**ABSTRACT**

Spam emails have been a chronic issue in computer security. They are very costly economically and extremely dangerous for computers and networks. Despite of the emergence of social networks and other Internet based information exchange venues, dependence on email communication has increased over the years and this dependence has resulted in an urgent need to improve spam ﬁlters. Although many spam ﬁlters have been created to help prevent these spam emails from entering a user’s inbox, there is a lack or research focusing on text modiﬁcations. Currently, Naive Bayes is one of the most popular methods of spam classiﬁcation because of its simplicity and efﬁciency. Naive Bayes is also very accurate; however, it is unable to correctly classify emails when they contain leetspeak or diacritics. Thus, in this proposes, we implemented a novel algorithm for enhancing the accuracy of the Naive Bayes Spam Filter so that it can detect text modiﬁcations and correctly classify the email as spam or ham. Our Python algorithm combines semantic based, keyword based, and machine learning algorithms to increase the accuracy of Naive Bayes compared to Spam assassin by over two hundred percent.

## TABLE OF CONTENTS

#### Page No

[Table of contents vii](#_TOC_250019)

[List of Figures viii](#_TOC_250018)

1. INTRODUCTION 1
   1. Relevance of Project 1
   2. Scope of Project 2
   3. [Problem Statement 2](#_TOC_250017)
2. LITERATURE SURVEY 3
   1. [Paper 1 3](#_TOC_250016)
   2. [Paper 2 3](#_TOC_250015)
   3. [Paper 3 4](#_TOC_250014)
   4. Comparison 5
3. SYSTEM REQUIRMENTS SPECEFICATION 6
   1. [Hardware Requirements 6](#_TOC_250013)
   2. [Software Specification 6](#_TOC_250012)
   3. Software Requirement 6
      1. [Anaconda Distribution 6](#_TOC_250011)
      2. [Python Libraries 7](#_TOC_250010)
4. SYSTEM ANALYSIS AND DESIGN 8
   1. [System Architecture 8](#_TOC_250009)
   2. [Dataset 9](#_TOC_250008)
   3. [Design 11](#_TOC_250007)
5. IMPLEMENTATION 12
   1. [Algorithm 12](#_TOC_250006)
   2. [Pre-Processing The Data 13](#_TOC_250005)
   3. Creating A Model And Training 14
6. RESULTS AND DISCUSSION 16
   1. [Screenshots 16](#_TOC_250004)
7. TESTING 18
8. CONCLUSION AND FUTURE SCOPE 20
   1. [Conclusion 20](#_TOC_250003)
   2. [Contribution 20](#_TOC_250002)
   3. [Future Scope 20](#_TOC_250001)

[REFERENCES 21](#_TOC_250000)

vii

## LIST OF FIGURES

#### Page NO

#### Fig 4.1 Proposed System 8

#### Fig 4.2 Dataset With Leets peak 9

#### Fig 4.3 Dataset Without Leets peak 10

#### Fig 4.4 Flow Chart 11

#### Fig 5.3.1 Naive Bayes Model 14

#### Fig 5.3.2 Logistic Regression Model 15

#### Fig 6.1 80-20 Split Ratio Comparison 16

#### Fig 7.1 60-40 Split Ratio Comparison 18

viii

**CHAPTER 1**

# INTRODUCTION

Spam refers to an email aimed manipulating an individual to whom it is aimed at or just randomly flooding the inbox. It is also called as junk mail and it floods Internet clients Inboxes. Today spam emails are of a variety of types ranging from ads to business promoting to doubtful products to some objectionable services. Therefore it is difficult to identify and classify an email as spam or non-spam.

Usenet also called as User Network is an email service that distributes group talks or emails aimed at a particular group of people associated with a certain service or product and are mostly informative but do crowd up the inbox of the user. The data that goes over the Internet is called Netnews” an accumulation of these data that is aimed at providing message about a specific topic is called a ”newsgroup”. People that read such news from these newsgroups are the prime target of Spammers. Spammers use these news groups for the promotion of certain unrelated ads or unrelated posts. Usenet spam robs clients of the utility of the newsgroups by promoting other unrelated posts.

### Relevance of the Project

As the digitization of communication grows, electronic mail, or emails, has become increasingly popular; in 2016, an estimated 2.3 million people used email. In 2015, 205 billion emails were sent and received daily, which is expected to grow at an annual rate of 3% and reach over 246 billion by 2019. However, the growth in emails has also led to an unprecedented increase in the number of illegitimate mail, or spam - 49.7% of emails sent is spam - because current spam detection methods lack an accurate spam classifier. Spam is problematic not only because it often is the carrier of malware, but also because spam emails hoard network bandwidth, storage space, and computational power. Additionally, the commercial world has significant

interests in spam detection because spam causes loss of work productivity and financial loss. It is estimated that American firms and consumers lose 20 billion annually, even while sustained by the private firms’ investment in anti-spam software. On the other hand, spam advertising earns 200 million per year. Although extensive work has been done on spam filter improvement over the years, many of the spam filters today have limited success because of the dynamic nature of spam. Spammers are constantly developing new techniques to bypass filters, some of which include word obfuscation and statistical poisoning. Although these two text classification issues are recognized, research today has largely neglected to provide a successful method to improve spam detection by counteracting word obstruction and Bayesian poisoning, and many common spam filters are unable to detect them.

### Scope of the project

Naïve Bayes is very accurate; however, it is unable to correctly classify emails when they contain leetspeak or diacritics. Thus in this proposes, we implemented a novel algorithm for enhancing the accuracy of the Naive Bayes Spam Filter so that it can detect text modifications and correctly classify the email as spam or ham.

### Problem statement

This project proposes “Spam Message Filtering using Intelligent Text Modification method”. This project predicts spam messages more accurately than the existing projects. This project uses Naïve Bayes algorithm to modify the text.

**CHAPTER 2**

# LITERATURE SURVEY

### Paper 1

**Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization[1]**

* + - Naïve Bayes algorithm is a Bayes theorem based statistical machine learning based approach having properties of strong independence, probability distribution and ability to handle large datasets.
    - In NB, probability distribution is evaluated from the frequency distribution of dataset.
    - Particle Swarm Optimization (PSO) is swarm intelligence based concept derived in 1995 by Eberhart and Kennedy
    - PSO work on the property of stochastic distribution and initially find the local search solution, then individual particle share their solution and global solution is obtained.
    - NB having probability distribution property determines the possible class for the email content from the spam class or non-spam class on the basis of keywords present in the email textual data.
    - PSO is used to further optimize the parameters of NB approach to improve the accuracy, search space and classification process.

### Paper 2

**Email Spam Classification by Support Vector Machine[2]**

* + - This paper uses Support Vector Mechanism algorithm to identify spam emails.
    - Descriptions as provided on Spam Assassin website for the dataset used in this paper.
    - SVM is also considered as an important kernel methods, which is one of the most important areas in machine learning concepts.
    - Smart Trafﬁc Control System with Application of Image Processing Techniques
    - In this work they have also compared Linear and Gaussian as two of the very popular kernel and employed them for the problem of email spam detection
    - The two models have been proposed, trained and tested using popular and often used standard database.

### Paper 3

**Intelligent Model for Classification of SPAM and HAM[3]**

* + - In this paper they have used machine learning and non machine learning approaches.
    - Machine learning approaches like support vector mechanism, neural network etc. Non machine learning approaches like strong key word searching and whitelisting and blacklisting of words**.**
    - The sets so formed are further used as training set and the classification set.
    - The process is to use the first set as training set and the remaining N-1 sets as the sets to be classified.
    - In the next iteration the second set is used as the training set and the remaining sets are sets to be classified. The process is repeated until all the sets are used as training sets.
    - The emails are classified based on the spam percentage each mail gains.

**CHAPTER 3**

# REQUIREMENTS SPECIFICATION

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

### Hardware requirements

* + - A PC with Windows/Linux OS
    - Processor with 1.7-2.4gHz speed
    - Minimum of 8gb RAM

### Software specification

* + - Anaconda distribution package
    - Python libraries

### Software requirements

#### Anaconda distribution:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data- science packages suitable for Windows, Linux and MacOS.

### Python libraries:

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Matplotlib, Flask framework, etc are needed.

**SKlearn:** It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**NumPy:** NumPy is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays. It **is** the fundamental package for scientific computing with Python.

**Pandas:** Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

**Matplotlib:** matplotlib pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pilot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac :**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

## How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

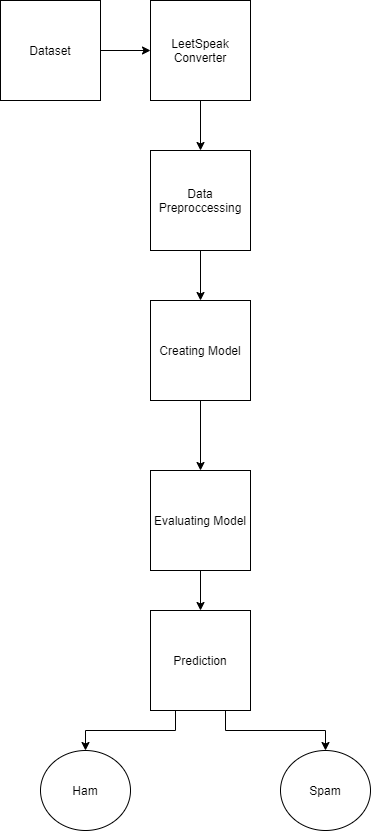
**Note:** The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. [Download the Python Cheatsheet here.](https://myelearninghub.com/python-cheat-sheet/)The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

**CHAPTER 4**

# SYSTEM ANALYSIS AND DESIGN

### System Architecture



#### Fig 4.1 Proposed System

The above figure represents the Proposed System of the project. We first take the dataset then it is passed through the leetspeak converter which converts all the leetspeak characters into normal text. Then that data set is set for data pre processing where in which all the unwanted and stop words are removed in order to minimize the dataset as much as possible. Then a model is created to test the data which is evaluated. Finally the dataset is fed to this model which predicts the output whether the given text is spam or ham.

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

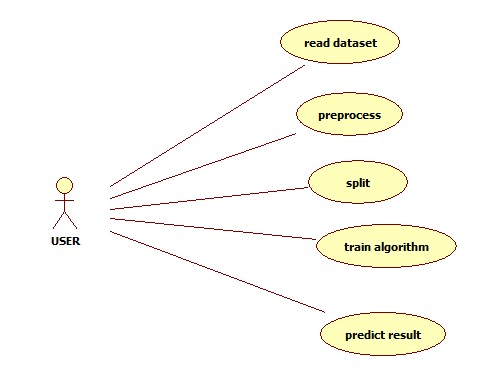
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

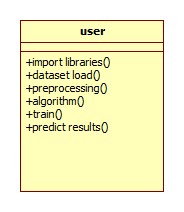
**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



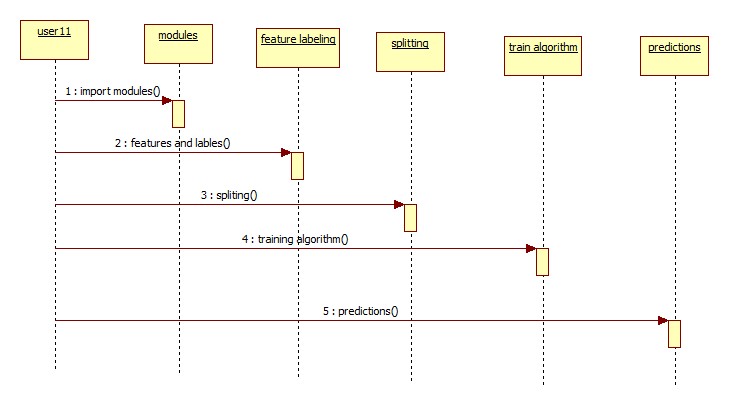
**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

****

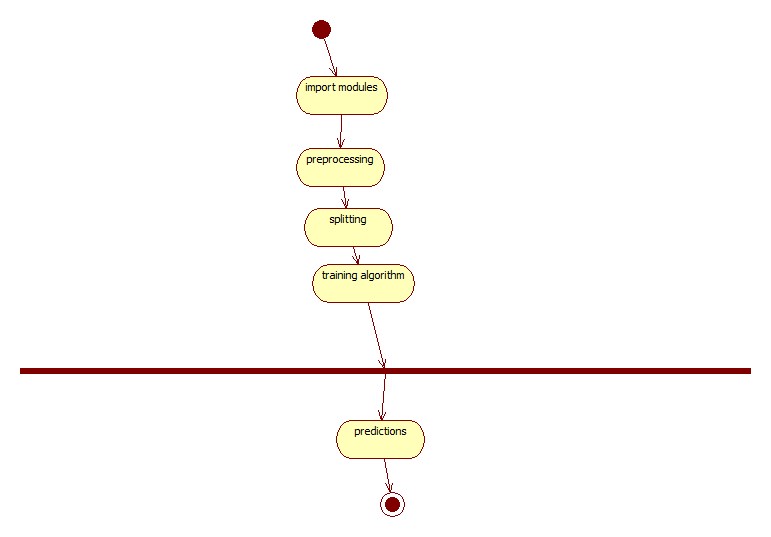
**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Modules**

**1.Load Dataset:**

Load data set using pandas read\_csv() method. Here we will read the excel sheet data and store into a variable.

**2.Split Data Set:**

Split the data set to two types. One is train data test and another one is test data set.here we will remove missing values from the dataset.

**3.Train data set:**

Train data set will train our data set using fit method. 80% of data from dataset we use for training the algorithm.

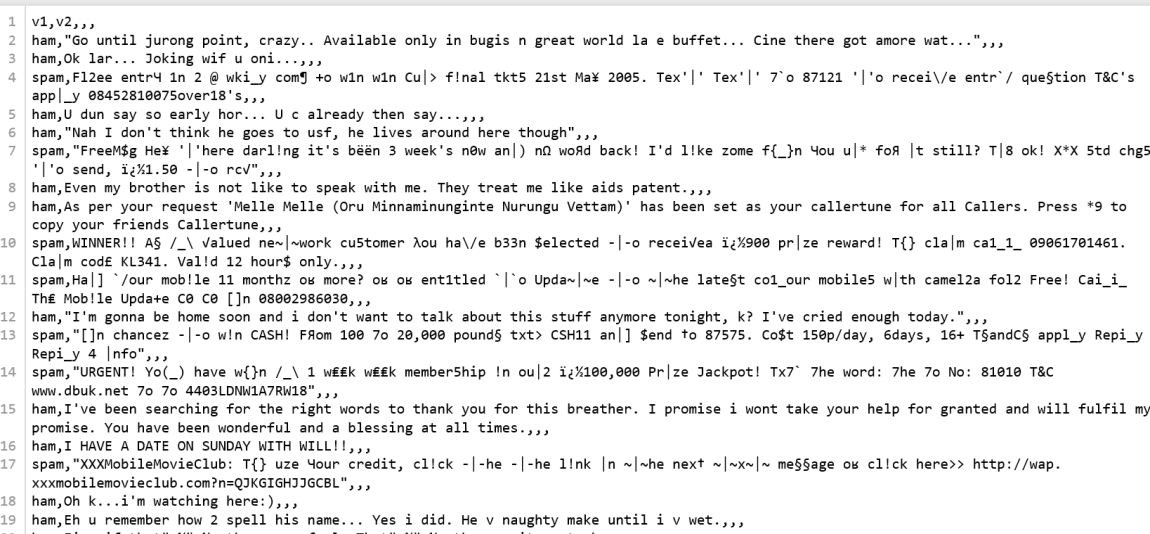
**4.Test data set:**

Test data set will test the data set using algorithm. 20% of data from dataset we use for testing the algorithm.

**5.Predict data set:**

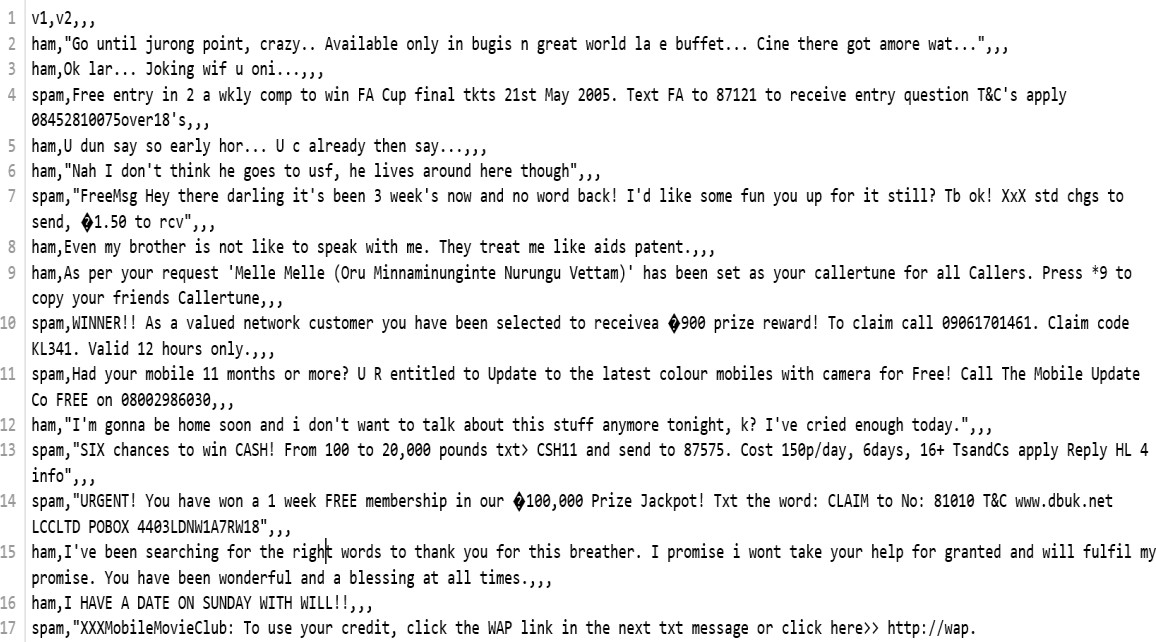
Predict() method will predict the results. In this step we will predict the ranking of the google play store app.

### Dataset



#### Fig 4.2 Dataset with leetspeak

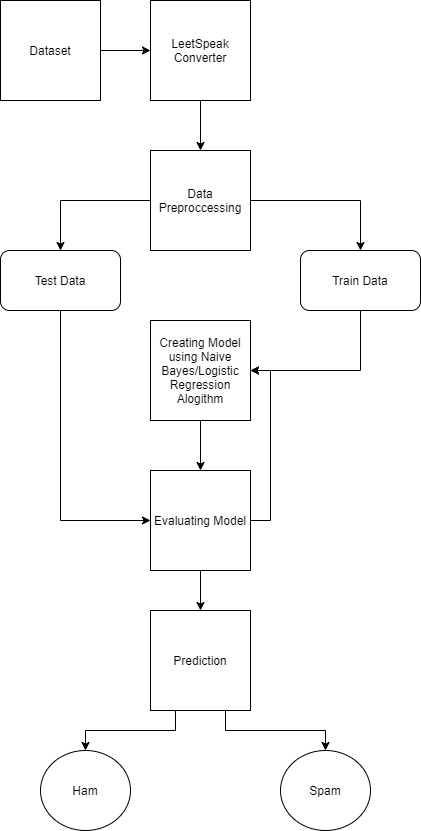
The above screenshot consists of sample dataset that contains leetspeak words which needs to be removed.



#### Fig 4.3 Dataset without leetspeak

The above screenshot consists of sample dataset where all the leetspeak words are removed and converted to the corresponding/similar words.

### Design



#### Fig 4.4 Flow chart

The dataset is sent to the leetspeak converter where all the leetspeak words are converted to corresponding words. The it is sent to data pre processor where all the stop and un wanted words are removed. The data is split into test data and train data. Each of these split data are sent to the model to evaluate their efficiency for the test and train data. Finally the out put is predicted for the test data.

**CHAPTER 5**

# IMPLEMENTATION

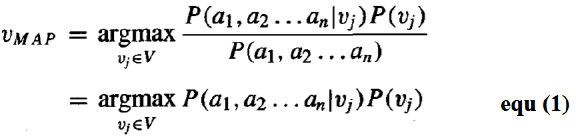
### Algorithm

Naïve Bayes Classifier applies to learning tasks where each instance x is described by conjunction of attribute values and where the target function f(x) can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by tuple of attribute values (a1,a2,…) . The learner is asked to predict the target value, or classification, for this new instance.

The Bayesian approach to classifying the new instance is to design the most portable target value, Vmap, given the attribute values(a1,a2,..) that describe the instance.



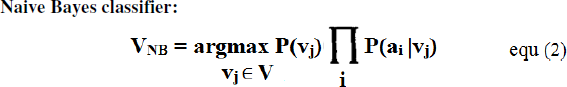
Use Bayesian theorem to rewrite this expression as



The Naïve Bayes classifier is based on the assumption that the attribute values are conditionally independent given the target value. Mean, the assumption is that given the target value of the instance, the probability of observing the conjunction is just product of the probabilities for the individual attributes.



Substituting this into equation 1

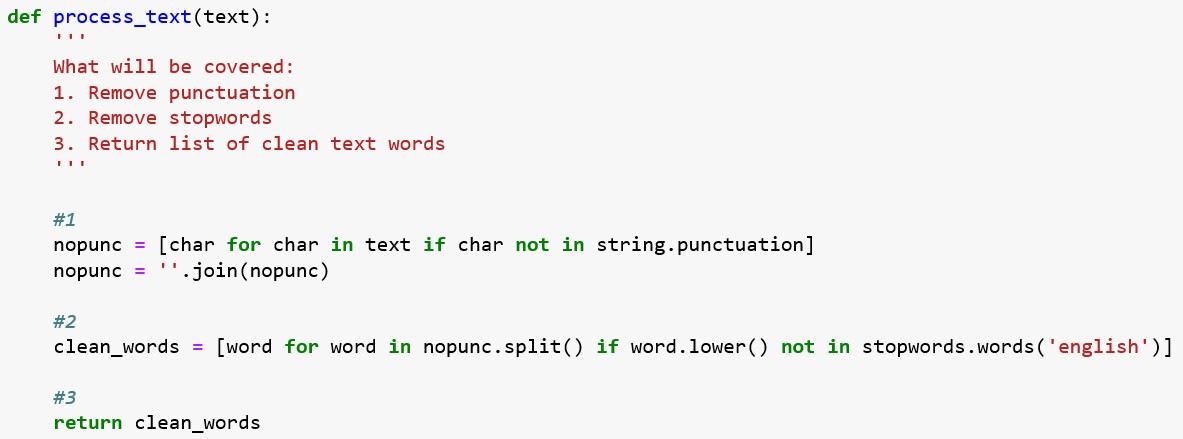


Where, VNB denotes the target value output by the naïve Bayes Classifier.

### Pre processing the data

Remove punctuations. Remove

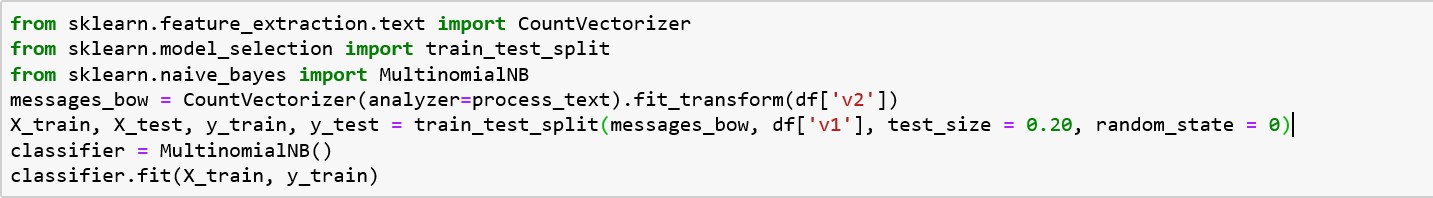
stop words :- Stop words like “and”, “the”, “of”, etc are very common in all English sentences and are not very meaningful in deciding spam or legitimate status, so these words have been removed from the emails.



In pre processing we remove all the unwanted words and punctuations such as comma, full stops, extra spaces and other repeated words. This will help to reduce the data to be processed. This helps to process the data much faster than before due to small amount of data.

### Creating Model and Training

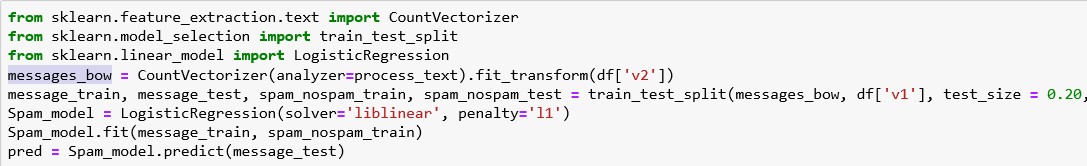
* Naïve Bayes Classifier



#### Fig 5.3.1 Naive Bayes Model

Here we split the data into train and test data and then we convert our data into the desired matrix format. To do this we will be using Count Vectorizer(). There are two steps to consider here: Firstly, we have to fit our training data (X\_train) into Count Vectorizer() and return the matrix. Secondly, we have to transform our testing data (X\_test) to return the matrix. Note that X\_train is our training data for the 'v2' column in our dataset and we will be using this to train our model. X\_test is our testing data for the 'v2' column and this is the data we will be using(after transformation to a matrix) to make predictions. Import the Multinomial classifier and fit the training data into the classifier using fit(). Name your classifier 'classifier'. Now that our algorithm has been trained using the training data set we can now make some predictions on the test data stored in 'X\_test' using predict().

### Logistic Regression model



#### Fig 5.3.2 Logistic regression Model

Here we split the data into train and test data and then we convert our data into the desired matrix format. To do this we will be using Count Vectorizer(). There are two steps to consider here: Firstly, we have to fit our training data (message train) into Count Vectorizer() and return the matrix. Secondly, we have to transform our testing data (message test) to return the matrix. Note that message train is our training data for the 'v2' column in our dataset and we will be using this to train our model. message test is our testing data for the 'v2' column and this is the data we will be using(after transformation to a matrix) to make predictions.

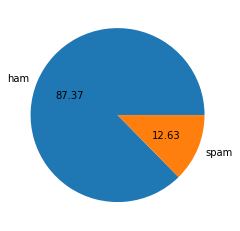
Import the Logistic Regression algorithm and fit the training data into the model using fit(). Name your model 'spam model'. Now that our algorithm has been trained using the training data set we can now make some predictions on the test data stored in 'message test' using predict().

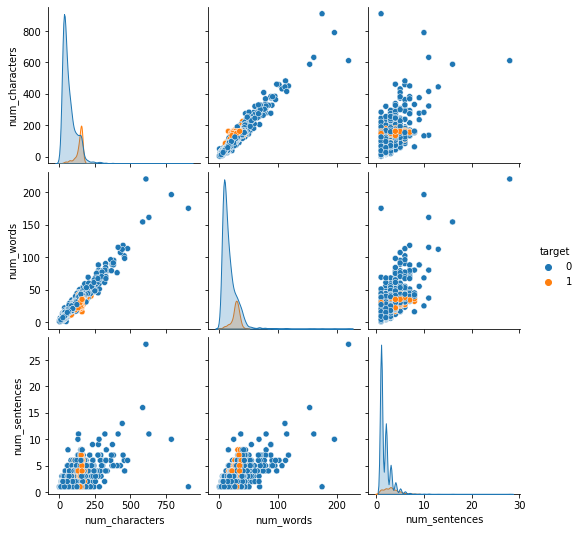
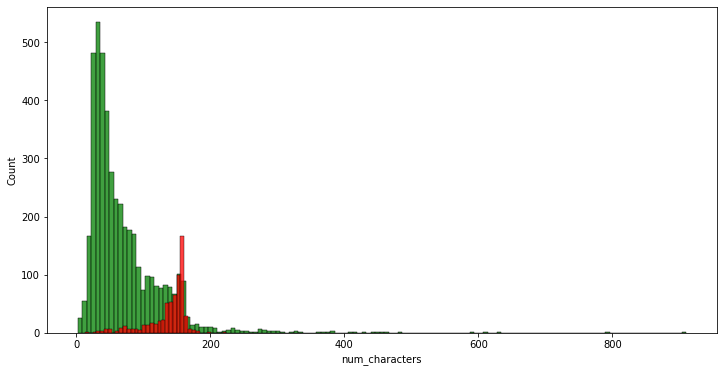
**CHAPTER 6**

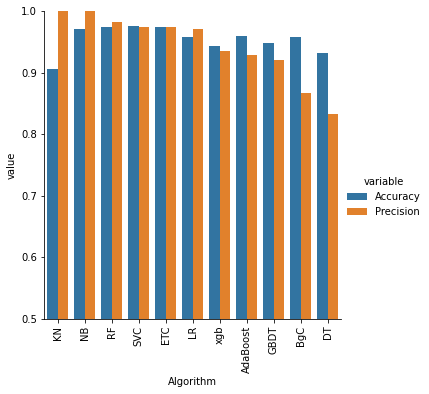
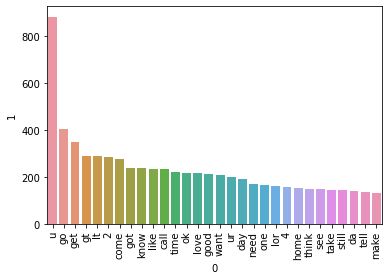
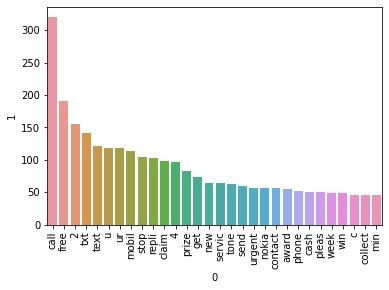
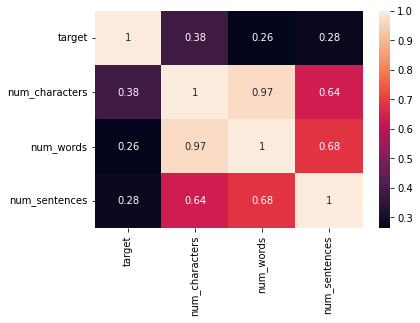
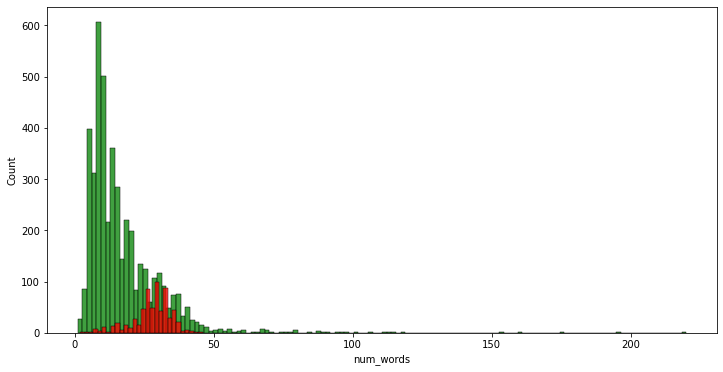
# RESULTS AND DISCUSSION

Comparing Results for Naïve Bayes and logistic Regression model with 80-20 split ratio.

### 6.1 SCREENSHOTS

****

**Fig:- dataset with spam and ham emails **

****

#### Fig algorithm comparison

The above screenshots shows the results of the output when the naïve bayes classifier and logistic regression models are tested for 80-20 split ratio that is the percentage of data provided while training the model and testing the model respectively.

**CHAPTER 7**

# TESTING

**7.1 INTRODUCTION TO TESTING**

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

* 1. **TESTING STRATEGIES**

A technique for programming testing coordinates the outline of programming experiments into an all around arranged arrangement of steps that outcome in fruitful improvement of the product. The procedure gives a guide that portrays the means to be taken, when, and how much exertion, time, and assets will be required. The procedure joins test arranging, experiment configuration, test execution, and test outcome gathering and assessment. The procedure gives direction to the specialist and an arrangement of points of reference for the chief. Due to time weights, advance must be quantifiable and issues must surface as ahead of schedule as would be prudent

Keeping in mind the end goal to ensure that the framework does not have blunders, the distinctive levels of testing techniques that are connected at varying periods of programming improvement are:

**Unit Testing**

Unit Testing is done on singular modules as they are finished and turned out to be executable. It is restricted just to the planner's prerequisites. It centers testing around the capacity or programming module. It Concentrates on the interior preparing rationale and information structures. It is rearranged when a module is composed with high union

• Reduces the quantity of experiments

• Allows mistakes to be all the more effectively anticipated and revealed**Black Box Testing**

It is otherwise called Functional testing. A product testing strategy whereby the inward workings of the thing being tried are not known by the analyzer. For instance, in a discovery test on a product outline the analyzer just knows the information sources and what the normal results ought to be and not how the program touches base at those yields. The analyzer does not ever inspect the programming code and does not require any further learning of the program other than its determinations. In this system some experiments are produced as information conditions that completely execute every single practical prerequisite for the program. This testing has been utilizations to discover mistakes in the accompanying classifications:

• Incorrect or missing capacities

• Interface blunders

• Errors in information structure or outside database get to

• Performance blunders

• Initialization and end blunders.

In this testing just the yield is checked for rightness.

**White Box testing**

It is otherwise called Glass box, Structural, Clear box and Open box testing . A product testing procedure whereby express learning of the inner workings of the thing being tried are utilized to choose the test information. Not at all like discovery testing, white box testing utilizes particular learning of programming code to inspect yields. The test is precise just if the analyzer comprehends what the program should do. He or she would then be able to check whether the program veers from its expected objective. White box testing does not represent blunders caused by oversight, and all obvious code should likewise be discernable. For an entire programming examination, both white box and discovery tests are required.

In this the experiments are produced on the rationale of every module by drawing stream diagrams of that module and sensible choices are tried on every one of the cases. It has been utilizations to produce the experiments in the accompanying cases:

• Guarantee that every single free way have been Executed.

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

**Integration Testing**

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

Two Approaches of Integration Testing

• Non-incremental Integration Testing

• Incremental Integration Testing

**System Testing**

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

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

**Acceptance Testing**

Acknowledgment testing, a testing method performed to decide if the product framework has met the prerequisite particulars. The principle motivation behind this test is to assess the framework's consistence with the business necessities and check in the event that it is has met the required criteria for conveyance to end clients. It is a pre-conveyance testing in which whole framework is tried at customer's site on genuine information to discover blunders. The acknowledgment test bodies of evidence are executed against the test information or utilizing an acknowledgment test content and afterward the outcomes are contrasted and the normal ones.

The acknowledgment test exercises are completed in stages. Right off the bat, the essential tests are executed, and if the test outcomes are palatable then the execution of more intricate situations are done.

**7.3 TEST APPROACH**

A Test approach is the test system usage of a venture, characterizes how testing would be done. The decision of test methodologies or test technique is a standout amongst the most intense factor in the achievement of the test exertion and the precision of the test designs and gauges.

Testing should be possible in two ways

• Bottom up approach

• Top down approach

**Bottom up Approach**

Testing can be performed beginning from littlest and most reduced level modules and continuing each one in turn. In this approach testing is directed from sub module to primary module, if the fundamental module is not built up a transitory program called DRIVERS is utilized to recreate the principle module. At the point when base level modules are tried consideration swings to those on the following level that utilization the lower level ones they are tried exclusively and afterward connected with the already inspected bring down level modules

**Top down Approach**

In this approach testing is directed from fundamental module to sub module. in the event that the sub module is not built up an impermanent program called STUB is utilized for mimic the sub module. This sort of testing begins from upper level modules. Since the nitty gritty exercises more often than not performed in the lower level schedules are not given stubs are composed. A stub is a module shell called by upper level module and that when achieved legitimately will restore a message to the calling module demonstrating that appropriate association happened.

**7.4 VALIDATION**

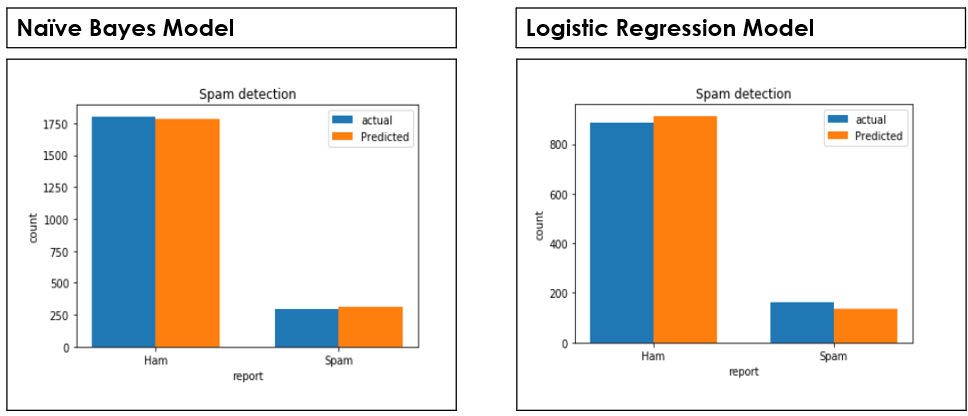
The way toward assessing programming amid the improvement procedure or toward the finish of the advancement procedure to decide if it fulfills determined business prerequisites. Approval Testing guarantees that the item really addresses the customer's issues. It can likewise be characterized as to exhibit that the item satisfies its proposed utilize when sent on proper condition.

The framework has been tried and actualized effectively and along these lines guaranteed that every one of the prerequisites as recorded in the product necessities determination are totally satisfied.

**7.5 Test Cases**

Experiments include an arrangement of steps, conditions and sources of info that can be utilized while performing testing undertakings. The principle expectation of this action is to guarantee whether a product passes or bombs as far as usefulness and different perspectives. The way toward creating experiments can likewise help discover issues in the prerequisites or plan of an application. Experiment goes about as the beginning stage for the test execution, and in the wake of applying an arrangement of information esteems, the application has a conclusive result and leaves the framework at some end point or otherwise called execution post condition.

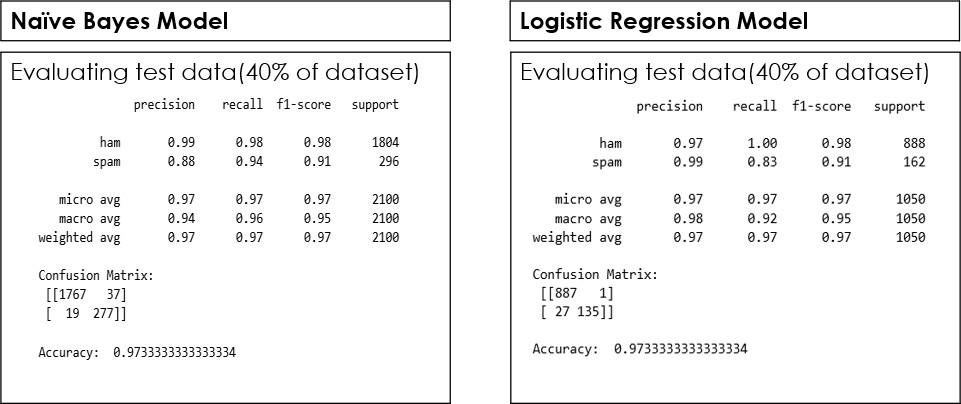
* We tested the dataset and found out which is ham and spam indicated as spam and ham.
* We calculated the accuracy of the predicted output.
* We finally compared the output with the model created using logistic regression algorithm.



#### Fig 7.1 60-40 split ration comparison

The above screenshots shows the results of the output when the naïve bayes classifier and logistic regression models are tested for 60-40 split ratio that is the percentage of data provided while training the model and testing the model respectively.

Comparing Results for Naïve Bayes and logistic Regression model with 60-40 split ratio.



**CHAPTER 8**

# CONCLUSION AND FUTURE SCOPE

### Conclusion

We proposed a novel algorithm for enhancing the accuracy of the Naive Bayes Spam Filter. The algorithm was implemented as an enhancement for Naive Bayes Classifier and also tested with logistic regression model. Naive Bayes has a very fast processing speed and allows for a small training set, hence is suitable for real-time spam filtering. We are also using Intelligent Text Modification method to identify messages containing leetspeak and diacritic. We are able to classify email as spam or ham. By creating an addition to Naive Bayes Classifier. We also found that our new addition helped improve ham classification due to the high recall and precision rates. We demonstrated that our algorithm consistently reduced the amount of spam emails misclassified as ham email.

### Contribution

We have tried to introduce leetspeak characters into the dataset and then convert them to the normal text. We then predict the output of the dataset. This is our contribution to the project.

### Future Scope

In the future we would like to create an API for the same and test that in real world environment. We will try to optimize this project for much larger amount of dataset. Since our addition successfully enhances the Naive Bayes spam filter, we will try to implement the addition onto other machine learning spam filters such as Vector Space Models, clustering, and artificial neural networks. Combining these other methods will allow the improvement of spam detection across many different systems to ultimately create a well developed spam detector for text modifications.

# REFERENCES

* + 1. Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization
    2. Email Spam Classification by Support Vector Machine
    3. Intelligent Model for Classification of SPAM and HAM
    4. Team, Radicati. "Email Statistics Report, 2015-2019. The Radicati Group." (2015).
    5. Androutsopoulos I., J. Koutsias, K. V. Chandrinos, G. Paliouras,and C. D. Spyropoulos, "An evaluation of naive bayesian antispam filtering", In: *11th European Conference on Machine Learning,* pp.9-17, Barcelona, Spain, 2000.
    6. GitHub, Inc, "Spam Assassin," 21 April 2016. [Online]. Available: https://github.com/dmitrynogin/SpamAssassin.git. [Accessed 20 August 2017].

Appendix

#!/usr/bin/env python

# coding: utf-8

# # Spam email detection

# In[1]:

#importing required libaries

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

import matplotlib.pyplot as plt

# In[2]:

#import the dataset

data=pd.read\_csv("spam\_data.csv")

# In[3]:

data

# In[4]:

data.describe()

# In[5]:

data['Category'].value\_counts().plot.bar(color=['g','r'])

print("total spam and ham in the dataset")

plt.show()

# ## Preprocessing the data

# In[6]:

data.isnull()

# In[7]:

data.shape

# In[8]:

data.isna().sum()

# In[9]:

# text preprocessing- remove the puncuation and convert the letters to lowercase and removes the words that do not contribute much for building the model which are stopwords

import string

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

def text\_preprocessing(text):

text=text.translate(str.maketrans('','',string.punctuation))

text=[i.lower() for i in text.split() if i.lower() not in stopwords.words('english')]

return " ".join(text)

# In[10]:

msg\_copy=data.Message.copy()

msg\_copy=msg\_copy.apply(text\_preprocessing)

msg\_copy

# In[11]:

vectorizer=CountVectorizer()

# now to convert the text to matrix we use fit\_transform method

msg\_matrix=vectorizer.fit\_transform(msg\_copy)

print(msg\_matrix.toarray())

# ## Spliting the data

# In[12]:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(msg\_matrix,data['Category'],random\_state=2,test\_size=0.3)

print(x\_train.shape)

print(y\_train.shape)

print(x\_test.shape)

print(y\_test.shape)

# ## Fitting the model using logistic regression

# In[13]:

from sklearn.linear\_model import LogisticRegression

logistic\_reg=LogisticRegression(solver='liblinear',penalty='l1')

logistic\_reg.fit(x\_train,y\_train)

lrprediction=logistic\_reg.predict(x\_test)

# ## Fitting the model using Naive bayes classifier

# In[14]:

from sklearn.naive\_bayes import MultinomialNB

nbclassifier=MultinomialNB()

nbclassifier.fit(x\_train,y\_train)

nbprediction=nbclassifier.predict(x\_test)

# ## Comparison of the performance of both the classifiers

# In[15]:

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score

import seaborn as sns

# performance of logisctic regression

print("Performance of logistic regression:")

print()

print('classification report:\n',classification\_report(y\_test,lrprediction))

print()

print("confuion matrix:\n",confusion\_matrix(y\_test,lrprediction))

print()

print("Accuracy score:",accuracy\_score(y\_test,lrprediction))

print("\n\nHeatmap on prediction:\n")

sns.heatmap(confusion\_matrix(y\_test,lrprediction),annot=True)

# In[16]:

# performance of Naive bayes

print("Performance of Naive bayes:")

print()

print('classification report:\n',classification\_report(y\_test,nbprediction))

print()

print("confuion matrix:\n",confusion\_matrix(y\_test,nbprediction))

print()

print("Accuracy score:",accuracy\_score(y\_test,nbprediction))

print("\n\nHeatmap on prediction:\n")

sns.heatmap(confusion\_matrix(y\_test,nbprediction),annot=True)