A Project Report on

Email/ SMS Spam Detection

Submitted in partial fulfillment of award of

BACHELOR OF TECHNOLOGY

degree
in
COMPUTER SCIENCE & ENGINEERING

By

SHUBHAM SHARMA (1900820100145)
SHUBHAM (1900820100143)
SHIVANI (1900820100138)
HARSHITA NAILWAL (2000820109001)
TABISH ABSAR (1900820100153)

Session: 2019- 2023

DR. SATENDRA KUMAR
(Assistant Professor)

SUPERVISOR

Department of Computer Science and Engineering

Moradabad Institute of Technology
Moradabad (U.P.)

MAY 2023

CERTIFICATE

Certified that the Project Report entitled **"EMAIL SPAM DETECTION"** submitted by **Shubham Sharma (1900820100145)**, **Shubham (1900820100143)**, **Shivani (1900820100138)**, **Harshita Nailwal (2000820109001)**, **Tabish Absar (1900820100153)** is their own work and has been carried out in our supervision. It is recommended that the candidates may now be evaluated for their project work by the university.

Date:

Dr. Satendra Kumar
Assistant Professor
(Project Supervisor)
Department of Computer Science and Engineering
Moradabad Institute of Technology, Moradabad (082)

ABSTRACT

The project titled “Email Spam Detection” is implemented using the CRISP-DM methodology. You will get to know Business understanding, Data Understanding (Data Description and Exploration), Data Preparation, Modelling, and Evaluation steps. Project is implemented using Python class object-based style. Email spam detection is done using machine learning algorithms Naive Bayes and SVM (Support vector machines). Further, it shows the complete program flow for Python-based email spam classifier implementation such as Data Retrieval Flow, Data Visualization Flow, Data Preparation Flow, Modeling, and Evaluation Flow. Also, including the section regarding Data ethics.

This whole project is divided into these steps: Data Cleaning (Remove unwanted/ useless data from the dataset), EDA (Exploratory data Analysis (EDA) is an analysis approach that identifies general patterns in the data by perform various operations), Text Preprocessing (In this step there are different steps we have perform like tokenization, lower case, remove special characters, remove stop words, and punctuations, etc.), Model Building (Build Machine learning model with machine learning algorithms), Evaluation, Improvements (Depending upon the evaluation), Website (in this step we perform various steps

like study about streamlit, APIs, etc.), Deploy (In this step we perform various steps like study about Render and make this project machine independent).

Short_Message Service (SMS), which allows users to send and receive messages, has become a multi-billion dollar industry as mobile phone usage has soared. The cost of messaging services has also decreased, which has led to an increase in the amount of spam that is delivered to mobile devices. Up to 40% of SMS messages in some regions of Asia were spam in 2012. Due to short message lengths, lack of reliable databases for SMS spams, informal language, and brief message characteristics, the current email filtering algorithms may not perform well in their. In this project, real SMS spam databases from the ML repository are used. Following feature extraction and preprocessing, On the databases, numerous machine learning methods are used. After comparing the results, the best algorithm for text message spam filtering is then presented. The results utilising that in this study decreases the total error rate of the best model in the original research referencing this. The following algorithms are used in this technique: Spam communications are categorised in mobile device communication using decision trees, K-Nearest Neighbour, and logistic regression The SMS spam collecting set is used to test the approach.

ACKNOWLEDGEMENT

Firstly, we would like to thank **Dr. Rohit Garg**, Director (MIT College) for permit us to make this project. Secondly, we would like to thank **Dr. Manish Gupta**, HOD (Department of CSE) for giving us opportunity to making this project. Thirdly, we would like to thank **Dr. Satendra Kumar (**student mentor), without their guidance this project seems far away from our reach. He helped us a lot regarding any technical issue and give their precious time and best advices which we have followed in this project. A paper is not enough for us to express the support and guidance we received from them almost all the work we did there. Although this project is a good example of team work and we are thankful for our team. Finally, we apologized all other unnamed who helped us in various ways to have a good training.

Last but not the least, we wish to thank our **Parents** for financing our studies in this college as well as for constantly encouraging us to learn engineering. Their personal sacrifice in providing this opportunity to learn engineering in gratefully acknowledged.

SHUBHAM SHARMA (1900820100145)

SHUBHAM (1900820100143)

SHIVANI (1900820100138)

HARSHITA NAILWAL (2000820109001)

TABISH ABSAR (1900820100153)

TABLE OF CONTENTS

CONTENTS **PAGE NO.**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Chat technology is simply one aspect of SMS. SMS technology was made possible by standard, an accepted international standard. Spam is the term for the abuse of electronic messaging services to send large numbers of unwanted messages to anybody. Even though email spam is the most well-known example, identical offences in other media and mediums are frequently referred to as "spam."

SMS In this sense, spam is frequently unsolicited bulk communications that contains some commercial interest and is quite similar to email spams. Phishing URLs and business promotion are spread via SMS spam. Commercial spammers use malware to transmit SMS since most countries outlaw the practice. Since it is challenging to pinpoint the origin of spam when it is sent from a hacked computer, spammers take less of a risk while doing so. Only letters, numbers, and a few symbols are permitted in SMS messages. A brief glance at the mails identifies [1]. Almost all spam message direct users to call a phone number or go to a website. A simple SQL query on the spam yields results that reveal this trend. Due of the low cost and large bandwidth of the SMS network, SMS spam is widely used.

Every time a user receives an SMS spam message, their mobile phone notifies them of the message's arrival. The consumer will be unhappy when they realise the message is unwanted, and SMS spam uses up some of the storage space on their mobile device.

There are several notable differences emails and text messages. Contrary, which may access a range of sizable datasets, actual databases for SMS spams are quite scarce. The number of criteria that can be utilized to classify text messages is also considerably less than those of emails due to the shorter duration of text messages. There is also no header in this case. In addition, text messages use significantly less

professional language than emails do and are chock full of acronyms. All of these elements could lead to a significant decline in the effectiveness of the most important Short text message spam filtering algorithms are utilized.

ML algorithms to the problem of classifying SMS spam, compare their results to learn more and further research the problem, and create a program based on one of these approaches that can precisely filter SMS spams [2]. A number of machine learning algorithms are then implemented using the module in Python after performing data feature extraction and basic analysis in MAT_LAB. Data is first analyzed in MAT_LAB, and then several machine learning techniques are applied using the learn module in python.

In the third installment of a three-part series, we'll examine the spam or ham classifier from the standpoint of AI ideas, experiment with several classification algorithms in a based on performance criteria [3]. A web-based Python.

Applications of machine learning in modern internet technology. Service providers have integrated spam detection algorithms that label such content as "Junk Mail" when it is received.

In this project, the naïve-bayes approach is utilized to create a model that, depending on the training data we provide the model, can classify dataset. The words "free," "win," "winner," "cash," "prize," and similar expressions are frequently used in these letters because they are meant to catch your attention and in a sense persuade you to open them.

Exclamation marks and writing in all capitals are other characteristics of spam communications. Since spam texts are often pretty evident to the receiver, we want to train a model to identify them for us. Finding spam mails is a binary classification issue since messages can only be categorized and nothing else. This is a supervised learning problem as well because we will be giving the model a tagged.

1.2 Aim and Objectives

The main objective of machine learning models is to learn automatically without any intervention from humans. Machine learning consists of three major kinds, used for numerous tasks.

For the last decade, researchers have been trying to make email communication better than today. Spam filtering of emails is one of the most critical ways of protecting email networks [4]. Many research articles have been published using various machine learning approaches to identify and process spam emails, but there are still some research gaps. Junk mail is one of the central, attractive research fields for filling the gaps. For this reason, many spam classification studies have already been carried out using several methods to make email communication more trustworthy and valuable for users. That is why, this paper is presented to make a summarized version of different existing machine learning models and approaches that are being used for email spam detection. This paper also evaluates the most common machine learning approaches like KNN, SVM, random forest, and Naïve Bayes.

In machine learning method, supervised machine learning algorithms are machine learning models that need labeled data. Initially, labeled training data is provided to these models for training, and after training models predict future events. In other words, these models begin with the analysis of an existing training dataset, and they generate a method to make predictions of success values. Upon proper training, the system can provide the prediction on any new data related to the user's data at the training time. Furthermore, the learning algorithm accurately compares the output to the expected output and identifies errors to modify the model.

Supervised learning uses labeled data for training, and then it can predict the new data. This type of learning can be used in solving various problems, i.e., advertisement popularity, spam classification, face recognition, and object classification.

CHAPTER 2

LITERATURE SURVEY

2.1 Data Mining

As Literature survey is that the most vital step in code development method. Before developing the tool it's necessary to see the time issue, economy and company strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool once the programmers begin building the tool the programmers would like heap of external support. This support is obtained from senior programmers, from book or from websites before building the system the on top of thought area unit taken under consideration for developing the projected system. We have to analyze the Data mining Outline Survey:

2.1.1 Data Mining Survey

Data mining is a data analysis technique which allows us to study and identify different patterns and relationships between the data. In other words, data mining is a technique which can be employed to extract information from large and extensive datasets and convert the information into a prominent structure so that it can be used further for gaining inference and knowledge on the data [5].

Data mining contains techniques for analysis which involve various domains. For instance, some of the domains involved in data mining are Statistics, Machine Learning and Database systems. Data mining is additionally spoken as “Knowledge discovery in databases (KDD)”.

The real assignment of data mining systems is the semi-automatic or computerized analysis of huge volumes of data to extract earlier unknown relationships such as

groups of data members (clustering analysis), unusual records (outlier or anomaly detection), and dependencies. Normally, this contains database techniques like spatial indices.

These relationships that are discovered can be used as input data or may also be used in depth analysis for example, in machine learning or predictive analysis.

Data mining can identify multiple groups in the data that can be put to further than use for accurate projections by a decision support system [6].

Fig. 2.1: Data Mining

2.1.2 Stages in Data Mining

There are 4 major steps in data mining which are described as follows:

- **Data Sources:** This stage includes gathering the data or making a dataset on which the analysis or the study has performed. The datasets can be of many forms for instance, they can be new letters, databases, excel sheets or various other sources like websites, blogs and social media.

An appropriate dataset must be chosen in order to perform an efficient study or analysis. The dataset must be chosen which is appropriate and well suited with respect to the problem definition.

• **Data Exploration:** This step includes preparing the data properly for analysis and study. This step is mainly focused on cleaning the data and removing the anomalies from the data. As there is a large amount of data there is always a great chance that some of the data might be missing or some data might be wrong. Thus, for efficient analysis we require the data to be maintained properly. This process includes removing the incorrect data and replacing the data which is missing with either mean or median of the whole data. This step is also generally known as data pre-processing.

• **Data Modeling:** In this step the relationships and patterns that were hidden in the data are examined and extracted from the datasets. The data can be modeled based on the technique that is being used. Some of the different techniques that can be used for modeling data are clustering, classification and association and decision trees.

• **Deploying Models:** Once the relationships and patterns present in the data are discovered we need to put that knowledge to use. These patterns can be used to predict events in the future and they can be used for further analysis. The discovered patterns can be used as inputs for machine learning and predictive analysis for the datasets.

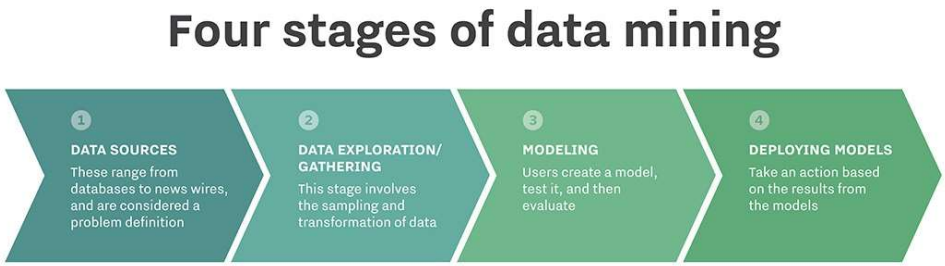


Fig. 2.2: Stages in Data Mining

2.1.3 Techniques in Data Mining

- **Classification:** This technique is used to divide various data into different classes. This process is also similar as clustering. It segments data records into various segments which are known as classes. Unlike clustering, here we have knowledge of different clusters. Ex: Outlook email, they have an algorithm to categorize an email as legitimate or spam.

- **Association:** This technique is used to discover hidden patterns in the data and identifying interesting relations between the variables in a database. Ex: It is used in retail industry.

- **Prediction:** This technique is used only for uses. It is used extract relationships between independent and dependent variables in the dataset. Ex: We use this technique to predict profit obtained from sales for the future.

- **Clustering:** A cluster is referred to as a group of data objects. The data objects that are similar in properties are kept in the same cluster. In other words we can tell that clustering is a process of discovering groups or clusters. Here we do not have prior knowledge of the clusters [7]. Ex: It can be used in consumer profiling.

- **Sequential Patterns:** This is an essential aspect of data mining techniques its main aim is to discover similar patterns in the dataset. Ex: E-commerce websites suggestions are based on what we have bought previously.

- **Decision Trees:** This technique is a vital role in data mining because it is easier to understand for the users. The decision tree begins with a root which is a simple question. As they can have multiple answers we get our nodes of the decision tree also the questions in the root node might lead to another set of questions. Thus, the nodes keep adding in the decision tree. At last, we are allowed for making a final decision on it. Apart from these techniques there are certain other techniques which allow us to remove noisy data and clean the dataset. This helps us to get accurate analysis and prediction results.

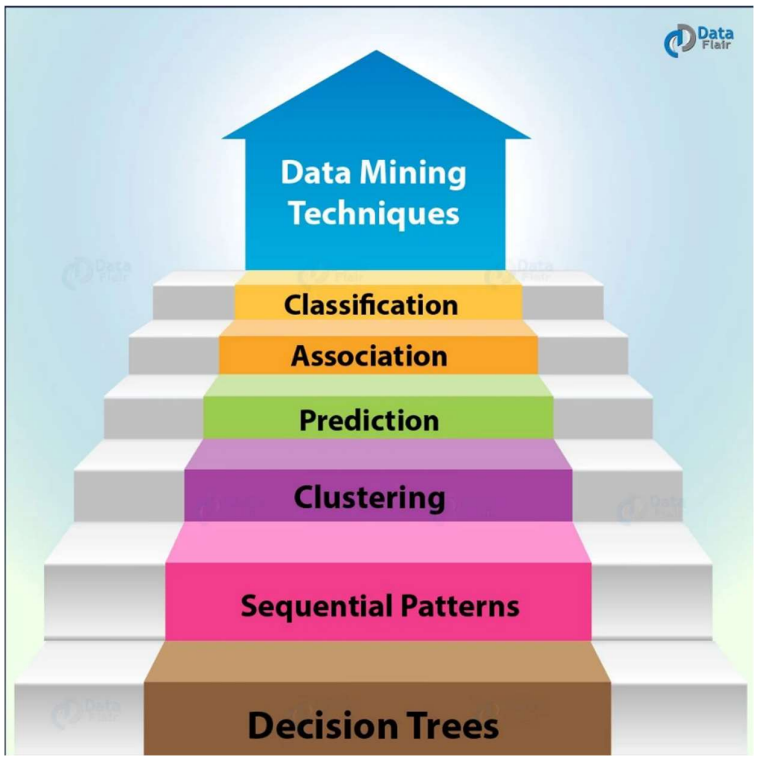


Fig. 2.3: Techniques in Data Mining

2.1.4 Benefits of Data Mining

Data mining has various uses in various sectors of the society:

- In finance sector, it can be used for modeling risks accurately regarding loans and other facilities.
- In marketing, it can be used for predicting profits and can be used for creating targeted advertisements for various customers.
- In retail sector, it is used for improving consumer experience and increasing the amount of profits.
- Tax governing organizations use it to determine frauds in transactions.

2.2 Existing System

Mainly, the existing spam detection methods are divided into two types: behavior pattern-based approaches and semantic pattern-based approaches. These approaches have their limitations and drawbacks. There has been significant growth in spam emails, along with the rise of the Internet and communication around the globe. Spams are generated from any location of the world with the Internet's help by hiding the attacker's identity. There are a plenty of antispam tools and techniques, but the spam rate is still very high.

The most dangerous spams are malicious emails containing links to malicious websites that can harm the victim's data. Spam emails can also slow down the server response by filling up the memory or capacity of servers. To accurately detect spam emails and avoid the rising email spam issues, every organization carefully evaluates the available tools to tackle spam in their environment. Some famous mechanisms to identify and analyze the incoming emails for spam detection are Whitelist/Blacklist, mail header analysis, keyword checking, etc. Social networking experts estimate that 40% of social network accounts are used for spam [8]. The spammers use popular social networking tools to target specific segments, review pages, or fan pages to send hidden links in the text to pornographic or other product sites designed to sell something from fraudulent accounts.

Kaur and Verma present a survey on email spam detection using a supervised approach with feature selection. They discuss the knowledge discovery process for spam detection systems. They also elaborate various techniques and tools proposed for spam detection. The choice of features based on N-Gram is also addressed in this survey. N-Gram is a predictive-based algorithm used to predict the probability of the next word occurrence after finding $N-1$ terms in a sentence or text corpus. N-Gram uses probability-based techniques for the next word prediction. They compare various machine learning (multilayer perceptron neural network support vector machine, Naïve Bayes) and non-machine learning (Signatures, Blacklist and Whitelist, and mail header checking) approaches for email spam detection.

2.3 Proposed System

As per the things seen it is necessary to propose the mechanism in which mail are going to cross verify the mail content in which we are going to filter both content and links of shared email. Most probably the spam mails contain the malicious link in which URL classification or parsing need to be work out. So that in proposed we analyze the URL data as well as mail content [9]. This research will experiment Bio-inspired algorithms alongside Machine learning models. This will be conducted on different spam email corpora that are publicly available. The paper aims to realize the subsequent objectives:

1) To investigate machine learning algorithms for the spam detection problem.

2) to research the workings of the algorithms with the acquired datasets.

3) To implement the bio-inspired algorithms.

4) to see and compare the accuracy of base models with bio-inspired implementation

5) To implement the framework using Python.

Scikit-Learn library will be instigated to perform the experiments with Python, and this will enable to edit the models, conduct pre-processing, and calculate the results. The program scripts will be implemented further with the optimization techniques and compared with the base results i.e., with default parameters.

2.3.1 Modules of proposed System

The necessary stages that must be observed:

ADDING CORPUS: This section will load all the email datasets within the program and distribute into training and testing data. This process will be accepting the datasets in'*.txt' format for all email (Ham and Spam). This is to help understand the real-world issues and how can they be tackled.

Pre-processing: this is often the primary stage that's executed whenever an incoming mail is received. This step consists of tokenization.

Tokenization: this is often a process that removes the words the body of an email. It also translates a message to its meaningful parts. It takes the e-mail and divides it into a sequence of representative symbols called tokens. In a tokenization phase every word is assigned a singular token.

Stemming: subsequent step to be performed is stemming. Stemming is employed to seek out 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 JAVA implementation of Porter stemmer which is slightly modified to satisfy our needs. The files are named with an extension 'words stemmed'.

Stop Words: This was wont to remove the unnecessary words and characters within each email and creates a bag of words for the algorithms to match against.

Feature selection: Sequel to the pre-processing stage is that the feature selection phase. Feature selection a sort of reduction within the measure of spatial coverage that effectively exemplifies fascinating fragments of email message as a compressed feature vector. The technique is useful when the dimension of the message is large, and a condensed feature representation is required to form the task of text or image matching snappy.

2.4 Software Description

2.4.1 Jupyter Notebook

The Jupyter Notebook App is a server-customer application that permits altering and running note pad records by means of an internet browser. The Jupyter Notebook App can be executed on a nearby work area requiring no web access as portrayed in this report or can be introduced on a remote server and got to through

the web. A scratch pad part is a computational motor that executes the code contained in a Notebook record.

When you open a Notebook report, the related part is consequently propelled. At the point when the scratch pad is executed either cell-by-cell, the portion plays out the calculation and produces the outcomes. Contingent upon the sort of calculations, the piece may expend critical CPU and RAM. Note that the RAM isn't discharged until the part is closed down, he Notebook Dashboard is the part which is indicated first when you dispatch Jupyter Notebook App. The Notebook Dashboard is essentially used to open note pad archives, and to deal with the running portions [10]. The Notebook Dashboard has different highlights like a record director, in particular exploring organizers, renaming and erasing documents.

Fig. 2.4: Jupyter Notebook

2.4.2 Pycharm

PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester,

integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains.

It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

PyCharm was released to the market of the Python-focused IDEs to compete with PyDev (for Eclipse) or the more broadly focused Komodo IDE by ActiveState.

The beta version of the product was released in July 2010, with the 1.0 arriving 3 months later. Version 2.0 was released on 13 December 2011, version 3.0 was released on 24 September 2013, and version 4.0 was released on November 19, 2014.

PyCharm became Open Source on 22 October 2013. The Open Source variant is released under the name Community Edition – while the commercial variant, Professional Edition, contains closed-source modules.

Fig. 2.5: Pycharm

2.5 Python Library Description

2.5.1 Numpy

NumPy is one of the bundles that we can't miss when we are learning information science, principally in light of the fact that this library gives us a cluster information structure that holds a few advantages over Python records, for example, being increasingly reduced, quicker access in perusing and composing things, being progressively advantageous and increasingly productive.

NumPy is a Python library that is the center library for logical registering in Python. It contains an accumulation of apparatuses and strategies that can be utilized to settle on a PC numerical models of issues in Science and Engineering. One of these apparatuses is an elite multidimensional cluster object that is an incredible information structure for effective calculation of exhibits and lattices.

To work with these clusters, there's a tremendous measure of abnormal state scientific capacities work on these grids and exhibits. Since you have set up your condition, it's the ideal opportunity for the genuine work. In fact, you have officially gone for some stuff with exhibits in the above Data camp Light pieces. We haven't generally gotten any genuine hands-on training with them, since we originally expected to introduce NumPy all alone pc [11] . Since we have done this current, it's a great opportunity to perceive what you have to do so as to run the above code pieces without anyone else. A few activities have been incorporated underneath with the goal that you would already be able to rehearse how it's done before we begin our own. To make a numpy exhibit, we can simply utilize the np.array () work. There's no compelling reason to proceed to retain these NumPy information types in case we are another client, but we do need to know and mind what information we are managing. The information types are there when we need more power over how our information is put away in memory and on plate. Particularly in situations where we are working with broad information, it's great that we know to control the capacity type.

Fig. 2.6: NumPy

2.5.2 Pandas

Pandas is an open-source, BSD-authorized Python library giving elite, and simple to-utilize information structures and information examination instruments for the Python programming language. Python with Pandas is utilized in a wide scope of fields including scholastic and business areas including money, financial matters, Statistics, examination, and so on. In this instructional exercise, we will get familiar with the different highlights of Python Pandas and how to utilize them practically speaking. This instructional exercise has been set up for the individuals who try to become familiar with the essentials and different elements of Pandas. It will be explicitly valuable for individuals working with information purging and examination. In the wake of finishing this instructional exercise, we will wind up at a moderate dimension of ability from where you can take yourself to more elevated amounts of skill [12]. We ought to have a fundamental comprehension of Computer Programming phrasing. Library utilizes vast majority of the functionalities of NumPy. It is recommended that we experience our instructional exercise on NumPy before continuing with this instructional exercise.

Fig. 2.7: Pandas

2.5.3 Matplotlib

People are exceptionally visual animals, we comprehend things better when we see things envisioned. The progression to showing investigations, results or bits of knowledge can be a bottleneck, we probably won't realize where to begin or you may have as of now a correct configuration as a top priority, however then inquiries will have unquestionably gone over your brain.

When we are working with the Python plotting library Matplotlib, the initial step to responding to the above inquiries is by structure up information on themes. Plot creation, which could bring up issues about what module we precisely need to import pylab, how we precisely ought to approach instating the figure and the Axes of our plot, how to utilize matplotlib in Jupyter note pads.

Plotting schedules, from straightforward approaches to plot your information to further developed methods for picturing your information. Essential plot customizations, with an emphasis on plot legends and content, titles, tomahawks marks and plot format [13].

Since all is set for us to begin plotting your information, it's an ideal opportunity to investigate some plotting schedules. We'll regularly go over capacities like plot() and disperse(), which either draw focuses with lines or markers interfacing them, or draw detached focuses, which are scaled or shaded. In any case, as you have just found in the case of the primary area, we shouldn't neglect to pass the information that you need these capacities to utilize.

In conclusion, we will quickly cover two manners by which we can alter Matplotlib, with templates and the settings.

2.5.4 Scikit-Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010 [14].

It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

Fig. 2.8: Scikit-Learn

2.5.5 Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

Different categories of plot in Seaborn:

Plots are basically used for visualizing the relationship between variables. Those variables can be either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories –

Relational plots: This plot is used to understand the relation between two variables.

Categorical plots: This plot deals with categorical variables and how they can be visualized.

Distribution plots: This plot is used for examining univariate and bivariate distributions.

Regression plots: The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.

Matrix plots: A matrix plot is an array of scatterplots.

Multi-plot grids: It is an useful approach is to draw multiple instances of the same plot on different subsets of the dataset.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Functional Requirements

The functions of software systems are defined in functional requirements and the behavior of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality [15].

- Our system should be able to read the data and preprocess data.
- It should be able to analyze the Spam data.
- It should be able to group data based on hidden patterns.
- It should be able to assign a label based on its data groups.
- It should be able to split data into train set and test set.
- It should be able to train model using train set.
- It must validate trained model using test set.
- It should be able to classify the Spam or Ham Message.

3.2 Non-Functional Requirements

Nonfunctional requirements illustrate how a system must behave and create constraints of its functionality. This type of constraints is also known as the system's quality features. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the user are described by the specification. We must contain only those needs that are appropriate for our design.

Some Non-Functional Requirements are as follows:

- Reliability
- Maintainability
- Performance
- Portability
- Scalability
- Flexibility

3.2.1 Accessibility

Availability is a general term used to depict how much an item, gadget, administration, or condition is open by however many individuals as would be prudent. In our venture individuals who have enrolled with the cloud can get to the cloud to store and recover their information with the assistance of a mystery key sent to their email ids. UI is straightforward and productive and simple to utilize.

3.2.2 Maintainability

In programming designing, viability is the simplicity with which a product item can be altered as:

- Correct absconds
- Meet new necessities

New functionalities can be included in the task based the client necessities just by adding the proper documents to existing venture utilizing ASP. Net and C# programming dialects. Since the writing computer programs is extremely straightforward, it is simpler to discover and address the imperfections and to roll out the improvements in the undertaking.

3.2.3 Scalability

Framework is fit for taking care of increment all out throughput under an expanded burden when assets (commonly equipment) are included. Framework can work ordinarily under circumstances, for example, low data transfer capacity and substantial number of clients.

3.2.4 Portability

Portability is one of the key ideas of abnormal state programming. Convenient is the product code base component to have the capacity to reuse the current code as opposed to making new code while moving programming from a domain to another. Venture can be executed under various activity conditions gave it meet its base setups. Just framework records congregations would need to be designed in such case.

3.3 Hardware Requirements

- Processor : Any Processor above 500 MHz
- RAM : 4 GB
- Hard Disk : 500 GB
- System : Pentium IV 2.4 GHz

Any system with above or higher configuration is compatible for this project.

3.4 Software Requirements

- Operating system : Windows 7/8/9/10
- Programming language : Python
- IDE: Jupyter Notebook, Pycharm

CHAPTER 4

SYSTEM DESIGN

4.1 Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part [16].

Before you begin making an activity diagram, you should first understand its makeup.

Some of the most common components of an activity diagram include:

- Action: A step in the activity wherein the users or software perform a given task. In Lucid chart, actions are symbolized with round-edged rectangles.
- Decision node: A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.

- Control flows: Another name for the connectors that show the flow between steps in the diagram.
- Start node: Symbolizes the beginning of the activity. The start node is represented by a black circle.
- End node: Represents the final step in the activity. The end node is represented by an outlined black circle.

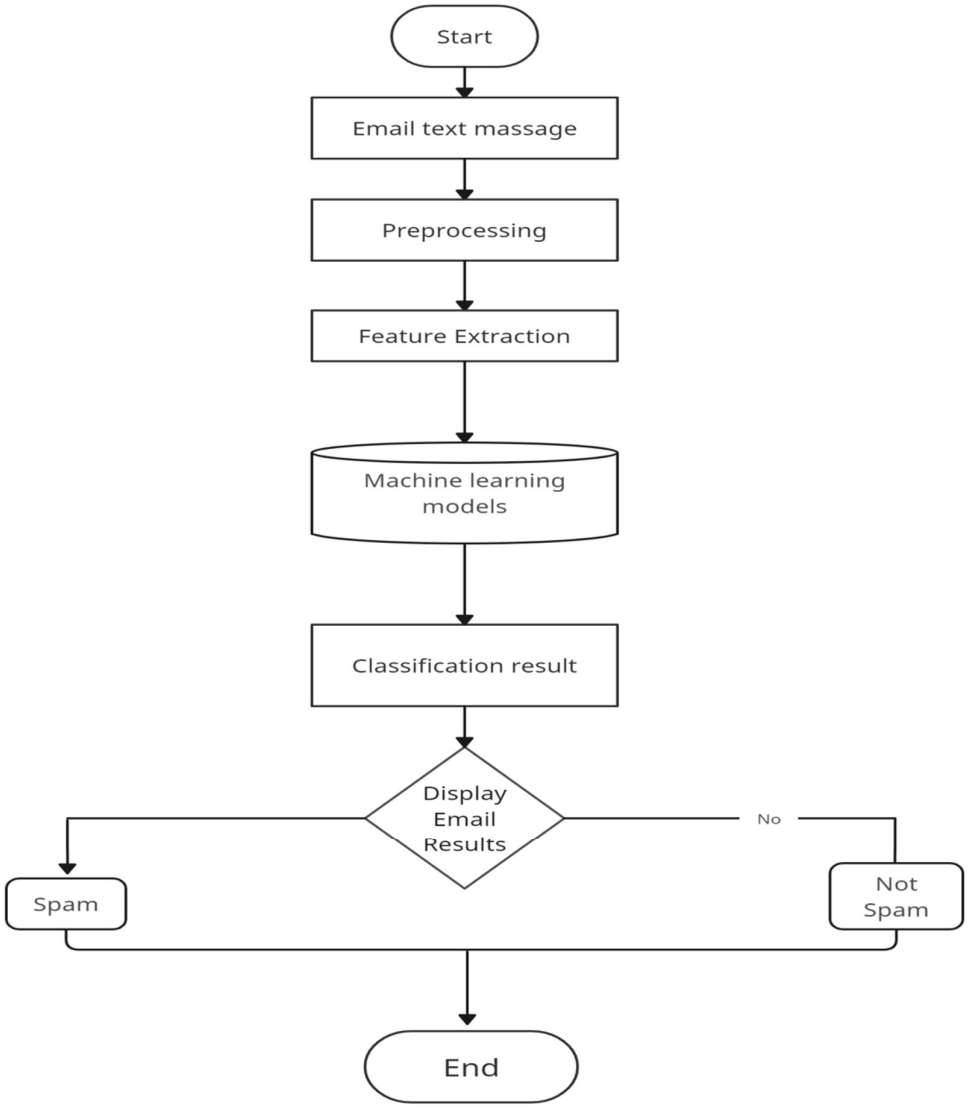


Fig. 4.1: Activity diagram

4.2 Class Diagram:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

The purpose of class diagram is to model the static view of an application [17]. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different.

It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram −

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

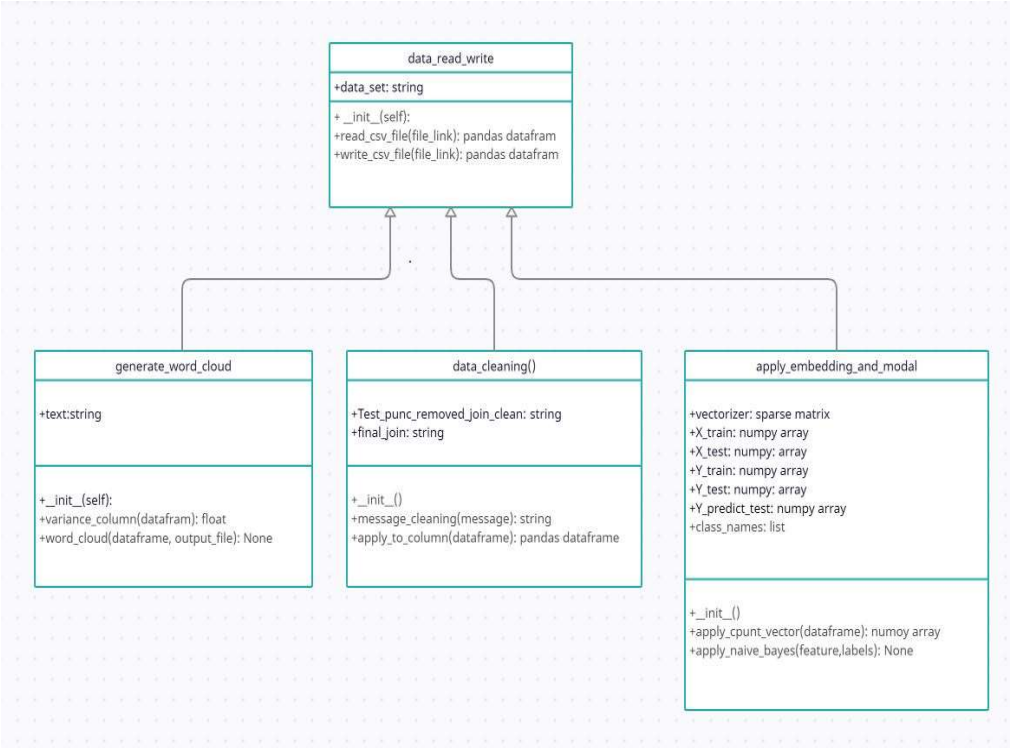


Fig. 4.2: Class Diagram

4.3 ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes [18]. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

There are usually three models people refer to based on the level of detail you want to show: conceptual ERD, logical ERD, and physical ERD.

- **Conceptual ERD or data model**: This model has the most abstraction and least amount of detail, as such it's appropriate for large projects that need a higher level view used by business analysts.
- **Logical ERD or data model**: This model adds more detail to the conceptual model by defining additional entities that are operational and transactional.
- **Physical ERD or data model**: This model serves as the actual design or blueprint of the database with lots of technical details including defining

cardinality and showing primary and foreign keys of entities instead of just their abstract semantic names.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types.
- **Ellipses:** Symbol represent attributes.
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types.
- **Primary key:** attributes are underlined.
- **Double Ellipses:** Represent multi-valued attributes.

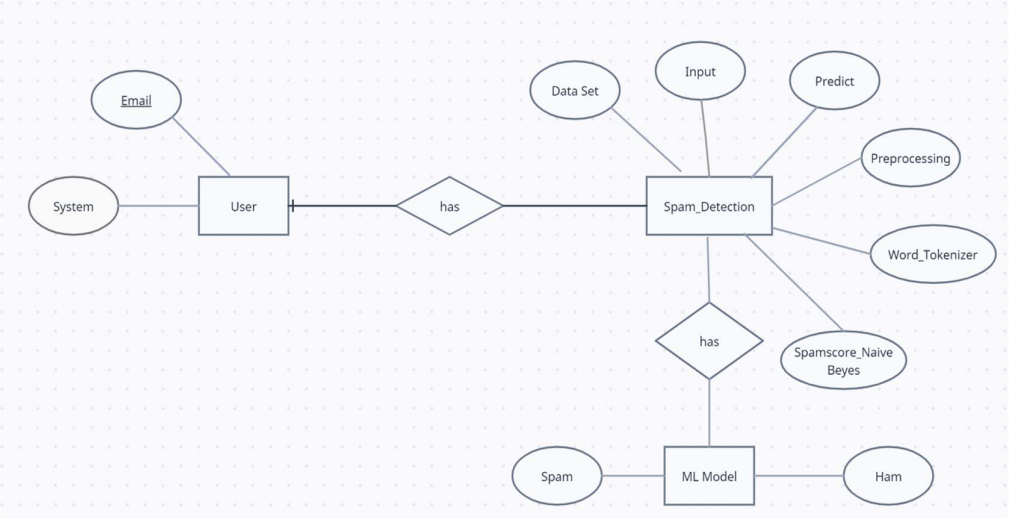


Fig. 4.3: ER Diagram

4.4 Data Flow Diagram

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present [19]. Specific operations depending on the type of data can be explained by a flowchart.

It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analyzing existing as well as proposed system.

Symbols Used in DFD:

- **Square Box:** A square box defines source or destination of the system. It is also called entity. It is represented by rectangle.
- **Arrow or Line:** An arrow identifies the data flow i.e. it gives information to the data that is in motion.
- **Circle or bubble chart:** It represents as a process that gives us information. It is also called processing box.
- **Open Rectangle:** An open rectangle is a data store. In this data is store either temporary or permanently.

Levels of DFD:

DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

- 0-level DFD: It represents the entire system as a single bubble and provides an overall picture of the system.
- 1-level DFD: It represents the main functions of the system and how they interact with each other.

- 2-level DFD: It represents the processes within each function of the system and how they interact with each other.
- 3-level DFD: It represents the data flow within each process and how the data is transformed and stored.

4.4.1 Zero Level DFD Diagram:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

This is the highest-level DFD, which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes.

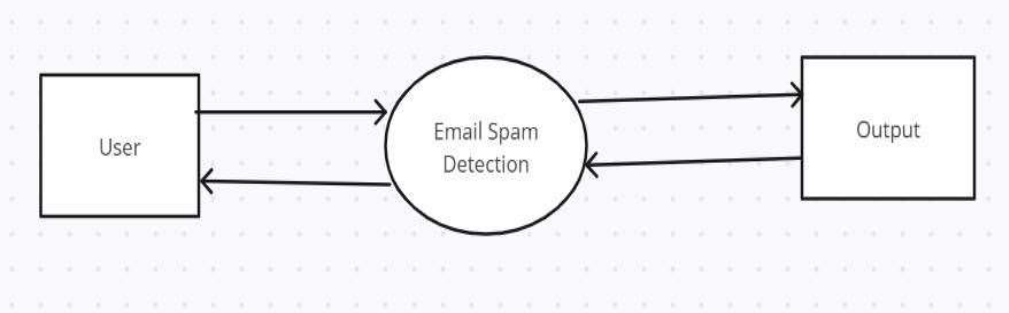


Fig. 4.4: Zero Level DFD Diagram

4.4.2 First Level DFD Diagram:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub processes.

This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes. Each sub-

process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown.

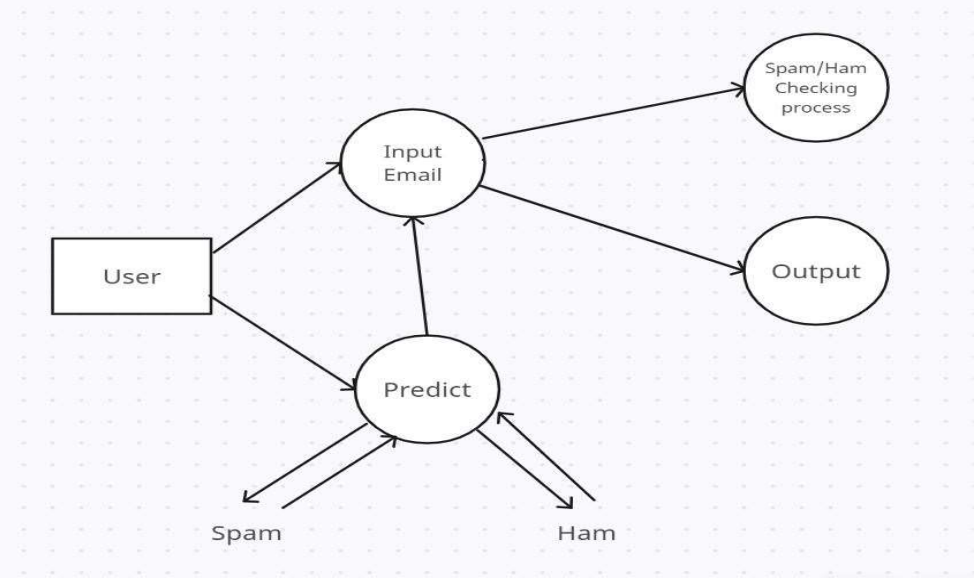


Fig. 4.5: First Level DFD Diagram

4.4.3 Second Level DFD Diagram:

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes. Each sub-process is depicted as a separate process on the level 2 DFD. The data flows and data stores associated with each sub-process are also shown.

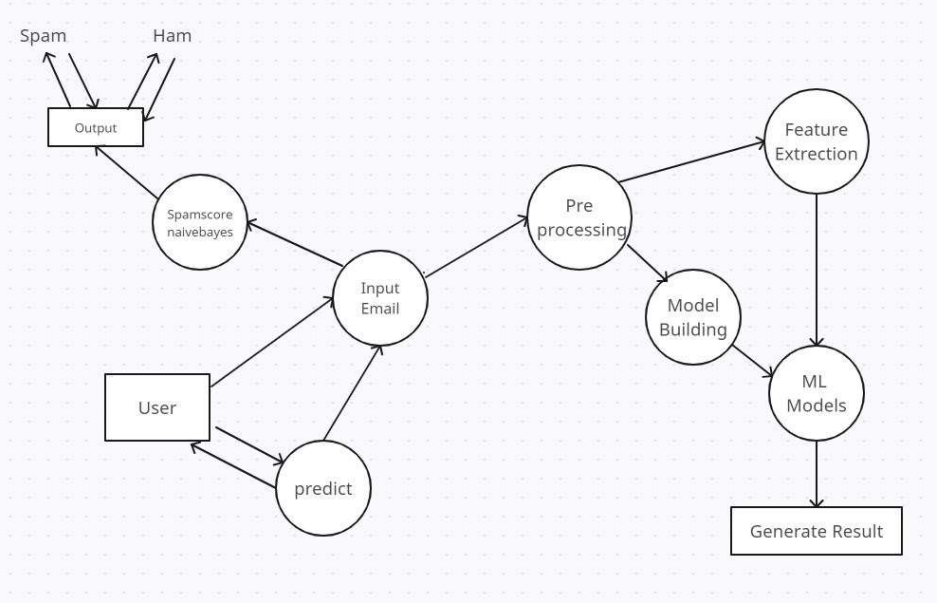


Fig. 4.6: Second Level DFD Diagram

CHAPTER 5

IMPLEMENTATION

5.1 DATA CLEANING

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time [20].

Data cleaning, also known as data cleansing or data preprocessing, is a crucial step in the data science pipeline that involves identifying and correcting or removing errors, inconsistencies, and inaccuracies in the data to improve its quality and usability. Data cleaning is essential because raw data is often noisy, incomplete, and inconsistent, which can negatively impact the accuracy and reliability of the insights derived from it.

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3920	ham	Do 1 thing! Change that sentence into: \Becaus...	NaN	NaN	NaN
785	ham	She was supposed to be but couldn't make it, s...	NaN	NaN	NaN
2111	ham	Yar he quite clever but aft many guesses lor. ...	NaN	NaN	NaN
4920	ham	Its so common hearin How r u? Wat r u doing? H...	NaN	NaN	NaN
3095	ham	We walked from my moms. Right on stagwood pass...	NaN	NaN	NaN

Fig. 5.1: Dataset before cleaning.

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

Fig. 5.2: Dataset after cleaning.

5.2 EDA (Exploratory Data Analysis)

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions. The main purpose of EDA is to help look at data before making any assumptions [21]. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

The role of data exploration analysis is based on the use of objectives achieved as above. After formatting the data, the performed analysis indicates patterns and trends that help to take the proper actions required to meet the expected goals of the business. As we expect specific tasks to be done by any executive in a particular job position, it is expected that proper EDA will fully provide answers to queries related to a particular business decision. As data science involves building models for prediction, they require optimum data features to be considered by the model. Thus, EDA ensures that the correct ingredients in patterns and trends are made available for training the model to achieve the correct outcome, like a successful recipe. Therefore, carrying out the right EDA with the correct tool based on befitting data will help achieve the expected goal.

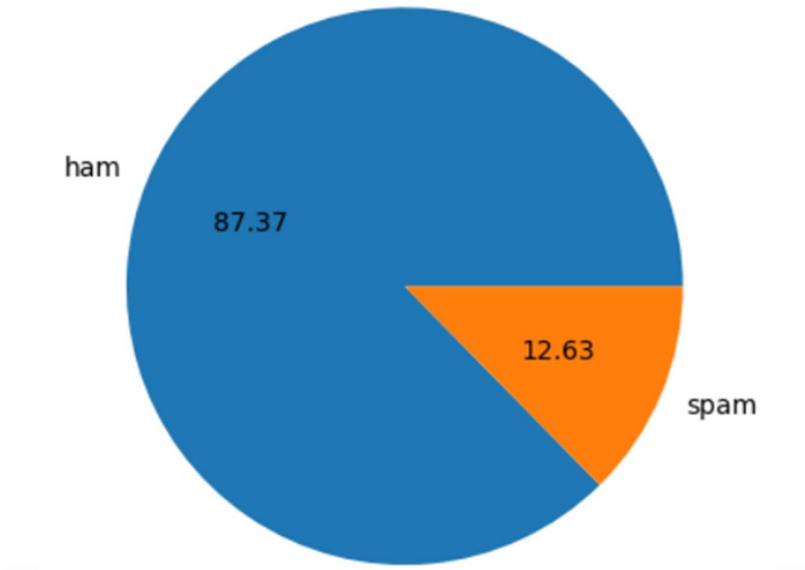


Fig. 5.3: Pie chart for spam and ham messages.

In this pie chart, the blue color represents ham messages, and the orange color represents spam messages. As we can see, the majority of messages (87.37%) are ham, while only a small proportion (12.63%) are spam.

5.3 Data Preprocessing

Data preprocessing refers to the series of steps taken to clean, transform, and prepare raw data for analysis by machine learning algorithms. The process involves converting data from its original format into a form that can be easily understood and analyzed by machine learning models.

Data preprocessing is essential in machine learning because it can greatly affect the accuracy of the resulting models. By cleaning and preparing the data, machine learning algorithms can more accurately identify patterns and relationships within the data, resulting in better predictions [22].

The following some common techniques are used in data preprocessing:

5.3.1 Lower Case

Lowercase in data preprocessing refers to the process of converting all letters in a string to their corresponding lowercase forms. This is a common step in text data preprocessing, where text data is transformed into a format suitable for natural language processing (NLP) or other machine learning tasks.

Lowercasing is important in NLP because it helps to reduce the complexity of text data by treating different letter cases as the same. For example, by converting all letters in a text document to lowercase, the machine learning algorithm will treat the words "The" and "the" as the same word, reducing the number of unique words in the dataset and simplifying the text analysis process.

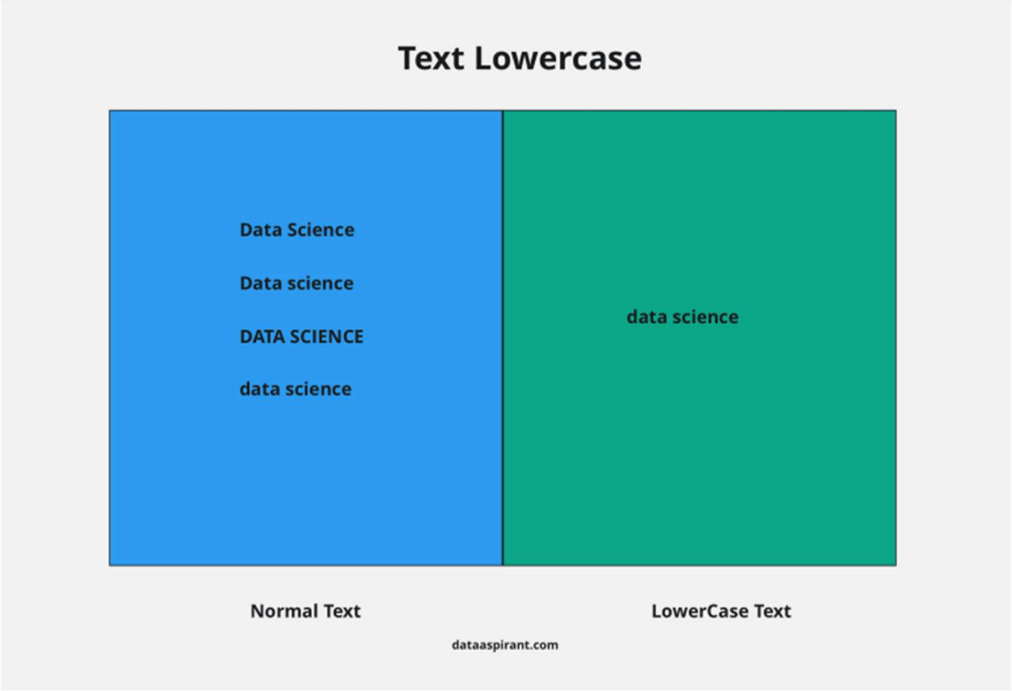


Fig. 5.4: Convert text to Lover Case

5.3.2 Tokenization

Tokenization in data preprocessing refers to the process of breaking down a text document or string into individual units, called tokens. These tokens can be individual words, phrases, or even sentences, depending on the level of granularity needed for the task at hand. Tokenization is an important step in text data preprocessing, as it allows machine learning algorithms to work with discrete units of text data rather than entire documents or strings. This can help to simplify the analysis process and enable the use of more advanced text analysis techniques.

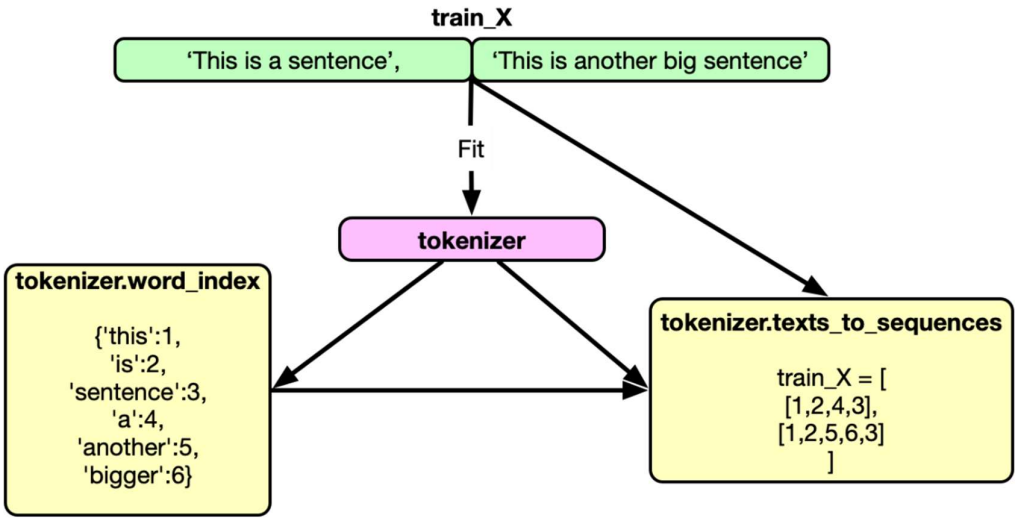


Fig. 5.5: Tokenization

5.3.3 Removing Special Characters

Removing special characters is an important step in data preprocessing, particularly when working with text data. Special characters are characters that are not letters, numbers, or punctuation marks, such as emojis, symbols, or other non-standard characters. These special characters can cause issues during text analysis or machine learning tasks, as they may not be recognized by the algorithms or may interfere with the analysis process. Removing special characters is typically done

by replacing them with spaces or removing them entirely from the text. This can be accomplished using a variety of programming languages and text processing tools. For example, in Python, regular expressions can be used to identify and remove special characters from a string or text document.

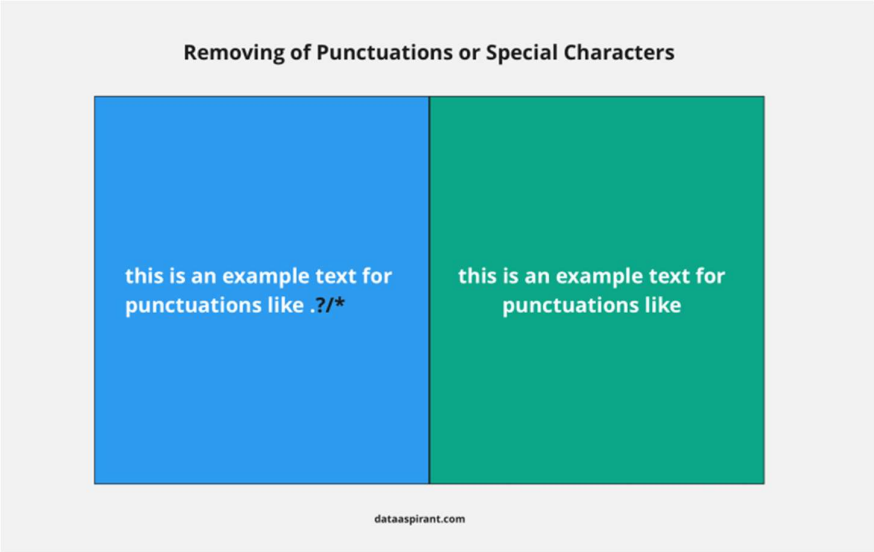


Fig. 5.6: Removing Special Characters

5.3.4 Removing stop words and Punctuation

Removing stop words and punctuation is a common step in text data preprocessing, particularly in natural language processing (NLP) applications. Stop words are commonly used words in a language that typically do not carry significant meaning, such as "a", "an", "the", "in", "on", "and", and "but". Punctuation marks are symbols used to clarify the meaning of text, such as commas, periods, question marks, and exclamation points. Removing stop words and punctuation can help to reduce the dimensionality of the text data and improve the accuracy of text analysis or machine learning tasks [23] . By removing stop words, we can focus on the more meaningful words in the text that are likely to carry the most important information. Removing punctuation can also help to simplify the text data and eliminate potential sources of noise.

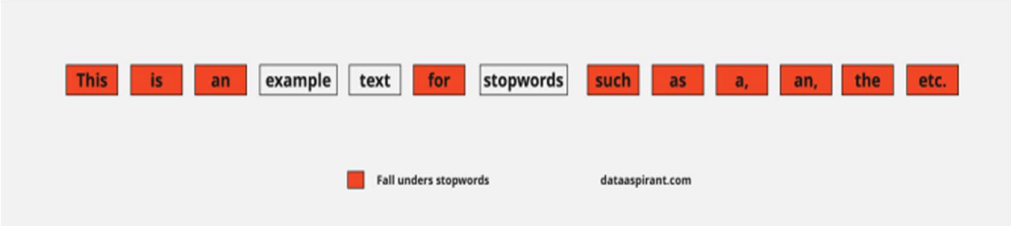


Fig. 5.7: Removing Stopwords

5.3.5 Stemming

Stemming is a technique used in data preprocessing to reduce words to their base or root form, by removing suffixes or prefixes from the word. The resulting word may not be an actual word, but it is a standard form of the original word that is used to capture its meaning. For example, the stem of the words "running", "runs", and "runner" is "run". Stemming is commonly used in natural language processing (NLP) and text mining to simplify text data and reduce its dimensionality [24] . By reducing words to their base form, stemming can help to group together words with similar meanings and reduce the number of unique words in a dataset. This can make text analysis or machine learning tasks more efficient and accurate.

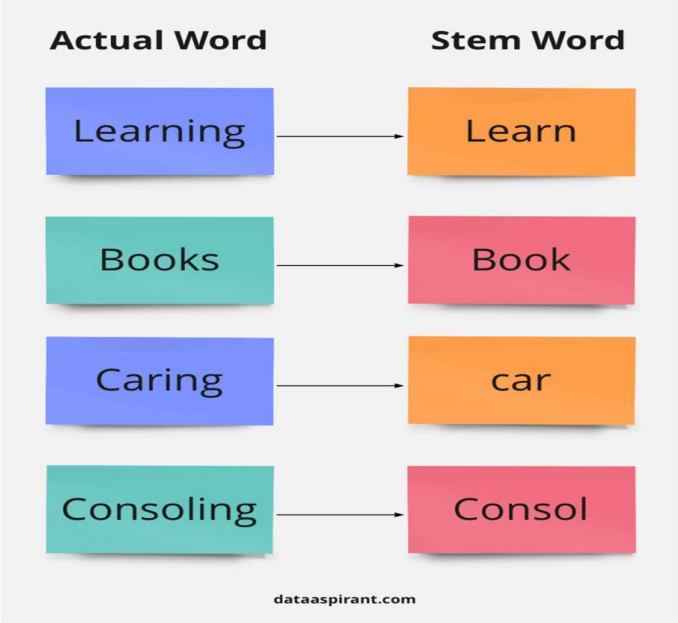


Fig. 5.8: Stemming

5.4 Model Building

The process of building a model for a spam email classifier involves a number of key steps. Common features used in spam email classifiers include the frequency of certain words, the presence of certain keywords or phrases, and the length or structure of the email. Once the features have been extracted, an appropriate machine learning algorithm is chosen to build the classifier. Naive Bayes, Support Vector Machines (SVMs), and Random Forests are among the most common algorithms used in spam email classification. The chosen algorithm is then trained on the preprocessed and feature-extracted email data, and evaluated using a separate test dataset to assess its accuracy, precision, recall, and F1 score. If necessary, the model may be optimized by adjusting hyper parameters or exploring alternative algorithms or feature sets. Finally, the optimized model is deployed to classify new incoming emails as spam or non-spam. The overall goal of model building for a spam email classifier is to develop an accurate and reliable model that can effectively distinguish between spam and non-spam emails, and help to protect users from unwanted and potentially harmful email messages.

5.4.1 Logistic Regression

Logistic regression is a machine learning algorithm used for binary classification tasks, where the goal is to predict the probability that an input belongs to one of two possible classes (e.g., spam vs. non-spam emails). The algorithm works by fitting a logistic function (also known as the sigmoid function) to the input data, which maps the input values to a probability score between 0 and 1. To train a logistic regression model, the algorithm uses a set of labeled training data to estimate the optimal values of the model parameters, which are the coefficients of the logistic function. The optimization process involves minimizing a loss function, which measures the difference between the predicted probabilities and the true class labels in the training data. One common loss function used in logistic regression is the binary cross-entropy loss.

Once the model has been trained, it can be used to make predictions on new, unlabeled data. To do this, the algorithm takes the input features and applies the logistic function to compute the predicted probability score [25]. If the probability score exceeds a certain threshold (e.g., 0.5), the algorithm predicts that the input belongs to the positive class (e.g., spam), otherwise it predicts that it belongs to the negative class (e.g., non-spam).

Logistic regression is a simple and interpretable algorithm that can be trained quickly on large datasets. It is also relatively robust to noisy or irrelevant features, and can be regularized to prevent overfitting. However, it may not perform well on highly non-linear or complex datasets, and may require feature engineering or preprocessing to achieve optimal performance.

5.4.2 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning [26]. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane [27]. It becomes difficult to imagine when the number of features exceeds 3.

SVM chooses the vectors that helps in creating the hyperplane. These extreme

cases are called as support vectors, and hence this algorithm is termed as Support Vector Machine
SVM can be of two types:

SVM: It is used for the linearly separable data that if dataset can be classified into two classes by using a straight line, then such data is termed as linearly separable data, and classifier called as Linear SVM classifier.

Non-linear SVM: It is used for non-linearly separated data, which means if dataset cannot be classified by straight line, then such data is termed as non-linear data and classifier called as Non-linear SVM classifier.

5.4.3 Naïve Bayesian classifier

A naive Bayes classifier is a simple probabilistic classifier with strong assumptions of independence. Simply put,a naive bayes classifier assumes that the presence / absence of a particular property of a class is not related to the presence / absence of any other feature, considering the class variable as a function of the Class Probability Model, Trained in a supervised learning environment [28]. An advantage of the naive bayes classification is that it requires only a small amount of training data to estimate the parameters required for classification. The Bayesian classification assumes that the data belongs to a particular class. We then calculate the probability that the assumption is true. Bayesian classmates are basically statistical classifiers, that is, they can predict probabilities of class membership, such as the probability that a given test belongs to a particular class [29].

The naïve technique of Bayes is based on a Bayesian approach, it is therefore a simple, clear and fast classifier. Before reaching the main term of the Bayes theorem, we will first analyze certain terms used in the theorem. P (A) is the probability that event A occurs. P (A / B) is the probability that event A occurs because event B has already occurred or we can define it as the conditional probability of A as a function of the condition that B has already occurred. The

Bayes theorem is defined in Equation.

$$P(A|B) = P(B|A) * P(A)/P(B)$$

Naive Bayes is a machine learning algorithm used for classification tasks, which works by computing the probability of a given instance belonging to a certain class, given the observed feature values. It is based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event.

In supervised learning, the Naive Bayes algorithm is used to predict the class label of a new instance, given a set of features that describe the instance. The algorithm makes a strong assumption that all the features are independent of each other, which is often not true in real-world datasets. Despite this simplifying assumption, Naive Bayes has been shown to work well in many practical applications, including text classification, image recognition, and spam filtering [30].

5.4.4 AdaBoost Classifier

AdaBoost, or Adaptive Boosting, is a popular machine learning algorithm that combines multiple weak classifiers to create a strong classifier. The basic idea of AdaBoost is to iteratively train a series of weak classifiers on a dataset by selectively focusing on the misclassified examples in each iteration, while adjusting the weights of the examples based on their previous classification performance. In each iteration, the algorithm generates a new weak classifier that focuses on the most challenging examples, and combines it with the previous classifiers by assigning weights to each classifier based on its classification accuracy. The final output of the AdaBoost algorithm is a weighted combination of the outputs of all the weak classifiers. AdaBoost is useful for solving binary classification problems where the goal is to classify input data into one of two categories. AdaBoost can use a variety of weak classifiers, such as decision trees, artificial neural networks, or support vector machines, to capture different patterns in the data. AdaBoost has

been widely used in various applications, such as computer vision, speech recognition, and natural language processing, due to its ability to handle complex and high-dimensional datasets.

AdaBoost is also called as Adaptive boosting is a strategy in Machine learning utilized as a Gathering Technique. The most widely recognized calculation utilized with AdaBoost is decision trees with one level that implies with decision trees with just 1 split. These trees are also known as Decision stumps.

5.4.5 K Neighbors Classifier

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

The parameter k in kNN refers to the number of labeled points (neighbors) considered for classification. The value of k indicates the number of these points used to determine the result. Our task is to calculate the distance and identify which categories are closest to our unknown entity. The main concept behind k-nearest neighbors is as follows. Given a point whose class we do not know, we can try to understand which points in our feature space are closest to it [31]. These points are the k-nearest neighbors. Since similar things occupy similar places in feature space, it’s very likely that the point belongs to the same class as its neighbors. Based on that, it’s possible to classify a new point as belonging to one class or another.

5.5 Evaluation

Different machine learning algorithms we are used for spam email classification, and the evaluate accuracy and precision of algorithm which is differ depending on the algorithm used.

For example, if a Naive Bayes classifier is used, evaluation metrics such as accuracy, precision, can be used. Naive Bayes is a probabilistic algorithm that assumes independence among the features, and it is a popular algorithm for text classification tasks such as spam email classification.

If a Support Vector Machine (SVM) classifier is used, evaluation metrics such as accuracy, precision, can be used. SVM is a binary classification algorithm that works by separating the data points into two classes using a hyperplane. It is known for its ability to handle high-dimensional data and is commonly used for text classification tasks.

If a decision tree classifier is used, evaluation metrics such as accuracy, precision, recall, and F1-score can be used. Decision trees are a popular algorithm for classification tasks because they are easy to interpret and visualize. However, they can suffer from overfitting if not properly tuned.

In general, the choice of evaluation metrics for spam email classification depends on the algorithm used and the problem at hand. It is important to choose the appropriate metrics that capture the relevant aspects of the problem and the algorithm's performance. Additionally, using multiple evaluation metrics can provide a more comprehensive understanding of the classifier's performance.

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.974855	0.982759
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
10	xgb	0.971954	0.943089
6	AdaBoost	0.960348	0.929204
9	GBDT	0.947776	0.920000
7	BgC	0.957447	0.867188
3	DT	0.929400	0.828283

Fig. 5.9: Accuracy and Precision

5.6 Improvement

Improvement for a spam email classifier using TF-IDF can be achieved using several techniques including scaling, voting, and stacking.

5.6.1 TF-IDF

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. TF-IDF (Term Frequency - Inverse Document Frequency) is a commonly used technique in machine learning for text classification tasks. It is used to assign weights to the words in a document based on their importance in the document and their frequency across all documents in the dataset.

The term frequency (TF) component of TF-IDF measures the frequency of a word in a document. This is calculated by dividing the number of times the word appears in the document by the total number of words in the document [32]. The idea is that

words that appear more frequently in a document are more important to the document's meaning.

The inverse document frequency (IDF) component of TF-IDF measures the rarity of a word in the dataset. This is calculated by taking the logarithm of the total number of documents in the dataset divided by the number of documents that contain the word. The idea is that words that appear in fewer documents are more important to the meaning of a particular document.

5.6.2 Scaling

Scaling in machine learning refers to the process of transforming the input features so that they are on the same scale. This is done to ensure that all input features are given equal importance during the model training process, which can improve the performance of the model.

In many machine learning algorithms, input features with larger numerical ranges can have a disproportionate impact on the output of the model compared to features with smaller numerical ranges. Scaling helps to overcome this issue by transforming the input features to a common scale.

5.6.3 Voting

Voting Classifier is a machine-learning algorithm often used by Kagglers to boost the performance of their model and climb up the rank ladder. Voting Classifier can also be used for real-world datasets to improve performance, but it comes with some limitations. The model interpretability decreases, as one cannot interpret the model using shap, or lime packages. Voting is an ensemble method where multiple classifiers are trained and their outputs are combined to make a final decision. This can be done using techniques such as majority voting or weighted voting. Majority voting simply takes the most commonly predicted class as the final output, while

weighted voting takes into account the confidence of each classifier's prediction. This helps to improve the accuracy and robustness of the classifier by reducing the impact of individual classifier's errors.

5.6.4 Stacking

Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance. Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance. Ensemble learning is a subset of machine learning. It is used to optimize the performance of a model by integrating the outputs of multiple models. Ensemble learning also improves the accuracy of the model [33]. Stacking in machine learning is an ensemble algorithm used for prediction models where we can get efficient outputs.

5.7 Website

To create a local website for a spam email classifier using Streamlit, we start by building the classifier model using a machine learning algorithm such as Naive Bayes or Support Vector Machines (SVM). Once the model is built, we integrate it into a Streamlit application by creating a Python script that accepts email input from the user, processes it using the classifier model, and returns the predicted spam or not-spam label. Streamlit is an open-source framework for building interactive data science web applications. It allows developers to easily create and deploy web applications that can be accessed by users through a web browser.

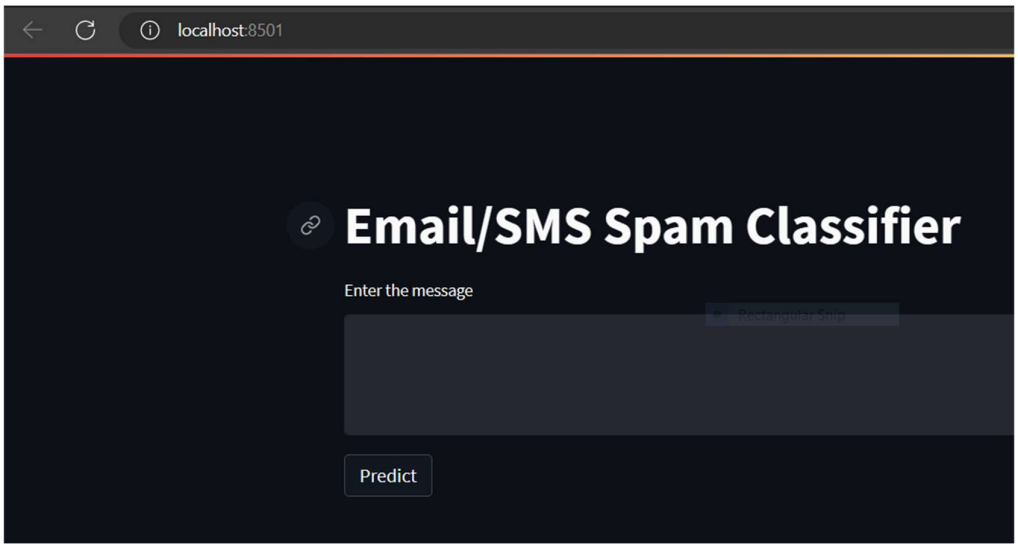


Fig. 5.10: local host website

5.8 Deploy

Deployment is a critical process in software development that involves making an application available for use by end-users. This process typically involves moving the application from a development environment to a production environment where it can be accessed by users. Deploying an application can be a complex process that involves several steps, such as building the application, configuring the production environment, deploying the application to the server, and testing and verifying that the application is working as intended in the production environment. It is important to have a robust deployment process in place to ensure that the application is stable, secure, and meets the requirements of the end-users [34]. Deploying an application can involve multiple technologies and tools, and the process can vary depending on the application's complexity and the infrastructure being used.

In our project we used Render for deploy our project, Render is a cloud platform that simplifies web application deployment. It supports various languages and frameworks and offers scalable infrastructure. Deploying a website on Render involves connecting the code to a Git repository, configuring the deployment

settings, and deploying the site to the Render platform. Render offers additional features like automatic scaling and monitoring to ensure the website runs smoothly.

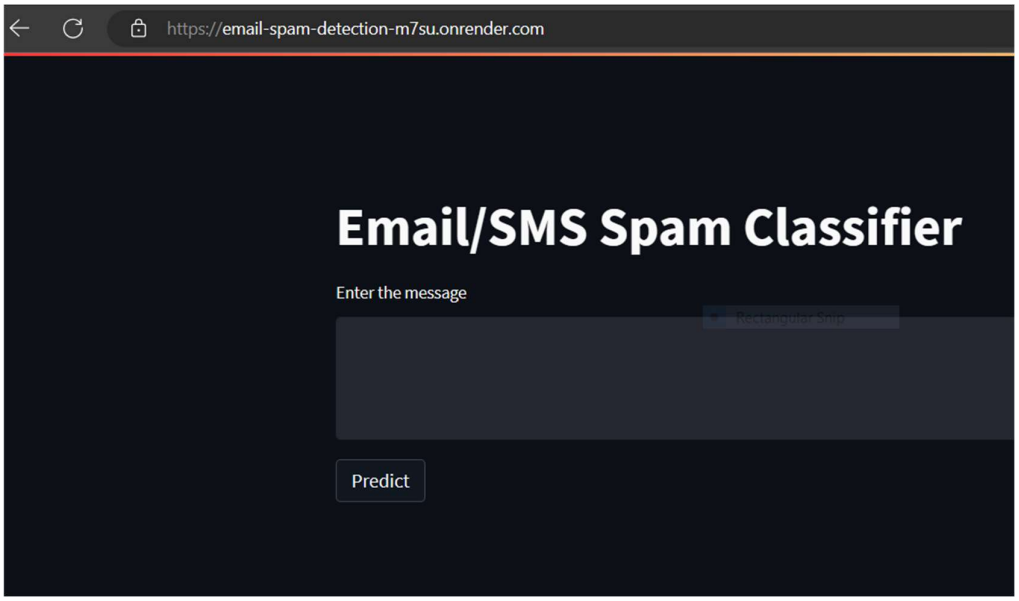


Fig. 5.11: Deployment Website

CHAPTER 6

TESTING

Testing is a crucial part of any email spam classifier project. The purpose of testing is to ensure that the classifier can accurately identify spam emails and separate them from legitimate ones. To achieve this goal, the testing process typically involves several steps.

First, a representative sample of emails is collected, both spam and non-spam, to serve as the basis for the testing data. This dataset is then used to train the classifier and evaluate its performance. During testing, the classifier's accuracy, precision, recall, and F1-score are calculated to determine how well it can classify emails. Additionally, scalability testing may be performed to ensure that the classifier can handle large volumes of incoming emails without experiencing performance issues. User experience testing may also be conducted to ensure that the classifier is user-friendly and effective at reducing clutter in users' inboxes.

Finally, maintenance testing may be performed periodically to ensure that the classifier remains effective and up-to-date with the latest spam email tactics and techniques. Overall, the testing process is essential for ensuring that an email spam classifier is accurate, scalable, user-friendly, and maintainable over time.

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability[35]. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

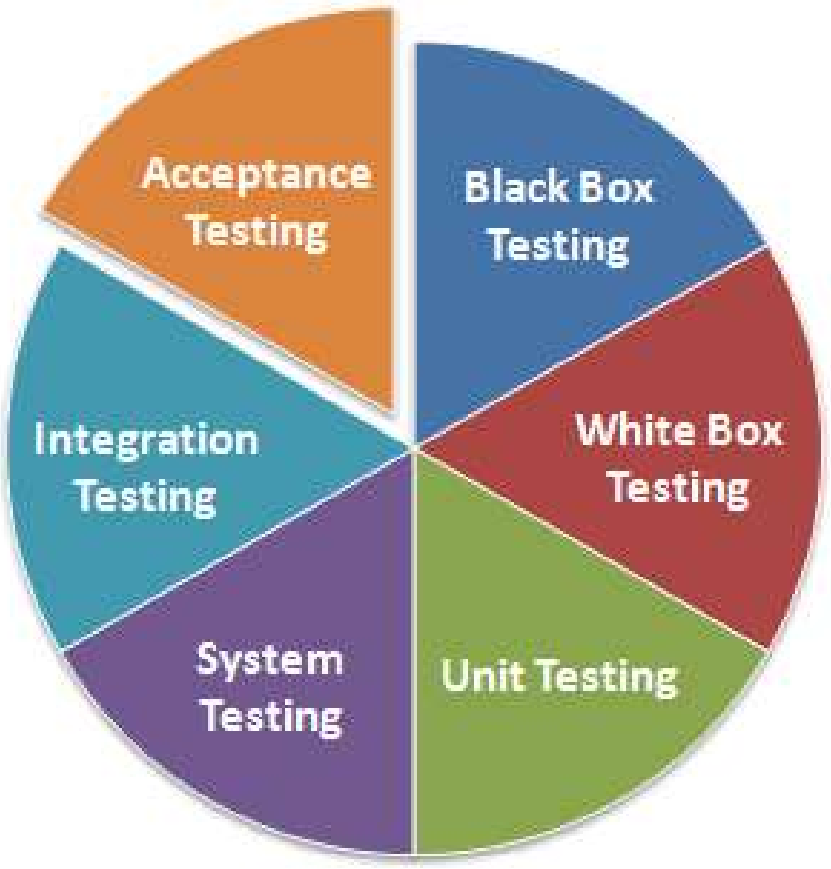


Fig. 6.1: Testing Techniques

6.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually scrutinized for proper operation. Software developers and sometimes QA staff complete unit tests during the development process. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself [36]. It is correlated with the functional correctness of the independent modules.

Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure.

Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

Unit testing is a software testing approach that involves testing individual units or components of a system to ensure they are functioning as intended. In the case of a spam email classifier, unit testing would involve testing each component of the system separately to ensure that they are correctly identifying and classifying spam emails.

This would typically involve creating a set of test cases that cover different scenarios, such as emails with different types of spam content or different email structures. The tests would be designed to verify that the classifier is correctly identifying spam emails and rejecting legitimate emails. By conducting unit tests on each component of the spam email classifier, developers can identify and address any issues early in the development process, ensuring that the final product is accurate and effective in identifying and filtering out spam emails.

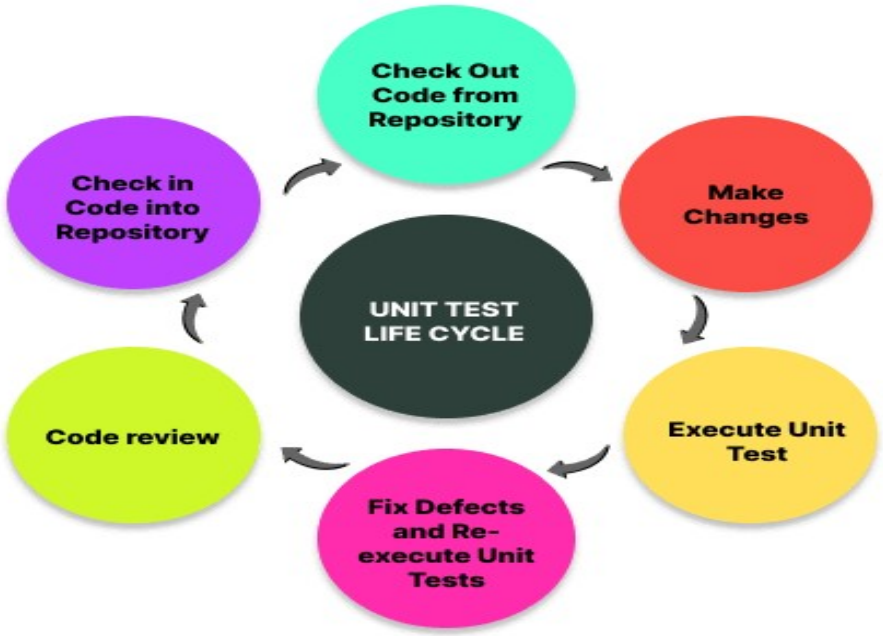


Fig. 6.2: Unit Testing

6.2 Integration Testing

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

In the context of a spam email classifier, integration testing involves testing that the classifier can correctly identify whether a given email is spam or not. This involves feeding a set of test emails with known labels, both spam and non-spam, into the classifier and verifying that it correctly classifies them.

Integration testing also verifies that the text preprocessing step correctly removes irrelevant information from the email text, such as stop words and punctuation, and that the feature extraction step correctly extracts relevant features from the preprocessed email text, such as word frequencies and n-grams.

Overall, integration testing in a spam email classifier is a critical step in ensuring that the classifier functions correctly and produces accurate results in real-world scenarios. This testing can help identify and resolve any issues or bugs in the classifier before it is deployed in production.

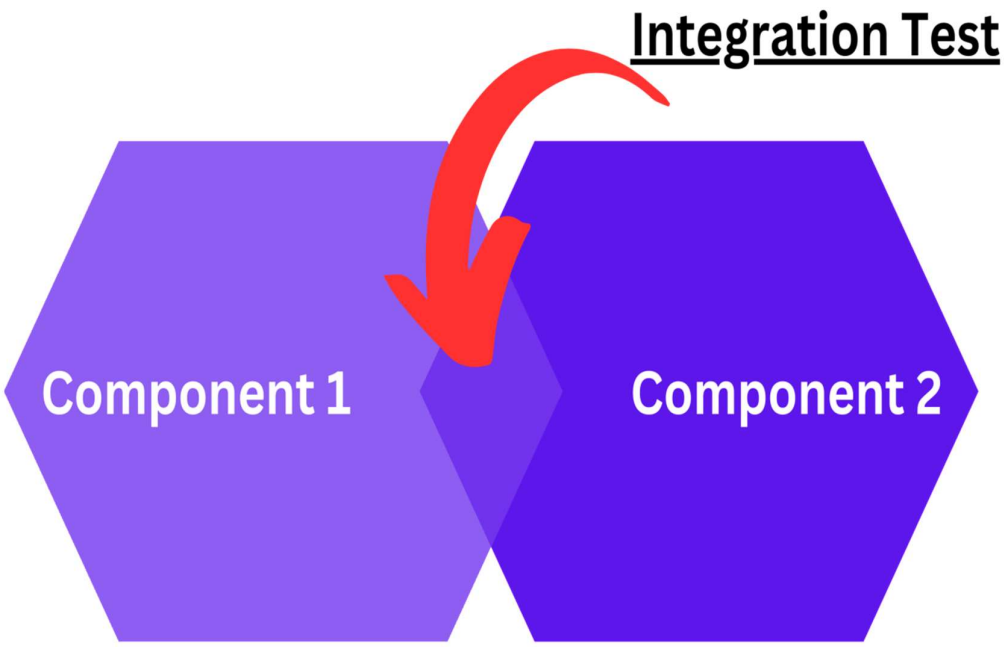


Fig. 6.3: Integration testing

6.3 White box testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing.

It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

In white box testing, the tester has access to the source code and the application's internal architecture. This enables the tester to design test cases that ensure that all lines of code, branches, and paths are executed as expected. White box testing also helps to identify defects in the code's logic, security vulnerabilities, and performance issues.

White box testing is typically performed by software developers themselves or by a dedicated testing team. It can be automated or performed manually using various testing tools and techniques, such as unit testing, integration testing, and system testing.

When it comes to testing a spam email classifier system, white box testing can be an effective approach to ensure the system's reliability and security. This type of testing requires a thorough understanding of the code and internal structures of the classifier system. Test cases are designed based on this understanding to verify the correctness, completeness, and robustness of the code.

The testing process can include unit testing, integration testing, and system testing, with a focus on testing the interaction between different components of the classifier system [37]. In addition, security vulnerabilities should be checked for, as the spam email classifier system needs to be secure and immune to attempts at

bypassing it or exploiting weaknesses in the code. By following these steps, white box testing can help ensure that the spam email classifier system is functioning correctly, is secure, and meets the users' requirements.

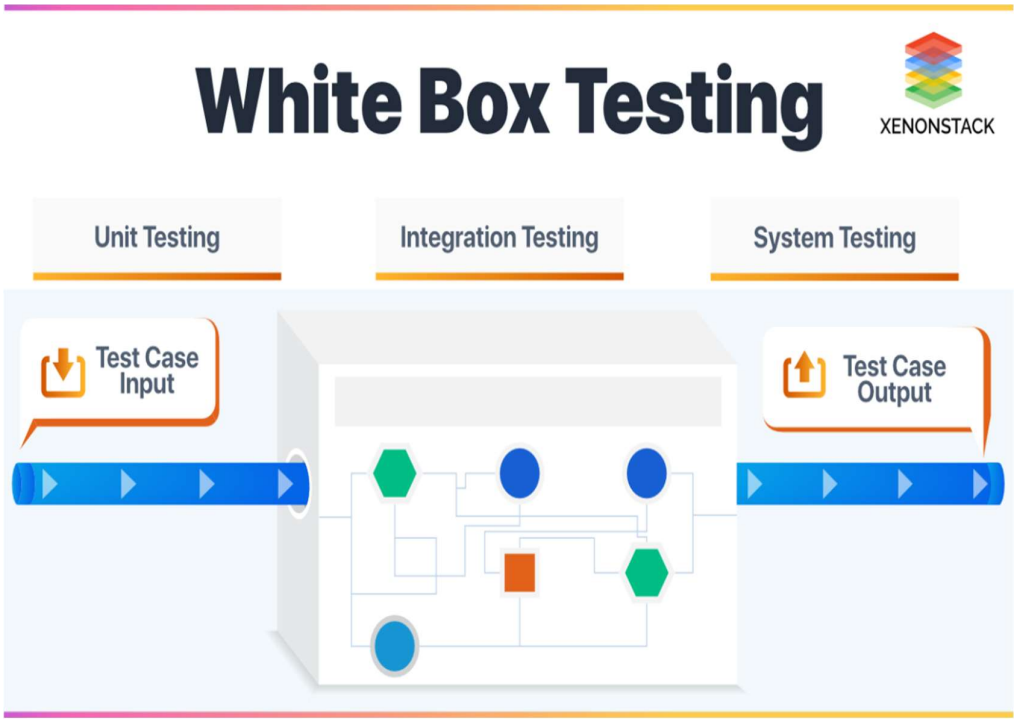


Fig. 6.4: White Box Testing

6.4 Black Box Testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly

focuses on input and output of software applications and it is entirely based on software requirements and specifications [38]. It is also known as Behavioral Testing.

In black box testing of a spam email classifier, the tester would typically supply a set of sample emails, some of which are known to be spam and some of which are known to be legitimate. The tester would then observe the classifier's output for each email and compare it to the known correct classification.

Black box testing can be a useful way to evaluate the performance of a spam email classifier because it focuses on the classifier's behavior as a whole, rather than on the specific details of how it works. However, it may not reveal any specific weaknesses or problems with the classifier's algorithms or rules.

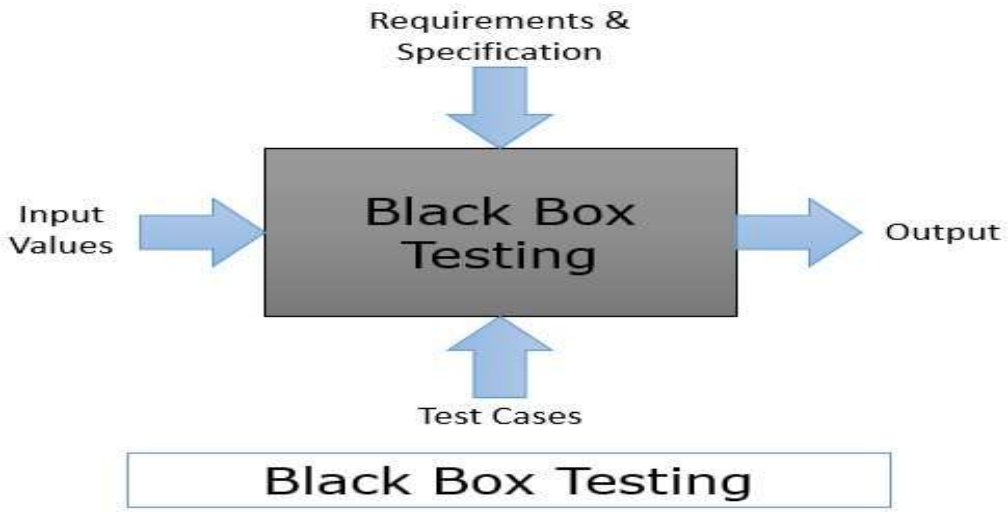


Fig. 6.5: Black Box Testing

6.5 System Testing

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the

end-users. This type of testing is performed after the integration testing and before the acceptance testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together.

System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer [39]. It is performed to test the system beyond the bounds mentioned in the software requirements specification.

System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing.

System testing for a spam email classifier involves evaluating the performance of the system in accurately identifying spam emails from legitimate ones. The testing process typically involves feeding a large set of emails, some of which are known to be spam, to the system and assessing its ability to correctly classify them.

System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user story testing and then each component through integration testing.

If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the

software. An app development team logs all defects and establishes what kinds and numbers of defects are tolerable.

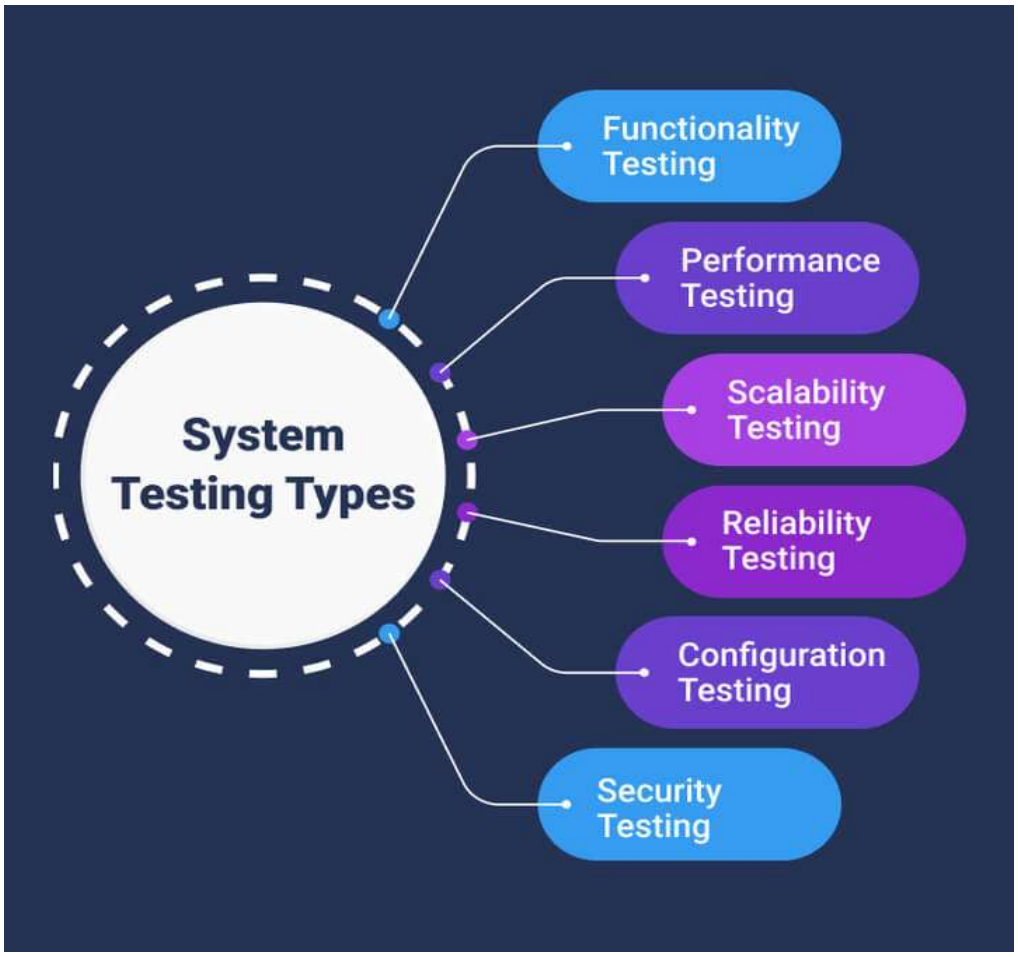


Fig. 6.6: System Testing

6.6 Testing Performed

In our project, we have implemented white box testing in which tester have the complete knowledge of the application tested and the source code of it.

In this implementation, we identified the individual code cells or functions within the notebook that we want to be tested, considering breaking down the complex logic into smaller and testable units for easier testing and maintenance.

We have implemented different test cases and debug the code and fixed the issues or error occurred in the code, also we use print statements to check for the expected output of the code.

Following are some errors and test-cases occur while testing the codes:

Case 1:

The code given below is the definition of the transform function.

```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)


    text = y[:]


    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

Fig. 6.7: Transform_Text() Error

The transform_text() function is being called with a sequence of words as a parameter passed inside it as given below.

```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.")
```

```
'i gon na be home soon and i do want to talk about thi stuff anymor tonight k i cri enough today gon na home soon want talk stu
ff anymor tonight k cri enough today'
```

Fig. 6.8: Transform_Text() Actual Output

This transform_text() function failed to meet the desired output.

After debugging the above code we find that there is a bug that we didn't pop elements from 'y' before appending other information into it. So we regenerate the code as given below.

```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)



    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

Fig. 6.9: Transform_Text() Error Resolved

After this we got the expected output from the transform text function.

```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.")
```

```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

Fig. 6.10: Transform_Text() Expected Output

Case 2:

Here we trigger the issue that dataframe object has no attribute 'tolist'. We had checked the length and got 0 as output.

```
: spam_corpus = []
  for msg in df[df['target'] == 1].tolist():
      for word in msg.split():
          spam_corpus.append(word)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_26976\459943197.py in <module>
      1 spam_corpus = []
----> 2 for msg in df[df['target'] == 1].tolist():
      3     for word in msg.split():
      4         spam_corpus.append(word)
      5

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5573         ):
   5574             return self[name]
-> 5575         return object.__getattribute__(self, name)
   5576
   5577     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'tolist'
```

```
: len(spam_corpus)
: 0
```

Fig. 6.11: Dataframe Error

The code snippet has a missing dataframe value transform text as there is no any way to apply tolist() method onto the 'target' dataframe.

After resolving, we had checked the length of spam corpus array and we got the expected output.

```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

```
len(spam_corpus)
```

```
9939
```

Fig. 6.12: Dataframe Error Resolved

Case 3:

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'])
```

```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```

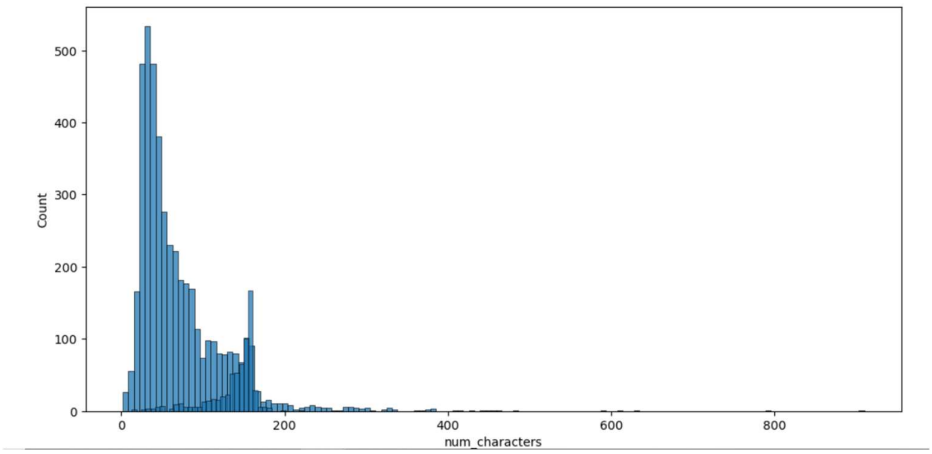


Fig. 6.13: Bad Visualization

The code used here has no any error but it seems blue color coincides two different clusters(spam and ham). For better visualization a color different from blue is assigned to another cluster.

Here red color is used to show the two different clusters(spam and ham) of dataframe.

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
```

```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```

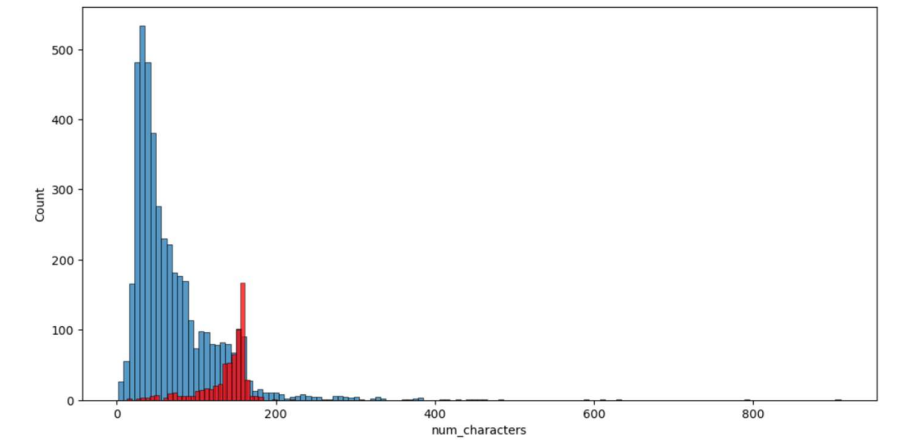


Fig. 6.14: Better Visualization

Case 4:

Here an error is encountered as the code has a missing method to extend the length of the ‘text’ dataframe to the dataframe ‘numwords’.

```
df['num_words']=df['text'].len(nltk.word_tokenize(x)
```

```
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_8892\1807989544.py", line 1
    df['num_words']=df['text'].len(nltk.word_tokenize(x)
                                                      ^
SyntaxError: unexpected EOF while parsing
```

Fig. 6.15: Apply method Error

After triggering the error and regenerating the code using .apply() method, the code snippet gives the expected output and shows no deviation as given below.

```
df['num_words']=df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

```
df.head()
```

	target	text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13

Fig. 6.16: Apply method Error Resolve

CHAPTER 7

OUTPUT SCREEN

7.1 Home Page Output

Firstly, open your web browser and visit the website of this project that offer online services to analyze and detect spam messages. Type the message, In most cases, you'll need to copy the content of the suspicious email or message that you want to analyze. Select the text of the message, right-click, and choose "Copy" (or use the appropriate keyboard shortcut). Then, go to the spam detection website and find the designated text field or box to paste the message content.

Submit the message for analysis, Paste the copied message into the provided text field on the spam detection website. Once you've done that, submit the message for analysis by clicking the appropriate button or initiating the process.

Analyze the results, the spam detection website will process the submitted message and provide a result. It may identify the message as spam or ham.

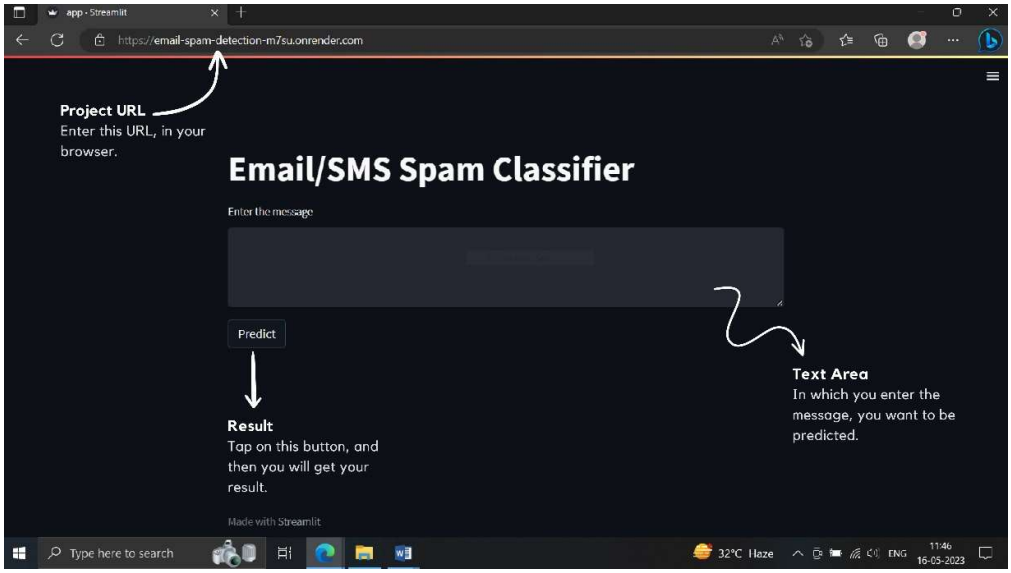


Fig. 7.1: Home Page Output

7.2 Spam Output

Spam email refers to unsolicited email messages, which are unwanted junk emails sent indiscriminately in bulk to a large group of recipients. Typically, spam emails or messages are sent for commercial purposes in massive volume by a botnet, which is a network of infected computers. Spam can be distributed by email, text messages or social media.

Spam is often related to malware and phishing scams. Phishing scams are fraudulent. They look like a legitimate service and are designed to imitate the services people mostly use in daily life. So people do not suspect about the true nature and origin of the e-mail or websites it is related to. Many people fall victims to such scams and enter sensitive personal details like passwords and online banking details or credit card information.

The spam could be linked to malware sites or spywares. Sometimes they mislead people to download some attachments or files that can also serve above mentioned nefarious purposes. They can also cause damage to your device and data and install ransomware which takes your device and data as hostage until you pay the ransom to mentioned accounts through various cryptocurrencies.

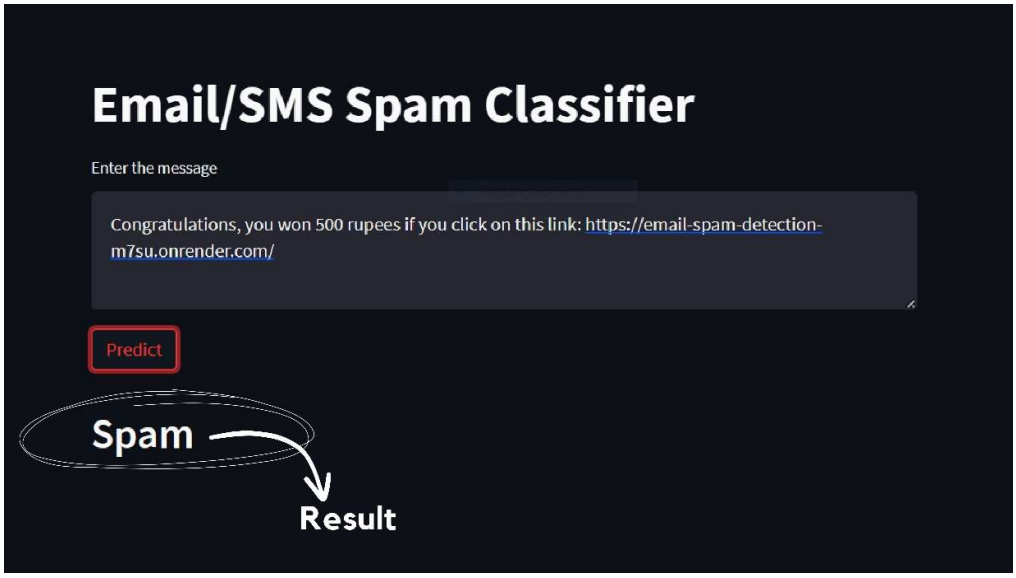


Fig. 7.2: Spam Output

7.3 Not Spam/ Ham Output

"Ham" message or a "not spam" message refers to a legitimate and desired message that is not classified as spam. It is a message that typically originates from a trusted source and contains relevant and meaningful content.

Ham messages can include various types of communication, such as personal emails, professional correspondence, newsletters or updates from subscribed services, notifications, and other non-spam messages. These messages are typically sent with the consent of the recipient or are expected as part of an ongoing relationship or communication.

Unlike spam messages, ham messages are relevant to the recipient's interests, needs, or previous interactions. They are not intended to deceive, promote fraudulent activities, or inundate recipients with unwanted or unsolicited content.

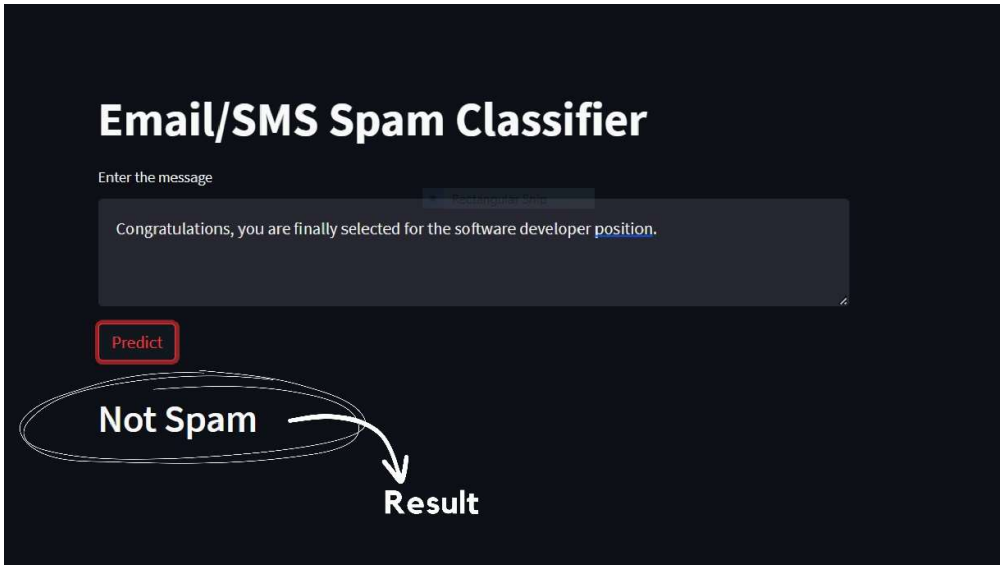


Fig. 7.3: Not Spam/ Ham Output

CHAPTER 8

CONCLUSION

Detection of spam is important for securing message and e-mail communication. The accurate detection of spam is a big issue, and many detection methods have been proposed by various researchers. However, these methods have a lack of capability to detect the spam accurately and efficiently. To solve this issue, we have proposed a method for spam detection using machine learning predictive models.

The method is applied for the purpose of detection of spam. The experimental results obtained show that the proposed method has a high capability to detect spam. The proposed method achieved 99% accuracy which is high as compared with the other existing methods. Thus, the results suggest that the proposed method is more reliable for accurate and on-time detection of spam, and it will secure the communication systems of messages and e-mails.

In this project we have proposed procedure to identify an email as spam or ham based on text categorization. Different methods for pre-processing of email organize are connected, for example, applying stop words expelling, stemming, include decrease and highlight choice strategies to bring the catchphrases from every one of the qualities lastly utilizing distinctive classifiers to isolate mail as spam or ham.

CHAPTER 9

FUTURE SCOPE

There are several potential areas for future research and improvement in the field of spam email classification using machine learning. Some of these areas include:

1) Deep learning approaches: Applying deep learning techniques, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), to capture complex patterns and relationships in the email data for more accurate classification.

2) Adapting to evolving spam techniques: Continuously updating and refining the spam classification models to adapt to new and evolving spamming techniques, such as the use of mystification, image-based spam, or social engineering tactics.

3) Incorporating user feedback: Designing feedback mechanisms that allow users to provide input on misclassified emails, which can be used to improve the classification models and further enhance their accuracy.

By addressing these areas of research, it is possible to develop more effective and robust spam email classification systems that can better protect users from unwanted and malicious emails.

Overall, the use of machine learning techniques for spam email classification shows promise in reducing the spam problem and improving the email experience for users. Further advancements and research in this field can contribute to the development of more revolutionary and accurate spam filtering systems.

REFERENCES

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

[2] H. Faris, A. M. Al-Zoubi, A. A. Heidari et al., "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," Information Fusion, vol. 48, pp. 67–83, 2019.

[3] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," Artificial Intelligence Review, vol. 29, no. 1, pp. 63–92, 2008.

[4] M. Bassiouni, M. Ali, and E. A. El-Dahshan, "Ham and spam e-mails classification using machine learning techniques," Journal of Applied Security Research, vol. 13, no. 3, pp. 315–331, 2018.

[5] Rathi, M. and Pareek, V. "Spam Mail Detection through Data Mining A Comparative Performance Analysis", I.J. Modern Education and Computer Science, 2013, 12, 31-39.

[6] Tretyakov, K. Machine learning techniques in spam filtering: Data Mining Problem-oriented Seminar, MTAT.03.177, May 2004.

[7] Kishore, R. K., Poonkuzhali, G. and Sudhakar, P. "Comparative Study on Email Spam Classifier using Data Mining Techniques", Proceedings of the International MultiConference of Engineers and Computers Science Scientists, 2012.

[8] G. Jain, M. Sharma, and B. Agarwal, "Optimizing semantic lstm for spam detection," International Journal of Information Technology, vol. 11, no. 2, pp. 239–250, 2019.

[9] Sunil B. Rathod, Tareek M. Pattewar "Content Based Spam Detection in Email using Bayesian Classifier", presented at the IEEE ICCSP 2015 conference.

[10] K. M. Mendez, L. Pritchard, S. N. Reinke, and D. I. Broadhurst, "Toward collaborative open data science in metabolomics using jupyter notebooks and cloud computing," Metabolomics, vol. 15, no. 10, Sep. 2019.

[11] van der Walt, S., Colbert, S. C. & Varoquaux, G. The NumPy array: a structure for efficient numerical computation. Comput. Sci. Eng. 13, 22–30 (2011).

[12] M. Harrison & T. Petrou, Pandas 1.x cookbook: Practical recipes for scientific computing, time series analysis, and exploratory data analysis using Python (Second edition) (2020).

[13] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[14] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., …others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[15] Jackson, M. Problems and requirements [software development], Proceedings of the second international symposium on requirement engineering, pp.2, (1995).

[16] McGrath, J.R., & MacEwan, G. (2011). Linking pedagogical practices of activity-based teaching. The International Journal of Interdisciplinary Social Sciences, 6(3), 261-274.

[17] Oliveira, A., Seco N. and Gomes P. 2006. A CBR Approach to Text to Class Diagram Translation. TCBR Workshop at the 8th European Conference on CaseBased Reasoning, Turkey, September 2006.

[18] Teorey, T.J., Yang, D., and Fry, J.P., (1986), "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," Computing Sur\,eys, 18: 12, June, pp. 197-222.

[19] Leavens, G.T., Wahls, T. and Bakar, A.L. (1999). Formal Semantics for SA Style Data Flow Diagram Specification Languages. Proceedings of the 1999 ACM Symposium on Applied Computing. Oregon, US: IEEE Computer Society. pp. 526–532.

[20] M.A. Hernandez and S.J. Stolfo,:Real-World Data Is Dirty: Data Cleansing and the Merge/Purge Problem. Data Miningand Knowledge Discovery, Vol. 2, pp. 9-37, 1998.

[21] Kaski, Samuel (1997) "Data exploration using self-organizing maps."Acta polytechnicascandinavica: Mathematics, computing and management in engineering series no. 82. 1997.

[22] Srividhya V, Anitha R. Evaluating preprocessing techniques in text categorization. International Journal of Com-puter Science and Application Issue 2010.

[23] Ghag KV, Shah K. Comparative analysis of effect of stopwords removal on sentiment classifcation. In: IEEEInternational Conference on Computer, Communication and Control; Indore, India; 2015.

[24] Lee, B.; Park, J.; Kwon, L.; Moon, Y.; Shin, Y.; Kim, G.; Kim, H. About relationship between business text patterns and financial performance in corporate data. J. Open Innov. Technol. Mark. Complex. 2018.

[25] Austin, J. T.,Yaffee, R. A., & Hinkle, D. E. (1992). Logistic regression forresearch in higher education. Higher Education: Handbook of Theoryand Research, 8, 379–410.

[26] Feng, W., Sun, J., Zhang, L., Cao, C. and Yang,Q., "A support vector machine based naive Bayes algorithm for spam filtering," 2016 IEEE 35thInternational Performance Computing and Communications Conference (IPCCC), Las Vegas, NV, 2016, pp. 1-8.

[27] S. O. Olatunji, "Extreme Learning machines and Support Vector Machines models for email spam detection," in Proceedings of the 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, Windsor, Canada, April 2017.

[28] Tariq, M., B., Jameel A. Tariq, Q., Jan, R. Nisar, A. S., "Detecting Threat E-mails using Bayesian Approach", IJSDIA International Journal of Secure Digital Information Age, Vol. 1. No. 2, December 2009.

[29] Patil, T. and Sherekar, S. "Performance Analysis of Na¨ıve Bayes and Classification Algorithm for Data Classification", International Journal Of Computer Science And Applications, 2013.

[30] J. K. Kruschke and T. M. Liddell, "Bayesian data analysis for newcomers," Psychonomic Bulletin & Review, vol. 25, no. 1, pp. 155–177, 2018.

[31] M. Kubat, M. Jr.: Voting Nearest-Neighbour Subclassifiers. Proceedings of the17th International Conference on Machine Learning, ICML-2000, pp.503-510,Stanford, CA, June 29-July 2, (2000).

[32] Fan, H., and Qin, Y. (2018). "Research on Text Classification Based on Improved TF-IDF Algorithm," International Conference on Network, Communication, Computer Engineering (NCCE 2018), vol. 147.

[33] Sikora R. and Al-laymoun O.H., "A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms", Journal of International Technology and Information Management. vol.23, No.1, 2014.

[34] Jay Lee, Behrad Badheri, Hung-An Kao. 2015. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manufacturing Letters, 3, pp 18-23,doi:10.1016/j.mfglet.2014.12.001.

[35] Shivkumar Hasmukhrai Trivedi, "Software Testing Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012, ISSN: 2277 128X.

[36] E. Daka and G. Fraser, "A survey on unit testing practices andproblems," in 2014 IEEE 25th International Symposium on Software Reliability Engineering. IEEE, 2014.

[37] Khan, M. E., 2011. "Different Approaches to White Box Testing Technique for Finding Errors". International Journal of Software Engineering and Its Applications, 5(3).

[38] Harsh Bhasin, E. K., 2014. "Black Box Testing based on Requirement Analysis and Design Specifications". International Journal of Computer Applications, 87(18), pp. 0975-8887.

[39] C. Michael, "Generating software test data by evolution, Software Engineering", IEEE Transaction, Volume: 27, 200.

APPENDIX

A. Jupyter Notebook Code (To build Model)

```
import numpy as np
import pandas as pd
df=pd.read_csv("spam.csv")
df.sample(5)
df.shape
## 1.Data Cleaning
df.info()
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed:
4'],inplace=True)
df.sample(5)
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['target'] = encoder.fit_transform(df['target'])
df.head()
df.isnull().sum()
df.duplicated().sum()
df = df.drop_duplicates(keep='first')
df.duplicated().sum()
df.shape
## 2.EDA
df.head()
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(),
labels=['ham','spam'],autopct="%0.2f")
plt.show()
import nltk
nltk.download('punkt')
df['num_characters'] = df['text'].apply(len)
df.head()
df['num_words'] = df['text'].apply(lambda
x:len(nltk.word_tokenize(x)))
df.head()
df['num_sentences'] = df['text'].apply(lambda
x:len(nltk.sent_tokenize(x)))
df.head()
df[['num_characters','num_words','num_sentences']].describe()
#ham
```

```
df[df['target'] == 0]
[['num_characters','num_words','num_sentences']].describe()
#spam
df[df['target'] == 1]
[['num_characters','num_words','num_sentences']].describe()
import seaborn as sns
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] ==
1]['num_characters'],color='red')
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
sns.pairplot(df,hue='target')
sns.heatmap(df.corr(),annot=True)
## 3.Data Preprocessing
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in
string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
from nltk.corpus import stopwords
stopwords.words('english')
import string
string.punctuation
transform_text("I'm gonna be home soon and i don't want to talk
about this stuff anymore tonight, k? I've cried enough today.")
df['text'][10]
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
df['transformed_text'] = df['text'].apply(transform_text)
df.head()
## Word Cloud
```

```
pip install wordcloud
from wordcloud import WordCloud
wc =
WordCloud(width=500,height=500,min_font_size=10,background_color
='white')
spam_wc = wc.generate(df[df['target'] ==
1]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
ham_wc = wc.generate(df[df['target'] ==
0]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
df.head()
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
len(spam_corpus)
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0
],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
len(ham_corpus)
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0]
,pd.DataFrame(Counter(ham_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
# Text Vectorization
# using Bag of Words
df.head()
## 4.Model Building
from sklearn.feature_extraction.text import
CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
X = scaler.fit_transform(X)
# appending the num_character col to X
X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
X.shape
y = df['target'].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.naive_bayes import
GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import
accuracy_score,confusion_matrix,precision_score
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
```

```
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt =
GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)
    return accuracy,precision
train_classifier(svc,X_train,y_train,X_test,y_test)
accuracy_scores = []
precision_scores = []
for name,clf in clfs.items():
    current_accuracy,current_precision = train_classifier(clf,
X_train,y_train,X_test,y_test)
    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
performance_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores
,'Precision':precision_scores}).sort_values('Precision',ascendin
g=False)
performance_df
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1,
kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
temp_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':acc
uracy_scores,'Precision_max_ft_3000':precision_scores}).sort_val
ues('Precision_max_ft_3000',ascending=False)
temp_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accurac
y_scores,'Precision_scaling':precision_scores}).sort_values('Pre
cision_scaling',ascending=False)
new_df = performance_df.merge(temp_df,on='Algorithm')
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
temp_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accur
acy_scores,'Precision_num_chars':precision_scores}).sort_values(
'Precision_num_chars',ascending=False)
new_df_scaled.merge(temp_df,on='Algorithm')
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
from sklearn.ensemble import VotingClassifier
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb),
('et', etc)],voting='soft')
voting.fit(X_train,y_train)
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
from sklearn.ensemble import StackingClassifier
clf = StackingClassifier(estimators=estimators,
final_estimator=final_estimator)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

B. Pycharm Code (To Build App)

```
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer
import nltk
nltk.download('stopwords')
import sklearn
import nltk
nltk.download('punkt')
def tokenize(token):
    return nltk.word_tokenize(token);
tokenize("why is this not working?");
ps = PorterStemmer()
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in
string.punctuation:

    y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
tfidf = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))
st.title("Email/SMS Spam Classifier")
input_sms = st.text_area("Enter the message")
if st.button('Predict'):
```

```
# 1. preprocess
transformed_sms = transform_text(input_sms)
# 2. vectorize
vector_input = tfidf.transform([transformed_sms])
# 3. predict
result = model.predict(vector_input)[0]
# 4. Display
if result == 1:
    st.header("Spam")
else:
    st.header("Not Spam")
```

Email/SMS Spam Classifier on supervised learning

Satendra Kumar, Shubham Sharma, Shubham, Shivani, Harshita Nailwal, Tabish Absar
Department of Computer Science and Engineering,
Moradabad Institute of Technology
Moradabad, Uttar Pradesh, INDIA

satendra04cs41@gmail.com,shubham4in@gmail.com,shubhamsingh.rsd@gmail.com,shivanisinghdilari@gmail.com, harshitanailwal28@gmail.com,tabishabsar52@gmail.com

Abstract: - Email, which is widely accessible, typically quick to send messages, and cost-free, is one of the most prominent and frequently utilised communication strategies. Email-based hazards have increased as a direct result of the weaknesses in email procedures and growing percentage of automated commercial and commercial trades. One of the challenging problems with the modern Internet is email spam, which annoys individual customers and causes financial havoc for businesses. Without the customers' permission, spam communications target them and clog their mailboxes. When checking and removing spam emails, they use up more time and organisation resources. Despite the fact that a large majority of Web users publicly dislike spam, sufficient of them nevertheless answer to profitable deals for spam to remain a real issue. Although the majority of Web users are clear about their hatred of spam, the fact that enough of them still click on commercial offers means that spammers may still make money from it. While most customers are aware of what they should be doing, they need clear instructions on how to avoid and delete spam. Whatever steps are made to eradicate spam, they are in no way successful. Filtering is the most straightforward and practical technique among the strategies developed to stop spam. The more recent classifier-related challenges have been the focus of many studies in spam separation. Machine learning for spam detection is a crucial area of research in modern times. Nowadays, spam detection using machine learning is a crucial area for inquiry. The appropriateness of the suggested effort is examined, and it recognises the application of several learning estimations for extracting spam mails from email. Similar to this, a close review of the estimates has been provided.

Keywords: – *Machine Learning, MLP, NaiveBayesian, Spam Classification,.*

1 INTRODUCTION

The usage of the internet has been steadily growing ended the previous ten years and is still rising. We might therefore conclude that the Internet is progressively suitable a necessary component of daily life. Email has evolved into a helpful tool for data transfer and web use is expected to continue growing. A rare of the many benefits that email likes over other physical systems are a slight interval delay through conduction, safety of the information being transported, and inexpensive costs. However, there aren't many problems that prevent messages from being used effectively. One of them is spam email [1]. Unsolicited Bulk Email (UBE), sometimes known as spam email, has recently become a major problem online. Due to how inexpensive it is to send spam email, it is recklessly sent to a big amount of recipients. Additionally, it is crucial to takings some period to separate spam from non-spam email when a big volume of spam is expected because the mail server may fall down if the latter is not done. Several attempts have been made to recognise in order to address the spam problem. Numerous machine learning techniques have been used in previous research to address the problem, including Naive Bayes, Support Vector Machine (SVM), and many more Bayesian classifiers [2]. With the aim of being widely used to a few separating programming's, Bayesian classifiers in these approaches achieved excellent outcomes by many examiners. Nearly all techniques identify and distinguish between the list of capabilities seen in spam and non-spam messages. Spam email comes in various forms today, including advertisements with the intention of making money or selling goods, urban legends that spread hoaxes or urban legends, and so on.

2 LITERATURE SURVEY

Spam mail, commonly referred to as junk mail or UBE, is sent to a group of recipients without their consent. Spontaneous communications that arise organically from a client's mail stream are to be managed by spam filtering. These impulsive sends have historically resulted in a variety of concerns, including overflowing letter boxes, wasting business data transmission, requiring clients to invest time in sorting through it, and the vast spectrum of different issues associated to spam [3]. In 2001, there were 10,847 spams as opposed to 1753 spams in 2000, per a series of reviews put together by CAUBE. Every few years, according to AU 1, the total quantity of spam that 41 email addresses have received has climbed by a factor of six. [4].

Puniskis [5] used the brain network approach to deal with the characterization of spam in his investigation. Instead of relying on the message's unique context or keyword repetition, his method makes use of ascribes that

are based on the graphic elements of the shady examples used by spammers. The data used is a corpus of 1812 spam messages and 2788 real communications that were collected over a period of time. That is how the results demonstrate that while ANN is fantastic, it shouldn't be used by itself as a spam filter.

In [6], Four distinct classifiers—Brain Organisation, SVM Classifier, Guileless Bayesian Classifier—were used to organise email data.

On the basis of various information sizes and varied element sizes, the analysis was conducted. In the event that the final grouping is spam, the final grouping result should be "1," else, it should be "0."

This study demonstrates the viability of a simple classifier that creates a binary tree on a dataset that may be referred to as tree such as binary tree.

3 DATASET DESCRIPTION

The dataset which is used here has been purchased from the Kaggle website. Five or so characteristics of the spams present in emails/sms has been found and incorporated in the dataset. Among the attributes used were the addresses that the spam came from, the sort of spam that was sent, and the organisation that sent the spam.

Kaggle offers data sets that can be used for machine learning algorithms. Data from 5527 email messages make up the spam dataset obtained from Kaggle. There are 5 attributes per instance in the Spam dataset. In the majority of the belongings, the occurrence of a specific term or character in the email that resembles to the example is shown. When comparing the distributions of the SPAM and HAM classes, it becomes clear that the data sets are unbalanced and that the SPAM class is the rare class. A fresh stable exercise data set has its data preprocessed in instruction to prevent biassing for the main class, the HAM class. It has a lot of variables, yet some of the dataset's columns are not necessary. Eliminate any unnecessary columns by doing so. The column names require an update. In order to avoid this, stop words must be removed at the preprocessing stage. Tokenization is the procedure of flouting a string of characters into words, numbers, punctuation, and other symbols, and then identifying tokens that do not need to be deconstructed in further processing, such as punctuation marks, are discarded in the tokenization process.

4 METHODOLOGY

A Data cleaning

Accurate, wrong, duplicate, and incomplete data are removed from the dataset through the process of "data cleaning." This type of information is often not useful for data analysis because it could produce erroneous results.

Finding strategies to increase data set consistency without erasing critical information is what is meant by "data cleaning," which is different from "information erasure to make room for new information." Generally speaking, data cleaning removes erroneous data and improves the data's quality.

Numerous methods can be used to clean data, and the methods used will vary based on the type of data being cleaned. We can perform the following operations to clean the data:

- Remove duplicate or pointless observations,
- correct structural issues,
- filter undesirable outliers,
- handle missing data,
- validate
- quality-assure observations.

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3920	ham	Do 1 thing! Change that sentence into: \Becaus...	NaN	NaN	NaN
785	ham	She was supposed to be but couldn't make it, s...	NaN	NaN	NaN
2111	ham	Yar he quite clever but aft many guesses lor. ...	NaN	NaN	NaN
4920	ham	Its so common hearin How r u? Wat r u doing? H...	NaN	NaN	NaN
3095	ham	We walked from my moms. Right on stagwood pass...	NaN	NaN	NaN

Fig. 3.1

	v1	v2
4864	ham	I'm really sorry I lit your hair on fire
3967	ham	Did u turn on the heater? The heater was on an...
2548	ham	Honestly i've just made a lovely cup of tea an...
1333	ham	Oh... Icic... K lor, den meet other day...
5167	ham	Oh did you charge camera

Fig.3.2

B EDA (Exploratory Data Analysis)

EDA, which commonly makes use of visual approaches, is a method for examining datasets in order to pinpoint and describe their main properties. EDA is used to looking at what the data have to say before starting a modelling job. It takes a thorough comprehension of each column in the spreadsheet or number to identify the important elements in the data.

EDA can be used to comprehend the structure of the data, identify patterns that distinguish spam from ham messages, and cover any underlying trends or features that can be applied to further in-depth research or modelling when it comes to spam and ham communications.

B.1 Pie chart

Pie Chart: Spam and Ham Messages

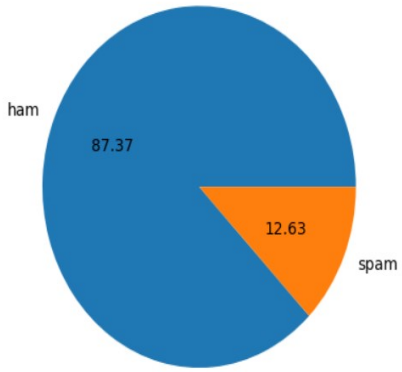


Fig. 4.1

The amount of emails falling into the ham (non-spam) and spam categories is depicted in the pie chart above.

Spam Messages (12.63%):

According to the graph, 12.63% of the messages are considered spam. Spam is a term used to describe unsolicited or unwelcome messages that are frequently delivered for commercial, malicious software distribution, phishing, or other purposes. They may annoy recipients and maybe pose a risk of damage**.**

Ham Messages (87.37%):

According to the graph, 30% of the communications are considered to be "ham," which denotes that they are genuine, non-spam letters. Ham communications are often those that the recipients want to receive and are relevant to them, such as emails from friends or family members, information about their jobs, or messages from reliable sources.

B.2 Historical Graph

The use of colours can be a useful visual aid to help differentiate among the two sorts of communications inside a historical graph showing spam and ham transmissions.

Red for Spam Messages: Spam messages are frequently unsolicited and unwelcome communications delivered in large numbers for commercial, phishing, or other malevolent purposes. As a result, using the colour red to represent spam communications in a historical graph might help visually express the concept that these messages are unwanted and possibly hazardous.

Blue for Ham Messages: Contrarily, ham transmissions are lawful communications that are not categorised as spam. They can be emails from well-known contacts, subscription-based newsletters, or additional messages that the receiver has requested and expected. Consequently, using the colour blue to denote ham signals in a historical graph.

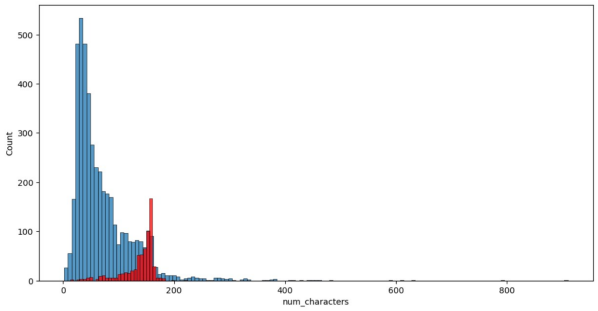


Fig. 4.2 historical graph for num_characters

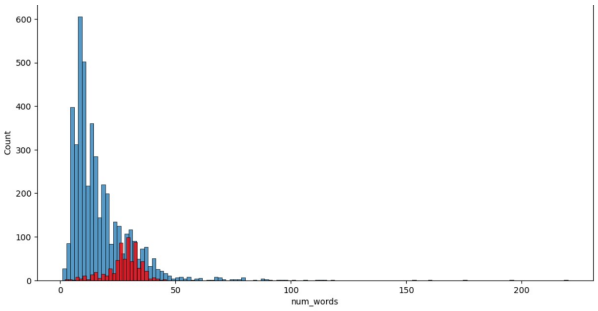


Fig.4.3 historical graph for num_words

You can distinguish between spam and ham communications in a history graph by using different colours to represent the two sorts of messages..

B.3 Pair Plot

A sort of scattering matrix called a pair plot shows scatter plots of various variables in a grid-like arrangement. Each grid cell shows a scatter plot of a pair of variables combined, usually with the same axes.

In a pair plot graph, you can utilise two different markers or colours to distinguish between ham and spam messages. For instance, you could use different colours (like using blue for ham and red for spam) to distinguish between the two groups, or you could use blue for ham and red for spam.

The correlation between two variables or attributes taken from the messages would be depicted by each scatter plot in the pair plot graph. These factors could include details like message length, the existence of specific words or phrases, the presence of particular letters or symbols, or any other pertinent details which can help classify communications as ham or spam.

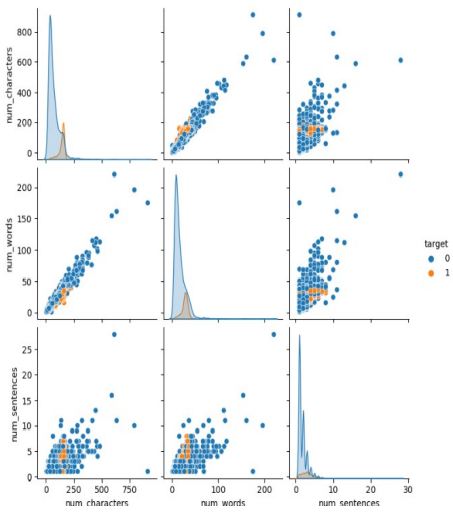


Fig 4.4: Pair plot

C Data Preprocessing

To convert unstructured text input into a form that machine learning algorithms can understand, data preprocessing is a crucial step in natural language processing. The following steps comprise the majority of data preparation:

Lower case: For the sake of maintaining uniformity throughout the data, we transform all the text to lower case in this stage.

Tokenization: The next step is to tokenize the text data, which requires breaking the text down into individual words or tokens. This stage facilitates the examination of the textual material.

Removing special characters: Punctuation marks, symbols, and other non-alphabetic characters are examples of special characters that can amplify the noise in the data and make analysis more challenging. To avoid confusion, we eliminate any special characters from the text data.

Removing stop words and punctuation: Stop words, such as "the," "and," "a," etc., are frequent words that don't offer much value to the text data. We eliminate all stop words from the text data in this stage. Punctuation is likewise eliminated because it serves no purpose in text analysis.

Stemming: In order to decrease the number of unique terms in the text data, in this stage, we transform words into their root form. By doing so, the vocabulary can be condensed and the data analysis process is facilitated.

By using these data pretreatment techniques on the unstructured text data, we can convert the unstructured text data into a format that machine learning algorithms can analyse.

Fig.4.5: Spam Word Cloud

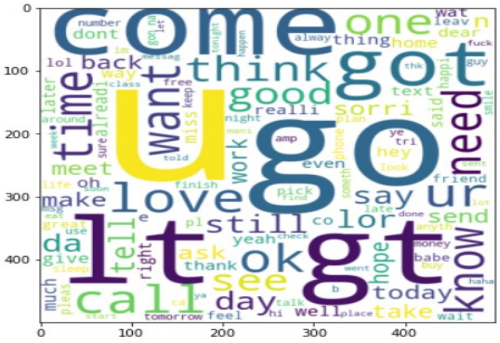


Fig.4.6: Ham Word Cloud

D Model Building

These supervised learning methods were put into place after a dataset analysis to decide which performance would be better [7]. Different representations of the knowledge are generalised by using a variety of techniques and biases. As a result, there is a propensity for this to display an error on various regions of the instance space.The hypothesis that uses multiple algorithms together might be more successful in correcting errors that are unrelated.

Classifier fusion and classifier selection are two paradigms that may be used to govern the ensemble types of various classification algorithms. A single algorithm is selected from the available classifier options in this case to categorise the examples (new instances), and another combines the algorithmic choices. We highlight the most crucial techniques from both categories in this section. One method of decision-making is classified selection.

The best classification algorithms are chosen for use on the test set after this approach evaluates the algorithms on the training set.

The fusion approach has the capacity to incorporate several kinds of classifiers as input and learn from the data.

E Classification Algorithms

To screen spam emails, classification techniques are utilised. It has keywords, phase and characteristics based studies. For filtering spam emails machine learning techniques have been used. Using the data set models build for classification algorithms. In the dataset taken from the UCI repository, there are 1813 spam emails and 2788 valid emails that were sent over the course of several months. Models for classification methods are constructed using this dataset as the training dataset.

- Support Vector Machine
- Naive Bayesian classifier
- Adaboost classifier
- GradientBoosting classifier
- Logistic regression classifier
- KNeighbours classifier
- Bagging classifier
- XGB classifier
- ExtraTrees classifier
- Random Forest classifier
- Decision Tree classifier

E.1 Naïve Bayesian classifier

The Naive Bayes Classifier, which assists in developing machine learning models that can rapidly predict outcomes, is the most simple and effective classification algorithm. The Bayes theorem serves as the establishment for the supervised learning approach used by the Nave Bayes algorithm to solve classification problems.

For the reason that it usages a probablistic classifier, it bases its predictions on the likelihood that a assumed occasion will take place. To determine the possibility of a theory assumed certain earlier info, the Bayes theorem—also known as Bayes' Rule or Bayes' law—is utilised. This can be determined via the conditional probability. The following is the formula for the Bayes theorem:

P(A|B) = P(B|A) * P(A) / P(B)

The posterior probability is P(A|B): Probability of hypothesis A with respect to the observed occurrence B.

P(B|A) is likelihood probability: Probability of the hypothesis B with respect to the observed occurrence

Priority probability, often known as P(A): likelihood theory.

P(B) is Marginal Probability: Probability of Evidence.

The Naive Bayes is a collection of three algorithms:

* MultinomialNB

* BernoulliNB

* GaussianNB

E.1.1 Gaussian Naive Bayes classifier

When deal with the constant data, it is common to make the assumption that the constant values related to every class are circulated according to a usual (or Gaussian) circulation. When working with continuous data, it's common to make the assumption that the constant standards connected to every class are dispersed according to a usual (or Gaussian) circulation. The traits' probability of occurrence is predicated on

$$p(X|Y-c) - \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

It is common to suppose that variance-

- is independent of Y (i.e., i)
- independent of Xi (i.e., k)
- or both (i.e., σ).

The Gaussian Naive Bayes model allows continuous valued structures and assumes that altogether of them follow a Gaussian (normal) circulation.

Assuming that the data is distributed in accordance with a Gaussian distribution with no covariance (independent dimensions) between dimensions is one method for developing an understandable model. This model can be fitted by calculating the nasty and normal deviancy of every point's value within each label, which is all that is necessary to create such a distribution.

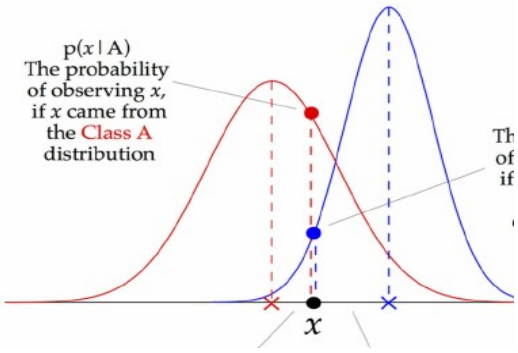


Fig. 4.7

The above image serves as an example of the Gaussian Naive Bayes (GNB) classifier. The z-score, which is the distance from the class mean divided by the class standard deviation, is calculated for each data point.

As a result, we can observe that the Gaussian Naive Bayes is effective and has a somewhat different methodology.

E.1.2 Multinomial Naïve Bayes classifier

The Multinomial Naive Bayes method is a popular Bayesian learning technique in Natural Language Processing (NLP). The programme produces an educated guess regarding a writing's label, such as that of an email or news item, via the Bayes principle. It computes the probabilities of every label for a specific example and outputs the label with the peak probabilities.

Multinomial Naive Bayes is a variation of the Naive Bayes method that is used in machine learning, and it is a great choice for use with datasets that are spread among several nodes. When there are several classes into which the text can be categorised, this approach can be used to predict the label of the text. This is accomplished by figuring out the likelihood of each label for the input text, and then producing the label with the highest likelihood as an output.

$$p(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

E.1.3 Bernoulli -NB classifier

IT is one of its variations. As shown by the Nave Bayes classification algorithm for machine learning, which provides the probability of an event occurring. Because it is a probabilistic classifier, the NB classifier forecasts the likelihood that input will be classified into all classes given the supplied data. It is also known as conditional probability.

Bernoulli Naive Bayes is a member of the Naive Bayes family. Only binary input is permitted. When determining if a value matches a word that appears in the document, this is the instance. That model is quite basic. When assessing word frequency is not the main factor, Bernoulli might yield more precise results. For example, whether a word appears in a document or not, each value that indicates a binary term occurrence feature must be counted. These attributes are used in place of counting the amount of intervals a term looks in the document.

Let p represent success probability and q represent failure probability as we work with binary values; q=1-p. With regard to a random variable 'X' with a Bernoulli distribution,

The Bernoulli distribution,

$$p(x) - P\lfloor X - x \rfloor - \begin{cases} q = 1 - p & x = 0 \\ p & x - 1 \end{cases}$$

E.2 Adaboost classifier

Adaptive boosting, often known as AdaBoost, is a machine learning technique used for data collection. The most well-known calculation utilised with AdaBoost is one-level decision trees or decision trees with only one split. Additionally called "Decision stumps," these trees.

This algorithm creates a model and assigns equal weights to each data point by creating a fictitious model. Then, it assigns higher weights to incorrectly designated points. Currently, any point with a larger load is given more weight in the model that follows. Model preparation will continue until a lower error is evaluated, at which point it will stop.
Algorithm:

1. Give every data point in the dataset an equal weight when initialising the dataset.
2. Enter this as model input and find the data points that were misclassified.
3. Increase the significance of the data items that were misclassified..
4. If (received the necessary findings) move to step 5, otherwise go to step 2.
5. End.

AdaBoost is use to integrate weak base beginners, but it has too been shown to integrate powerful base beginners, alike bottomless decision trees, successfully, producing an even more accurate model.

E.3 Gradient Boosting classifier

Gradient Boosting classifier is a ML Algorithm that coming feeble models together and then generate a solid analytical models. Gradient boosting are becoming prevalent because of the their efficiency of classifying complex datasets and also applied on several kaggle datasets to the strength of the models.

Gradient Boosting Algorithm is used to build models and try to reduce the error of the previous model.

An approach similar to gradient descent that reduces the loss function by computing the calculated loss and applying gradient descent to the loss function in order to minimise the error between parameters.

E.4 Logistic Regression classifier

Logistic regression is a grouping method use in machine learning. In order to model the reliant variable, a logistic task is use. Here only binary possible categories for the reliant variable because it is dicotomous (for example, the cancer could be malignant or not), leaving no other options. Therefore, while working with binary data, this method is used. There are only two feasible classes because of the dichotomous character of the dependent variable.

The reliant variable is, to place it simple, a binary variable, with data noted as any 1 (which show success/yes) or 0 (which show failure/no).

A logistic regression model predicts P(Y=1) as a function of X mathematically. One of the most straightforward ML methods, it might be applied to a many of classification problems, including spam detection, diabetes estimation, cancer analysis, etc.

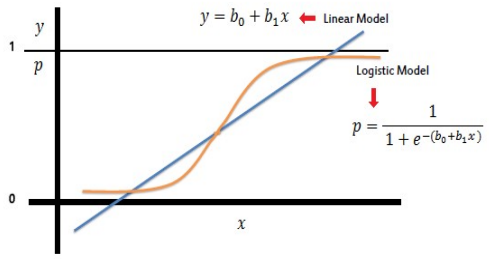


Fig.4.8

E.5 KNeighbors classifier

The most popular algorithm based on the supervised technique is the KNeighbors classifier. It assesses how similar the new case data is to the current case dataset and incorporates it into the current case data in associated categories. Using the available similarity dataset, KNN-stores the existing data and categorises a new data point.

It can be as the Regression as well as classification. KNN algorithm does not absorb by the training dataset instead, it provides the dataset and information on the classification phase. It makes a change to the dataset so, it can be also called Lazy algorithm.

E.6 SVM classifier

Support Vector Machine, or SVM, is the very popular supervised learning method, and this is use to point classification or regression difficulty. This is use in machine learning, nevertheless, to address classification difficulty.

The aim of this approach is to build the most suitable limit or line of decision that can create classes in n-dimension space, allowing us to fastly graph subsequent data facts in the relevant category. This ideal decision margin is known as the hyperplane.

The vector chosen by SVM are used to construct the hyperplane. These extreme occurrences are known as support vectors, which is why this method is called a machine using support vectors.

SVM is available in two forms:

A dataset is considered to have been linearly separated when it can be split into two groups along a straight line. A linear SVM classifier is then used to classify the dataset.

For un-linear divided data, un-linear SVM is utilised. A dataset is called un-linear when it is unable to be classified using a plane that is straight, in which case the nonlinear SVM classifier is employed to do it.

In n-dimensional space, there may be several different decision borders that can be used to separate classes, then we necessity identify the optimum choice margin that can accurately classify the data tally. The name for this ideal boundary is the hyperplane.

The dimensions of the hyperplane are determined by the features in the dataset; hence, even though there are two characteristics, the hyperplane is a straight line. Where there are three features, the hyperplane will also be a 2-D plane. We consistently create a hyperplane with maximum margin, or the widest gap among the data points.

Support vectors are a set of points or vectors that are closest to their respective hyperplane and have an effect on the position of the hyperplane.

E.7 XGBoost classifier

XGBoost is the execution of the gradient boosted decision trees which is being planned for speediness and performance that is good machine learning.

A decision tree-based machine learning approach called Extreme Gradient Boosting, or XGBoost, employs boosting to improve performance. It has been one of the most effective machine learning algorithms since it was first developed, routinely outperforming other algorithms.

Although XGBoost also offers libraries for Python and integrates nicely with scikit-learn, scikit-learn is a machine learning platform that Python data scientists typically use. It may be used to address classification and regression problems and is ideal for a broad range of data science applications.

E.8 Extra Trees classifier

Extremely Randomised Trees Classifiers, often referred to as Extra Trees Classifiers, are an ensemble learning method that combines the output of several decorrelated decision trees gathered in a "forest" to produce classification results. The way the decision trees within the forest are constructed is the only way that it is distinct from the Random Forest Classifier.

Every single decision tree in an Extra Trees Forest is built using the original data used for training. The most suitable feature to split the information at each node must then be selected by each decision tree from a randomly selected group of k features drawn from a feature collection. This random sample of features results in a large number of de-correlated decision trees.

To execute the feature selection using the aforementioned forest structure, the standardised overall reductions in the mathematical criteria used in the split decision feature is computed. Gini This value is called the feature's importance.

E.9 Random Forest classifier

The supervised learning approach includes the machine learning algorithm Random Forest. Machine learning uses it for both regression and classification problems. It is based on the concept that ensemble learning, which is the process of combining several classifiers to address complex problems and improve model performance.

Considering what the name implies, "Random Forest is a classifier in which the number of decision trees on various subsets of the dataset and takes average to improve the predictive accuracy of dataset." Instead of using a single decision tree, the random forest makes use of prediction from each tree and predicts the outcome based on which predictions garnered the most overall votes.

A majority trees there are in the forest, the more accurate it is and less overfitting occurs.

The method known as Random Forest should be utilised for the following reasons:

- Compared to other systems, it takes less time to train users.
- It is capable of making exact predictions regarding the outcome regardless of a larger dataset.
- Even when a sizable quantity of data is missing, it can still be correct.

E.10 Decision Tree classifier

An example of supervised learning is the Decision Tree Classifier. The decision tree classifier functions similarly to a typical tree with roots, branches, and leaves. Attributes are tested on each internal node, the results are displayed on the tree's branches, and the leaf node receives the results.

A decision Tree is a type of tree where each node represents a feature, each branch a choice, and each leaf an outcome. If, as its name suggests, a root node is the uppermost node, it is the parent node. The entire concept is aided by the creation of a tree for all of the data, which is then processed at each leaf node.

A decision tree refers to a tree-based approach in which the beginning of the tree is characterised by a data separation sequence or the shape of the tree's root up until a boolean outcome at the leaf node of the tree.

Divide and conquer tactics are used in decision tree learning to find the ideal split points within the dataset by using a greedy search approach.

We employ the CART technique, which stands for Classification and Regression Tree algorithm, to visually display all potential solutions based on the criteria. In a decision tree, the question is simply posed, the response is (Yes/NO), and the tree is then divided into subtrees.

E.11 Bagging classifier

A bagging classification is a group of meta-estimators that combine their separate forecasts to produce a final prediction after each meta-estimator fits a base classifier independently to a set of randomised subsets of the initial dataset. The black-box estimate as a decision tree can be made more accurate by introducing randomness to its development mechanism and then making a collection from it, a meta-estimator of this kind is frequently used as a technique for reducing the disparity between estimates.

A preparation set is used to build each base classifier. This preparation set is made by randomly extracting N data points with replacement via the first training dataset, in which N represents the total amount of the first training set. The training sets for each base classifier are distinct from one another. Numerous the initial details might be repeated in the ensuing preparation set, while others would be forgotten.

However, bagging reduces overfitting (fluctuation) by averaging or polling, despite the fact that this leads to an increase in bias that is offset by a reduction in dissimilarity.

5. RESULT EVALUATION

The dataset used here was divided into 2 portions, one of which served as the foundation for the prediction model, while the other part was used to assess the model's accuracy. Both the feature values and the classification of each record are included in the segment used to build the model. The 10-fold cross validation procedure was used to evaluate the model.

5.1 MEASURING THE PERFORMANCE

Depending on the particular field in which it is used, an effective classifier may be defined differently. For instance, when identifying spam, it's important to avoid classifying legitimate messages as spam because doing so could have detrimental effects on the user, such as causing them financial or emotional hardship.

5.2 PRECISION AND RECALL

In the context of recognising spam, precision and recall are often used metrics for assessing the effectiveness of information retrieval algorithms. The percentage of pertinent items—in this case, spam

messages—that are accurately recognised as such is known as recall. The fraction of communications accurately labelled as spam among all messages classified as spam, on the other hand, is measured by precision. A high recall means that the majority of spam communications are successfully identified, whereas a high precision means that few real messages are categorised as spam. However, the precision rate will drop if even one legitimate communication is labelled as spam.

Precision = (TP / TP + FP) and Recall = (TP / (TP + FN)) are the formulas used to compute precision and recall.

Let,

Ngg is also referred to as a false negative and is categorised as a good message as ham.

Ngs is also referred to as false positives and is categorised as both spam and good messages.

Nss are also referred to be genuine positives and are categorised as spam mails.

Nsgs are also referred to as false negatives and are categorised as spam communications and ham.

The precision counts the amount of false positives, or legitimate messages that are mistakenly labelled as spam. While a poor recall rate only indicates that there may be some spam messages in the inbox, a low precision rate can be harmful to the user. Therefore, high precision is more important than good recall [8]. When assessing classifiers, it can be difficult to strike a balance between recall and precision; however, utilising a combination score, such as weighted accuracy, can help with this.

Fig. 4.9

5.3 CROSS VALIDATION

There are various methods for assessing a classifier's performance after training. The holdout method, which divides the dataset into two portions for training and testing, is one straightforward technique. The outcomes of this strategy, however, greatly depend on how the sample set is divided, which is a disadvantage. K-fold cross-validation is another way that lowers the holdout method's variability.

The dataset is split into k distinct, non-overlapping portions for k-fold cross-validation, and the model is trained on one part and tested on another. Each component serves as the test set once during this procedure' repetition k times. The outcomes of all k tests are averaged to assess performance. The mean of the findings from each individual test is employed to determine the recall and precision for a k-fold test.

The accuracy p and recall r for the k-folded test are specified as,

$$Cv = \frac{1}{k}\sum_{i=1}^{K} MSE$$

The mean of the precision and recall from each individual test is what is referred to as the k-fold test's precision and recall. Research has demonstrated that k=10 is sufficient, and the experiments in this thesis employed this approach.

6. RESULTS AND DISCUSSION

Predictive accuracy was developed as the most important evaluation criterion in order to have a thorough picture of the scenario. The following formula was used to determine this.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ Number\ of\ predictions}$$

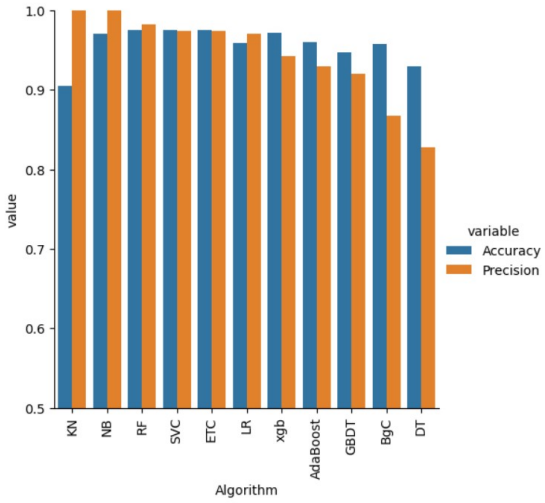
Correctly Classified Instances+ Incorrectly Classified Instances = Total Number of Instances .

Accuracy = (TrueNegative + TruePositive)/
(TruePositive + FalsePositive +
TrueNegative + TruePositive)

P.A. stands for prediction accuracy.

The sum of the cases that were correctly classified and those that were wrongly classified yields the total number of instances.

How well an algorithm anticipates the requested data is determined by its predictive accuracy. Three factors— accuracy of predictions, training time, and the error rate —were considered for evaluating the dataset's performance.



7. CONCLUSION AND FUTURE WORK

In this research, a common dataset is used to implement the thorough study of various classifiers. On the basis of the previously indicated evaluation criteria, the outcomes are compared. This study demonstrates that the performance of a classifier that is identical when carried out over an identical dataset using different software packages. The Nave Bayes classifier is a solid choice. The performance of some classifiers, such as Simple Logistic and Adaboost, is good. However, MLP is superior when compared to it. As a result, MLP makes a decision in every situation based on all comparisons and viewpoints.

ACKNOWLEDGMENT

The authors are appreciative of our department's and our organization's invaluable assistance.

REFERENCES

[1] C. Pu and S. Webb, "Observed trends in spam construction techniques: A case study of spam evolution", *Proceeding of 3rd Conference on E-Mail and Anti-Spam,* 2006.

[2] M. Embrechts, B. Szymanski, K. Sternickel, T. Naenna, and R. Bragaspathi,"Use of Machine Learning for Classification of Magnetocardiograms", *Proceedings of IEEE Conference on System, Man and Cybernetics, Washington DC*, pp. 1400-05, 2003.

[3] Duncan Cook, Jacky Hartnett, Kevin Manderson and Joel Scanlan, "Catching Spam before it arrives: Domain Specific Dynamic Blacklists", *in ACSW Frontiers, Australian Computer Society*, Vol. 54, pp. 193 –202,2006.

[4] Bekker S, "Spam to Cost U.S. Companies $10 Billion in 2003", ENTNews, http://www.entmag.com/news/article.asp?EditorialsID=565 1>.

[5] D. Puniškis, R. Laurutis and R. Dirmeikis, "An Artificial Neural Nets for Spam e-mail Recognition", *Electronics and electrical engineering*, Vol. 69, No. 5, pp. 73 – 76, 2006.

[6] Youn and Dennis McLeod, "A Comparative Study for Email Classification", *Proceedings of International Joint Conferences on Computer, Information, System Sciences and Engineering,* 2006.

[7] Witten I. & Frank E., "*Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*", Morgan Kaufmann Publishers, 2000.

[8] Upasana Pandey and S. Chakraverty "A Review of Text Classification Approaches for E-mail Management".