

**NATIONAL INSTITUTE OF TECHNOLOGY  
SILCHAR**



2020-2022

MINI PROJECT REPORT

ON

**ANALYSIS OF NAIVE BAYES ALGORITHM  
FOR SPAM FILTERING**

MASTER OF TECHNOLOGY

FIRST SEMESTER

SUBMITTED TO

**DR. BISWAJIT PURKAYASTHA**

SUBMITTED BY

**ROHIT THINGBAIJAM**

**2022120**

CONTACT NO.

**8794710048**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# ABSTRACT

## Analysis of Naive Bayes Algorithm for Spam Filtering

Spam continues to become a problem on the Internet. Spammed may contain many copies of the same message. In this research, we test Naive Bayes algorithm for spam filtering on 2 datasets and test its performance i.e. Spam and NotSpam. The performance of the datasets is evaluated based on their Bayes Estimate values. Our research will use C++ program for the evaluation of Naive Bayes algorithm for spam filtering on both datasets. The result will show us that the testing data set is classified to Spam or NotSpam by the implementation of Naive Bayes Estimation.

# CONTENT

1.INTRODUCTION

2.METHODOLOGY

3.EXPERIMENTAL SETUP

4.EXPERIMENTAL RESULT

5.PROJECT CODE

6.FUTURE WORK

7.DISCUSSION

8.REFERENCES

# 1.INTRODUCTION

We received mail from yahoo, gmail, outlook etc. and SMS in our phone which can be sent by anyone. But when a person wanted to receive particular mail or SMS only from the Institution that we studied we need to classified as two categories one is important and not important.

The not important part can be name as Spam and the important one as NotSpam. So the mail or SMS that we don't want it received can go to the spam categories and the important one can be go to NotSpam i.e. Inbox.

Several machine learning algorithms have been used in spam filtering, but Naive Bayes algorithm is particularly popular in commercial and open-source spam filters. This is because of its simplicity, which make them easy to implement and just need short training time or fast evaluation to filter spam. The filter requires training that can be provided by a previous set of spam and non-spam messages. It keeps track of each word that occurs only in spam, in non-spam messages, and in both. Naive Bayes can be used in different datasets where each of them has different features and attribute.

The research objectives are to implement the Naive Bayes algorithm for spam filtering on two datasets.

## 2.METHODOLOGY

The methodology used for the process of e-mail spam filtering is based on Naive Bayes algorithm.

### 2.1 Naive Bayes classifier

The Naive Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combination of values in a given dataset. In this research, Naive Bayes classifier use bag of words features to identify spam and a text is representing as the bag of its word. The bag of words is always used in methods of document classification, where the frequency of occurrence of each word is used as a feature for training classifier. This bag of words features are included in the chosen datasets. Naive Bayes technique used Bayesian Estimate theorem to determine that probabilities spam. Some words have particular probabilities of occurring in spam or not-spam.

Example, suppose that we know exactly, that the word “OFFER” could never occur in a non-spam. Then, when we saw a subject containing this word, we could tell for sure that were spam. Bayesian spam filters have learned a very high spam probability for the words such as “OFFER” and “PARTY”, but a very low spam probability for words seen in not-spam. So, to calculate the probability that spam or not-spam Naive Bayes technique used Bayesian Estimate theorem as shown in formula below.

$$P(C_x | \bar{A}) = P(\bar{A} | C_x) P(C_x)$$

A =Set of attribute

$C_x$  =Set of class

$$P(C_x) = \text{Priori probability} = \frac{|C_k| + 1}{m + \sum |C_i|}$$

$C_k$  =no. of patterns in  $C_k$

$C_i$  =total no. of patterns of all class

$$P(\bar{A} | C_x) = \text{Conditional probability} = \frac{|C_{ij}| + 1}{i_n + |C_k|}$$

$C_k$  =no. of patterns in  $C_k$

$i_n$  =no. of distinct value in the attribute

$C_{ij}$  =no. of patterns for class  $c_{ij}$  which satisfy the attribute value

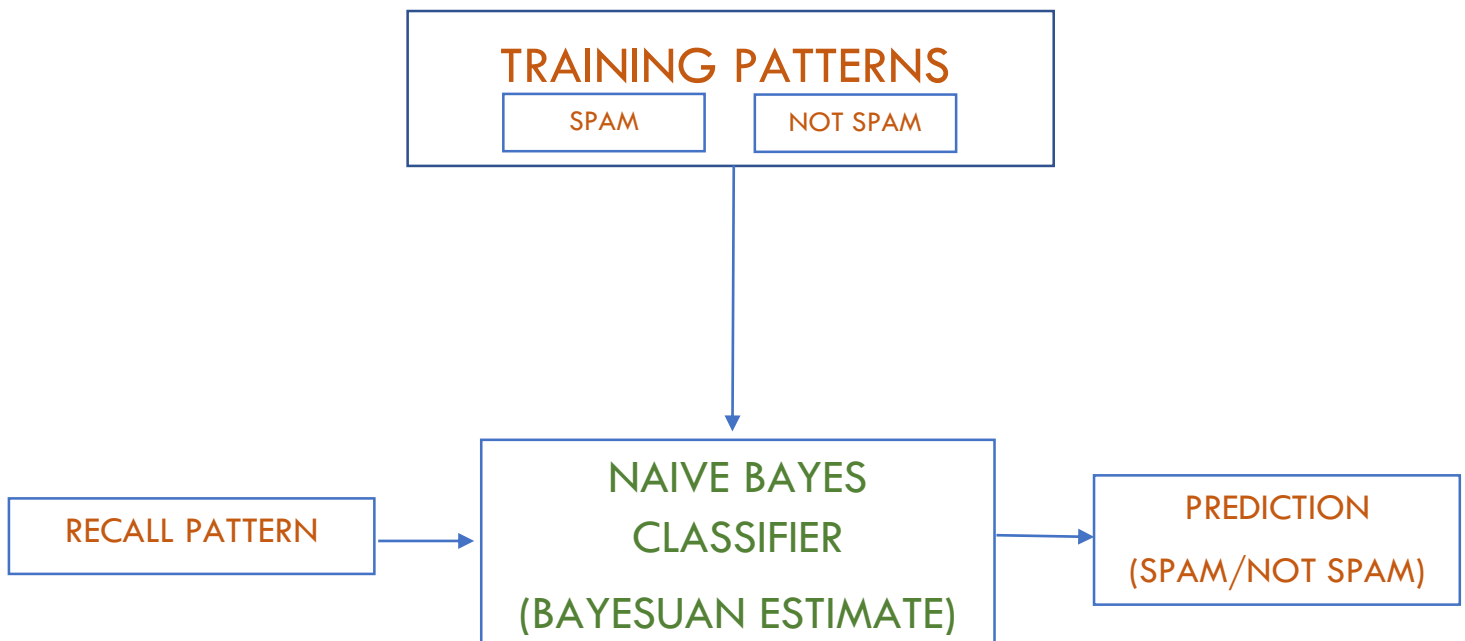


FIG. NO. 1

## 2.2 PROCEDURE OF MY PROJECT:

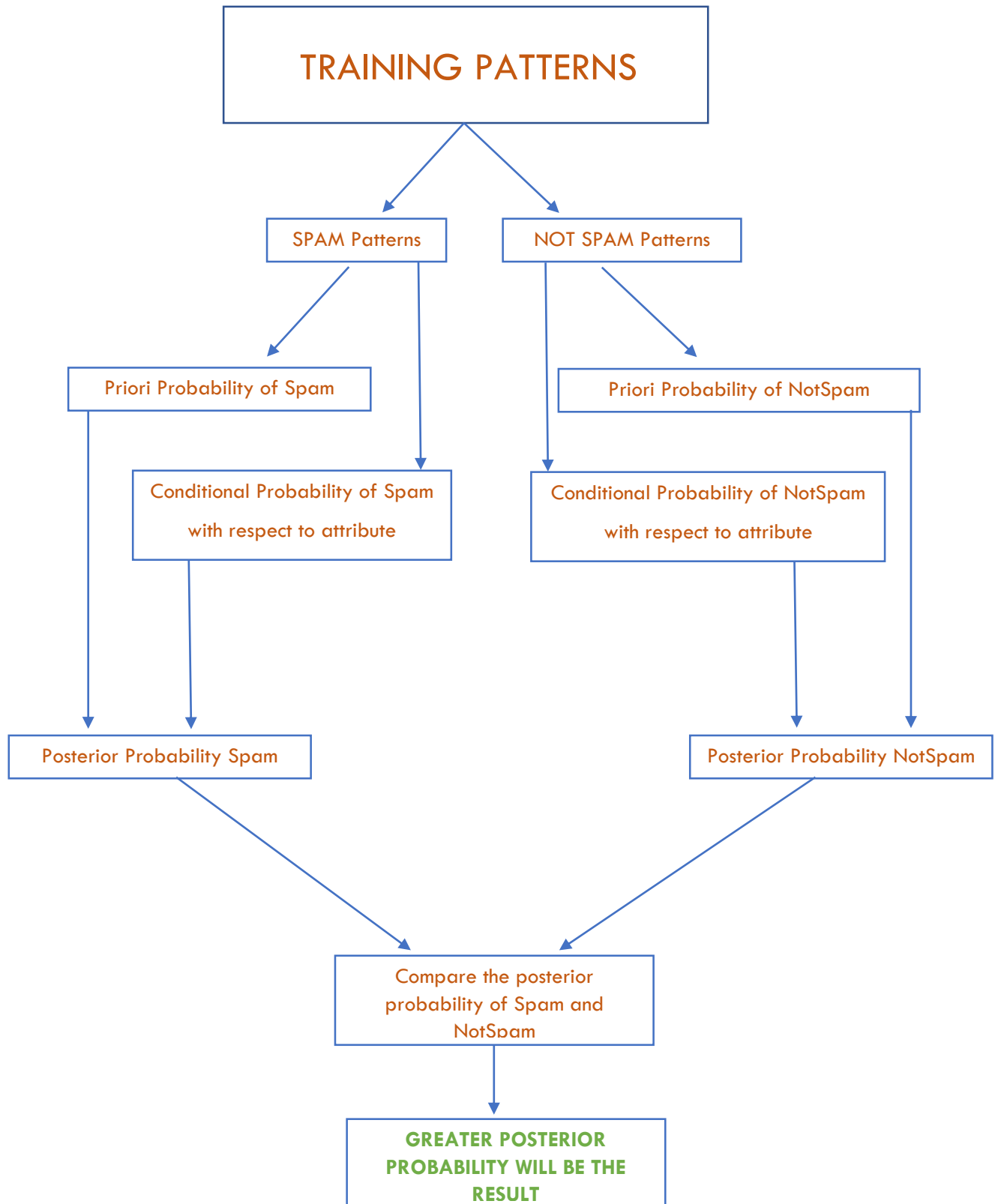


FIG. NO. 2

# 3.EXPERIMENTAL SETUP

## 3.1 The experiment setup are as follow:

### 3.1.1 Setup IDE (Dev C++)

Dev-C++, developed by Bloodshed\_Software, is a fully featured graphical IDE, which is able to create Windows or console-based C/C++ programs using the MinGW compiler system. MinGW uses GCC, which is essentially the same compiler system that is in Cygwin (the Unix environment program for Windows) and most versions of Linux.

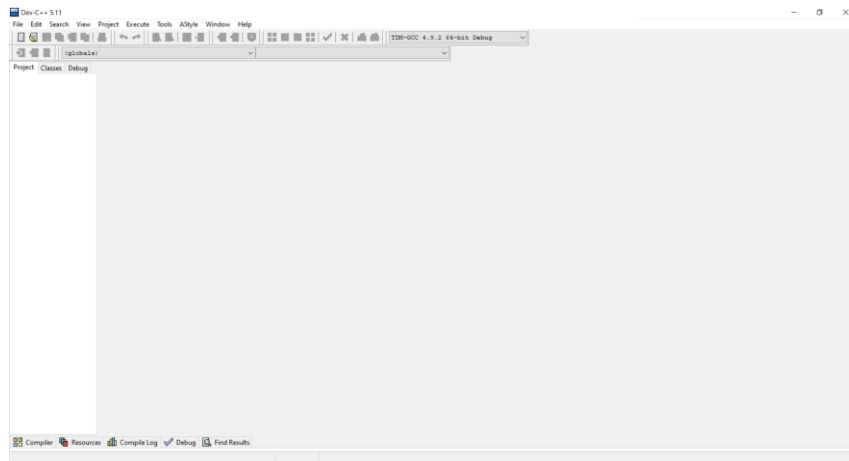


FIG. NO. 3

### 3.1.2 Setup Notepad++

Notepad++ is a text and source code editor for use with Microsoft Windows. It supports tabbed editing, which allows working with multiple open files in a single window. The project's name comes from the C increment operator.



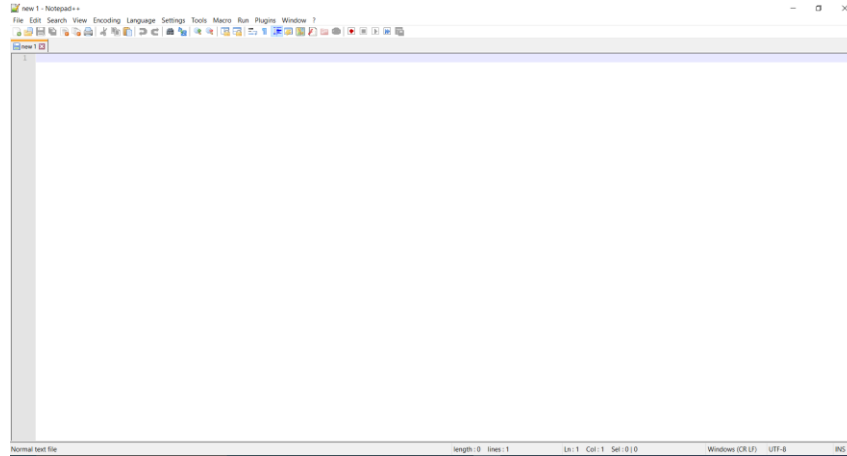


FIG. NO. 4

### 3.1.3 Setup training pattern

For this experiment we use two text file containing the training pattern i.e. Spam.text and NotSpam.text.

In this file we give 50 patterns each making it together 100 patterns for the experiment. For each patterns we provide 3 attributes.



FIG. NO. 5

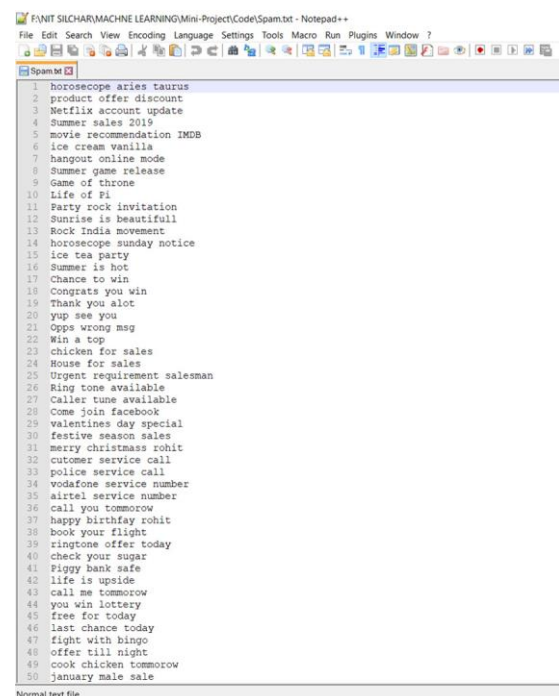


FIG. NO. 6

For calculating  $i_n$  = no. of distinct value in the attribute. We need to go through each column for both Spam and NotSpam. We need 3 text files for each column i.e.

column1 =Containing 1<sup>st</sup> column for both Spam and NotSpam  
i.e.100 rows

column2 =Containing 2<sup>nd</sup> column for both Spam and NotSpam  
i.e.100 rows

column3 =Containing 3<sup>rd</sup> column for both Spam and NotSpam  
i.e.100 rows

for calculating  $i_n$ .

## 4.EXPERIMENTAL RESULT

The result of the experiment by using Bayesian Estimate theorem and calculate using C++ with 2 recall pattern, we get

### 4.1 Recall 1: NIT rock meet

No. of training patterns for Spam: 50

No. of training patterns for NotSpam: 50

Total no. of training patterns: 100

Prior probability of class spam: 0.5

Prior probability of class spam: 0.5

No Posterior Probability. of values in first attribute :90

No. of values in second attribute :73

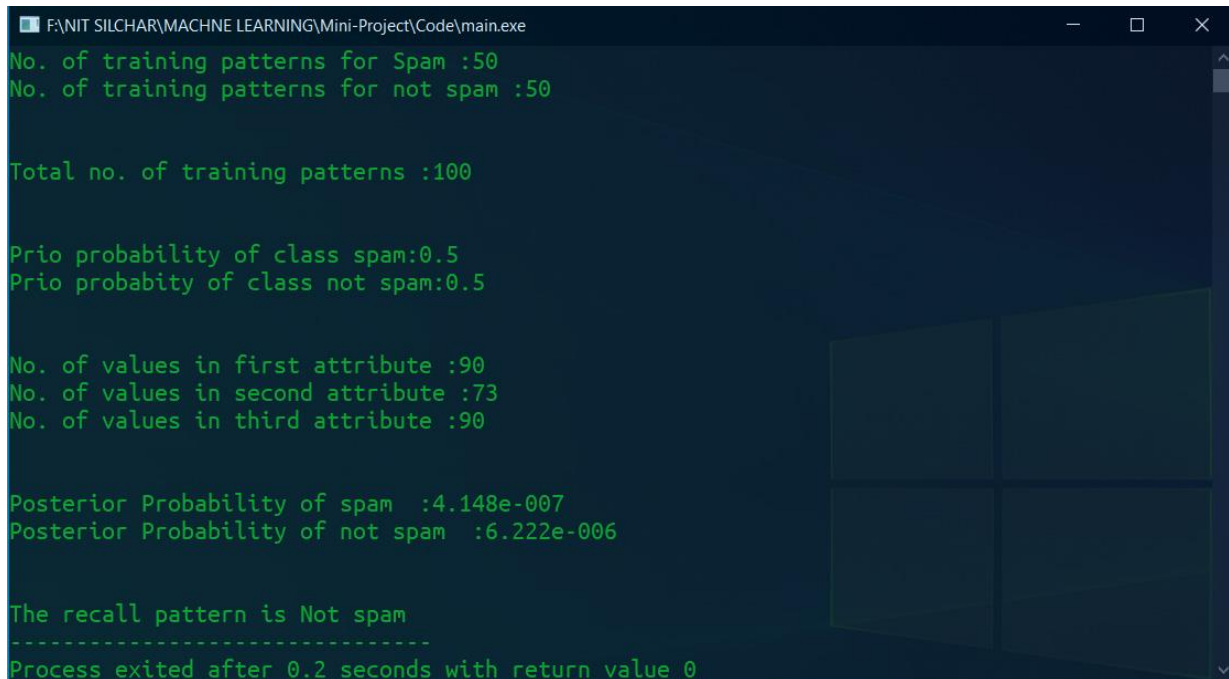
No. of values in third attribute :90

of spam:  $4.148e-007$

Posterior Probability of not spam:  $6.222e-006$

Thus, the recall pattern is Not spam.

The output is shown below after running in Dev C++



```
F:\NIT SILCHAR\MACHINE LEARNING\Mini-Project\Code\main.exe
No. of training patterns for Spam :50
No. of training patterns for not spam :50

Total no. of training patterns :100

Prio probability of class spam:0.5
Prio probability of class not spam:0.5

No. of values in first attribute :90
No. of values in second attribute :73
No. of values in third attribute :90

Posterior Probability of spam :4.148e-007
Posterior Probability of not spam :6.222e-006

The recall pattern is Not spam
-----
Process exited after 0.2 seconds with return value 0
```

FIG. NO. 7

## 4.2 Recall 2: piggy day party

No. of training patterns for Spam: 50

No. of training patterns for NotSpam: 50

Total no. of training patterns: 100

Prio probability of class spam: 0.5

Prio probability of class not spam: 0.5

No. of values in first attribute :90

No. of values in second attribute :73

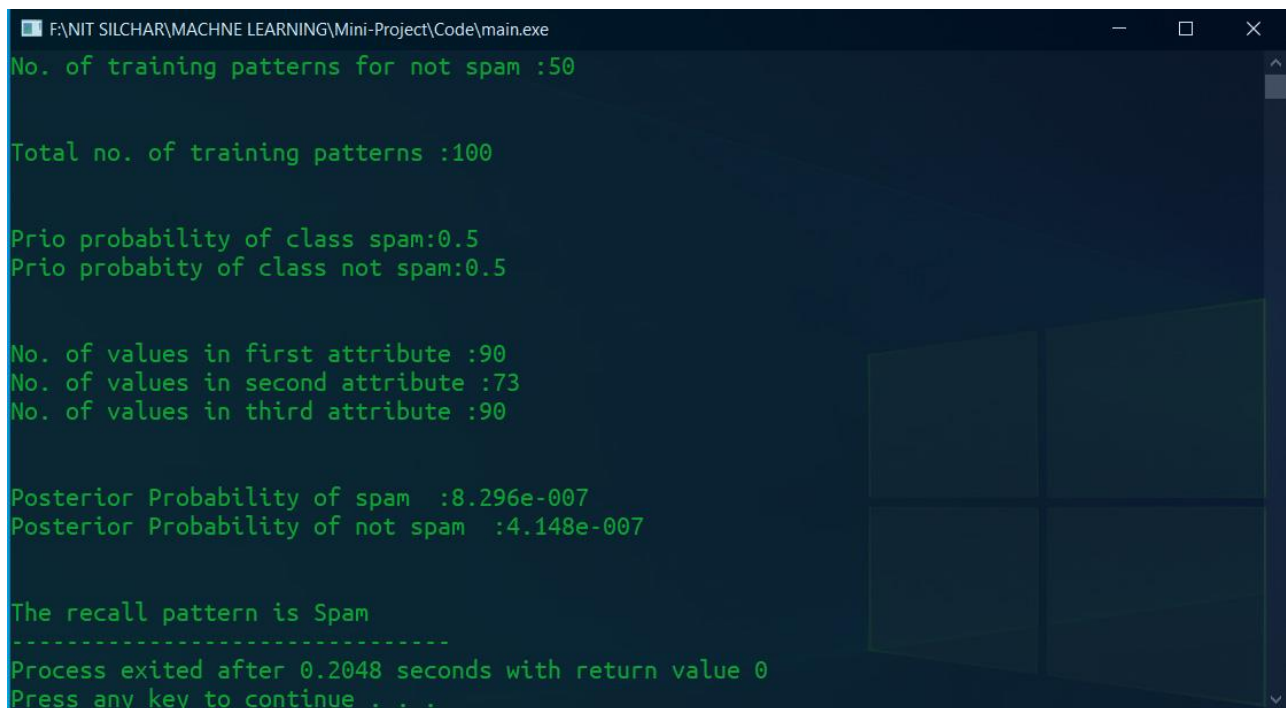
No. of values in third attribute :90

Posterior Probability of spam: 8.296e-007

Posterior Probability of not spam: 4.148e-007

Thus, the recall pattern is Not spam

The output is shown below after running in Dev C++



```
F:\NIT SILCHAR\MACHINE LEARNING\Mini-Project\Code\main.exe
No. of training patterns for not spam :50

Total no. of training patterns :100

Prio probability of class spam:0.5
Prio probability of class not spam:0.5

No. of values in first attribute :90
No. of values in second attribute :73
No. of values in third attribute :90

Posterior Probability of spam :8.296e-007
Posterior Probability of not spam :4.148e-007

The recall pattern is Spam
-----
Process exited after 0.2048 seconds with return value 0
Press any key to continue . . .
```

FIG. NO. 8

# 5.PROJECT CODE

## 5.1 PUSHING STRING FROM TEXT FILE TO STRING VARIABLE:

```
file.open("Spam.txt");
while(!file.eof())
{
    file>>str;
    Spam1.push_back(str);
}
file.close();
```

## 5.2 PRIORI PROBABILITY:

```
float probSpam =(float) ((NoOfSpam+1)/(2+(NoOfSpam+NoOfNotSpam)));
float probNotSpam = (float)((NoOfNotSpam+1)/(2+(NoOfSpam+NoOfNotSpam)));
```

## 5.3 CONDITIONAL PROBABILITY

```
(float)(noOfWords + 1) / (float)(frequency[count1-1] + NoOfSpam);
(float)(noOfWords + 1)/(float)(frequency[count2-1] + NoOfNotSpam);
```

## 5.4 POSTERIOR PROBABILITY:

```
probSpam *=(float)(noOfWords + 1) / (float)(frequency[count1-1] + NoOfSpam);
probNotSpam *=(float)(noOfWords + 1)/(float)(frequency[count2-1] + NoOfNotSpam);
```

## 5.5 FREQUENCY OF CLASS IN ALL PATTERNS:

### i.NotSpam

```
file.open("NotSpam.txt");
file.get(ch);
while(file)
{
    if( ch == '\n')
        NoOfNotSpam++;
    file.get(ch);
}
file.close()
```

### ii.Spam

```
file.open("spam.txt");
file.get(ch);
while(file)
{
    if( ch == '\n')
        NoOfSpam++;
    file.get(ch);
}
file.close();
```

## 5.6 $i_n$ = NO. OF DISTINCT VALUE IN THE ATTRIBUTE:

```
bool search(string str)
{
    for(string i : temp){
        if(i == str)
            return true;
    }
    return false;
}

file.open("column1.txt");
while(!file.eof()){
    file>>str;
    column1.push_back(str);
}
file.close();
for(string i : column1)
{
    if(!search(i))
        temp.push_back(i);
}
frequency[x]=(temp.size());
temp.clear();
cout<<"No. of values in first attribute : "<<frequency[x]
    <<endl;
break;
```



## 6.FUTURE WORK

In further, research we can increase the no. of attributes to more than 3 to increase the accuracy.

We can also make compare sentences having different no. of attributes.

Compare the accuracy level of Bayes Estimation with Bayes Maximum likelihood.

## 7.DISCUSSION

In our experiment, first we calculate the priori probability of spam and not-spam using Bayesian estimate formula by using our C++ program to calculate the no. of pattern for spam class and not-spam class by opening the text files for each and count the no. of new line to get the no. of training patterns for each class.

Again we need to find the Conditional probability. For this we in to calculate no. of distinct value that exist in the 3 attributes. For this we classified 3 text file i.e. column1, column2 and column 3 containing all values of each attributes to the text files and with the help of our program we are able to generate the no. of distinct for each attribute.

Lastly using Naive Bayes algorithm, we calculate the conditional probability with the recall pattern values for both the class and the recall patterns belongs to the probability which is high.

Thus the output of the experiment is classified by using Naive Bayes algorithm.

## 8.REFERENCES

[1] Rushdi, S. and Robet, M, “Classification spam emails using text and readability features”, IEEE 13th International Conference on Data Mining, 2013.

[2] Androutsopoulos, I., Paliouras, G., and Michelakis, “E. Learning to filter unsolicited commercial e- mail”, Technical report NCSR Demokritos, 2011.