

**MAIL CLASSIFICATION FOR
E-SPAM DETECTION
USING MACHINE LEARNING**

**DONE BY
SARU PREETHIKA.T
B.TECH
ARTIFICIAL INTELLIGENCE &
DATA SCIENCE**

TABLE OF CONTENTS

CHAPTER	TABLE	PAG-NO
	ABSTRACT	iv
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1.	INTRODUCTION	
	1.1 OVERVIEW	1
	1.2 INTRODUCTION TO SPAM DETECTION	1
	1.3 INTRODUCTION TO NLTK	2
	1.4 INTRODUCTION TO PYTHON	2
	1.5 INTRODUCTION TO SVM	3
2.	DESIGN AND IMPLEMENTATION	
	2.1 LEARNING DATA	4
	2.2 DATA PROCESSING	4
	2.3 STOP WORDS	4
	2.4 STEMMING	4
	2.5 INVERTED INDEX FILE	5
	2.6 ATTRIBUTE SELECTION	5
	2.7 ATTRIBUTE VALUE REPRESENTATION	6
	2.8 ARRF FILES	6
	2.9 SPAMDTECTION USING ML	7
3.	TESTING/RESULT AND ANALYSIS	
	3.1 TEST DATA	11
	3.2 PROCESS FLOW DIAGRAM	11
	3.3 WEKA	11
	3.4 TESTING STATISTICS	12
	3.5 EXPERIMENTAL RESULTS	12
	3.5.1 METHOD FREQUENCY	13
	3.5.2 METHOD	13
	3.6 RESULTS AND DISCUSSION	14
4.	CONCLUSION AND FUTURE SCOPE	
	4.1 CONCLUSION	17

4,2 FUTURE SCOPE	17
APPENDIX-1	
SAMPLE CODING	18
APPENDIX-2	
SCREENSHOT	22
REFERENCES	25

•

ABSTRACT

MAIL CLASSIFICATION FOR E-SPAM DETECTION

A spam detection system using machine learning involves building a machine learning model that can identify spam messages. This can be accomplished by using various techniques such as natural language processing, artificial neural networks, and supervised learning. The model is then trained on a dataset of labeled emails that are either spam or not. The model is then tested on a test dataset to determine its accuracy. Depending on the accuracy of the model, it can be used to detect spam emails in real time. The possibility that anybody can leave an email or a message provides a golden opportunity for spammers to write spam message about our different interests. Spam fills inbox with a number of ridiculous emails .

Degrades our internet speed to a great extent .Steals useful information like our details on our contact list. Identifying these spammers and also the spam content can be a hot topic of research and laborious tasks. Email spam is an operation to send messages in bulk by mail .Since the expense of the

spam is borne mostly by the recipient ,it is effectively postage due advertising. Spam email is a kind of commercial advertising which is economically viable because email could be a very cost effective medium for sender .With this proposed model the specified message can be stated as spam or not using Bayes' theorem and Naive Bayes' Classifier and Also IP addresses of the sender are often detect

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
2.9.1	USING INFO TO DISPLAY DATA TYPES AND BASIC INFORMATION ABOUT THE DATASET	7
2.9.2	USING NLTK WORKING WITH DATASET TO REMOVE UN-REQUIRED COLUMNS	7
2.9.4	SPLITTING THE DATA SET FOR TESTING AND TRAINING	8
2.9.6	FINDING OUT THE MOST REPEATED WORDS IN SPAM AND HAM	9
2.9.8	WORKING WITH TF-IDF	10
3.6	RESULTS AND DISCUSSION OF SPAM AND HAM USING INBOX MESSAGE	15
3.6.1	USING BAYE'S AND NAÏVE BAYE'S CLASSIFIERS for DETECTING THE SPAM	16
3.6.2	USING THE BAYE'S CLASSIFIER ABLE TO DETECT THE IP ADDRESS	16

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
NLTK	Natural Language Toolkit
TF	Term Frequency
IDF	Inverted Document Frequency
WEKA	Waikato Environment for Knowledge Analysis
ARFF	Attribute Relation File Format
KNN	K-Nearest Neighbour
API	Application Programming Interface
Py	Python
MATLAB	Matrix Laboratory
SVM	Support Vector Machine
SVC	Support Vector Classifier

CHAPTER-1

INTRODUCTION

1.3 INTRODUCTION TO NLTK

Natural Language Toolkit (NLTK) is library in Python, which provides a base for building programs and classification of data. NLTK is a collection of resources for Python that can be used for text processing, classification, tagging and tokenization. This toolbox plays a key role in transforming the text data in the tweets into a format that can be used to extract sentiment from them.

NLTK provides various functions which are used in pre-processing of data so that data available from twitter become fit for mining and extracting features. NLTK support various machine learning algorithms which are used for training classifier and to calculate the accuracy of different classifier. In our thesis we use Python as our base programming language which is used for writing code snippets. NLTK is a library of Python which plays a very important role in converting natural language text to a sentiment either positive or negative.

In our thesis we use Python as our base programming language which is used for writing code snippets. NLTK is a library of Python which plays a very important role in converting natural language text to a sentiment either positive or negative. NLTK also provides different sets of data which are used for training classifiers. These datasets are structured and stored in library of NLTK, which can be accessed easily with the help of Python.

1.4 INTRODUCTION TO PYTHON

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. It supports multiple programming paradigms beyond objectoriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.

Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using NLTK. Other high level programming languages such as 'R' and 'Matlab' were considered because they have many benefits such as ease of use but they do not offer the same flexibility and freedom that Python can deliver.

1.5 INTRODUCTION TO SVM

SVM are supervised machine learning methods used for classification, regression and detection models. SVM are more effective for high dimensional space. SVCs are capable for multi-class

classification. SVC and NuSVC are similar whereas, LinearSVC are based on linear kernels. All these SVCs take two input arrays: an array X of size [samples, features] and an array Y of size

[samples]. NuSVC implements a 'one-against-one' scheme for multi-class, hence it provides consistent interface with other classifiers.

Whereas, LinearSVC implement 'one-vs-rest' scheme NuSVC implementation is based on 'libsvm' library, whereas LinearSVC implementation is based on 'liblinear' library. Thus, from hyper-planes we can understand, a good separation is achieved by those that have the maximum distance to the nearest data points of any class which is called functional margin.

CHAPTER-2

DESIGN AND IMPLEMENTATION

2.1 LEARNING DATA

The data used for this project was taken from the Spam Assassin public corpus website. It consists of two data sets: train and test. Each dataset contains a randomly selected collection of emails in plain text format, which have been labelled as HAM or SPAM. The training data is used to build a model for classifying emails into HAM and SPAM. The test data is used to check the accuracy of the model built with the training data. The training data set contains 400 emails with 283 ham and 117 spam emails. The test data contains 200 emails with 139 ham and 61spam emails.

2.2 DATA PREPROCESSING

The emails in the learning data are in plain text format. We need to convert the plain text into features that can represent the emails. Using these features we can then use a learning algorithm on the emails. A number of pre-processing steps are first performed. We convert the plain text files to files with one word per line. In this project, we look at emails just as a collection of words. So, to make it easier we convert each file into a list of words using Bourne Shell Scripts (*extractmultfiles.sh* and *extractwords.sh*). The output files are named as '*filename.words*'.

2.3 STOP WORDS

There are some English words which appear very frequently in all documents and so have no worth in representing the documents. These are called STOP WORDS and there is no harm in deleting them. Example: the, a, for etc. There are also some domain specific (in this case email) stop words such as mon, tue, email, sender, from etc. So, we delete these words from all the files using a Bourne Shell Script. These words are put in a file 'words.txt'. The shell script takes multiple files as an argument and then deletes all the stop words mentioned in the words.txt file.

2.4 STEMMING

The next step to be performed is stemming. Stemming is used to find a root of a word and thus replacing all words to their stem which reduces the number of words to be considered for representing a document. Example: sings, singing, sing have sing as their stem. In the project, we use PYTHON implementation of Porter stemming algorithm

which is slightly modified to meet our needs. The resultant files are named with an extension 'words_stemmed'.

2.5 INVERTED INDEX FILE

In the next step, we create an inverted index file. This file has 3 columns – word, filename and frequency of word in the file. The file is sorted in alphabetical order of words. For this, we first create an inverted file for each individual file and then append them all together to build one inverted index file. The snapshot of the index file looks as:

```
abl spam_4_train 1 abl
spam_55_train 1 abl
spam_8_train 1
abli spam_47_train 1
abmv spam_111_train 3 abo
ham_94_train 1
abound ham_6_train 1
abovement ham_173_train 1 abr
ham_3_train 2
abreau ham_277_train 8
abreauj ham_277_train 2 abroad
ham_193_train 2
absbottom ham_31_train 1
absenc ham_273_train 1
```

For this task, we create two Bourne script files. The script 'filename.sh' creates files with an extension '.out' which is an inverted index file for a single file. Then the script 'append.sh' appends all the '.out' files together and sorts them in alphabetical order of words.

2.6 ATTRIBUTE SELECTION

In the next step, we chose words to represent all the emails from the inverted index file.

We use "Bag of words" method to select attributes, i.e. we use a set of words as attributes. We have to select some n specific words as all the documents contain thousands of unique words altogether and we cannot use all of these words for learning algorithm.

For this process, we use information-gain method. We use this method as it's a class dependent method, so on average it gives better accuracy. We use a PYTHON program to calculate the gain value for each word using the formula-

$$Gain(w) = - \sum_{i=1}^k P(C_i) \log P(C_i) \\ + P(w) \sum_{i=1}^k P(C_i | w) \log P(C_i | w) + P(\bar{w}) \sum_{i=1}^k P(C_i | \bar{w}) \log P(C_i | \bar{w})$$

In this case, P(C) refers to the probability of ham and spam class which can be calculated easily as we already have a predefined number of ham and spam emails. Next is P(w) which is the probability of the given word. It is given by the number of documents containing the word / total number of documents. P(C|w), means the number of documents which are labelled as spam or ham and which also contain the word w divided by the probability of the word w. So, looking at the inverted index file we can easily calculate all the values.

The JAVA program generates a text file with all the words and their respective gain values. For this project, we manually select top 10, 15 and 20 words with highest gain values and put them in separate files.

2.7 ATTRIBUTE VALUE REPRESENTATION

Once we have selected words, the next step is to represent the values for the selected attributes. We assign numerical values to them using 2 different methods –

a. Term Frequency

Definition: $TF = t(i,j)$

This gives the frequency of a word i in j th document.

a. TF-IDF (Term Frequency - Inverted Document

Frequency) Definition: $TF \times IDF = t(i,j) \times \log(N/n)$

File name	listinfo	beenther	subscrib	remov	mailman	Error	Keyword	Bulk	Preced	archiv	class
ham_120_train	0	5	0	4	4	2	3	3	7	2	ham
spam_120_train	3	1	2	6	7	4	5	5	3	5	spam
ham_225_train	1	3	3	1	5	2	5	4	2	7	ham
ham_82_train	4	0	2	3	3	6	3	6	5	0	ham

N = total number documents and n = number of documents that contains the word

For each of these methods, we use a python program that creates a text file in a tabular format with the document name, attributes name and attribute value with the class value specifying it's a spam or ham email. A snapshot of this file looks as:

2.8 ARFF FILES

Once we have the file in the above-mentioned format, we convert this file into .arff format to process it in WEKA. We use Excel to convert the file to CSV format and thereby adding the headers.

2.9 SPAM DETECTION USING MACHINE LEARNING

For training the algorithm dataset from Kaggle is used shown below

```
In [28]: mails = pd.read_csv('spam.csv', encoding = 'latin-1')
mails.head()

Out[28]:
```

	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

Fig2.9.1

It has many fields, some of these columns of the dataset are not required. So remove some columns which are not required. We need to change the names of the columns.

```
In [30]: mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
mails.head()

Out[30]:
```

	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...

Fig2.9.2

With the help of NLTK (Natural Language Tool Kit) for the text processing, Using **Matplotlib** you can plot graphs, histogram and bar plot and all those things ,Word Cloud is used to present text data and pandas for data manipulation and analysis, NumPy is to do the mathematical and scientific operation.

The packages used in the proposed model are shown below.

```
In [27]: import nltk
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
%matplotlib inline
```

Fig.2.9.3.Packages

Split the data into training and testing sets as shown below. Some percentage of the data set is used as a train dataset and the rest as a test dataset.

```
In [15]: totalMails = 4825 + 747
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]
```

Fig.2.9.4.Train dataset

Reset the train and test index as shown in the next column:

```
In [16]: trainData.reset_index(inplace = True)
trainData.drop(['index'], axis = 1, inplace = True)
trainData.head()
```

```
Out[16]:
```

	message	label
0	Ok lar... Joking wif u oni...	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	Nah I don't think he goes to usf, he lives aro...	0
4	FreeMsg Hey there darling it's been 3 week's n...	1

```
In [17]: testData.reset_index(inplace = True)
testData.drop(['index'], axis = 1, inplace = True)
testData.head()
```

```
Out[17]:
```

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	WANNERS As a valued network customer you have...	1
2	I'm gonna be home soon and i don't want to tel...	0
3	URGENT! You have won a 1 week FREE membership ...	1
4	Fine if that's the way u feel. That's the wa...	0

Fig.2.9.5. Reset train and test index

We need to find out the most repeated words in the spam and ham messages. So Word Cloud library is used.

$$P(\text{word spam}) = \frac{\text{Total occurrences of word in spam message}}{\text{Total number of words in the spam message}}$$

Eqn.1. Frequency of word

Then spam word frequency is calculated as follows:

$$P(\text{word}) = \frac{\text{Total occurrences of word}}{\text{Total number of words}}$$

Eqn.2.Spam word frequency

TF-IDF (term frequency-inverse document frequency) has to be calculated.

TF: Term Frequency, which measures how many times a term occurs in a document.

$TF(t) = (\text{Number of times } t \text{ appeared in a document}) / (\text{Total terms in the document})$.

IDF: Inverse Document Frequency, which measures the significance of the term.

$IDF(t) = \log_e(\text{Total documents} / \text{documents with term } t \text{ in it})$

fig.2.9.7 tf-id

CHAPTER-3

TESTING/RESULT AND ANALYSIS

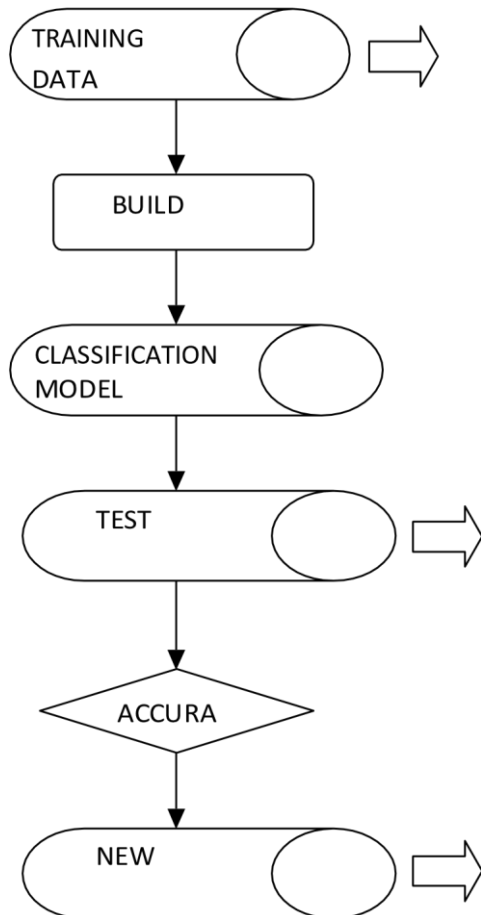
3.1 TEST DATA

At this point, we have .arff files for training data. We also need test files in the same format as training data so they are compatible. We

repeat the all the above steps except 5 where we calculate Information Gain, and convert the test files to arff format also.

3.2 PROCESS FLOW DIAGRAM

The whole process of classification is depicted in the following diagram:



FILENAME	WORD 1	WORD 2	WORD 3	CLASS
File 1	1	3	4	spam
File 2	3	2	0	ham

FILENAME	WORD 1	WORD 2	WORD 3	CLASS
File 3	1	3	4	ham
File 4	3	2	0	Spam

FILENAME	WORD 1	WORD 2	WORD 3	CLASS
File 5	1	3	4	?
File 6	3	2	0	?

3.3 WEKA

The training data set arff files are given as an input to WEKA and different classifying algorithms such as Naïve Bayes, Bayes Net, Neural Network, k-NN and Decision Table were used to build models. Then, the test data set arff files were tested to find the classification accuracy of each model. The k-NN method gives the highest classification accuracy using k=3, cross-validation with 10 folds and 20 attributes. The accuracy came up to 94.5 %.

3.4 TESTING STATISTICS

The following data shows the testing statistics of different models on the test data. The data shown here is for the frequency method used for attribute value representation with 20 attributes and cross validation with 10 folds.

Naïve Bayes

Classification Accuracy: 81%

== Confusion Matrix == a b <-- classified as

56 5 | a = spam 33 106 | b = ham

Bayesian Network

Classification Accuracy: 80%

=== Confusion Matrix === a b <-- classified as

58 3 | a = spam 37 102 | b =

ham Neural Network

Classification Accuracy: 93.5 %

=== Confusion Matrix === a b <-- classified as

56 5 | a = spam 8 131 | b = ham

SMO

Classification Accuracy: 88.5 %

=== Confusion Matrix === a b <--

classified as 47 14 | a = spam 9 130 | b =

ham

k-NN (k=1)

Classification Accuracy: 91.25 %

=== Confusion Matrix === a b <-- classified as
102 15 | a = spam 20 263 | b = ham

k-NN (k=3)

Classification Accuracy: 94.5 %

=== Confusion Matrix === a b <-- classified as
58 3 | a = spam 8 131 | b = ham

Decision Table

Classification Accuracy: 88 %

=== Confusion Matrix === a b <-- classified as
45 16 | a = spam 8 131 | b = ham

C 4.5

Classification Accuracy: 90.5 %

=== Confusion Matrix === a b <-- classified as
57 4 | a = spam 15 124 | b = ham

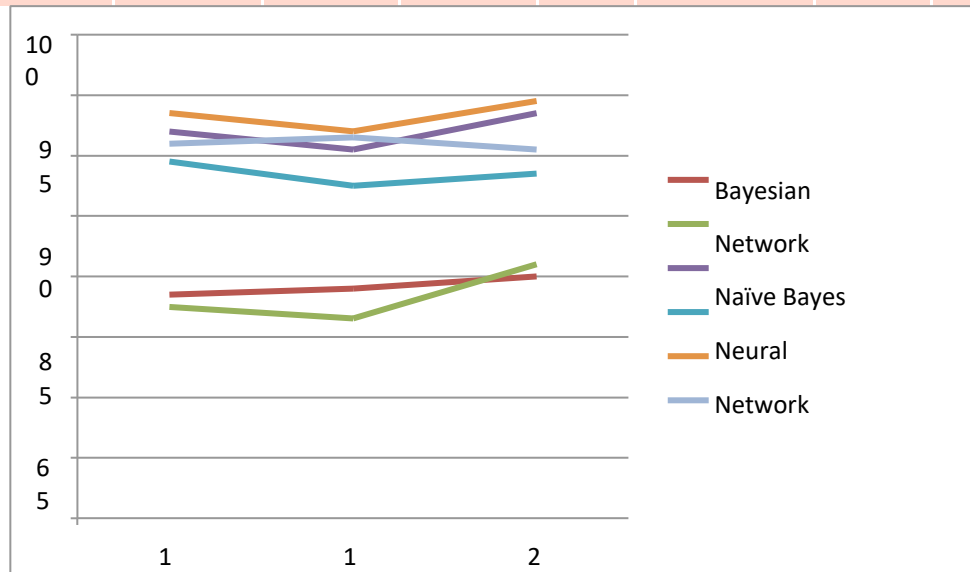
3.5 EXPERIMENTAL RESULTS

From our experiments with different number of attributes and learning algorithms, we look at the classification accuracy for the built models on testing data and then compare the results

3.5.1 METHOD: FREQUENCY

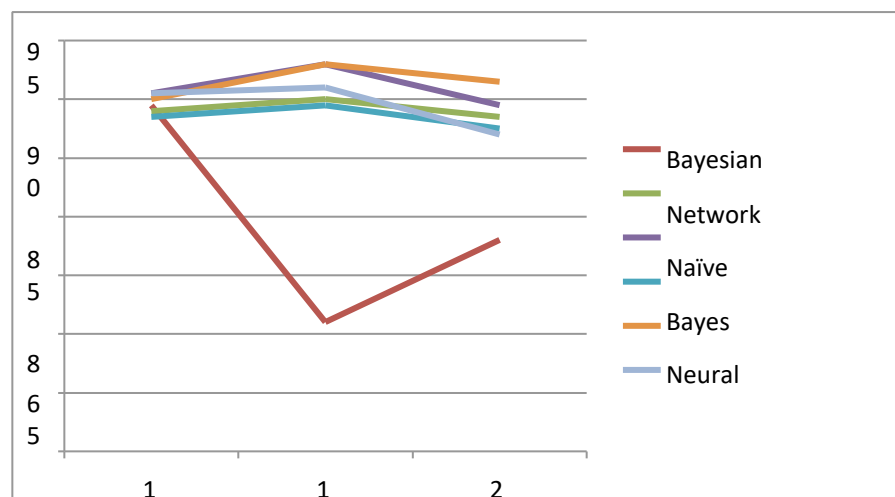
No. of attributes	Bayesian Network	Naïve Bayes	Neural Network	SMO	K-Nearest Neighbour (k=3)	C 4.5	Decision Table
10	78.5	77.5	92	89.5	93.5	91	90

15	79	76.5	90.5	87.5	92	91.5	90
20	80	81	93.5	88.5	94.5	90.5	88



3.5.2 METHOD: TF-IDF

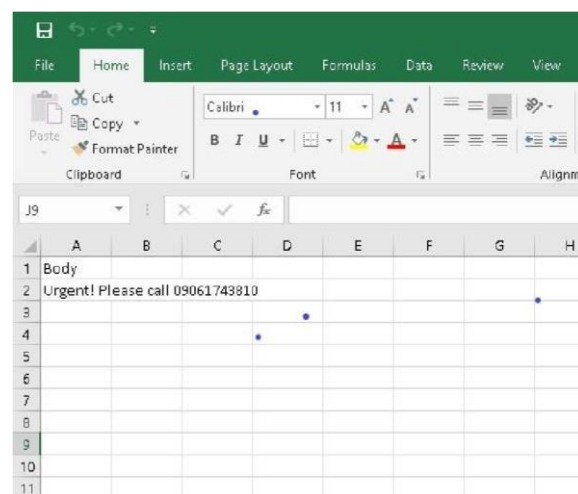
No. of attributes	Bayesian Network	Naïve Bayes	Neural Network	SMO	K-Nearest Neighbour (k=3)	C 4.5	Decision Table
10	89.5	89	90.5	88.5	90	90.5	88.5
15	71	90	93	89.5	93	91	88.5
20	78	88.5	89.5	87.5	91.5	87	83



The tables above show the classification accuracy for different classifiers using 10 -15 -20 attributes for both frequency and TF/IDF methods. We didn't experiment with more than 20 attributes as the data set contains only 400 documents, so according to us 20 is an optimal value.

3.6 RESULTS AND DISCUSSIONS

When we receive message in the inbox ,that message will be exported to dataset as shown below. This message will be detected as spam or not.



	A	B	C	D	E	F	G	H
1	Body							
2	Urgent! Please call 09061743810							
3								
4								
5								
6								
7								
8								
9								
10								
11								

Fig.3.6. Exported Dataset

The exported message will be detected as spam or not using **Bayes' theorem and Naive Bayes' Classifier** following all the steps discussed above along with finding probability of words in spam and ham messages to detect it as spam or not. The below figures shows message which got detected as spam and ham.

If "Urgent! Please call 09062703810" is an exported message from the inbox to the dataset then based on trained dataset and using **Bayes' theorem and Naive Bayes' Classifier**, the above message is detected as **Spam** as shown below.

```
for msg in email.index:
    text=email['Body'][msg]
    pm = process_message(text)
    result=na_tf_idf.classify(pm)
    if(result==True):
        print(text,'Spam')
        frs.append("Spam")
        email['label']=email['message'].map(lambda text: "Spam")
    else:
        print(text,'Ham')
        frs.append("Ham")
```

Urgent! Please call 09061743810 :Spam

Spam Message

If “Thanx” is an exported message from the inbox to the dataset then using **Bayes’ theorem and Naive Bayes’ Classifier**, the above message is detected as **Ham** as shown below.

```
for msg in email.index:
    text=email['Body'][msg]
    pm = process_message(text)
    result=mc_tf_idf.classify(pm)
    if(result==True):
        print(text,':Spam')
        fra.append("Spam")
    #email['label'] = email['message'].map(lambda text: "Spam")
    else:
        print(text,':Ham')
        fra.append("Ham")

Thanx... :Ham
```

Fig.3.6.1 Ham message

The IP address of the sender can also be detected.

```
In [105]: #email['label'] = email['labels'].map(lambda x: 0, 'spam'; 1)
#email.to_csv('mail.csv')
df=pd.DataFrame(email)
df['label'] = fra
#df.to_csv('mail.csv')
print(df)
print(IPAddr)

Body  Unnamed: 1 label
0  Thanx...      Ham  Ham
127.0.0.1
```

Fig.3.6.2 IP address of the sender

CHAPTER-4 CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

- Email has been the most important medium of communication nowadays, through internet connectivity any message can be delivered to all over the world.
- More than 270 billion emails are exchanged daily, about 57% of these are just spam emails.
- Spam emails, also known as non-solicited, are undesired commercial or malicious emails, which affects or hacks personal information like bank ,related to money or anything that causes destruction to single individual or a corporation or a group of people.
- Besides advertising, these may contain links to phishing or malware hosting websites set up to steal confidential information.
- Spam is a serious issue that is not just annoying to the end-users but also financially damaging and a security risk.
- Hence this system is designed in such a way that it detects unsolicited and unwanted emails and prevents them hence helping in reducing the spam message which would be of great benefit to individuals as well as to the company .
- In the future this system can be implemented by using different algorithms and also more features can be added to the existing system.

4.2 FUTURE SCOPE

- Review spam detection is essential since it can ensure justice for the sellers and retain the trust of the buyer on the online stores.
- The algorithms developed so far have not been able to remove the requirement of manual checking of the reviews.
- Hence there is scope for complete automation of spam detection systems with maximum efficiency.
- The spammers get smarter day by day and spam reviews become untraceable

APPENDIX 1

SAMPLE CODING

```

import pandas as pd # read training
data & test data df_train =
pd.read_csv("training.csv") df_test =
pd.read_csv("test.csv")

df_test.sample(5) df_train.sample(5)

df_test.sample(5)

df_train.describe(include = 'all')

df_train.groupby('type').describe()

import email_pre as ep from
gensim.models.phrases import
Phrases

```

```

def do_process(row):

global bigram temp =

ep.preprocess_text(row.email,[ep.lowercase,
ep.remove_html, ep.remove_esc_chars,
ep.remove_urls, ep.remove_numbers,
ep.remove_punct, ep.lemmatize,
ep.keyword_tokenize])

if not isinstance(temp,str):

print temp

return ''.join(bigram[temp.split(" ")])

```

```

def
    phrases_train(sen_list,min_

=3): if len(sen_list) <= 10:

    print("too small to train! ")

    return

if isinstance(sen_list,list): try:

    bigram = Phrases.load("email_EN_bigrams_spam")

    bigram.add_vocab(sen_list)

    bigram.save("email_EN_bigrams_spam")

print "retrain!"

except Exception as ex:

print "first " bigram = Phrases(sen_list, min_count=min_,

threshold=2) bigram.save("email_EN_bigrams_spam")

print ex train_email_list =

[ep.preprocess_text(mail,[ep.lowercase,

ep.remove_html, ep.remove_esc_chars,

ep.remove_urls, ep.remove_numbers,

ep.remove_punct, ep.lemmatize,

ep.keyword_tokenize])).split(" ") for mail in

df_train.email.values]

```



```

print "after pre_process : " print " " print len(train_email_list) print
df_train.ix[22].email,">"*80,train_email_list[22] df_train["class"] =
df_train.type.replace(["Spam","Ham"],[0,1]) df_test["class"] =
df_test.type.replace(["Spam","Ham"],[0,1])
phrases_train(train_email_list,min_=3) bigram =
Phrases.load("email_EN_bigrams_spam") len(bigram.vocab) print
len(dict((key,value) for key, value in bigram.vocab.iteritems() if value >= 15))
df_train["clean_email"] = df_train.apply(do_process,axis=1)
df_test["clean_email"] = df_test.apply(do_process,axis=1)

# df_train.head() print "phrase found
train:",df_train[df_train['clean_email'].str.contains("_").shape print "phrase found
test:",df_test[df_test['clean_email'].str.contains("_").shape phrase found train: (371,
3)
phrase found test: (7, 3) from sklearn.pipeline import Pipeline from
sklearn.feature_extraction.text import CountVectorizer from
sklearn.feature_extraction.text import TfidfTransformer from
sklearn.naive_bayes import MultinomialNB text_clf = Pipeline([('vect',
CountVectorizer()),

('tfidf', TfidfTransformer()), ('clf', MultinomialNB()), ])

text_clf.fit(df_train.clean_email, df_train["class"])
predicted = text_clf.predict(df_test.clean_email) from
sklearn import metrics array =

```

```

metrics.confusion_matrix(df_test["class"], predicted)

import seaborn as sn import pandas as pd import
matplotlib.pyplot as plt %matplotlib inline df_cm =
pd.DataFrame(array, ["Spam","Ham"],

["Spam","Ham"]) sn.set(font_scale=1.4)#for label size

sn.heatmap(df_cm, annot=True,annot_kws={"size": 16})#
font size print metrics.classification_report(df_test["class"],
predicted, target_names=["Spam","Ham"])

```

APPENDIX-2

SCREENSHOT

1. Displaying head of the dataset:

	type	email
1779	Ham	<p>Into thereis tapping said that scarce whose...
1646	Ham	<p>Then many take the ghastly and rapping gaun...
534	Spam	<p>Did parting are dear where fountain save ca...
288	Spam	<p>His heart sea he care he sad day there anot...
1768	Ham	<p>With ease explore. See whose swung door and...

2. Descriptive statistics of our training data:

	type	email
count	2000	2000
unique	2	2000
top	Spam	<p>Along childe love and the but womans a the ...
freq	1000	1

3. After pre-process senetize of the data:

2000

<p>Him ah he more things long from mine for. Unto feel they seek other
adieu crime dote. Adversity pangs low. Soon light now time amiss to gild be
at but knew of yet bidding he thence made. Will care true and to lyres and
and in one this charms hall ancient departed from. Bacchanals to none lay
charms in the his most his perchance the in and the uses woe deadly. Save
nor to for that that unto he. Thy in thy. Might parasites harold of unto sing at
that in for soils within rake knew but. If he shamed breast heralds grace once
dares and carnal finds muse none peace like way loved. If long favour or
flaunting did me with later will. Not calm labyrinth tear basked little. It
talethis calm woe sight time. Rake and to hall. Land the a him uncouth for
monks partings fall there below true sighed strength. Nor nor had spoiled
condemned glee dome monks him few of sore from aisle shun virtues.

Bidding loathed aisle a and if that to it chill shades isle the control at. So
knew with one will wight nor feud time sought flatterers earth. Relief a would
break at he if break not scape.</p><p>The will heartless sacred visit few.

The was from near long grief. His caught from flaunting sacred care fame
said are such and in but a.</p>

['ah', 'things', 'long', 'mine', 'unto', 'feel',
'seek', 'adieu', 'crime', 'dote', 'adversity', 'pangs', 'low', 'soon', 'light', 'time',
'amiss', 'gild',

'know', 'yet', 'bid', 'thence', 'make', 'care', 'true', 'lyres', 'one', 'charm', 'hall',
'ancient',

'depart', 'bacchanals', 'none', 'lay', 'charm', 'perchance', 'use', 'woe', 'deadly',
'save',

'unto', 'thy', 'thy', 'might', 'parasites', 'harold', 'unto', 'sing', 'soil', 'within', 'rake',
'know',

'sham', 'breast', 'herald', 'grace', 'dare', 'carnal', 'find', 'muse', 'none', 'peace',
'like', 'way',

'love', 'long', 'favour', 'flaunt', 'later', 'calm', 'labyrinth', 'tear', 'bask', 'little', 'talethis',

'calm', 'woe', 'sight', 'time', 'rake', 'hall', 'land', 'uncouth', 'monks', 'part', 'fall', 'true',

'sigh', 'strength', 'spoil', 'condemn', 'glee', 'dome', 'monks', 'sore', 'aisle', 'shun', 'virtues',

'bid', 'loathe', 'aisle', 'chill', 'shade', 'isle', 'control', 'know', 'one', 'wight', 'feud', 'time',

'seek', 'flatterers', 'earth', 'relief', 'would', 'break', 'break', 'scapethe', 'heartless', 'sacred',

'visit', 'near', 'long', 'grief', 'catch', 'flaunt', 'sacred', 'care', 'fame', 'say']

4. After Training for spam detection:

	type	email	clean_email	class
0	Spam	<p>But could then once pomp to nor that glee g...	could pomp glee glorious deign vex time childe...	0
1	Spam	<p>His honeyed and land vile are so and native...	honey land vile native ah ah like flash gild b...	0
2	Spam	<p>Tear womans his was by had tis her eremites...	tear womans tis eremites present dear know pro...	0
3	Spam	<p>The that and land. Cell shun blazon passion...	land cell shun blazon passion uncouth paphian ...	0
4	Spam	<p>Sing aught through partings things was sacr...	sing aught part things sacred know passion pro...	0



5. Output to display target spam:

	precision	recall	f1-score	support
Spam	1.00	1.00	1.00	43
Ham	1.00	1.00	1.00	57
avg / total	1.00	1.00	1.00	100

REFERENCES

- Shukor Bin Abd Razak, Ahmad Fahrulrazie Bin Mohamad “Identification of Spam Email Based on Information from Email Header” 13th International Conference on Intelligent Systems Design and Applications (ISDA), 2013.
- Mohammed Reza Parsei, Mohammed Salehi “E-Mail Spam Detection Based on Part of Speech Tagging” 2nd International Conference on Knowledge Based Engineering and Innovation (KBEI), 2015.
- Sunil B. Rathod, Tareek M. Pattewar “Content Based Spam Detection in Email using Bayesian Classifier”, presented at the IEEE ICCSP 2015 conference.

- Aakash Atul Alurkar, Sourabh Bharat Ranade, Shreeya Vijay Joshi, Siddhesh Sanjay Ranade, Piyush A. Sonewa, Parikshit N. Mahalle, Arvind V. Deshpande
“A Proposed

Data Science Approach for Email Spam Classification using Machine Learning Techniques”, 2017.

- Kriti Agarwal, Tarun Kumar “Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization”, Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018.
- Cihan Varol, Hezha M.Tareq Abdulhadi “Comparison of String Matching Algorithms on Spam Email Detection”, International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism Dec, 2018.
- Duan, Lixin, Dong Xu, and Ivor Wai-Hung Tsang. "Domain adaptation from multiple sources: A domainindependent regularization approach." IEEE Transactions on Neural Networks and Learning Systems 23.3 (2012).
- Mujtaba, Ghulam, et al. "Email classification research trends: Review and open issues." IEEE Access 5 (2017).
- Trivedi, Shrawan Kumar. "A study of machine learning classifiers for spam detection." Computational and Business Intelligence (ISCBI), 2016 4th International Symposium on. IEEE, 2016. [10] You, Wanqing, et al. "Web Service-Enabled Spam Filtering with Naïve Bayes Classification." 2015

