SHILLONG COLLEGE
BOYCE ROAD, LAITUMKHRAH
SHILLONG-793003, MEGHALAYA

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF BACHELOR OF COMPUTER APPLICATION

Submitted by,
Student name:Narola. D .Dhar
Roll number: P1500073
Registration Number:10471 of 2014-2015

Under the Guidance of
Mrs A Mitri

NORTH EASTERN HILL UNIVERSITY
CERTIFIED THAT THIS IS A BONAFIDE RECORD OF THE PROJECT

ENTITLED

A SIMPLE SPAM FILTER
USING NAIVE BAYES CLASSIFIER

17/4/17 | 17/4/17 | 17/04/17
PROJECT GUIDE | HEAD OF DEPARTMENT | EXAMINER

Viva Voce held on: 17th of April, 2017

CERTIFICATE

This is to certify that the Project Report entitled **"A simple spam filter using Naive Bayes Classifier"** submitted by Narola. D. Dhar in partial fulfillment for the award of the Degree of Computer Application for the year 2017 to the "North Eastern Hill University" is a record of bonafide work carried out by her under our guidance and supervision.

The results embodied on this project have not been submitted to any other University or Institute for the award of any Degree or Diploma.

ACKNOWELEDGEMENT

I deeply express my gratitude to our Head of Department , Mrs A. Mitri who have allowed me to do a project based on Machine Learning. I thank all the people whose ceaseless cooperation made it possible, whose constant guidance and encouragement help me to not give up in the process of working on the project..

I am grateful to my project guide Mrs. A. Mitri for the guidance, inspiration and constructive suggestions that helped me in the preparation of this project. I am incredibly thankful to all my dear teachers who has been a great inspiration to all of us.

I am also thankful to my dear classmates, friends and family who have helped me in the working of the project.

TABLES OF CONTENT

PAGES

1. INTRODUCTION AND OBJECTIVE

1.1 Introduction

Spam is flooding the Internet with many copies of the same message, in an attempt to force the message on people who would not otherwise choose to receive it. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. Spam costs the sender very little to send -- most of the costs are paid for by the recipient or the carriers rather than by the sender.

Email spam is any email that meets the following three criteria:

- **Anonymity:** The address and identity of the sender are concealed
- **Mass Mailing:** The email is sent to large groups of people
- **Unsolicited:** The email is not requested by the recipients

Email spam targets individual users with direct mail messages. Email spam lists are often created by scanning Usenet postings, stealing Internet mailing lists, or searching the Web for addresses. Email spams typically cost users money out-of-pocket to receive. Many people - anyone with measured phone service - read or receive their mail while the meter is running, so to speak. Spam costs them additional money. On top of that, it costs money for ISPs and online services to transmit spam, and these costs are transmitted directly to subscribers.

Anti-spam techniques are used to prevent email spam (unsolicited bulk email). There are many anti-spam techniques available and one of them is using machine learning techniques. And, the algorithm this project is based on is the Naive Bayes Theorem which is implemented in the simplest way possible and these methods are part of Machine Learning techniques.

1.2 OBJECTIVES OF THE PROJECT

The web is growing day by day. People are aware of the internet and it's technology. The high rise in spams in all web medias ha become a menace. Presently, there are many good spam filters available that provides accurate ways to negate spam from ham. The objective of this project is to understand Machine Learning and how we can do things through it.So, we start first off with a simple spam filter using Naive Bayes Classifier. The main objective is to understand and from this small project we will be able to progress more with time in order to be able to create super smart applications using the methods of Machine Learning creating simpler solutions for people.

2. SYNOPSIS

2.1 **Synopsis**

PROJECT TITLE
“A SIMPLE SPAM FILTER USING NAIVE BAYES CLASSIFIER”

We humans are all tired of spam, those unwanted mails that we get everytime we open up a computer, our phones, our laptops. These spam are a menace leading to malicious activities sometimes even loss of ones own identity. We need to solve this.
The project main objective is to classify ham and spam. In this project we use machine learning techniques called the supervised learning to classify the mails. We use concepts like classification. It includes dataset that are divided into the training set and the testing set. But in this project, we can also test by inputing our own message to be classified. We use the Naive bayes classifier which is an algorithm based on Bayes Theorem. It is a statistical based algorithm. What we actually did was “multiply each feature give spam together and multiply that by the prior probability for spam and repeat the same for ham(i.e multiply each feature give ham together and multiply that by the prior probability for ham).
Now we have two numbers which can be normalized to probabilities by dividing both each by the total of both. That will give the desired probabilities of both ham and spam which we can compare and the higher probability is the result of the outcome of the mail.
The testing set is used to calculate the accuracy of the classified results.
The training dataset used is comprised of total of 2501 mails where 501 are ham and 2000 are spam which are already labeled. The testing set are comprised of four test path where the first test path comprised of hard spam mails, the second,third and fourth all comprised of hard spam mails.

2.2 **Requirements**

HARDWARE

- OS: minimum 64 bit Ubuntu 15.10
- Ram Memory: minimum 1.8 GB (the higher the better)
- Processor: Intel core i3
- CPU:2.20 GHz X 4

SOFTWARE

- Anaconda2 (Link to download- https://www.continuum.io/anaconda)

  Anaconda is a free open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its package management system is *conda.*

- *Python 2.7*

  Python is a programming language. It's used for many different applications. It's used in some high schools and colleges as an introductory programming language because **Python**

is easy to learn, but it's also used by professional software developers at places such as Google, NASA, and Lucasfilm Ltd.

- *Spyder IDE*

  Spyder (formerly Pydee) is an open source cross-platform integrated development environment (IDE) for scientific programming in the **Python** language. **Spyder** integrates NumPy, SciPy, Matplotlib and IPython, as well as other open source software.

2.3 **Conclusion**

Using the above concepts and algorithms, we will create the SIMPLE SPAM FILTER. This project will be able to provide insight on the interesting field of Machine Learning irrespective of the final results.

3. ANALYSIS

3.1 Information Gathering

What is an email spam?

Email spam, also known as junk email, is unsolicited bulk messages sent through email. The use of spam has been growing in popularity since the early 1990s and is a problem faced by most email users. Recipients of spam often have had their email addresses obtained by spambots, which are automated programs that crawl the internet looking for email addresses. Spammers use spambots to create email distribution lists. A spammer typically sends an email to millions of email addresses, with the expectation that only a small number will respond or interact with the message.

What is a spam filter?

A *spam filter* is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox. Like other types of *filter*ing programs, a *spam filter* looks for certain criteria on which it bases judgments.

What is machine learning?

Machine learning is turning data into information.
Machine learning lies at the intersection of computer science, engineering, and statistics and often appears in other disciplines. It can be applied to many fields from politics to geosciences. It's a tool that can be applied to many problems. Any field that needs to interpret and act on data can benefit from machine learning techniques.
Machine learning uses statistics. It is a subfield of artificial intelligence where the main objective is to teach machines to learn like humans but instead of learning from experience like humans, they will learn from data.

Machine Learning will be more popular in the future

In the last half of the twentieth century the majority of the workforce in the developed world has moved from manual labor to what is known as knowledge work. The clear definitions of "move this from here to there" and "put a hole in this" are gone. Things are much more ambiguous now; job assignments such as "maximize profits," "minimize risk," and "find the best marketing strategy" are all too common. The fire hose of information available to us from the World Wide Web makes the jobs of knowledge workers even harder. Making sense of all the data with our job in mind is becoming a more essential skill. Machine learning will help us get through all the data and extract some information from it.

Supervised Learning

Where a program is "trained" on a pre-defined dataset. Based off its training data the program can make accurate decisions when given new data. Example: Using a training set of human tagged positive, negative and neutral tweets to train a sentiment analysis classifier.

Classification

A sub-category of Supervised Learning, Classification is the process of taking some sort of input and assigning a label to it. Classification systems are usually used when predictions are of a discrete, or "yes or no" nature. Example: Mapping a picture of someone to a male or female classification, classifying a mail to be ham or spam.

The main formula's used are:-

1. Bayes Theorem or Bayes Rule

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{P(B \cap A)}{P(B \cap A) + P(B^c \cap A)}$$

2.Naive Bayes

$$\underset{\text{Posterior Probability}}{P(c\,|\,x)} = \frac{\overset{\text{Likelihood}}{P(x\,|\,c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c\,|\,X) = P(x_1\,|\,c) \times P(x_2\,|\,c) \times \cdots \times P(x_n\,|\,c) \times P(c)$$

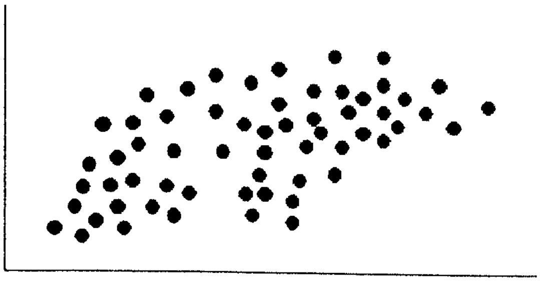
Naive Bayes Classifier

Naive Bayes is a collection of classification algorithms based on Bayes Theorem. It is not a single algorithm but a family of algorithms that all share a common principle, that every feature being classified is independent of the value of any other feature. So for example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A Naive Bayes classifier considers each of these "features" (red, round, 3" in diameter) to contribute independently to the probability that the fruit is an apple, regardless of any correlations between features. Features, however, aren't always independent which is often seen as a shortcoming of the Naive Bayes algorithm and this is why it's labeled "naive".

Although it's a relatively simple idea, Naive Bayes can often outperform other more sophisticated algorithms and is extremely useful in common applications like spam detection and document classification.

In a nutshell, the algorithm allows us to predict a class, given a set of features using probability. So in another fruit example, we could predict whether a fruit is an apple, orange or banana (class) based on its colour, shape etc (features).

To demonstrate the concept of Naïve Bayes Classification, consider the example given below:



As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently existing objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

Prior Probability of GREEN: `number of GREEN objects / total number of objects`

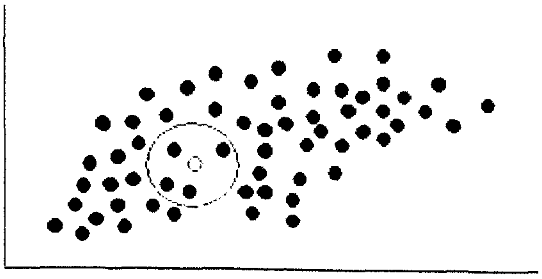
Prior Probability of RED: `number of RED objects / total number of objects`

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:

Prior Probability for GREEN: `40 / 60`

Prior Probability for RED: `20 / 60`

Having formulated our prior probability, we are now ready to classify a new object (WHITE circle in the diagram below). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:



$$Likelihood\ of\ X\ given\ GREEN \propto \frac{Number\ of\ GREEN\ in\ the\ vicinity\ of\ X}{Total\ number\ of\ GREEN\ cases}$$

$$Likelihood\ of\ X\ given\ RED \propto \frac{Number\ of\ RED\ in\ the\ vicinity\ of\ X}{Total\ number\ of\ RED\ cases}$$

From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:

$$Probability\ of\ \ X\ \ given\ GREEN \propto \frac{1}{40}$$

$$Probability\ of\ \ X\ \ given\ RED \propto \frac{3}{20}$$

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

$$Posterior\ \ probability\ of\ \ X\ being\ GREEN \propto$$
$$Prior\ \ probability\ of\ GREEN \times Likelihood\ of\ \ X\ \ given\ GREEN$$
$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$Posterior\ \ probability\ of\ \ X\ being\ RED \propto$$
$$Prior\ \ probability\ of\ RED \times Likelihood\ of\ \ X\ \ given\ RED$$
$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Finally, we classify X as RED since its class membership achieves the largest posterior probability.

We will use this concept on classifying a mail into spam or ham.

*In simple terms, a **naive Bayes classifier** assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class* *variable. ... It's **called naive** because it makes the assumption that all attributes are independent of each other.*

Pros and cons of Naive Bayes:

Advantages

- It's relatively simple to understand and build
- It's easily trained, even with a small dataset
- It's fast!
- It's not sensitive to irrelevant features

Disadvantages

- **It assumes every feature is independent, which isn't always the case**

3.2 ALREADY EXISTINING SOFTWARES

1. SPAM ASSASIN

2. SPAMBAYES

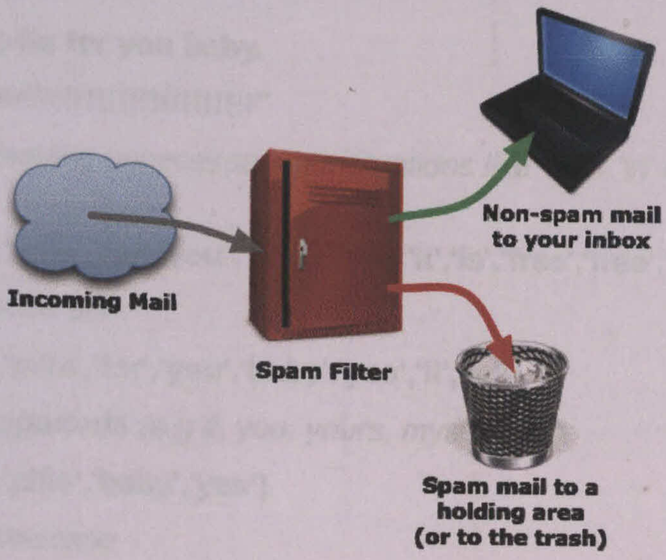
3. Spamihilator

4.POPFile

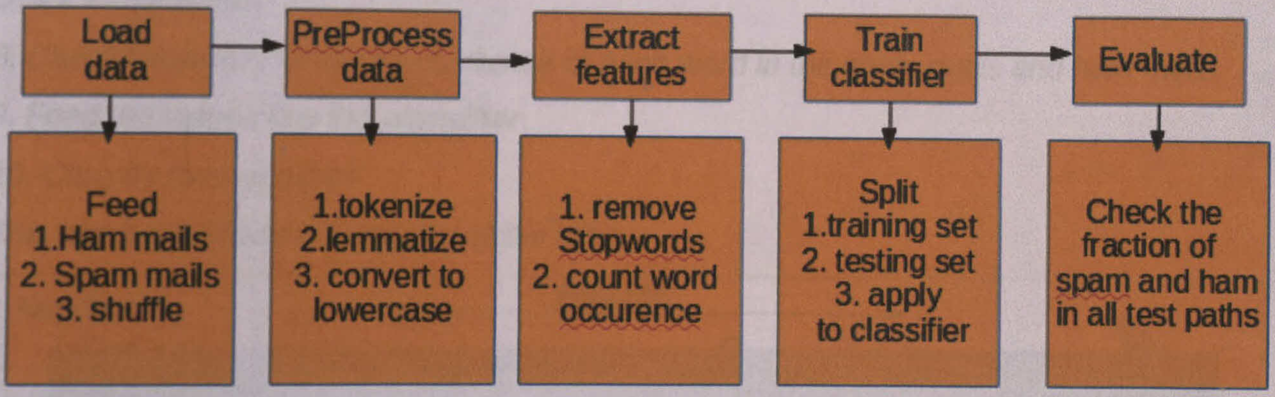
5.Spamfence and so on

4. DESIGN AND IMPLEMENTATION

4.1 DESIGN


Incoming Mail
Spam Filter
Non-spam mail to your inbox
Spam mail to a holding area (or to the trash)

A Block diagram to explain the implementation


Load data
PreProcess data
Extract features
Train classifier
Evaluate
Feed 1.Ham mails 2. Spam mails 3. shuffle
1.tokenize 2.lemmatize 3. convert to lowercase
1. remove Stopwords 2. count word occurence
Split 1.training set 2. testing set 3. apply to classifier
Check the fraction of spam and ham in all test paths

We will be working with text data, we will be using the Python-based library Natural Language Toolkit (NLTK), which has rich functionality in natural language processing tasks.

** We import the toolkit*
**We import the corpus which contains the stopwords*
**We import os so that we can import folders,files etc.*
**We insert the spam and ham mails and shuffle them*

How is the spam and ham classified?
1.We entered message

“100% free Choco pills for you baby.

Yes it is free free free!!!!!!!!!!!!!!!!!!!!!”

*2. We **tokenize** it removing unnecessary punctuations like '=' or '\n' using the wordpunct.tokenizer*

{'100','free','Choco','pills','for','you','baby','yes''it','is','free','free','free'}

*3. **lemmatize** the above set*

{'100','free','Choco','pills','for','you','baby','yes','it','is'}

*4. We remove the **stopwords** (e.g if, you, yours, myself etc.)*

{'100','free','Choco','pills','baby','yes'}

5. We convert it to lowercase

{'100','free','choco','pills','baby','yes'}

6. We tabulate it with the training set.

*7. These non-stopwords are compared with a **dictionary** of ham and spam words to check it's likeliness*

8. Check frequency or word occurrence of each word in the spam mails and ham mails

*9. Feed the values into the **classifier***

10. Classify ham or spam

This can be illustrated by looking at the table:-

New Data

Features

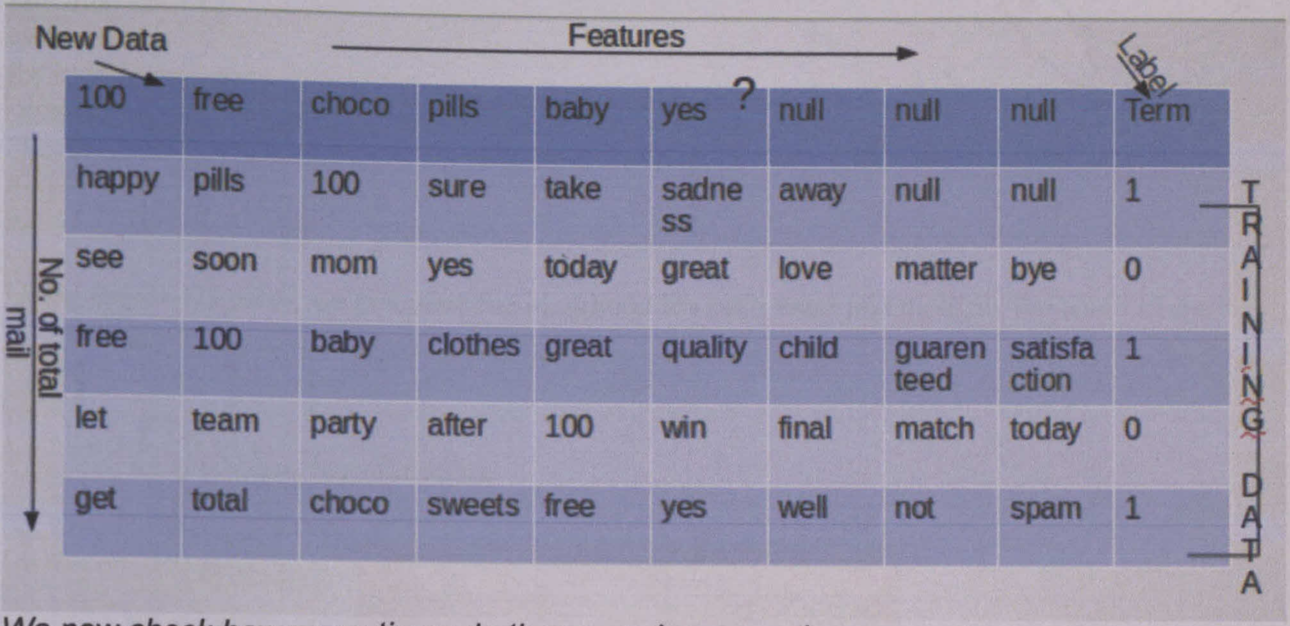
Label

100	free	choco	pills	baby	yes ?	null	null	null	Term
happy	pills	100	sure	take	sadness	away	null	null	1
see	soon	mom	yes	today	great	love	matter	bye	0
free	100	baby	clothes	great	quality	child	guarenteed	satisfaction	1
let	team	party	after	100	win	final	match	today	0
get	total	choco	sweets	free	yes	well	not	spam	1

No. of total mail

TRAINING DATA

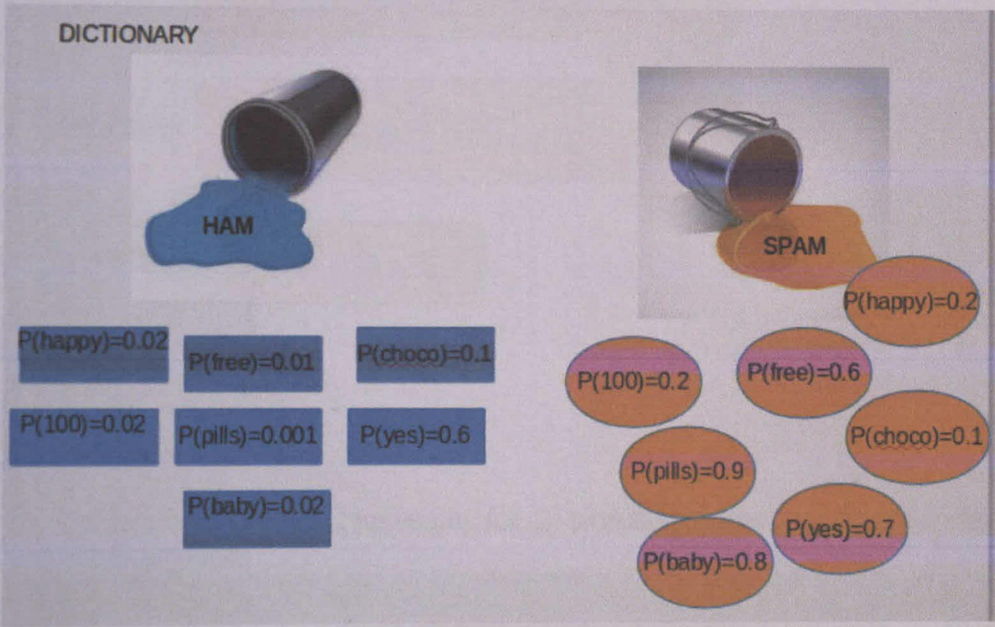
This can be illustrated by looking at the table:-



New Data
Features
Label
No. of total mail
TRAINING DATA

100	free	choco	pills	baby	yes ?	null	null	null	Term
happy	pills	100	sure	take	sadness	away	null	null	1
see	soon	mom	yes	today	great	love	matter	bye	0
free	100	baby	clothes	great	quality	child	guarenteed	satisfaction	1
let	team	party	after	100	win	final	match	today	0
get	total	choco	sweets	free	yes	well	not	spam	1

We now check how many times do these word occur as ham or spam

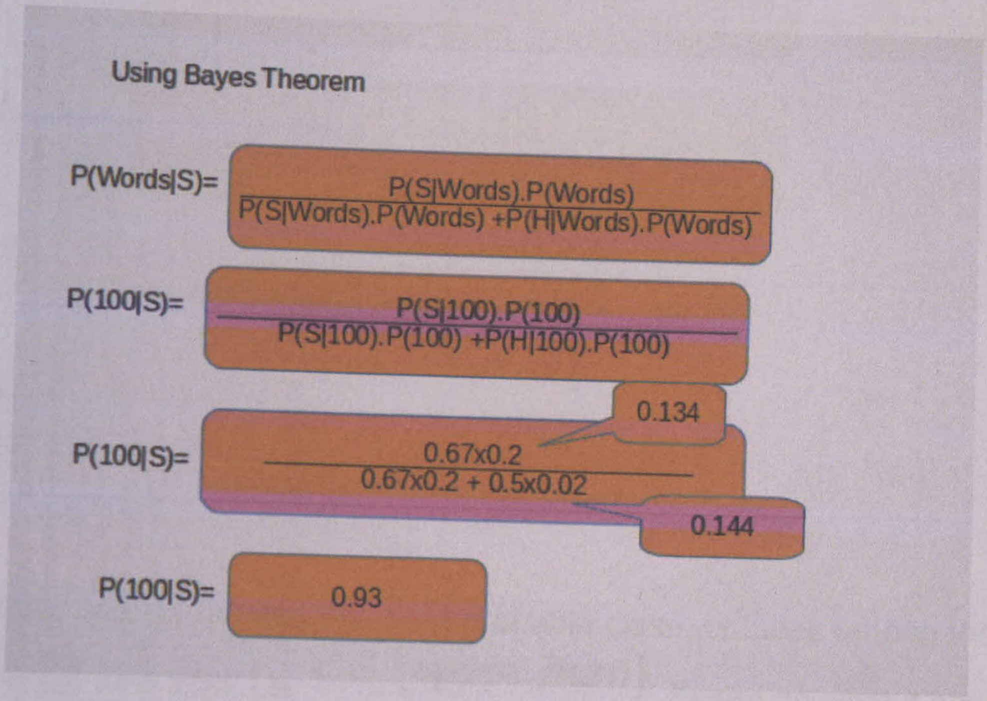


The dictionary is are already trained data set where two folders of spam and ham and their probabilities are calculated .Already trained data where prior probability of Spam is 0.2 and prior probability of Ham is 0.8

The dictionary approach is used to create this spamfilters. There are many of such dictionaries available online that contains words and phrases that trigger spam
(e.g. E-Commerce Spam Triggers List 18). Python enable developers to
easily create lexicons and dictionaries as it has its own dedicated data structure for building and retrieving dictionaries using key:value pairs. The following example illustrates how a dictionary can be created and retrieved in Python:

```
my_dict = {
'key1': 'value1',
'key2': 'value2',
'key3': 'value3'
}
my_dict['key1']
# Out: 'value
for item in my_dict:
print item
#key3
#key2
#key1
```

Using Bayes Theorem, we calculate the likelihood for each word and multiply the result of each feature together



Using Bayes Theorem

$$P(Words|S)= \frac{P(S|Words).P(Words)}{P(S|Words).P(Words) +P(H|Words).P(Words)}$$

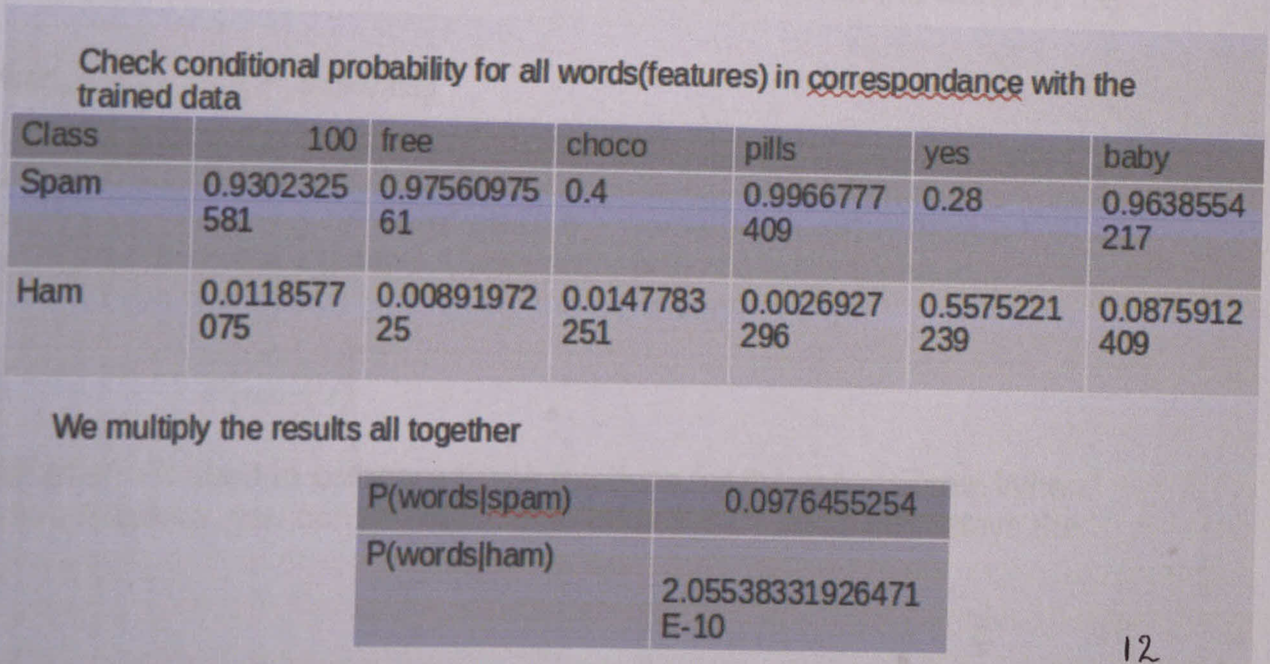
$$P(100|S)= \frac{P(S|100).P(100)}{P(S|100).P(100) +P(H|100).P(100)}$$

$$P(100|S)= \frac{0.67x0.2}{0.67x0.2 + 0.5x0.02}$$

0.134

0.144

P(100|S)= 0.93



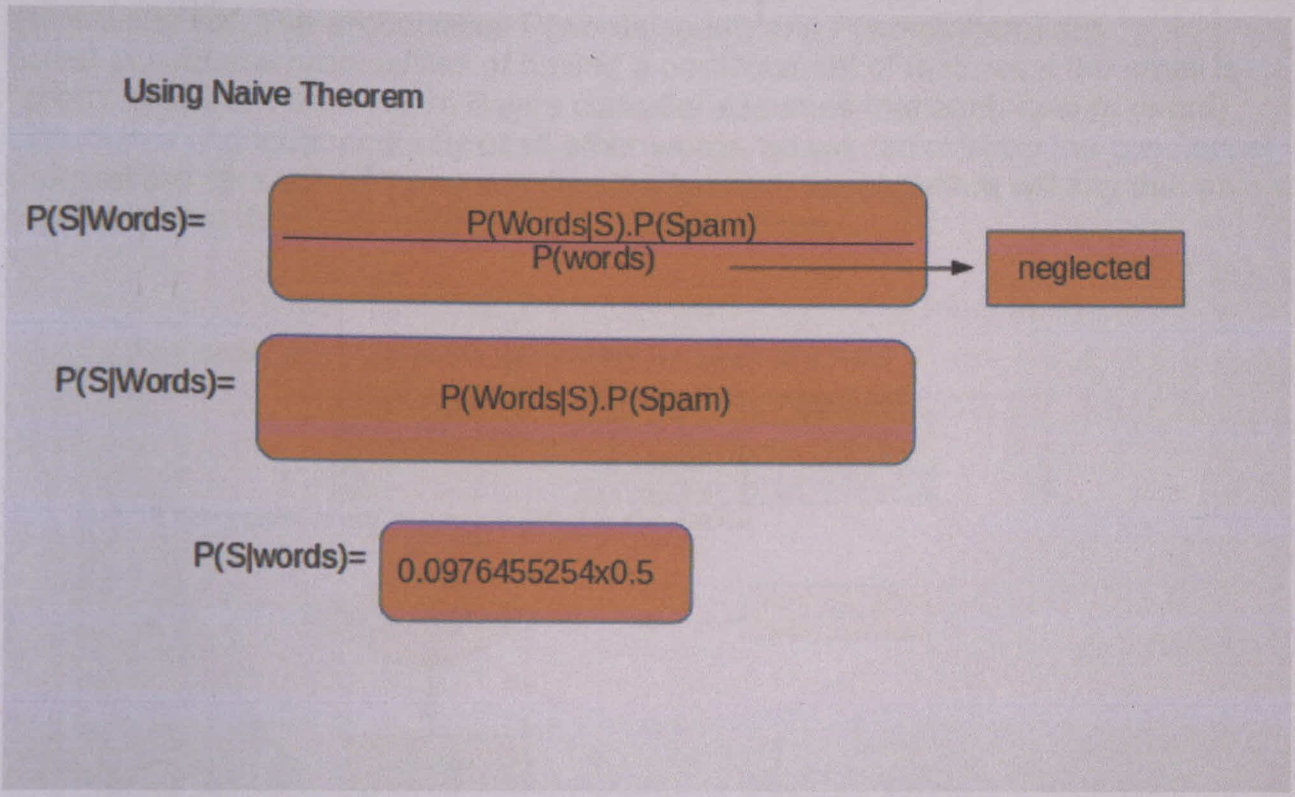
Check conditional probability for all words(features) in correspondance with the trained data

Class	100	free	choco	pills	yes	baby
Spam	0.9302325581	0.9756097561	0.4	0.9966777409	0.28	0.9638554217
Ham	0.0118577075	0.0089197225	0.0147783251	0.0026927296	0.5575221239	0.0875912409

We multiply the results all together

P(words|spam)	0.0976455254
P(words|ham)	2.05538331926471E-10

Feed the values into the classifier



Using Naive Theorem

P(S|Words)= $\frac{P(Words|S).P(Spam)}{P(words)}$ → neglected

P(S|Words)= P(Words|S).P(Spam)

P(S|words)= 0.0976455254x0.5

The classifier tries to choose the most probable class, or label, among the two classes, spam and ham, i.e. $c \in \{spam, ham\}$ based on what it has learned about the features (presence or frequency of words in the emails of each type). More precisely, it's trying to choose the most probable class given the words in the e-mail:

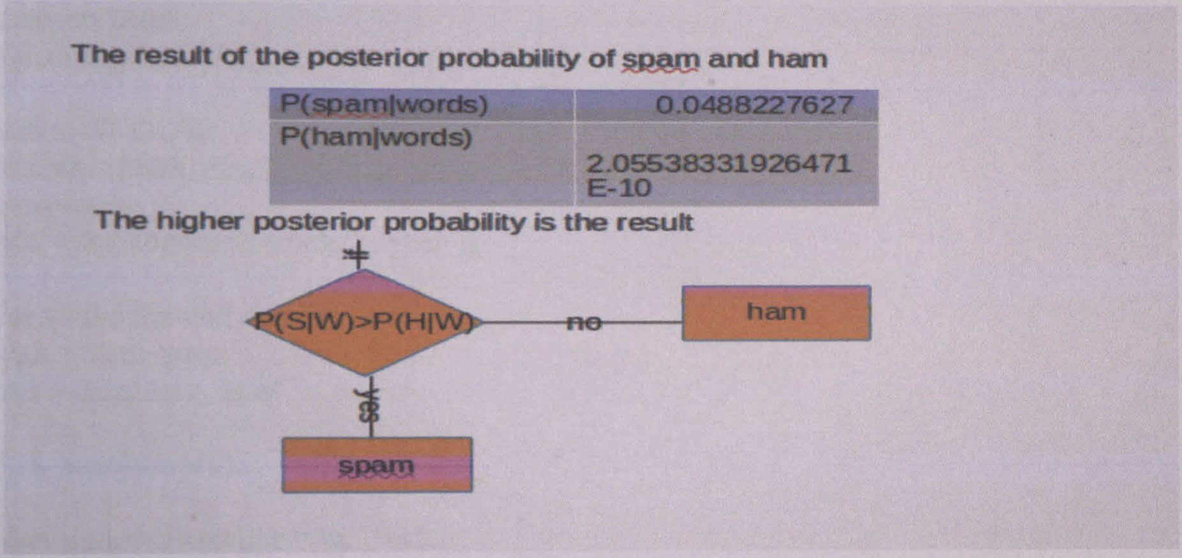
$$\hat{c} = argmax_{c\in\{spam,ham\}} P(c|words)$$

The classifier will assign the class (denoted as **c** with a hat) it will choose among the two classes by looking which of the two probabilities – "spam" given the words in the email **P(spam | words)**, or "ham" given the words in the email **P(ham | words)** – is higher (thus the *argmax*). These probabilities cannot be directly estimated, but Bayes rule allows us to swap the conditions and get:

$$P(c|words) = \frac{P(words|c)P(c)}{P(words)}$$

Now the classifier will need to compare these fractions for the two classes. When comparing two fractions, you can disregard the denominator because it stays the same for both classes, and directly compare the product **P(words|spam)P(spam)** with **P(words|ham)P(ham)**.

$\hat{c} = argmax_{c\in\{spam,ham\}} P(words|c)P(c)$ Probabilities P(spam) and P(ham) are called the *prior probabilities,* and they show the distribution of "spam" and "ham" classes in the training set. The probabilities P(words|spam) and P(words|ham) are called *conditional probabilities* of having a particular set of features if the email is "spam" or if it is "ham". Naive Bayes classifier assumes that each feature (word) occurs in a text independently of all other words, so we can multiply the conditional probabilities for each of the words directly. In short, the algorithm will say that an email is **spam** if P(S|W) > P(H|W) and ham otherwise.



The result of the posterior probability of spam and ham

P(spam|words)	0.0488227627
P(ham|words)	2.05538331926471 E-10

The higher posterior probability is the result

In the training set, P(SPAM)= 0.2 and P(HAM)=0.8 because our training set consists of 2501 mails 501 spam and 2000 ham mails.

*** On testing the data, our test path is made up of four mails folders and we run the classifier on [5:-1] of the folders and we get the fractions of all folder which is calculated as:-**

```
if spam_probability > ham_probability:
results[SPAM] += 1
else:
results[HAM] += 1

total_files = results[SPAM] + results[HAM]
spam_fraction = float(results[SPAM]) / total_files
ham_fraction = 1 - spam_fraction
```

Which shows the fraction of spam existing on each folder

5. SOURCE CODE

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 19 10:01:52 2016

@author: rhodel
"""
#training##########################################################################################
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import wordpunct_tokenize

# for reading all the files
from os import listdir
from os.path import isfile, join

# add path to NLTK file
nltk.data.path = ['nltk_data']
# load stopwords
stopwords = set(stopwords.words('english'))

# path for all the training data sets
spam_path = 'data/spam/'
ham_path = 'data/easy_ham/'

def get_all_words(email):

    #remove punctuations like =,\n, ' ' etc.
    all_words = set(wordpunct_tokenize(email.replace('=\\n', ").lower()))

    # remove the stopwords
    eml_words = [word for word in all_words if word not in stopwords and len(word) > 2]
    return eml_words

def get_mail_from_a_file(file_name):

    """
    Returns the entire mail as a string from the given file.
    """

    email = "

    with open(file_name, 'r') as mail_file:

        for line in mail_file:
            # the contents of the actual mail start after the first newline
            # so find it, and then extract the words
            if line == '\n':
```

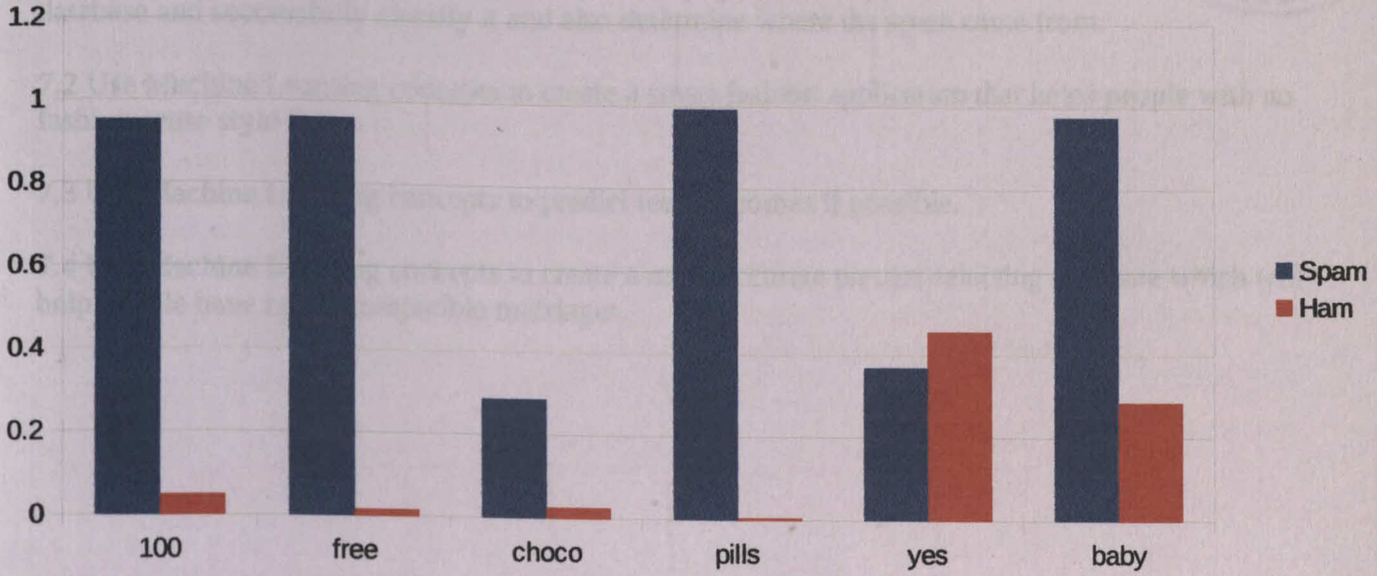
```
        # make a string out of the remaining lines
        for line in mail_file:
            email += line

    return email

def make_training_set(path):

    """
    Returns a dictionary of <term>: <occurrence> of all
    the terms in files contained in the directory specified by path.
    path is mainly directories to the training data for spam and ham folders.
    occurrence is the percentage of documents that have the 'term' in it.
    frequency is the total number of times the 'term' appears across all the
    documents in the path
    """

    # initializations
    training_set = {}

    mails_in_dir = [mail_file for mail_file in listdir(path) if isfile(join(path, mail_file))]

    # count of cmds in the directory
    cmds_count = 0
    # total number of files in the directory
    total_file_count = len(mails_in_dir)
    print ' '
    '''
    print 'The total file count is',total_file_count
    print '.'
    print '.'
    print '.'
    '''

    for mail_name in mails_in_dir:

        if mail_name == 'cmds':
            cmds_count += 1
            continue

        # get the message in the mail
        email = get_mail_from_a_file(path + mail_name)

        # we have the message now
        # get the words in the message
        terms = get_all_words(email)

        # what we're doing is tabulating the number of files
        # that have the word in them
        # add these entries to the training set
        for term in terms:
            if term in training_set:
                training_set[term] = training_set[term] + 1
                # print 'noelse'
                # print training_set[term]
            else:
                training_set[term] = 1
                #print 'else'
                #print training_set[term]
    # reducing the count of cmds files from file count
    total_file_count -= cmds_count
    # print 'The total file count is', total_file_count
    # calculating the occurrence for each term
    for term in training_set.keys():
        training_set[term] = float(training_set[term]) / total_file_count
        # print training_set[term]

    return training_set

print "
print 'Loading training sets...',
spam_training_set = make_training_set(spam_path)
ham_training_set = make_training_set(ham_path)
print 'done.'
'''
print spam_training_set
print '.'
print ham_training_set
print '.'
'''
print "
#classifier##################################################################
def classify(email, training_set, prior = 0.5, c = 3.7e-4):

    """
    Returns the probability that the given message is of the given type of
    the training set.
    """

    msg_terms = get_all_words(email)
    #print msg_terms

    msg_probability = 1

    for term in msg_terms:
        if term in training_set:
            msg_probability *= training_set[term]
            #print msg_probability
        else:
            msg_probability *= c
            #print msg_probability
    return msg_probability * prior




mail_msg = raw_input('Enter the message to be classified:')
print "

#
## 0.2 and 0.8 because the ratio of samples for spam and ham were the 0.2-0.8
spam_probability = classify(mail_msg, spam_training_set, 0.2)
ham_probability = classify(mail_msg, ham_training_set, 0.8)
if spam_probability > ham_probability:
   print 'Your mail has been classified as SPAM.'
else:
    print 'Your mail has been classified as HAM.'
print "
print 'The spam probability of this message is:', spam_probability
print 'The ham probability of this message is:', ham_probability
#testing
SPAM = 'spam'
HAM = 'ham'

# change it to the type of mails you want to classify
# path to the hard ham mails
spam2_path = 'data/spam_2/'
ham2_path = 'data/easy_ham_2/'
hard_ham2_path = 'data/hard_ham_2/'
hard_ham_path = 'data/hard_ham/'

test_paths = [spam2_path, ham2_path, hard_ham_path, hard_ham2_path]

for mail_path in test_paths:

    mails_in_dir = [mail_file for mail_file in listdir(mail_path) if isfile(join(mail_path, mail_file))]

    results = {}
    results[SPAM] = 0
    results[HAM] = 0

    print 'Running classifier on files in', mail_path[5:-1], '...'

    for mail_name in mails_in_dir:
```

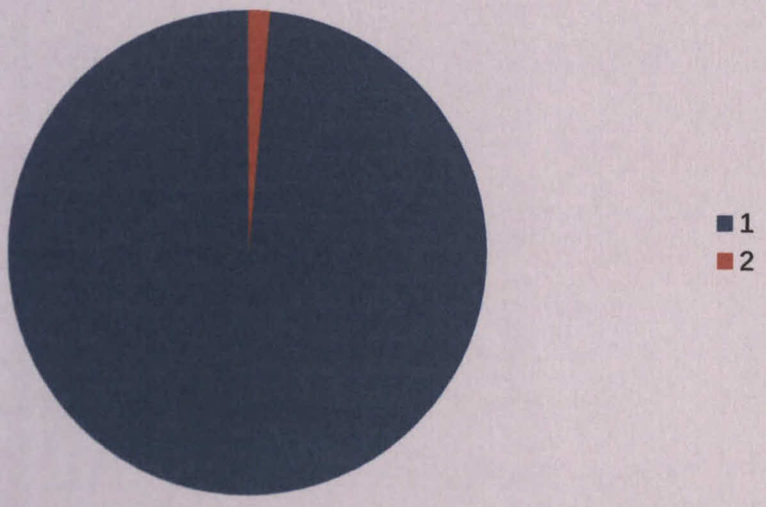
```
    if mail_name == 'cmds':
        continue

    mail_msg = get_mail_from_a_file(mail_path + mail_name)

    # 0.2 and 0.8 because the ratio of samples for spam and ham were the same
    spam_probability = classify(mail_msg, spam_training_set, 0.2)
    ham_probability = classify(mail_msg, ham_training_set, 0.8)

    if spam_probability > ham_probability:
        results[SPAM] += 1
    else:
        results[HAM] += 1

    total_files = results[SPAM] + results[HAM]
    spam_fraction = float(results[SPAM]) / total_files
    ham_fraction = 1 - spam_fraction

print 'The fraction of spam messages =', spam_fraction
print 'The fraction of ham messages =', ham_fraction
```

6. RESULT/OUTPUT

```
Loading training sets...

done.

Enter the message to be classified:100% free choco pills for you baby. It is free free free!!!!!!!

Your mail has been classified as SPAM.

The spam probability of this message is: 4.4077952e-13
The ham probability of this message is: 9.29144549376e-15

If New mail is not feeded by oneself
We test using the test path which are already created to test
Running classifier on files in spam_2 ...
The fraction of spam messages = 0.739441660702
The fraction of ham mesisages = 0.260558339298

If New mail is not feeded by oneself
We test using the test path which are already created to test
Running classifier on files in easy_ham_2 ...
The fraction of spam messages = 0.0385714285714
The fraction of ham mesisages = 0.961428571429

If New mail is not feeded by oneself
We test using the test path which are already created to test
Running classifier on files in hard_ham ...
The fraction of spam messages = 0.0763052208835
The fraction of ham mesisages = 0.923694779116

If New mail is not feeded by oneself
We test using the test path which are already created to test
Running classifier on files in hard_ham_2 ...
The fraction of spam messages = 0.0927419354839
The fraction of ham messages = 0.907258064516
```

BAR DIAGRAM COMPARING PROBABILITIES OF EACH FEATURE





Where, 1 is Spam and 2 is Ham

There are many loopholes that need to be fixed in this project like:-

1. false positives.
2. false negatives.
3. comparing with the threshold.
4. good visualization using anaconda packages like matplotlib.
5. accuracy test compared with other types of classifiers. Overall, this project has increased more understanding on this subject and created keen interests for future projects to be created based on this field.

7. FUTURE GOALS

7.1 Improving on this project to create user friendly spam filter application feed it to a real email database and successfully classify it and also determine where the spam came from.

7.2 Use Machine Learning concepts to create a smart fashion application that helps people with no fashion sense style better.

7.3 Use Machine Learning concepts to predict teer outcomes if possible.

7.4 Use Machine Learning concepts to create a more accurate partner selecting software which will help people have better compatible marriages.

8. CONCLUSION

On doing this project, it has brought about more understanding on Machine Learning. It is a step towards taking this field as a major tool for creating fun softwares and application for various platform. The project has been able to classify emails as ham or spam but it is not yet that efficient to be used professionally as a spam classification software. If this project finds fruitful development. It can be used in major . We will move forward and make the above future goals a reality. As we can see, in today's modern world, the rate of technological innovation is progressing fast and within a few more years with the development of newer computer languages, softwares and sophisticated computer hardware this “SPAM FILTER” will be more efficient and have less error rate and maybe find flawless application in networking and communication sectors.

9. Bibliography

1.www.wikipedia.com
2.www.tutorialspoint.com
3.www.continuum.io/Downloads
4.www.pypi.python.org/pypi/spyder
5.Machine Learning in Action by Peter Harrington
6.Machine Learning for Dummies by John Paul Mueller
7.Machine Learning in Python by Michael Bowles
8. www.stackoverflow.com
9.www.youtube.com
10.www.yahoo.co.in
11.www.quora.com
12.www.google.co.in
13.www.edX.com
14.www.udacity.com
15. Siraj Raval tutorials