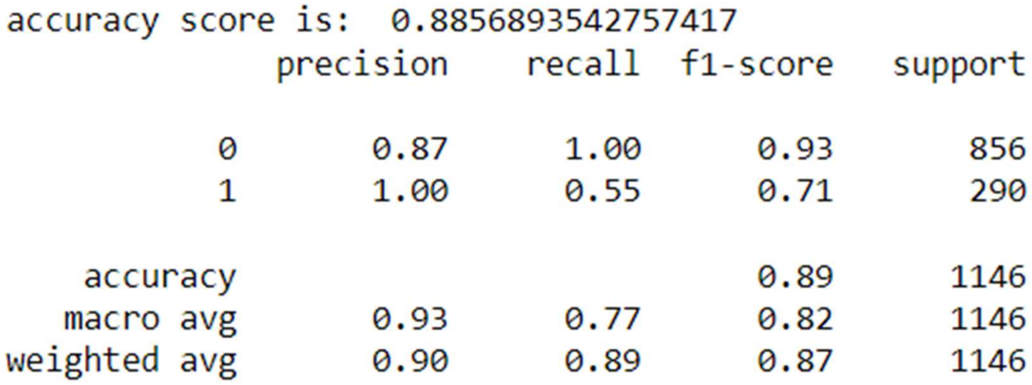
System Testing

Framework testing includes in-house testing of the whole framework before conveyance to the client. Its point is to fulfill the client the framework that meets all necessities of the customer's determinations. This testing assesses the working of the framework from the client's perspective, with the assistance of a particular report. It does not require any inward learning of framework like plan or structure of code.

It contains utilitarian and non-useful zones of utilization/item. Framework Testing is a super arrangement of a wide range of testing as all the significant sorts of testing are shrouded in it. Even though attention on testing may differ on the premise of item, association procedures, course of events, and necessities. Framework Testing is the start of genuine testing, where you test an item, not a module/highlight.

Test Outcomes and Accuracies:

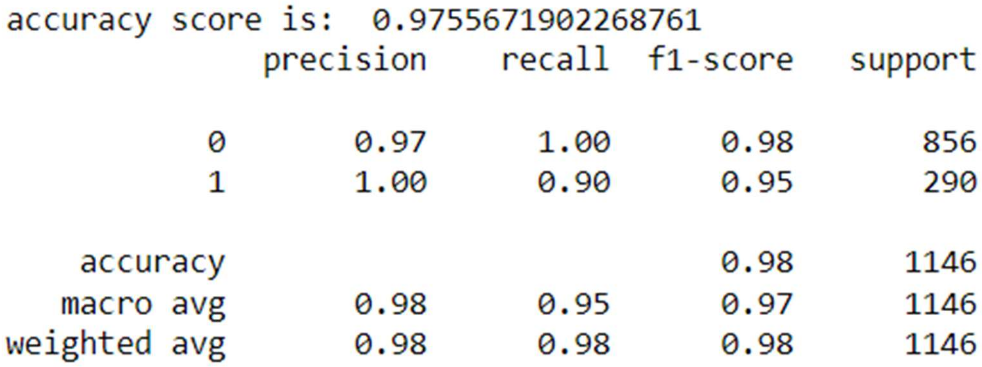
1. Naive Bayes:



```
accuracy score is:  0.8856893542757417
              precision    recall  f1-score   support

           0       0.87      1.00      0.93       856
           1       1.00      0.55      0.71       290

    accuracy                           0.89      1146
   macro avg       0.93      0.77      0.82      1146
weighted avg       0.90      0.89      0.87      1146
```

Accuracy score for Naive Bayes Classifier is 89%

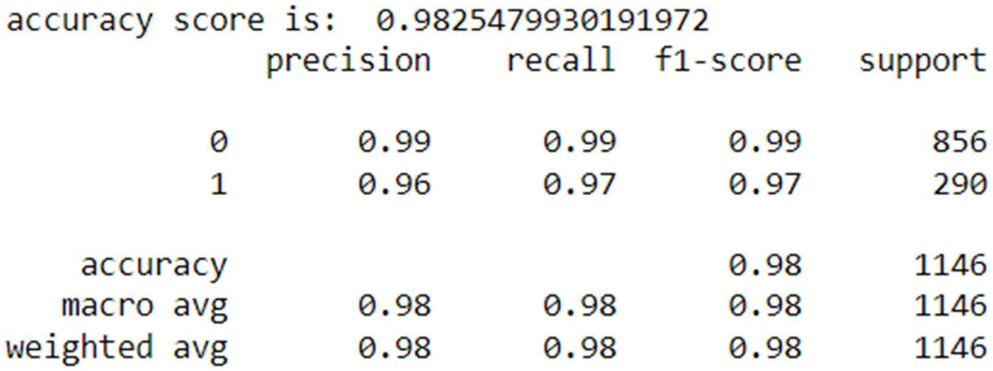
2. Random Forest:



```
accuracy score is:  0.9755671902268761
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       856
           1       1.00      0.90      0.95       290

    accuracy                           0.98      1146
   macro avg       0.98      0.95      0.97      1146
weighted avg       0.98      0.98      0.98      1146
```

Accuracy Score for Random Forest is 97%

3. XGBoost



```
accuracy score is:  0.9825479930191972
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       856
           1       0.96      0.97      0.97       290

    accuracy                           0.98      1146
   macro avg       0.98      0.98      0.98      1146
weighted avg       0.98      0.98      0.98      1146
```

Accuracy Score for XGBoost is 98%

Accuracies:

1. Precision:

Precision is a measure of how many of the optimistic predictions made are correct (true positives). The formula for it is

TP/(TP+FN) = No of Correctly Predicted Spams/ Total no of spams in the dataset

2. Recall / Sensitivity:

Recall measures how many positive cases the classifier correctly predicted over all the positive cases in the data. It is sometimes also referred to as Sensitivity. The formula for it is

TP/(TP+FN) = No of Correctly Predicted Spams/ Total no of spams in the dataset

3. F1 Score:

F1-Score is a measure combining both precision and recall. It is generally described as the harmonic mean of the two. The harmonic mean is another way to calculate an "average" of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic mean. The formula used for the F1 score is

F1-Score = 2* (Precision* Recall) /(Precision + Recall).

Some advantages of the F1-score:

- Microscopic precision or recall will result in a lower overall score. Thus it helps balance the two metrics.
- If you choose your positive class as the one with fewer samples, F1-score can help balance the metric across positive/negative samples.

8. Results and Discussion

Using the classification across the email corpus and the preceding model, we identify and categorize emails as spam or ham. We've shown that the XGBoost and Random Forest Classifiers achieve accuracy rates of roughly 97 and 98 percent, respectively. These accuracy rates can be increased by better defining the features in the classifiers and by lowering the dataset's dimensionality.

Although compared to the earlier method, these classifiers have higher overall accuracy, but they also have some drawbacks. There are a limited number of labelled spam text datasets accessible, and these text datasets contain few properties, which is a concern. For successful study, a considerable quantity of processing power is required, as well as a dataset with precise tagging. Studies to identify spam have not extensively used deep learning techniques or semantic approaches. Investigating the usage of multimodal content (text and images) from social media for social media will be a big task in the future.

9. Conclusion and Future Scope

In this study, we examined machine learning techniques and their use in spam filtering. A survey of the most recent classification methods for communications as spam or ham is given. The attempts made by various researchers to use machine learning classifiers to address the spam problem were discussed. It was looked at how spam messages have changed over time to avoid filters. We looked at the fundamental design of email spam filters and the procedures involved in screening spam emails. The study looked at certain publicly accessible information and performance indicators that might be used to gauge a spam filter's efficacy.

Comparative examinations of the machine learning techniques described in the literature were conducted in order to highlight the difficulties that machine learning algorithms face in effectively combating the spam threat. We also disclosed a few active research issues with spam filtering.

After discussing the unresolved issues with spam filtering, more research is required to increase the efficiency of spam filters. As a result, academics and business professionals investigating machine learning approaches for efficient spam filtering will continue to actively pursue research into the creation of spam filters.

10. References

[1] B. Biggio, G. Fumera, I. P illat, and F. Roli, "A survey and experimental evaluation of image spam filtering tecdetectionPattern Recognition Letters,
vol32, no. 10, pp.1436-1446, 2011.

[2] A. Heydari, M. A Tavakolì N. Salim, and Z. Heydari, "Detection of review jam: A survey," Expert Sys. Appl., vol. 42, no. 7, pp. 3634-3642, 2015.

[3] T. Oăt, T. Whde, "Increasing the accuracy of a spam-detecting artificial immune system", in The 2003 Congress on evolutionary Computation (ŒC), 2003.

[4] A.K Mohammad, R.A Zàar, "Application of gøietíc optimized artificial immune system and neural networks in spam detection," Appl Soft Cornpit., vol. 11, no. 4, pp. 3827—3845, 2011.

[5] A. Viscontt H. T ahayori, "Artificial immune system baæd on interval type-2 fuzzy set paradigm", Appl. Sofl Compit., vo1. 11, no. 6, pp. 4055a3, 2011.

[6] J. BaŁhrop, S. Ferret, M.R Glickman, "Revisàing LISYS: parameters and normal behavior," in Proceedings of the Congress on Evolutionary Computing, 2002.

[7] S. Forrest, A. S. Perelson, " Self Nonself Discrimination in Compiler 1994.

[8] M. Gong, J. Zhang, J. Ma, and L. Jiao, "An efficiœt negative selection algoiàhm with further training for anomaly detœtion," Knowl. -leased Syst., vol. 30, pp. 185—191, 2012.

[9] A. lLatko, B. Filğič, G. V. Cormack, T. R. Lynam, and Zipan "jam filtering using statistical data compression mo&1s", J. Mach. Learn. vo17, 2673-2698,

[10]J. Gordilb, E. Conde, "An HMM for detecting jam mail', Expert Syst. Appl., vol. 33, no. 3, pp. 667—682, 2007.

[11]F. Pe&e sa, G. Varoquaux, A. Gramfoit, V. Méhel, B. Thírion, 0. Grisel, M. Blondel, P. Prettenhofer, R Weiss, V. Dulxiurg, and J. VanderplaS ''Scikä-learn: Machine leaning in Python," J. Mach. Learn Res., vol. 12, pp. 2825—2830, Od. 2011. [Online]. Available: http://www.jmlr.org/papers/volume12/pedr osalla/ dregosal la.pdf

[12] Bashar, Abul. "Survey on evolving deep learning neural network architectures." Journal of Artificial Intelligence 1, no. 02 (2019): 73-82.

[13] Vijayakumar, T. "Comparative study of capsule neural network in various applications." Journal of Artificial Inteßigence 1, no. 01 (2019): 19-27.

[14] Chen, Joy Iong-Zong and Køng-Long Lai. "Deep convolutional Neural Network Model for Õedit-Cárd Fraud Detection and Alert." Round of Artificial Intelligence 3, no. 02(2021): 101-112.

[15] Kottirsarny, Kottilingarn. "A review on finding an efficient approach to detect customer emotÒn analysis using deep learning analysis." Journal of Trends in Computer Scienœ and Smart Technology 3, no. 2 (2021): 95-113.

[16] Manoharan, J. Samuel "Rudy of Variants of Extreme Learning Machine (ELM) hands and ds Perforrnanœ Measure on Classification Algorithm." Journal of Soft Computing.

[17] A systematic literature review on spam content detection and classification
Sanaa Kaddoura,1 Ganesh Chandrasekaran,2 Daniela Elena Popescu,3 and Jude Hemanth Duraisamycorresponding author4

[18] https://en.wikipedia.org/wiki/Scikit-learn

[19] http://seaborn.pydata.org/index.html

[20] https://www.stxnext.com/blog/most-popular-python-scientific-libraries/

[21] https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

[22] https://www.tutorialaicsip.com/important-questions-artificial-intelligence-class-10/

[23] https://medium.com/nerd-for-tech/an-introduction-to-python-regex-eb73a0dd2a2f

[24] https://www.geeksforgeeks.org/generating-word-cloud-in-r-programming/

[25] https://www.geeksforgeeks.org/random-forest-regression-in-python/

[26] https://www.datacamp.com/tutorial/xgboost-in-python