

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor in
Technology Degree in Information Technology

By
DIGANTA PANDA
REG. NO. 39120024



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI-600119, TAMILNADU

NOVEMBER 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **DIGANTA PANDA (Reg. No: 39120024)** who carried out the project entitled "**SPAM CLASSIFIER USING MACHINE LEARNING**" under my supervision from September 2021 to November 2021.

Internal Guide

Dr. S REVATHY, M.E., Ph.D.,

Head of the Department

Dr. R SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **DIGANTA PANDA** hereby declare that the project entitled **SPAM CLASSIFIER USING MACHINE LEARNING**” completed by me under the guidance of **Dr. S. Revathy, M.E., Ph.D**, is submitted in partial fulfilment of the requirements for the award of Bachelor in Technology Degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** at **Sathyabama Institute** for their encouragement and guidance in the completion of this project.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D. Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere gratitude to my Project Guide **Dr. S. Revathy, M.E., Ph.D**, for her valuable guidance, suggestions and constant encouragement which paved the way for the successful completion of my project work.

I wish to express my thanks to all teaching and non-teaching staff members of the **Department of Information Technology** who were helpful in many ways throughout the duration of the project.

ABSTRACT

Email and mobile messaging is the worldwide use of communication application. It is because of the ease of use and faster than other communication application.

However, its inability to detect whether the mail content is either spam or ham degrade its performance. Nowadays, lot of cases have been reported regarding stealing of personal information or phishing activities via email from the user.

This project will discuss how machine learning help in spam detection. Machine learning is an artificial intelligence application that provides the ability to automatically learn and improve data without being explicitly programmed. The objective of developing this model is to detect whether the message or email received is spam or ham and will automatically be detected and blocked.

Spam filtering is aimed at reducing to the barest minimum the volume of unsolicited emails. Email filtering is the processing of emails to rearrange it in accordance to some definite standards. Mail filters are generally used to manage incoming mails, filter spam emails, detect and eliminate mails that contain any malicious codes such as virus, trojan or malware. The workings of email is influence by some basic protocols which include the SMTP. Some of the widely used Mail User Agents (MUAs) are Mutt, Elm, Eudora, Microsoft Outlook, Pine, Mozilla Thunderbird, IBM notes, Kmail, and Balsa.

They are email clients that assists the user to read and compose emails. Spam filters can be deployed at strategic places in both clients and servers.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE No.
	ABSTRACT	v
	LIST OF FIGURES LIST OF ABBREVIATIONS LIST OF TABLES	viii
1	INTRODUCTION	1
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	3
	2.1 LITERATURE SURVEY	3
	2.2 AIM	8
3	EXPERTIMENTAL OR MATERIALS AND METHODS USED	9
	3.1 SOFTWARE REQUIRMENT	9
	3.1.1 Windows/Mac System	9
	3.1.2 Numpy	9
	3.1.3 Pandas	9
	3.1.4 Flask	10
	3.1.5 Metplotlib	10
	3.1.6 Jupyter Notebook	10
	3.1.7 Machine Learning	10
	3.1.8 Python	11
	3.2 ALGORITHMS USED	11

	3.2.1 Bayes Theorem	11
	3.2.2 Occurrence of a word	11
	3.2.3 Classification of greater	12
	3.2.4 Loading Dependencies	12
	3.2.5 Loading Data	12
	3.2.6 Train Test Split	13
	3.2.7 Visualizing Data	14
	3.2.8 Training the Model	14
	3.2.9 Stopping Words	15
	3.2.10 Additive Smoothing	16
	3.2.11 Classification	16
	3.2.12 Final Result	17
	3.3 IMPLEMENTATION AND DEPLOYMENT	17
4	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	18
	4.1 RESULT AND PERFORMANCE DISCUSSION	18
	SCREENSHOTS	18
	4.2 PERFORMANCE ANALYSIS	20
5	SUMMARY AND CONCLUSIONS	22
	5.1 SUMMARY	22
	5.2 CONCLUSIONS	22
	REFERENCES	23
	APPENDIX (source code)	24

LIST OF FIGURES

S. NO.	TITLE	PAGE NO.
1.1	Spam Detection Flow	2
3.2.4	Load Dependencies	12
3.2.7	Loading Data	12
3.2.6	Train Test Split	13
3.2.7	Visualizing Data	14
3.2.12	Final Result	17
4.1	Dataset	18
4.2	Spam Classifier running on web browser	19

LIST OF ABBREVIATIONS

ACRONYM	EXPANSION
Mac	Mac Operating System
ML	Machine Learning
AI	Artificial Intelligence
numpy	numpy python library
png	png format photo file

LIST OF TABLES

S. NO.	TITLE	PAGE NO.
4.1	Examples of Ham Vs Spam	21

CHAPTER 1

INTRODUCTION

Today, spam has become a big internet issue. Recent 2017, the statistic shown spam accounted for 55% of all e-mail messages and mobile messaging, same as during the previous year. Spam which is also known as unsolicited bulk email has led to the increasing use of email as email provides the perfect ways to send the unwanted advertisement or junk newsgroup posting at no cost for the sender. This chance has been extensively exploited by irresponsible organizations and resulting to clutter the mail boxes of millions of people all around the world.

Evolving from a minor to major concern, given the high offensive content of messages, spam is a waste of time. It also consumed a lot of storage space and communication bandwidth. End user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economical which led some countries to adopt legislation. Text classification is used to determine the path of incoming mail/message either into inbox or straight to spam folder. It is the process of assigning categories to text according to its content. It is used to organized, structures and categorize text. It can be done either manually or automatically.

Machine learning automatically classifies the text in a much faster way than manual technique. Machine learning uses pre-labelled text to learn the different associations between pieces of text and its output. It used feature extraction to transform each text to numerical representation in form of vector which represents the frequency of word in predefined dictionary. Text classification is important in the context of structuring the unstructured and messy nature of text such as documents and spam messages in a cost-effective way. A Machine learning platform has capabilities to improve the accuracy of predictions. With regard to Big Data, a Machine Learning platform has abilities to speed up analysing of gigantic data.

It is important especially to a company to analyse text data, help inform business decisions and even automate business processes. For example, text classification is used in classifying short texts such as tweets or headlines. It can be used in larger documents such as media articles. It also can be applied to social media monitoring, brand monitoring and etc. In this project, a machine learning technique is used to detect the spam message of a mail. Machine learning is where computers can learn to do something without the need to explicitly program them for the task.

A specific algorithm is used to learn the classification rules from these messages. Those algorithms are used for classification of objects of different classes. The algorithms are provided with input and output data and have a self-learning program to solve the given task. Searching for the best algorithm and model can be time consuming. The two-class classifier is best used to classify the type of message either spam or ham. This algorithm is used to predict the probability and classification of data outcome.

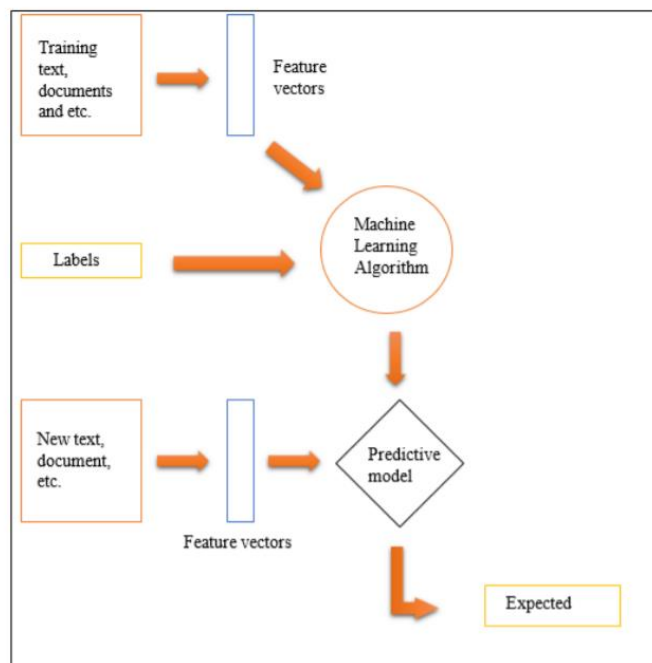


Fig 1.1 Spam detection Flow

CHAPTER-2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 LITERATURE SURVEY

2.1.1 Design and Evaluation of a Real-Time URL Spam Filtering Service

On the heels of the widespread adoption of web services such as social networks and URL shorteners, scams, phishing, and malware have become regular threats. Despite extensive research, email-based spam filtering techniques generally fall short for protecting other web services. To better address this need, we present Monarch, a real-time system that crawls URLs as they are submitted to web services and determines whether the URLs direct to spam. We evaluate the viability of Monarch and the fundamental challenges that arise due to the diversity of web service spam. We explore the distinctions between email and Twitter spam, including the abuse of public web hosting and redirector services. Finally, we demonstrate Monarch's scalability, showing our system could protect a service such as Twitter—which needs to process 15 million URLs/day—for a bit under \$800/day.

In recent years, the Internet has seen a massive proliferation of web services, including social networks, video sharing sites, blogs, and consumer review pages that draw in hundreds of millions of viewers. On the heels of the widespread adoption of these services, phishing, malware, and scams have become a regular threat. Bypassing protection mechanisms put in place by service operators, scammers are able to distribute harmful content through the use of compromised and fraudulent accounts. As spam evolves beyond email and becomes a regular nuisance of web services, new defenses must be devised to safeguard what is currently a largely unprotected space.

While email spam has been extensively researched, many of the solutions

fail to apply to web services. In particular, recent work has shown that domain and IP blacklists currently in use by social network operators and by URL shortening services perform too slowly (high latency for listing) and inaccurately for use in web services .

The architecture of Monarch consists of three core elements: a front-end that accepts URLs submitted by web services seeking a classification decision, a pool of browsers hosted on cloud infrastructure that visits URLs to extract salient features, and a distributed classification engine designed to scale to tens of millions of features that rapidly returns a decision for whether a URL leads to spam content. Classification builds upon a large foundation of spam characteristics and includes features drawn from the lexical properties of URLs, hosting infrastructure, and page content (HTML and links). We also collect new features including HTTP header content, page frames, dynamically loaded content, page behavior such as JavaScript events, plugin usage, and a page's redirection behavior. Feature collection and URL classification occur at the time a URL is submitted to our service, with the overall architecture of Monarch scaling to millions of URLs to satisfy the throughput expected of large social networks and web mail providers.

Monarch is a real-time system for filtering scam, phishing, and malware URLs as they are submitted to web services. We showed that while Monarch's architecture generalizes to many web services being targeted by URL spam, accurate classification hinges on having an intimate understanding of the spam campaigns abusing a service. In particular, we showed that email spam provides little insight into the properties of Twitter spammers, while the reverse is also true. We explored the distinctions between email and Twitter spam, including the overlap of spam features, the persistence of features over time, and the abuse of generic redirectors and public web hosting. We have demonstrated that a modest deployment of Monarch on cloud infrastructure can achieve a throughput of 638,000 URLs per day with an overall accuracy of 91% with 0.87% false positives. Each component of Monarch readily scales to the requirements of large web

services. We estimated it would cost \$22,751 a month to run a deployment of Monarch capable of processing 15 million URLs per day.

2.1.2 Instagram Spam Detection

In recent years, Instagram has become one of top 15 online social networks. However, popularity of Instagram also causes advertisement and spam posts flooding. Therefore, it is necessary to build a spam detection model to decrease number of spam posts in Instagram. We present a scheme applying feature-based method and supervised learning technique to detect spam posts from Instagram. We use K-fold cross validation to find best pair of supervised learning model and parameters of the model and accuracy of our best model is 96.27%.

2.1.3 Towards Web Spam Filtering Using a Classifier Based on the Minimum Description Length Principle

The steady growth and popularization of the Web has led spammers to develop techniques to circumvent search engines aiming good visibility to their web pages in search results. They are responsible for serious problems such as dissatisfaction, irritation, exposure to unpleasant or malicious content, and financial loss. Despite different machine learning approaches have been used to detect web spam, many of them suffer with the curse of dimensionality or require a very high computational cost impeding their employment in real scenarios. In this way, there is still a big effort to develop more advanced methods that at the same time are able to prevent overfitting and fast to learn.

To fill this gap, we present the MDLClass, a classifier technique based on the minimum description length principle, applied to the context of web spam filtering. The proposed method is very efficient, lightweight, multi-class, and fast. We also evaluated a new approach to detect web spam that combines the predictions obtained by the classifiers using content-based, link-based, and transformed link-based features. In our experiments, we employed two real, public and large datasets: the WEBSPAM-UK2006 and

the WEBSpam-UK2007. The results indicate that the proposed MDLClass and ensemble of predictions using different types of features are promising in the task of web spam filtering.

2.1.4 Scalable spam classifier for web tables

Internet mail spam is a problem for most organizations and individuals. Spam can be classified into two categories: fraud and commercial. The fraud category includes phishing, scams, malware, counterfeit products and any other criminal activities. The commercial category includes promotional messages and newsletters that we do not want to receive, being sent illegally from legitimate organizations. Fraud can be seen as being a high threat with high volume while commercial spam is the opposite.

Similar to mail, there are spam Web tables that do not have any useful content. Here we describe our machine-learning classifier for efficient and effective Web tables spam filtering that was tested on a large-scale Web tables corpus of ≈ 36 million tables.

2.1.5 A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms

Email is the most used source of official communication method for business purposes. The usage of the email continuously increases despite of other methods of communications. Automated management of emails is important in the today's context as the volume of emails grows day by day. Out of the total emails, more than 55 percent is identified as spam. This shows that these spams consume email user time and resources generating no useful output. The spammers use developed and creative methods in order to fulfil their criminal activities using spam emails, Therefore, it is vital to understand different spam email classification techniques and their mechanism. This paper mainly focuses on the spam classification approached using machine learning algorithms. Furthermore, this study provides a comprehensive analysis and review of research done on different machine learning techniques and email features used in different Machine

Learning approaches. Also provides future research directions and the challenges in the spam classification field that can be useful for future researchers.

2.1.6 Deployment of Deep Learning Models to Mobile Devices for Spam Classification

The advent of deep learning brings the possibility of better and faster applications in real world. In this work, deep learning models are used for application of spam classification in mobile devices. A Binary Classification model is trained with deep learning and is transformed to a graph using tensorflow and then, is converted to a protobuf file to be deployed on mobile devices. Instead of looking into the spam messages in an algorithmic way i.e. just with keywords, binary model deals with experience of learning and predicts if a text message is spam. The training was performed multiple times on resource-deficient devices and hyper-parameter optimization was performed to enhance the training accuracy to 99.87 %. The test accuracy of mobile application is 98.7 % and testing happens in real-time without any internet access. Our simulation shows that a model with an embedding layer (size 128), an LSTM layer (size 64, dropout 0.2) and a dense layer (sigmoid) yields the highest performance. Also, the comparative evaluation with state-of-the-art methods displayed that our model achieves higher accuracy.

2.1.7 SPAM DETECTION USING MACHINE LEARNING BASED BINARY CLASSIFIER

Email is the worldwide use of communication application. It is because of the ease of use and faster than other communication application. However, its inability to detect whether the mail content is either spam or ham degrade its performance. Nowadays, lot of cases have been reported regarding stealing of personal information or phishing activities via email from the user. This project will discuss how machine learning help in spam detection.

Machine learning uses pre-labelled text to learn the different associations between pieces of text and its output. It used feature extraction

to transform each text to numerical representation in form of vector which represents the frequency of word in predefined dictionary. Text classification is important in the context of structuring the unstructured and messy nature of text such as documents and spam messages in a cost-effective way. A Machine learning platform has capabilities to improve the accuracy of predictions. With regard to Big Data, a Machine Learning platform has abilities to speed up analysing of gigantic data. It is important especially to a company to analyse text data, help inform business decisions and even automate business processes. For example, text classification is used in classifying short texts such as tweets or headlines. It can be used in larger documents such as media articles. It also can be applied to social media monitoring, brand monitoring and etc.

The performance of a classification technique is affected by the quality of data source. Irrelevant and redundant features of data not only increase the elapse time, but also may reduce the accuracy of detection. Supervised ML is able to separate messages and classified the correct categories efficiently. It also able to score the model and weight them successfully. For instances, Gmail's interface is using the algorithm based on machine learning program to keep their users' inbox free of spam messages. This project only focus on filtering, analysing and classifying message and do not blocking them. Hence, the proposed methodology may be adopted to overcome the flaws of the existing spam detection.

From this project, it can be concluded that machine learning algorithm is one of the important parts in order to create spam detection application. To make it more efficient, improvement need to be implemented in future.

2.2 AIM

To build a system using machine learning which classifies whether the message which comes in our mobile from unknown number or emails received are ham or spam.

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS USED

3.1 SOFTWARE REQUIREMENT

- Windows/Mac based system
- Numpy
- Pandas
- Flask
- Matplotlib
- jupyter notebook
- Machine Learning
- Python

3.1.1 Windows/Mac System

Windows or Mac operating based system for personal computers or any other internet of things devices.

3.1.2 Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3.1.3 pandas

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

3.1.4 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

3.1.5 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

3.1.6 Project Jupyter

Jupyter Notebook with any of the supported web browsers. It aims to support the latest versions of these browsers:

- Chrome
- Safari
- Firefox

Up to date versions of Opera and Edge may also work.

3.1.7 Machine Learning

Machine learning (ML) is **a type of artificial intelligence (AI)** that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations.

3.1.8 Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

3.2 ALGORITHMS USED

3.2.1 We will use Bayes theorem. We have a message $m = (w_1, w_2, \dots, w_n)$, where $(w_1, w_2, \dots, w_n)$ is a set of unique words contained in the message. We need to find

$$P(spam|w_1 \cap w_2 \cap \dots \cap w_n) = \frac{P(w_1 \cap w_2 \cap \dots \cap w_n|spam) \cdot P(spam)}{P(w_1 \cap w_2 \cap \dots \cap w_n)}$$

3.2.2 If we assume that occurrence of a word are independent of all other words, we can simplify the above expression to

$$\frac{P(w_1|spam) \cdot P(w_2|spam) \dots P(w_n|spam) \cdot P(spam)}{P(w_1) \cdot P(w_2) \dots P(w_n)}$$

3.3.3 In order to classify we have to determine which is greater $P(spam|w_1 \cap w_2 \cap \dots \cap w_n)$ versus $P(\sim spam|w_1 \cap w_2 \cap \dots \cap w_n)$

3.2.4 Loading dependencies

```
In [1]: from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        import matplotlib.pyplot as plt
        from wordcloud import WordCloud
        from math import log, sqrt
        import pandas as pd
        import numpy as np
        %matplotlib inline
```

We are going to make use of NLTK for processing the messages, WordCloud and matplotlib for visualization and pandas for loading data, NumPy for generating random probabilities for train-test split.

3.2.5 Loading Data

```
In [2]: mails = pd.read_csv('spam.csv', encoding = 'latin-1')
        mails.head()
```

```
Out[2]:
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

We do not require the columns 'Unnamed: 2', 'Unnamed: 3' and 'Unnamed: 4', so we remove them. We rename the column 'v1' as 'label' and 'v2' as 'message'. 'ham' is replaced by 0 and 'spam' is replaced by 1 in the 'label' column. Finally we obtain the following dataframe.

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	Ok lar... Joking wif u oni...	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	U dun say so early hor... U c already then say...	0
4	Nah I don't think he goes to usf, he lives aro...	0

3.2.6 Train-Test Split

To test our model we should split the data into train dataset and test dataset. We shall use the train dataset to train the model and then it will be tested on the test dataset. We shall use 75% of the dataset as train dataset and the rest as test dataset. Selection of this 75% of the data is uniformly random.

```
In [8]: totalMails = mails['message'].shape[0]
trainIndex, testIndex = list(), list()
for i in range(mails.shape[0]):
    if np.random.uniform(0, 1) < 0.75:
        trainIndex += [i]
    else:
        testIndex += [i]
trainData = mails.loc[trainIndex]
testData = mails.loc[testIndex]

In [9]: trainData.reset_index(inplace = True)
trainData.drop(['index'], axis = 1, inplace = True)
trainData.head()
```

Out[9]:

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	Free entry in 2 a wkly comp to win FA Cup fina...	1
2	U dun say so early hor... U c already then say...	0
3	FreeMsg Hey there darling it's been 3 week's n...	1
4	As per your request 'Melle Melle (Oru Minnamin...	0

3.2.7 Visualizing data

Let us see which are the most repeated words in the spam messages! We are going to use WordCloud library for this purpose.

```
In [13]: spam_words = ' '.join(list(mails[mails['label'] == 1]['message']))
spam_wc = WordCloud(width = 512,height = 512).generate(spam_words)
plt.figure(figsize = (10, 8), facecolor = 'k')
plt.imshow(spam_wc)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

3.2.8 Training the model

We are going to implement two techniques: Bag of words and TF-IDF.

Preprocessing: Before starting with training we must preprocess the messages. First ,we shall make all the character lowercase('free' and 'FREE' mean the same).

Then we tokenize each message in the dataset. Tokenization is the task of splitting up a message into pieces and throwing away the punctuation characters.

The words like 'go', 'goes', 'going' indicate the same activity. We can replace all these words by a single word 'go'. This is called stemming. We are going to use Porter Stemmer, which is a famous stemming algorithm.

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

3.2.9 We then move on to remove the stop words. Stop words are those words which occur extremely frequently in any text. For example words like 'the', 'a', 'an', 'is', 'to' etc. These words do not give us any information about the content of the text.

```
In [19]: def process_message(message, lower_case = True, stem = True, stop_words = True, gram = 2):
    if lower_case:
        message = message.lower()
    words = word_tokenize(message)
    words = [w for w in words if len(w) > 2]
    if gram > 1:
        w = []
        for i in range(len(words) - gram + 1):
            w += [' '.join(words[i:i + gram])]
        return w
    if stop_words:
        sw = stopwords.words('english')
        words = [word for word in words if word not in sw]
    if stem:
        stemmer = PorterStemmer()
        words = [stemmer.stem(word) for word in words]
    return words
```

Bag of Words: In Bag of words model we find the 'term frequency', i.e. number of occurrences of each word in the dataset. Thus for word w ,

$$P(w) = \frac{\text{Total number of occurrences of } w \text{ in dataset}}{\text{Total number of words in dataset}}$$

and

$$P(w|spam) = \frac{\text{Total number of occurrences of } w \text{ in spam messages}}{\text{Total number of words in spam messages}}$$

TF-IDF: TF-IDF stands for Term Frequency-Inverse Document Frequency. In addition to Term Frequency we compute Inverse document frequency.

$$IDF(w) = \log \frac{\text{Total number of messages}}{\text{Total number of messages containing } w}$$

3.2.10 Additive Smoothing

So, if we encounter a word in test dataset which is not part of train dataset, in that case $P(w)$ will be 0, which will make the $P(\text{spam}|w)$ undefined (since we would have to divide by $P(w)$ which is 0). We introduce additive smoothing. In additive smoothing we add a number α to the numerator and add α times number of classes over which the probability is found in the denominator.

$$P(w|\text{spam}) = \frac{TF(w|\text{spam}) + \alpha}{\sum_{\forall \text{ words } x \in \text{spam in train dataset}} TF(x) + \alpha \sum_{\forall \text{ words } x \in \text{spam in train dataset}} 1}$$

When using TF-IDF

$$P(w|\text{spam}) = \frac{TF(w|\text{spam}) * IDF(w) + \alpha}{\sum_{\forall \text{ words } x \in \text{train dataset}} TF(x) * IDF(x) + \alpha \sum_{\forall \text{ words } x \in \text{spam in train dataset}} 1}$$

This is done so that the least probability of any word now should be a finite number. Addition in the denominator is to make the resultant sum of all the probabilities of words in the spam emails as 1. When $\alpha = 1$, it is called Laplace smoothing.

3.2.11 Classification

For classifying a given message, first we preprocess it. For each word w in the processed message we find a product of $P(w|\text{spam})$. If w does not exist in the train dataset we take $TF(w)$ as 0 and find $P(w|\text{spam})$ using above formula. We multiply this product with $P(\text{spam})$. The resultant product is the $P(\text{spam}|\text{message})$. Similarly, we find $P(\text{ham}|\text{message})$.

3.2.12 Final result

```

In [22]: sc_tf_idf = SpamClassifier(trainData, 'tf-idf')
          sc_tf_idf.train()
          preds_tf_idf = sc_tf_idf.predict(testData['message'])
          metrics(testData['label'], preds_tf_idf)

Precision: 0.8873239436619719
Recall: 0.6596858638743456
F-score: 0.7567567567567568
Accuracy: 0.94164265129683

In [23]: sc_bow = SpamClassifier(trainData, 'bow')
          sc_bow.train()
          preds_bow = sc_bow.predict(testData['message'])
          metrics(testData['label'], preds_bow)

Precision: 0.890625
Recall: 0.5968586387434555
F-score: 0.7147335423197492
Accuracy: 0.9344380403458213

In [24]: pm = process_message('I cant pick the phone right now. Pls send a message')
          sc_tf_idf.classify(pm)

Out[24]: False

In [25]: pm = process_message('Congratulations ur awarded $500 ')
          sc_tf_idf.classify(pm)

Out[25]: True

```

3.3 IMPLEMENTATION AND DEPLOYMENT

- 3.3.1 Open command prompt in project folder.
- 3.3.2 First create a virtual environment by using the command- pip install virtualenv then virtualenv base.
- 3.3.3 Activate the virtual environment by using the command- base\Scripts\activate.
- 3.3.4 Install all the libraries like flask , matplotlib, pandas, numpy by using the command- pip install numpy and similarly like this.
- 3.3.4 Run the server by using the command- python app.py
- 3.3.5 Open any web browser and paste this and press enter:
http://127.0.0.1:5000/spam_classifier
- 3.3.6 Spam classifier project will run and now you can test it.

CHAPTER 4

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 RESULTS & DISCUSSION

When any kind of message is typed into the spam classifier app, it determines and tells the user successfully it is spam or ham by using machine learning. It finally saves time, money from phishing, data, bandwidth and stops illegal and harm causing messages and emails.

SCREENSHOTS

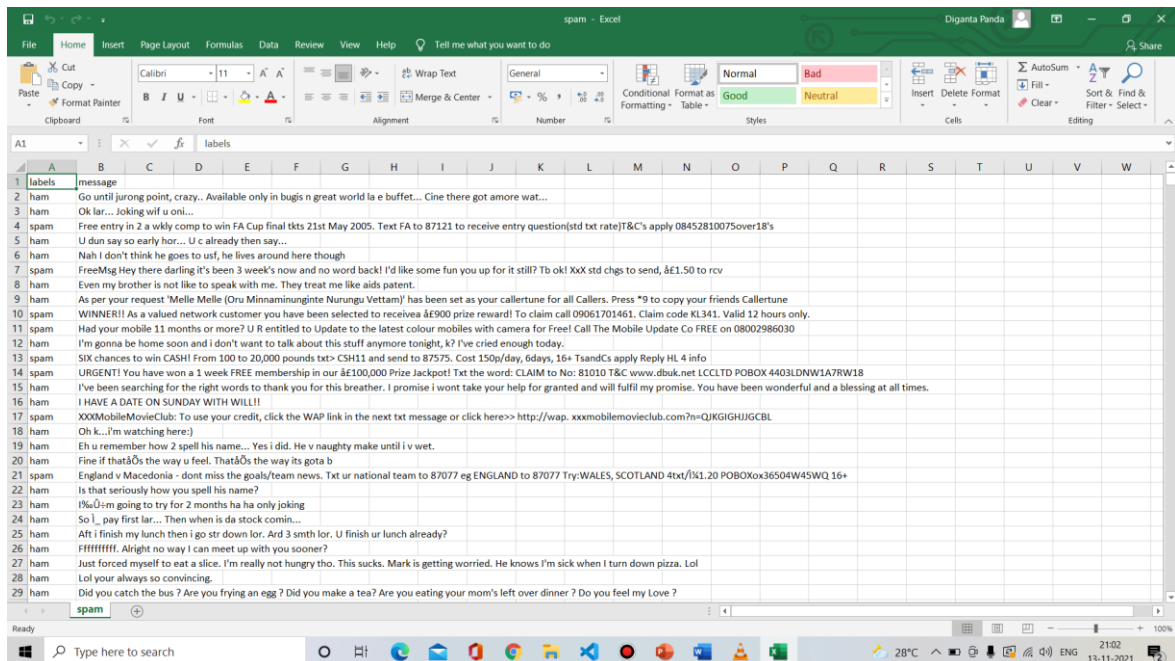


Fig 4.1 Dataset

4.1.1 We have run the server by

- Opening the command prompt in project folder.

- First create a virtual environment by using the command- pip install virtualenv then virtualenv base.
- Activate the virtual environment by using the command- base\Scripts\activate.
- Install all the libraries like flask , matplotlib, pandas, numpy by using the command- pip install numpy and similarly like this.
- Run the server by using the command- python app.py

4.1.2 Testing the Project:

- Now by opening any web browser and pasting this and pressing enter: http://127.0.0.1:5000/spam_classifier
- Spam classifier project will run and now we can test it.

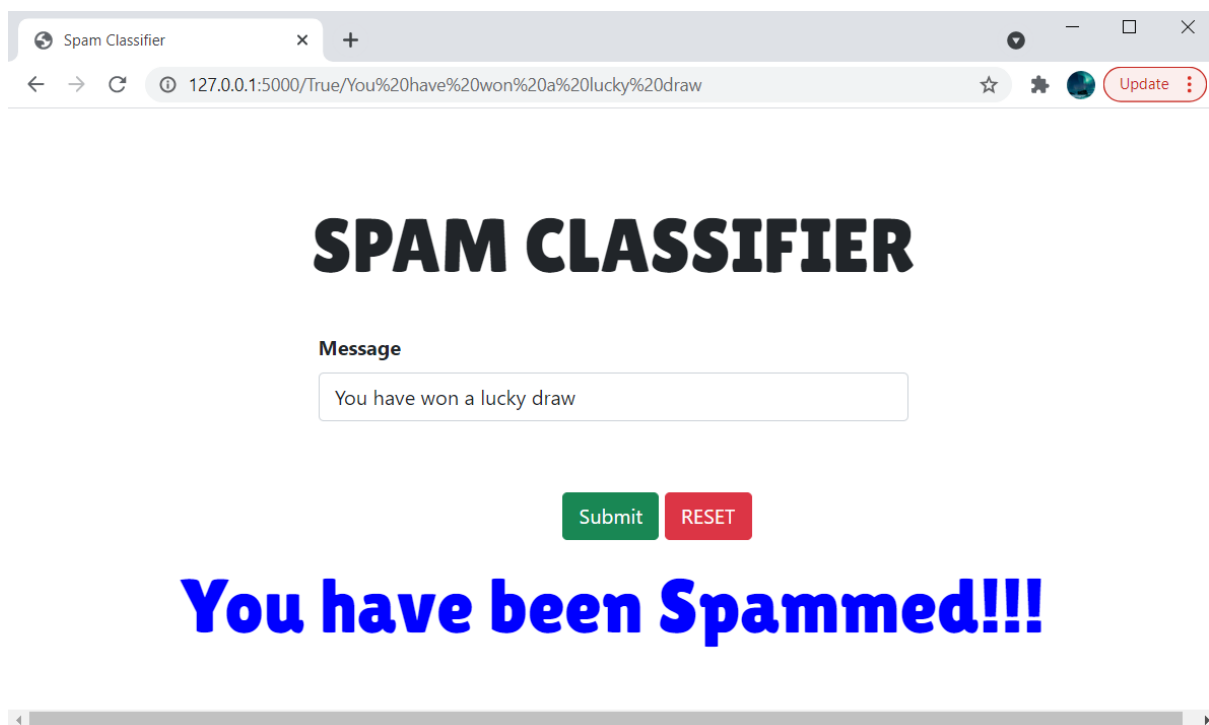


Fig 4.2 We have entered a spam message and the project successfully determines that it is a spam.

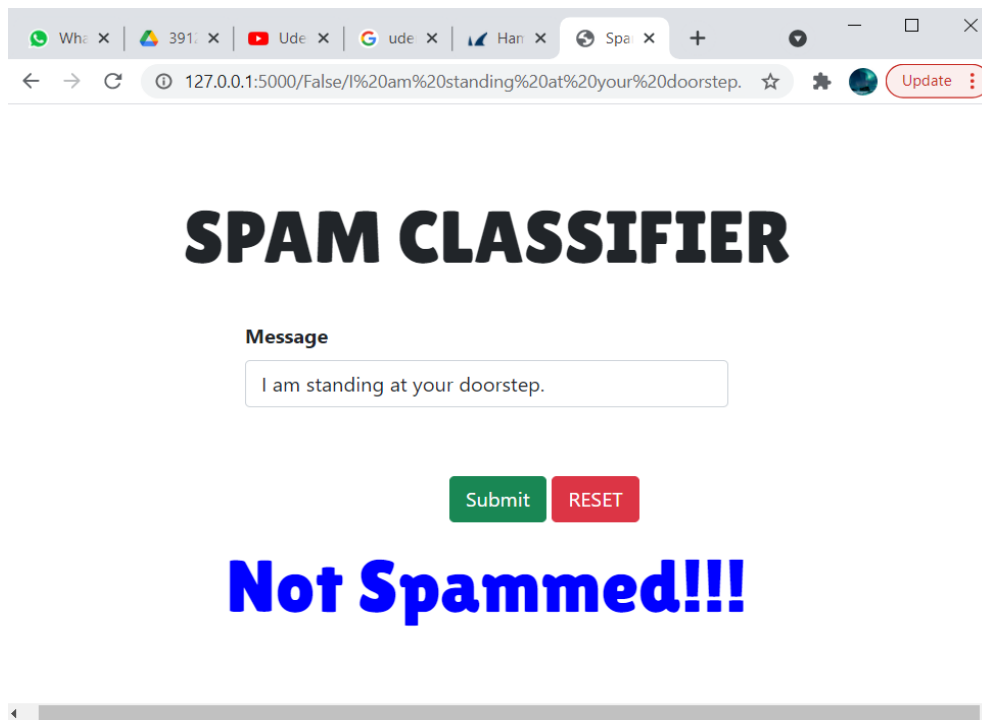


Fig 4.3 We have entered a ham message and the project successfully determines that it is a ham.

4.2 PERFORMANCE ANALYSIS

This spam classifier works on most of the messages which we get via mobile messaging or emails and identifies accurately whether it is spam or ham . Some examples of certain messages are given below which our classifier identifies pretty well:

Table 4.1 Examples of Ham Vs Spam

<u>HAM</u>	<u>SPAM</u>
1. I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.	1. Congratulations, you have won 1 crore rupees in MyRummyCircle.
2. You have been rewarded 1 st prize in our cultural meeting held at Sathyabama University.	2. Hurray! You have won a lottery trip to Maldives.
3. I can't pick the phone right now. Please leave a message.	3. You have been enrolled in Artificial Intelligence Course. Only 4 students have got selected from your college.
4. He told me that he will give his money back in time.	4. Thanks for your subscription to Ringtone UK your mobile will be charged å£5/month Please confirm by replying YES or NO. If you reply NO you will not be charged
5. I purchased a new house for my parents at 20% down payment.	5. Win exiting amazon vouchers by forwarding this message to 5 more friends.
6. Your amazon order will get delivered by 6pm today.	6. You have a won a Swift DZire. Enter the required details to get eligible.

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 SUMMARY

Email and mobile messaging is the worldwide use of communication application. It is because of the ease of use and faster than other communication application. However, its inability to detect whether the mail content is either spam or ham degrade its performance. Nowadays, lot of cases have been reported regarding stealing of personal information or phishing activities via email from the user.

This project discussed how machine learning helps in spam detection. Machine learning is an artificial intelligence application that provides the ability to automatically learn and improve data without being explicitly programmed. The objective of developing this model is to detect whether the message or email received is spam or ham and will automatically be detected and blocked.

5.2 CONCLUSION

The performance of a classification technique is affected by the quality of data source. Irrelevant and redundant features of data not only increase the elapse time, but also may reduce the accuracy of detection. Each algorithm has its own advantages and disadvantages. As state before, supervised ML is able to separate messages and classified the correct categories efficiently. It also able to score the model and weight them successfully.

For instances, Gmail's interface is using the algorithm based on machine learning program to keep their users' inbox free of spam messages. During the implementation, only text (messages) can be classified and score instead of domain name and email address. This project only focus on filtering, analysing and classifying message and do not blocking them. Hence, the proposed methodology may be adopted to overcome the flaws of the existing spam detection.

REFERENCES

- Zainab, D. Syed and D. Al-Thani, "Deployment of Deep Learning Models to Mobile Devices for Spam Classification," in *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*, Los Angeles, CA, USA, 2019 pp. 112-117.
 - Wuxain Zhang ,Hung-Min Sun,2017 *IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*
 - R. Silva, T. Almeida and A. Yamakami, "Towards Web Spam Filtering Using a Classifier Based on the Minimum Description Length Principle," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Anaheim, CA, 2016
 - S. Villasenor, T. Nguyen, A. Kola, S. Soderman and M. Gubanov, "Scalable spam classifier for web tables," in *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017
 - M. RAZA, N. Jayasinghe and M. Muslam, "A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms," in *2021 International Conference on Information Networking (ICOIN)*, Jeju Island, Korea (South), 2021
 - Anitha, PU & Rao, Chakunta & , T.Sireesha. (2013). A Survey On: E-mail Spam Messages and Bayesian Approach for Spam Filtering. *International Journal of Advanced Engineering and Global Technology (IJAEGT)*. 1. 124-136.
 - Attenberg, J., Weinberger, K., Dasgupta, A., Smola, A., & Zinkevich, M. (2009, July). Collaborative email-spam filtering with the hashing trick. In *Proceedings of the Sixth Conference on Email and Anti-Spam*.
 - Machine learning for email spam filtering: review, approaches and open research problems
- <https://www.sciencedirect.com/science/article/pii/S2405844018353404>
- www.geeksforgeeks.org

APPENDIX (Source code)

- app.py

```
from flask import Flask, request, redirect, url_for
from flask.templating import render_template
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import matplotlib.pyplot as plt
from math import log, sqrt
import pandas as pd
import numpy as np
import re
from joblib import dump, load
import nltk
nltk.download('punkt')

app = Flask(__name__)

@app.route("/spam_classifier", methods=["GET", "POST"])
def spam_classifier():
    if request.method == "POST":
        prob_spam, prob_ham, sum_tf_idf_spam, sum_tf_idf_ham, spam_words,
        ham_words, prob_spam_mail, prob_ham_mail = getData()
        message = request.form["message"]
        processed_message = process_message(message)
        result = classify(processed_message, 'tf-idf', prob_spam,
        prob_ham, sum_tf_idf_spam, sum_tf_idf_ham, spam_words,
        ham_words, prob_spam_mail, prob_ham_mail)
        return redirect(url_for("result", result=result, mssg=message))
    else:
        return render_template("index.html")

@app.route("/<result><mssg>")
def result(result, mssg):
    return render_template("result.html", content=result, mssg=mssg)

def getData():
    prob_spam = load("./Joblib/prob_spam_tf_idf.joblib")
    prob_ham = load("./Joblib/prob_ham_tf_idf.joblib")
    sum_tf_idf_spam = load("./Joblib/sum_tf_idf_spam.joblib")
    sum_tf_idf_ham = load("./Joblib/sum_tf_idf_ham.joblib")
    spam_words = load("./Joblib/spam_words.joblib")
    ham_words = load("./Joblib/ham_words.joblib")
    prob_spam_mail = load("./Joblib/prob_spam_mail.joblib")
    prob_ham_mail = load("./Joblib/prob_ham_mail.joblib")
```

```

        return prob_spam, prob_ham, sum_tf_idf_spam, sum_tf_idf_ham, spam_words,
        ham_words, prob_spam_mail, prob_ham_mail

def process_message(message, lower_case = True, stem = True, stop_words =
True, gram = 2):
    #converting all upper case characters to lower case
    if lower_case:
        message = message.lower()

    # word tokenize converts sentence into an array of words
    words = word_tokenize(message)

    #all the words of length less than or equal to 2 are discarded
    words = [w for w in words if len(w) > 2]
    return words

    #gram is used to slide the window of size 2
    #since gram is set to 2 we will have pair of two words
    if gram > 1:
        w = []
        for i in range(len(words) - gram + 1):
            w += [' '.join(words[i:i + gram])]
        return w

    # stop words are words like 'a' 'the' 'in' etc so below we are removing
all the stop words
    if stop_words:
        sw = stopwords.words('english')
        words = [word for word in words if word not in sw]

    # stemming is the process of reducing words to there base form like
reducing words like eating, eats, eaten to eat
    if stem:
        stemmer = PorterStemmer()
        words = [stemmer.stem(word) for word in words]

    return words

def classify(processed_message , method, prob_spam, prob_ham,
sum_tf_idf_spam, sum_tf_idf_ham, spam_words,
ham_words, prob_spam_mail, prob_ham_mail):
    pSpam, pHam = 0, 0
    for word in processed_message:
        if word in prob_spam:
            pSpam += log(prob_spam[word])
        else:
            if method == 'tf-idf':
                pSpam -= log(sum_tf_idf_spam +
len(list(prob_spam.keys()))))

```

```

        else:
            pSpam -= log(spam_words + len(list(prob_spam.keys()))))
        if word in prob_ham:
            pHam += log(prob_ham[word])
        else:
            if method == 'tf-idf':
                pHam -= log(sum_tf_idf_ham + len(list(prob_ham.keys()))))
            else:
                pHam -= log(ham_words + len(list(prob_ham.keys()))))
        pSpam += log(prob_spam_mail)
        pHam += log(prob_ham_mail)
    return pSpam >= pHam

if __name__ == "__main__":
    app.run(debug=True)

```

- **SMSSpamClassifier.ipynb**

```

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from math import log, sqrt
import pandas as pd
import numpy as np
import re
from joblib import dump, load
%matplotlib inline

import nltk
nltk.download('punkt')

... [nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True

###Preprocessing

mails = pd.read_csv('spam.csv', encoding = 'latin-1')
mails.head()
mails.shape

... (5572, 5)

```

```

mails.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis = 1, inplace = True)
mails.head()

...
  labels      message
0  ham  Go until jurong point, crazy.. Available only ...
1  ham              Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3  ham  U dun say so early hor... U c already then say...
4  ham  Nah I don't think he goes to usf, he lives aro...

mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
mails.head()

...
  labels      message
0  ham  Go until jurong point, crazy.. Available only ...
1  ham              Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3  ham  U dun say so early hor... U c already then say...
4  ham  Nah I don't think he goes to usf, he lives aro...

mails['labels'].value_counts()

...  ham    4825
     spam    747

```

```

Name: labels, dtype: int64

mails['label'] = mails['labels'].map({'ham': 0, 'spam': 1})
mails.head()

...
  labels      message  label
0  ham  Go until jurong point, crazy.. Available only ...    0
1  ham              Ok lar... Joking wif u oni...    0
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...    1
3  ham  U dun say so early hor... U c already then say...    0
4  ham  Nah I don't think he goes to usf, he lives aro...    0

mails.drop(['labels'], axis = 1, inplace = True)
mails.head()

...
      message  label
0  Go until jurong point, crazy.. Available only ...    0
1              Ok lar... Joking wif u oni...    0
2  Free entry in 2 a wkly comp to win FA Cup fina...    1
3  U dun say so early hor... U c already then say...    0
4  Nah I don't think he goes to usf, he lives aro...    0

totalMails = 4825 + 747
trainIndex, testIndex = list(), list()
for i in range(mails.shape[0]):
    if np.random.uniform(0, 1) < 0.75:

```


[illegible]

```

# stemming is the process of reducing words to there base form like reducing words like eating, eats, eaten to eat
if stem:
    stemmer = PorterStemmer()
    words = [stemmer.stem(word) for word in words]

return words

```

```

class SpamClassifier(object):
    def __init__(self, trainData, method = 'tf-idf'):
        self.mails, self.labels = trainData['message'], trainData['label']
        self.method = method

    def train(self):
        self.calc_tf_and_idf()
        if self.method == 'tf-idf':
            self.calc_tf_idf()
        else:
            self.calc_prob()

    def calc_prob(self):
        self.prob_spam = dict()
        self.prob_ham = dict()
        for word in self.tf_spam:
            self.prob_spam[word] = (self.tf_spam[word] + 1) / (self.spam_words + \
                                                                len(list(self.tf_spam.keys())))
        for word in self.tf_ham:
            self.prob_ham[word] = (self.tf_ham[word] + 1) / (self.ham_words + \
                                                                len(list(self.tf_ham.keys())))
        self.prob_spam_mail, self.prob_ham_mail = self.spam_mails / self.total_mails, self.ham_mails / self.total_mails
        dump(self.prob_spam, 'prob_spam_bow.joblib')

```

```

dump(self.prob_ham, 'prob_ham_bow.joblib')
dump(self.prob_spam_mail, 'prob_spam_mail.joblib')
dump(self.prob_ham_mail, 'prob_ham_mail.joblib')

def calc_tf_and_idf(self):
    noOfMessages = self.mails.shape[0]
    self.spam_mails, self.ham_mails = self.labels.value_counts()[1], self.labels.value_counts()[0]
    self.total_mails = self.spam_mails + self.ham_mails
    self.spam_words = 0
    self.ham_words = 0
    self.tf_spam = dict()
    self.tf_ham = dict()
    self.idf_spam = dict()
    self.idf_ham = dict()
    for i in range(noOfMessages):
        message_processed = process_message(self.mails[i])
        count = list() #To keep track of whether the word has ocured in the message or not.
        #For IDF
        for word in message_processed:
            if self.labels[i]:
                self.tf_spam[word] = self.tf_spam.get(word, 0) + 1
                self.spam_words += 1
            else:
                self.tf_ham[word] = self.tf_ham.get(word, 0) + 1
                self.ham_words += 1
            if word not in count:
                count += [word]
        for word in count:
            if self.labels[i]:
                self.idf_spam[word] = self.idf_spam.get(word, 0) + 1

```

```

            else:
                self.idf_ham[word] = self.idf_ham.get(word, 0) + 1
        dump(self.spam_words, 'spam_words.joblib')
        dump(self.ham_words, 'ham_words.joblib')

    def calc_tf_idf(self):
        self.prob_spam = dict()
        self.prob_ham = dict()
        self.sum_tf_idf_spam = 0
        self.sum_tf_idf_ham = 0
        for word in self.tf_spam:
            self.prob_spam[word] = (self.tf_spam[word]) * log((self.spam_mails + self.ham_mails) \
                                                                / (self.idf_spam[word] + self.idf_ham.get(word, 0)))
            self.sum_tf_idf_spam += self.prob_spam[word]
        for word in self.tf_ham:
            self.prob_ham[word] = (self.tf_ham[word]) * log((self.spam_mails + self.ham_mails) \
                                                                / (self.idf_spam.get(word, 0) + self.idf_ham[word]))
            self.sum_tf_idf_ham += self.prob_ham[word]
        for word in self.prob_spam:
            self.prob_spam[word] = (self.prob_spam[word] + 1) / (self.sum_tf_idf_spam + len(list(self.prob_spam.keys())))

        for word in self.prob_ham:
            self.prob_ham[word] = (self.prob_ham[word] + 1) / (self.sum_tf_idf_ham + len(list(self.prob_ham.keys())))

        self.prob_spam_mail, self.prob_ham_mail = self.spam_mails / self.total_mails, self.ham_mails / self.total_mails
        dump(self.sum_tf_idf_spam, 'sum_tf_idf_spam.joblib')
        dump(self.sum_tf_idf_ham, 'sum_tf_idf_ham.joblib')
        dump(self.prob_spam, 'prob_spam_tf_idf.joblib')
        dump(self.prob_ham, 'prob_ham_tf_idf.joblib')

    def classify(self, processed_message):

```

```

    pSpam, pHam = 0, 0
    for word in processed_message:
        if word in self.prob_spam:
            pSpam += log(self.prob_spam[word])
        else:
            if self.method == 'tf-idf':
                pSpam -= log(self.sum_tf_idf_spam + len(list(self.prob_spam.keys())))
            else:
                pSpam -= log(self.spam_words + len(list(self.prob_spam.keys())))
        if word in self.prob_ham:
            pHam += log(self.prob_ham[word])
        else:
            if self.method == 'tf-idf':
                pHam -= log(self.sum_tf_idf_ham + len(list(self.prob_ham.keys())))
            else:
                pHam -= log(self.ham_words + len(list(self.prob_ham.keys())))
    pSpam += log(self.prob_spam_mail)
    pHam += log(self.prob_ham_mail)
    return pSpam >= pHam

def predict(self, testData):
    result = dict()
    for (i, message) in enumerate(testData):
        processed_message = process_message(message)
        result[i] = int(self.classify(processed_message))
    return result

```

```

def metrics(labels, predictions):
    true_pos, true_neg, false_pos, false_neg = 0, 0, 0, 0
    for i in range(len(labels)):
        true_pos += int(labels[i] == 1 and predictions[i] == 1)

```

```

    true_pos, true_neg, false_pos, false_neg = 0, 0, 0, 0
    for i in range(len(labels)):
        true_pos += int(labels[i] == 1 and predictions[i] == 1)
        true_neg += int(labels[i] == 0 and predictions[i] == 0)
        false_pos += int(labels[i] == 0 and predictions[i] == 1)
        false_neg += int(labels[i] == 1 and predictions[i] == 0)
    precision = true_pos / (true_pos + false_pos)
    recall = true_pos / (true_pos + false_neg)
    Fscore = 2 * precision * recall / (precision + recall)
    accuracy = (true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg)

    print("Precision: ", precision)
    print("Recall: ", recall)
    print("F-score: ", Fscore)
    print("Accuracy: ", accuracy)

```

```

sc_tf_idf = SpamClassifier(trainData, 'tf-idf')
sc_tf_idf.train()
preds_tf_idf = sc_tf_idf.predict(testData['message'])
metrics(testData['label'], preds_tf_idf)

```

```

... Precision: 0.9722222222222222
Recall: 0.546875
F-score: 0.7
Accuracy: 0.933920704845815

```

```

sc_bow = SpamClassifier(trainData, 'bow')
sc_bow.train()
preds_bow = sc_bow.predict(testData['message'])
metrics(testData['label'], preds_bow)

```

```

... Precision: 0.9315068493150684
Recall: 0.3333333333333333
F-score: 0.4909747292418773
Accuracy: 0.8984881209503239

```

```

pm = process_message('I cant pick the phone right now. Pls send a message')
sc_tf_idf.classify(pm)

```

```

... False

```

```

pm = process_message('Congratulations ur awarded $500 ')
sc_tf_idf.classify(pm)

```

```

... True

```