PREDICTION OF HAM OR SPAM (EMAIL)

USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

GANESH J	**211518104035**
KABILAN B	**211518104058**
NISHANTH K	**211518104103**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR INSTITUTE OF TECHNOLOGY,

POONAMALLEE

ANNA UNIVERSITY: CHENNAI-600025

JUNE 2022

ANNA UNIVERSITY: CHENNAI - 600025

BONAFIDE CERTIFICATE

Certified that this project report titled **" PREDICTION OF HAM OR SPAM (EMAIL) USING MACHINE LEARNING"** is the bonafide work of **" GANESH J (211518104035), KABILAN B (211518104058), NISHANTH K (211518104103)"** who carried out the projectwork under my supervision.

SIGNATURE
Dr . V. SUBEDHA, M. Tech., Ph.D.

Professor and Head,
Department of CSE,
Panimalar Institute of Technology,
Ponnamalle, Chennai-600123.

SIGNATURE
MRS. S. ANNIE SHERYL, M.Tech.

Assistant Professor,
Department of CSE,
Panimalar Institute of Technology,
Ponnamalle, Chennai-600123.

Certified that the above candidates were examined in the university project work viva voce examination held on_____at Panimalar Institute of Technology, Chennai-600 123.

INTERNAL EXAMINER **EXTERNALEXAMINER**

ACKNOWLEDGEMENT

A project of this magnitude and nature requires the kind cooperation and support of many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We would like to express our deep gratitude to Our **Beloved Secretary and Correspondent, Dr P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation which inspired us a lot in completing the project.

We also express our sincere thanks to Our **Dynamic Directors Mrs.C.VIJAYARAJESWARI, Mr. C. SAKTHIKUMAR, M.E.,** and **Mrs.S.SARANYA SREE SAKTHIKUMAR, B.E.,** for providing us with the necessary facilities for the completion of this project.

We also express our gratefulness to our **Principal Dr T.JAYANTHY, M.E., Ph.D.,** who helped us in the completion of this project.

We wish to convey our thanks and gratitude to our **Head of the Department Dr. V. SUBEDHA, M Tech., Ph.D.,** Department of Computer Science and Engineering, for her support and providing us ample time to complete our project.

We express our indebtedness and gratitude to our Staff in charge , **Mr . K . SATHYAMOORTHY , M.Tech.,(CSE)**Assistant Professor**,** Department of Computer Science and Engineering, for her guidance throughout the Course of our project.

TABLE OF CONTENTS

ABSTRACT

Nowadays, we use frequently e-mails, one of the communication channels, in electronic environment. It play an important role in our lives because of many reasons such as personal communications, business-focused activities, marketing, advertising etc. E-mails make life easier because of meeting many different types of communication needs. On the other hand they can make life difficult when they are used outside of their purposes. Spam emails can be not only annoying receivers, but also dangerous for receiver's information security. Detecting and preventing spam e-mails has been a separate issue. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyse the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. To propose a machine learning-based method to classify the email in the form of spam or ham by best accuracy from comparing supervised classification machine learning algorithms.

Keywords: Machine learning, Ham, Spam, Dataset, Logistic Regression.

LIST OF FIGURES

LIST OF ABBREVATIONS

S.NO.	ABBREVATIONS	EXPANSIONS
1	ML	Machine Learning
2	AI	Artificial Intelligence
3	NES	Non-Stationary Environment
4	JIT	Just In Time
5	HCDT	Hierarchical Change Detection-Tests
6	RFA	Random Forest Algorithm
7	GUI	Graphical User Interface
8	IT	Information Technology
9	OS	Operating System
10	SQL	Structured Query Language
11	OOP	Object Oriented Programming
12	CASE	Computer Aided Software Engineering
13	OMT	Object Modelling Technique
14	OMG	Object Management Group
15	DFD	Data Flow Diagram
16	SUT	Software Under Test

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Over View

Users can automatically write evaluations or comments on e-commerce websites in the Web 2.0 era. Both customers and collaborations benefit greatly from user-generated content. On the one hand, reading these evaluations before purchasing a product or service can provide buyers with some knowledge about it. Business companies, on the other hand, might use these reviews to improve their goods and marketing methods. When it comes to purchase decisions, people are often influenced by reviews information, therefore favorable evaluations may bring a lot of money and recognition to businesses and individuals. This encourages the spread of false opinion spam (also known as false reviews). However, spam filtering helps to reduce recipient overload to some extent, but with these adjustments, it is feasible to construct an email system that is more efficient and accurate. In addition, a system that gives user-specific output has been sought. This ensures that everyone who utilizes the system has a positive experience. The objective is to create a machine learning model for predicting email spam or ham, which might eventually replace updatable classifier models by predicting outcomes in the form of the greatest accuracy by comparing supervised algorithms.

1.2 Machine Learning

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data to produce accurate results. Machine learning combines data with statistical tools t o predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers. A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation. Machine learning is also used for a variety of task like house price prediction, predictive maintenance, portfolio optimization, automatize task and so on.

Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer codes all the rules in consultation with an expert the industry for which software is being developed. Each rule is based on a logical foundation and the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

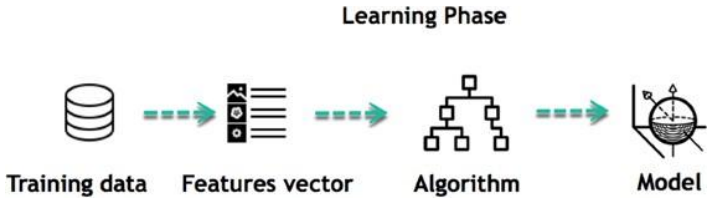
How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference.

1. Learning

First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.



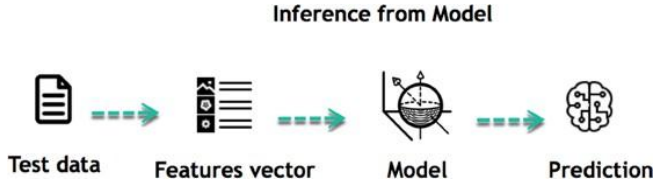
Learning Phase (Machine Learning)

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model.

2. Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.



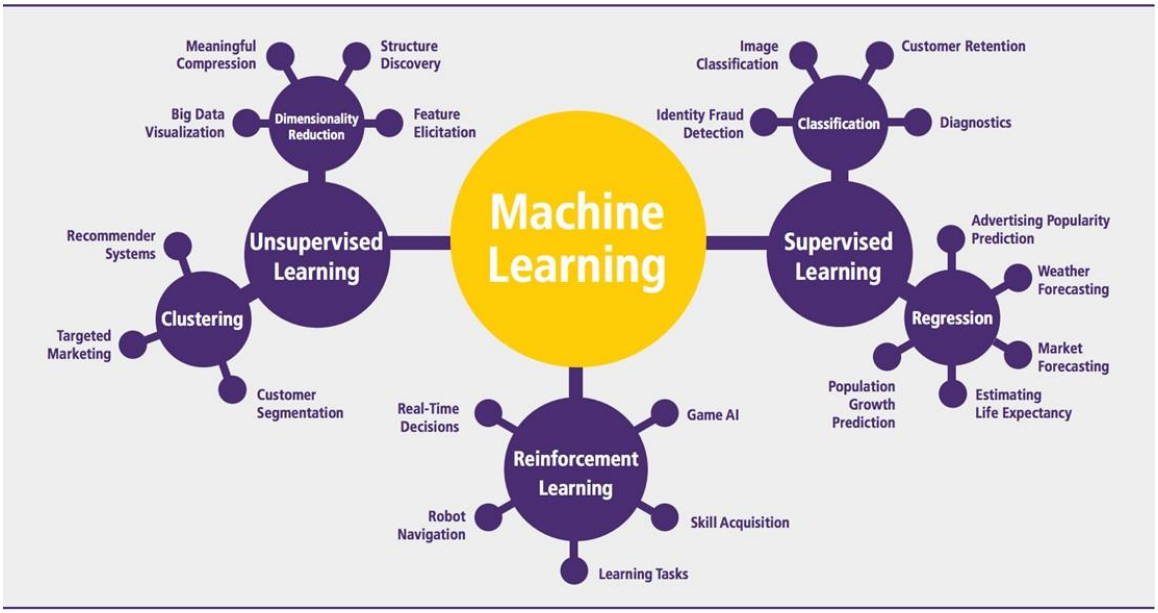
Inference Phase (Machine Learning)

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4–7 until the results are satisfying
9. Use the model to make a prediction

Machine learning Algorithms and where they are used?



Machine Learning (Unsupervised and Supervised)

Machine learning can be grouped into two broad learning tasks:

Supervised and **Unsupervised.** There are many other algorithms.

1. Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- ✓ Classification task
- ✓ Regression task

1.1. Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

1.2. Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index.

The system will be trained to estimate the price of the stocks with the lowest possible error.

2. Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you.

Application of Machine learning

Augmentation:

Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry:

Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.**Government organization:**

The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry:

Healthcare was one of the first industry to use machine learning withimage detection.

Marketing:

Broad use of AI is done in marketing thanks to abundant access todata. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

1.3 Artificial Intelligence

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading

AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many

challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or

an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language

processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

1.4 PREPARING THE DATASET:

This dataset contains 3000 records of features extracted which were then classified into 2 classes

1. Spam
2. Ham

1.4 OBJECTIVES

The goal is to develop a machine learning model for email spam or ham Prediction, to potentially replace the updatable supervised machine learning

classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

1.5PROJECT GOALS

- Exploration data analysis of variable identification
  - Loading the given dataset
  - Import required libraries packages
  - Analyze the general properties
  - Find duplicate and missing values
  - Checking unique and count values
- Uni-variate data analysis
  - Rename, add data and drop the data
  - To specify data type
- Exploration data analysis of bi-variate and multi-variate
  - Plot diagram of pairplot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
  - Pre-processing the given dataset
  - Splitting the test and training dataset
  - Comparing the Decision tree and Logistic regression model and random forest etc.
- Comparing algorithm to predict the result
  - Based on the best accuracy

1.5 SCOPE OF THE PROJECT

The main Scope is to detect the spam email, which is a classic text classification problem with a help of NLP and machine learning algorithm. It is needed to build a model that can differentiate the email between “Spam” and “Ham”.

CHAPTER 2

Literature Survey

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Review of Literature Survey

Title : MACHINE LEARNING METHODS FOR SPAM E-MAIL CLASSIFICATION

Author: W.A. Awad1 and S.M. ELseuofi2

Year : 2011

The increasing volume of unsolicited bulk e-mail (also known as spam) has generated a need for reliable anti-spam filters. Machine learning techniques now days used to automatically filter the spam e-mail in a very successful rate. In this paper we review some of the most popular machine learning methods (Bayesian classification, k-NN, ANNs, SVMs, Artificial immune system and Rough sets) and of their applicability to the problem of spam Email classification. Descriptions of the algorithms are presented, and the comparison of their performance on the SpamAssassin spam corpus is presented.

Title : Intelligent Model for Classification of SPAM and HAM

Author: Amandeep Singh Rajput, Vijay Athavale, Sumit Mittal

Year : 2019

In our study, we propose a collaborative approach by using cluster computing with the help of parallel machines for fast isolation of SPAM and HAM. A cluster approach can increase the computing power many folds with existing hardware and resources thus by increasing the speed of processing without incurring any extra cost. In this study, we only use header based filtering method, thus by keeping the privacy of the user intact. The standard test set for HAM and SPAM from Spam Assassin [1][2] is used. Two types of parallel environments are used in this research. First is where multiple Anti-Spam methods are used in the parallel environment against the test corpora and false positive and false negative accuracy recorded. The second parallel environment is where standard test corpora are divided into parts and fed into parallel machine environment with single anti-spam method used at all machines and the time saving is recorded

against standalone machine being used.

Title : A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques

Author: Hanif Bhuiyan, Akm Ashiquzzaman, Tamanna Islam Juthi, Suzit Biswas

Year : 2018

E-mail is one of the most secure medium for online communication and transferring data or messages through the web. An overgrowing increase in popularity, the number of unsolicited data has also increased rapidly. To filtering data, different approaches exist which automatically detect and remove these untenable messages. There are several numbers of email spam filtering technique such as Knowledge-based technique, Clustering techniques, Learning-based technique, Heuristic processes and so on. This paper illustrates a survey of different existing email spam filtering system regarding Machine Learning Technique (MLT) such as Naive Bayes, SVM, K-Nearest Neighbor, Bayes Additive Regression, KNN Tree, and rules. However, here we present the classification, evaluation and comparison of different email spam filtering system and summarize the overall scenario regarding accuracy rate of different existing approaches.

Title : Ham and Spam E-Mails Classification Using Machine Learning Techniques

Author: Mahmoud Bassiouni, Mayar Aly Shafaey

Year : 2018

Spam e-mail has become a very serious problem. Sending inappropriate messages to a large number of recipients indiscriminately has resulted in anger by users but large profits for spammers. This article looks at classifying spam e-mails from inboxes. Ten alternative classifiers are applied on one benchmark dataset to evaluate which classifier gives better result. A 10-fold cross validation is used to provide the accuracy. Results of the classification algorithms are compared with the spambase UCI dataset. The experimental results approve that the spam mails can be classified correctly, with accuracy reaching up to 95.45% for the Random Forest technique, compared to other classifiers used.

Title : An Intelligent Spam Detection Model Based on Artificial Immune System

Author: Abdul Jabbar Saleh, Asif Karim, Bharanidharan Shanmugam , Sami Azam

Year : 2019

Spam emails, also known as non-self, are unsolicited commercial or malicious emails, sent to affect either a single individual or a corporation or a group of people. Besides advertising, these may contain links to phishing or malware hosting websites set up to steal confidential information. In this paper, a study of the effectiveness of using a Negative Selection Algorithm (NSA) for anomaly detection applied to spam filtering is presented. NSA has a high performance and a low false detection rate. The designed framework intelligently works through three detection phases to finally determine an email's legitimacy based on the knowledge gathered in the training phase. The system operates by elimination through Negative Selection similar to the functionality of T-cells' in biological systems. It has been observed that with

the inclusion of more datasets, the performance continues to improve, resulting in a 6% increase of True Positive and True Negative detection rate while achieving an actual detection rate of spam and ham of 98.5%. The model has been further compared against similar studies, and the result shows that the proposed system results in an increase of 2 to 15% in the correct detection rate of spam and ham.

CHAPTER 3

System Analysis

3.1 Existing System

The growing consumerism has led to the importance of online reviews on the Internet. Opinions voiced by these reviews are taken into consideration by many consumers for making financial decisions online. This has led to the development of opinion spamming for profitable motives or otherwise. This work has been done to tackle the challenge of identifying such spammers, but the scale of the real-world review systems demands this problem to be tackled as a big data challenge. A big data approach was applied for the detection of review spammers. A metadata-based rating model for detecting review spammers was implemented on large datasets to study the effect of scale on such models. The discrepancies were identified, and mitigations for the same in the form of exponential smoothing were proposed. A distributed computing platform was set up and heterogeneous methods were applied to compute the various spam indicators

Drawbacks of the Existing System

- Machine Learning concept is not implemented.
- Accuracy and performance metrics are not calculated.

3.2 Proposed System

Exploratory Data Analysis:

Machine learning supervised classification algorithms will be used to give the given

dataset and extract patterns, which would help in classifying the reviews, thereby helping the apps for making better decisions of their features in the future.

Data Wrangling:

In this section of the report, you will load in the data, check for cleanliness, and then trim and clean your dataset for analysis.

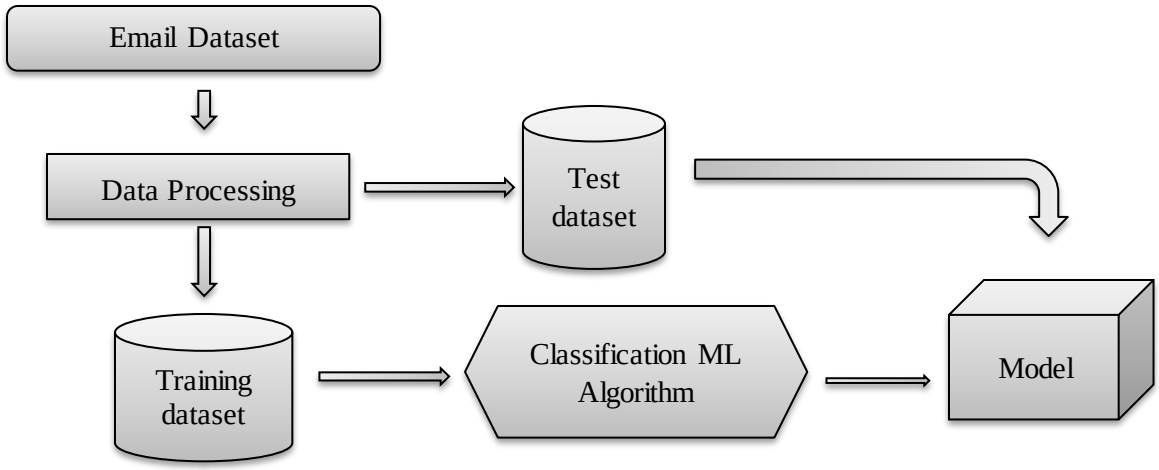
Data collection:

The data set collected for classifying the given data is split into Training set and Test set. Generally, 70:30 percentage are applied to split the Training set and Test set. The Data Model which was created using the SMLT is applied on the Training set and based on the test result accuracy, Test set prediction is done.

Building the classification model

The prediction of the email spam or ham is good in machine learning algorithm prediction model and is effective because of the following reasons the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.



Architecture of Proposed model

Advantages of Proposed System

- Machine Learning method is implemented.
- Pre-Processing and analyzing the data.
- Performance metrics of different algorithm are compared, and the better prediction is done.

3.3 System Requirement Analysis

Requirement analysis determines the requirements of a new system. This project lyses a product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

3.3.1 Hardware Requirement

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design.

- Processor : Core i3/i5/i7

- RAM : 4 GB
- HDD : 500 GB

3.3.2 Software Requirement

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

- Platform : Windows 7,Windows 8 & Windows 10 (32or64 bit)
- Software : Anaconda Navigator
- Tool : Jupyter Notebook
- Language : Python
- Libraries : Numpy,Pandas,Sklearn,Matplotlib,glob

3.4 Software Description

3.4.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack

as it can be used for everything from web development to software development and scientific applications.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games

History of Python

Python was developed by **Guido van Rossum** in the lute eighties and early. nineties at the **National Research Institute for Mathematics and Computer Science** in the Netherlands Python is derived from many other languages, including ABC, Modala-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Characteristics of Python

Python's characteristics include:

- ☑ **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- ☑ **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- ☑ **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- ☑ **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh
- ☑ **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- ☑ **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- ☑ **Extendable:** You can add low-level modules to the Python interpreter These modules enable programmers to add so or customize their tools to be more efficient.
- ☑ **Databases:** Python provides interfaces to all major commercial databases .
- ☑ **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX . Scalable: Python provides a better structure and support for large programs than shell scripting

Apart from the above-mentioned characters, Python has a big list of good features,few are listed below:

- ☑ IT supports function and structured programming methods as well as OOP . It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic types checking.
- ☑ IT supports automatic garbage collection
- ☑ It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java

Features of Python

A simple language which is easier to learn. Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

- **Free and open source:** You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.
- **Portability:** You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Limos
- **Extensible and Embeddable:** Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
- **A high-level, interpreted language:** Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower level operations.
- **Large standard libraries to solve common tasks** Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use My SQL db library using

import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

- **Object-oriented:** Everything in Python is an object Object oriented programming (OOP) helps you solve a complex problem intuitively. With OCP, you are able to divide these complex problems into smaller sets by creating object.

LIBRARIES USED IN PROJECT:

o Numpy Package

Numpy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. This encapsulates n-dimensional arrays of homogeneous data types, with many operation is being performed in compiled code for performance.

There are several important differences between NumPy arrays and the standard Python sequences: NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original. The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements. NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using

Python's built-in sequences. A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence type is insufficient one also needs to know how to use NumPy arrays. The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element.

- **Pandas Package**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.[4] Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010 .Developer Wes McKinney started working on pandas in 2008 while at AQR Capital Management out of the need for a high performance, flexible tool to perform quantitative analysis on financial data. Before leaving AQR he was able to convince management to allow him to open source the library.

- **Sklearn Package**

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Scikit-learn and sklearn both refer to the same package however, there are a couple of things you need to be aware of. Firstly, you can install the package by using either of scikit-learn or sklearn identifiers however, it is recommended to install scikit-learn through pip using the skikit -learn identifier.

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.

Both are 3rd party machine learning modules, and both are good at it. Tensorflow is the more popular of the two. Tensorflow is typically used more in Deep Learning and Neural Networks. SciKit learn is more general Machine Learning

- **Matplotlib Package**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib is the O.G. of Python data visualization libraries. Despite being over a decade old, it's still the most widely used library for plotting in the Python community

It is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter.

- **Matplotlib Included in Python?**

Matplotlib is not a part of the Standard Libraries which is installed by default when Python, there are several toolkits which are available that extend python matplotlib functionality. Matplotlib is a library for making 2D plots of arrays in Python. Although it has its origins in emulating the MATLAB graphics commands, it is independent of MATLAB, and can be used in a Pythonic, object oriented way.

- **Glob Package**

The glob module is a useful part of the Python standard library. glob (short for global) is used to return all file paths that match a specific pattern.

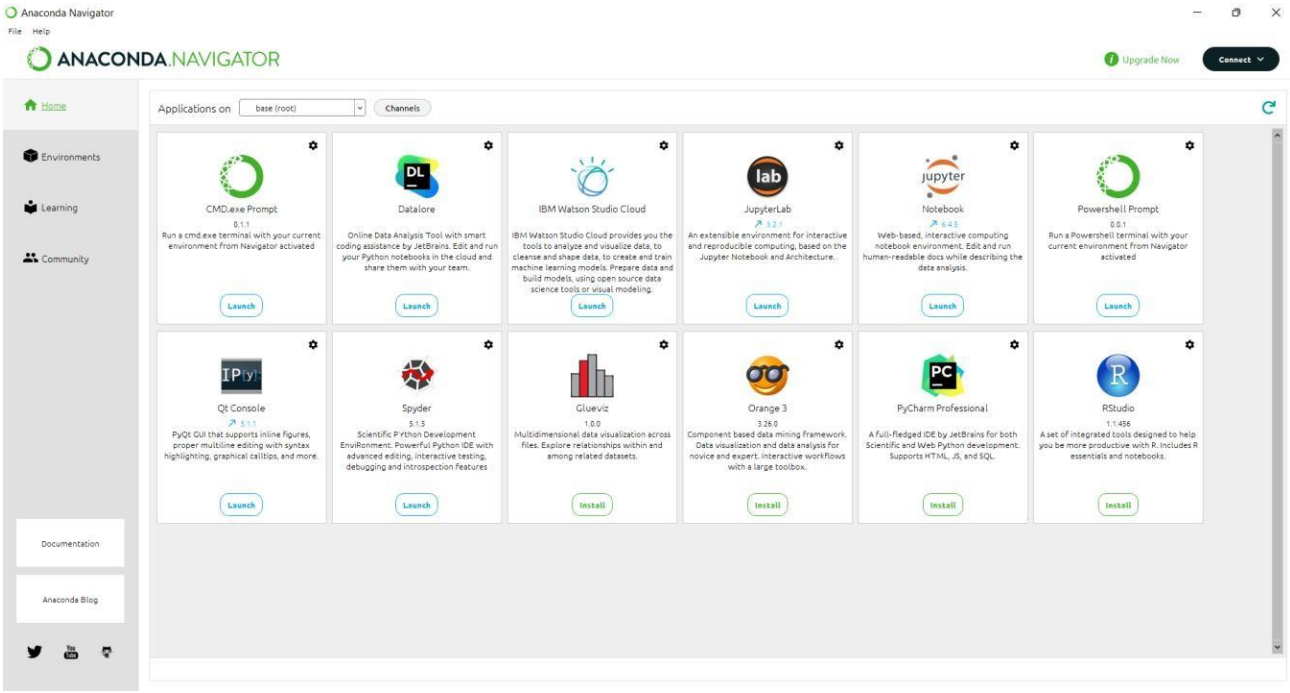
The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order

3.4.2 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local

Anaconda Repository. It is available for Windows, mac OS and Limoe. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator. Advanced conda users can also build your own Navigator applications. The simplest way to run code in the navigator is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code. You can also use Jupyter Notebooks the same way, Jupyter Notebooks are an increasingly popular system that combines your code, descriptive text, output, images and interactive



interfaces into a single notebook file that is edited, viewed and used in a web browser.

The following applications are available by default in Navigator.

- ◊ Jupyterlab
- ◊ Jupyter Notebook
- ◊ QTConsole
- ◊ Spyder
- ◊ VSCode
- ◊ Glueviz
- ◊ Orange 3 App
- ◊ RStudio
- ◊ Pycharm

Anaconda Enterprise

Anaconda Enterprise is an enterprise-ready, secure and scalable data science platform that empowers teanu to govern data science assets, collaborate and deploy data science projects:

Enterprise 5 includes these capabilities:

- Easily deploy your projects into interactive data applications, live notebooks and machine learning models with APIs.
- Share those applications with colleagues and collaborators
- Manage your data science assets: notebooks, packages, environments and projects in an integrated data science experience.

Anaconda Enterprise 5

With Anaconda Enterprise, you can do the following:

- **Develop:** ML/AI pipelines in a central development environment that scales from laptops to thousands of nodes
- **Govern:** Complete reproducibility from laptop to cluster with the ability to configure access control
- **Automate:** Model training and deployment on scalable, container-based infrastructure

Anaconda Distribution

Anaconda Distribution is a free, easy-to-install package manager, environment manager and Python distribution with a collection of 1,000+ open source packages with free community support. Anaconda is platform-agnostic, so you can use it whether you are on Windows, macOS or Linux

Anaconda Cloud

Anaconda Cloud is a package management service that makes it easy to find, access, store and share public notebooks, and environments, as well as conda and PyPI packages.

Conda Package

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux Conda quickly installs runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local

computer. It was created for Python programs, but it can package and distribute software for any language.

Conda as a package manager helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment. In its default configuration, conda can install and manage the thousand packages at repo.continuum.io that are built, reviewed and maintained by Anaconda. Conda can be combined with continuous integration systems such as Travis CI and AppVeyor to provide frequent, automated testing of your code.

The conda package and environment manager is included in all versions of Anaconda and Miniconda Anaconda Repository. Conda is also included in Anaconda Enterprise, which provides on-site enterprise package and environment management for Python, R, Node.js, Java and other application stacks. Conda is also available on PyPI, although that approach may not be as up to date.

CHAPTER 4

SYSTEM DESIGN

4.1 System Design

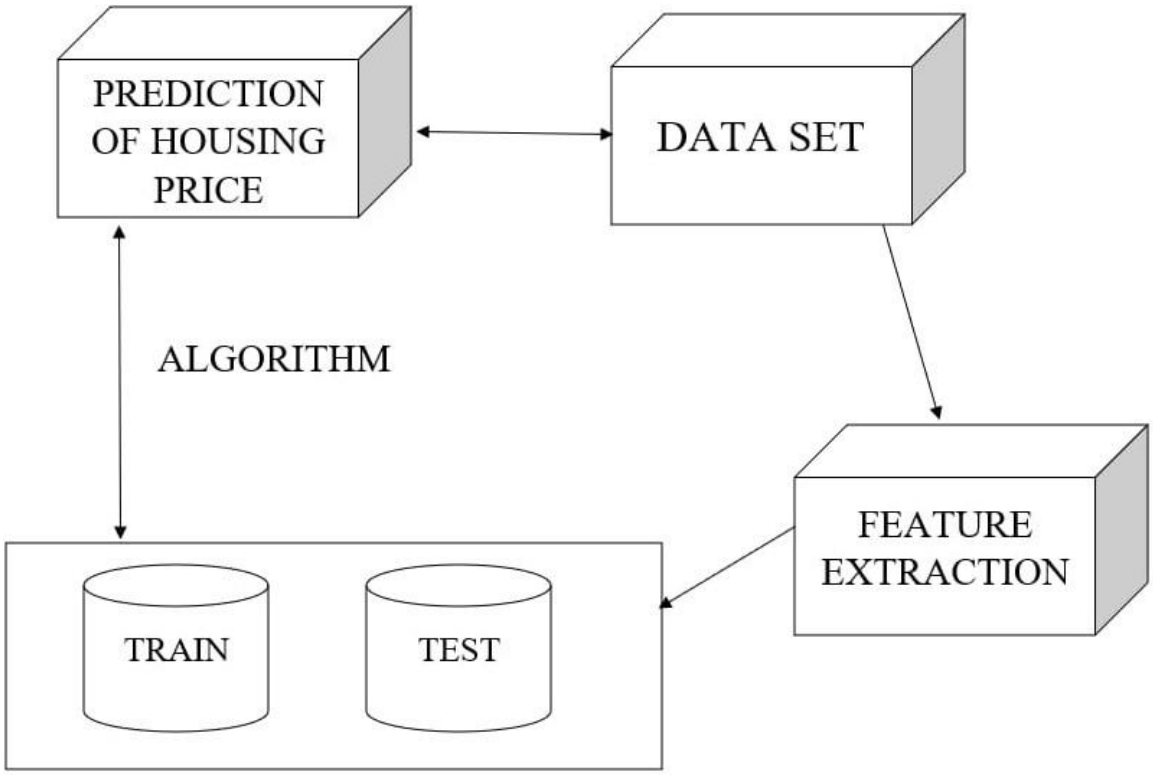
System Design Involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this tratuition. CASE tools take advantage of Mera modeling that is helpful only after the construction of the class diagram. In the FUSION method, some object-oriented approach likes Object Modeling Technique (OMT), Classes, and Responsibilities. Collaborators (CRC), etc., are used.

Objector used the term "agents" to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram The designer creates the class diagram.

4.2 System Architecture

System architecture diagram dataset is being collected and the collected data set is classified, formatted and sampled in the process called data preprocessing and after which the feature extraction process takes place and then the random forest algorithm is used to detect whether the data set in house price prediction or not. By using random forest algorithm we are splitting the datasets as trained and test datasets. The trained datasets will be more in amount than the test datasets. We will

compare the test datasets with the trained datasets and we will see whether it matches or not. If it matches it will display the result as zero.



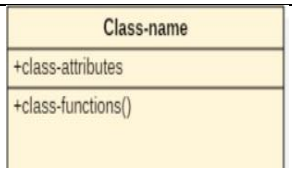
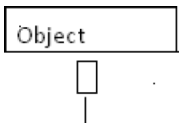
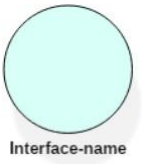
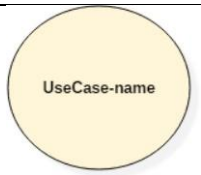
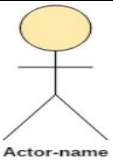
System architecture diagram from House Price Prediction

4.3 UML DIAGRAMS

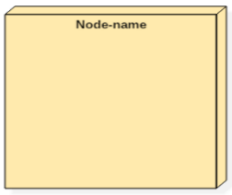
UML is a standard language for specifying, visualizing, and documenting of software systems and created by Object Management Group (OMG) in 1997. There are three important type of UML modeling are Structural model, Behavioral model, and Architecture model. To model a system the most important aspect is to capture the dynamic behavior which has some internal or external factors

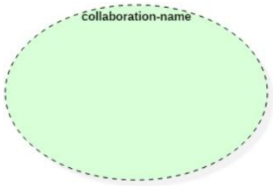
for making the interaction. These internal or external agents are known as actors. It consists of actors, use cases and their relationships

4.3.1 UML NOTATIONS

S.NO	SYMBOL NAME	NOTATION	DESCRIPTION
1	CLASS		A class is used to represent various objects. It is used to define the properties and operations of an object.
2	OBJECT		A real time entity.
3	INTERFACE		An interface is similar to a template without implementation details. A circle notation represents it.
4	USE-CASE		Use-cases are used to represent high-level functionalities and how the user will handle the system.
5	ACTOR		It is used inside use case diagrams. A user

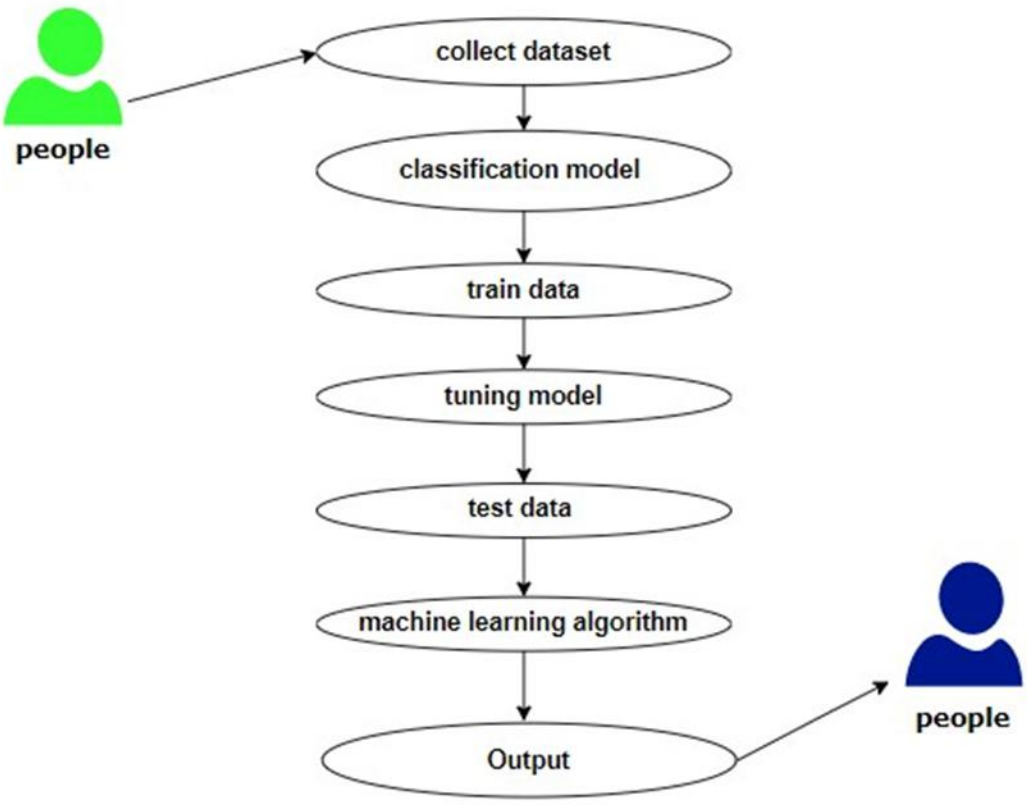
			is the best example of an actor.
6	INITIAL ACTIVITY		This shows the Starting point or first activity of flow.
7	ACTIVITY		Represented by a rectangle with rounded edges.
8	DECISION BOX		A logic where a decision is to be made.
9	FINAL ACTIVITY		The end of the Activity diagram is shown by a bull's eye symbol.
10	ASSOCIATION RELATIONSHIP		It is denoted as a dotted line with arrowheads on both sides. Both the sides contain an element which describes the relationship.

11	DEPENDENCY RELATIONSHIP		It is one of the most important notations of UML. It is denoted by a dotted line with an arrow at one side.
12	GENERALIZATION RELATIONSHIP		It is also called as a parent-child relationship. It is denoted by a straight line with a hollow arrowhead at one side.
13	REALIZATION RELATIONSHIP		Realization relationship is widely used while denoting **interfaces**. It is denoted as a dotted line with a hollow arrowhead at one end.
14	NODE		A node is used to describe the physical part of a system. A node can be used to represent a network, server, routers, etc.
15	COMPONENT		A component notation is used to represent a part of the system.

16	COLLABORATION		It is represented by a dotted ellipse with a name written inside it.

4.3.2 USE CASE DIAGRAM

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system. Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration. They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.



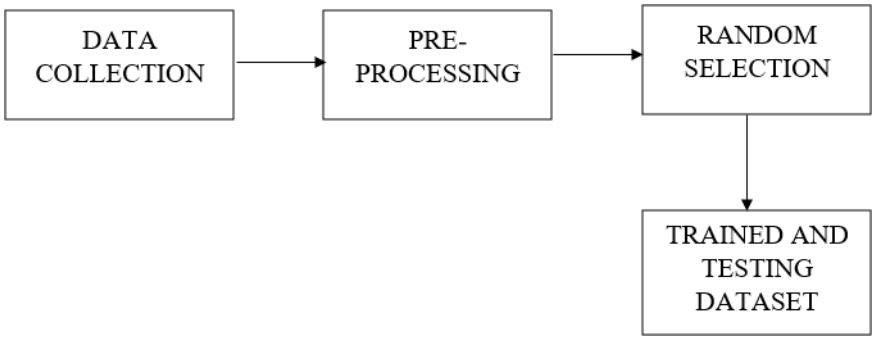
4.3.3 Data Flow Diagram

LEVELS IN DATA FLOW DIAGRAMS (DFD):

In Software engineering DFD(data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher-level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 3 levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0- LEVEL DFD:

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.



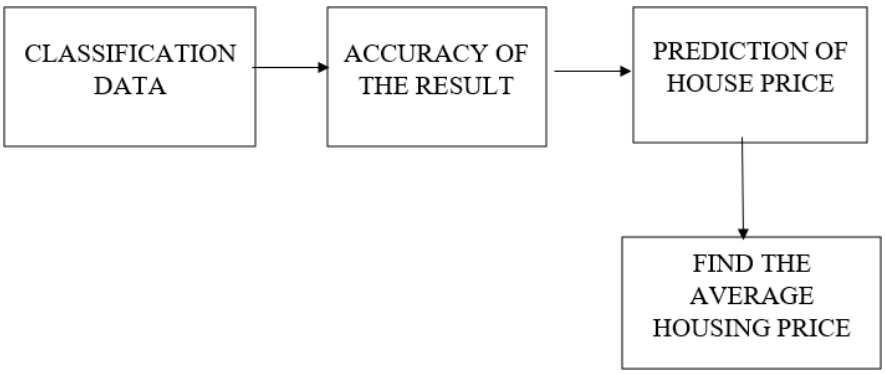
1- LEVEL DFD:

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 subprocesses.

2- LEVEL DFD:

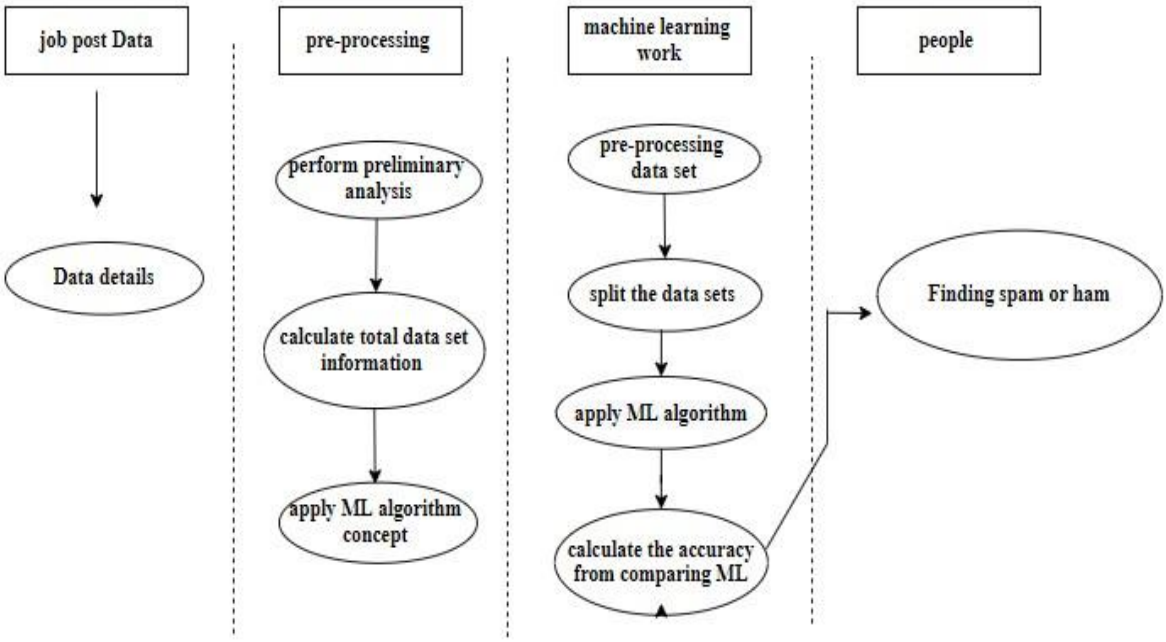
2-level DFD goes one step deeper into parts of 1-level DFD. It can

be used to plan or record the specific/necessary detail about the system's functioning.



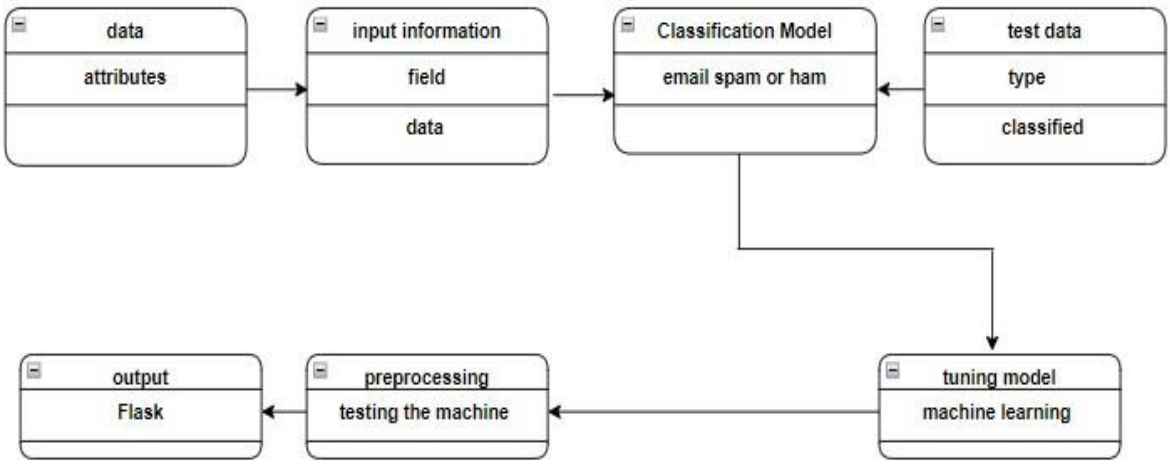
4.3.4 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. 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 behavior of the system.



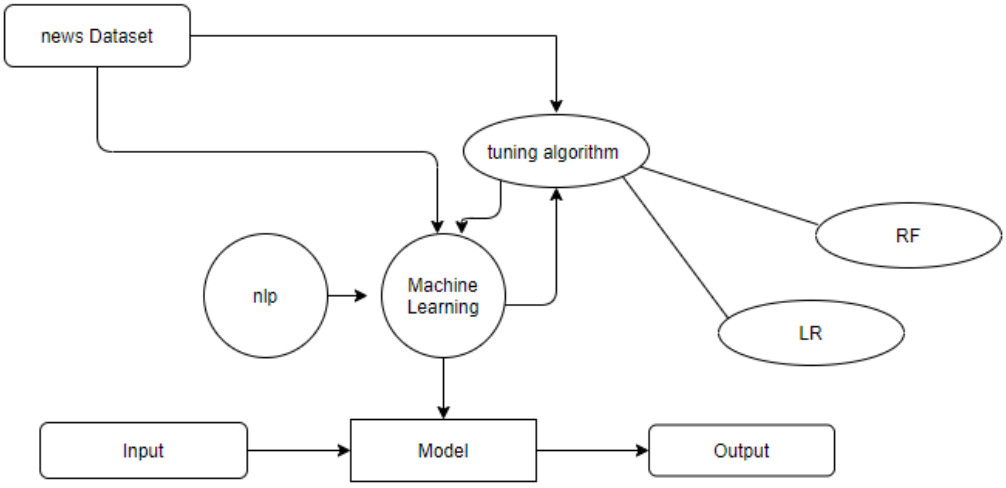
4.3.5 CLASS DIAGRAM

A Class diagram is a Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



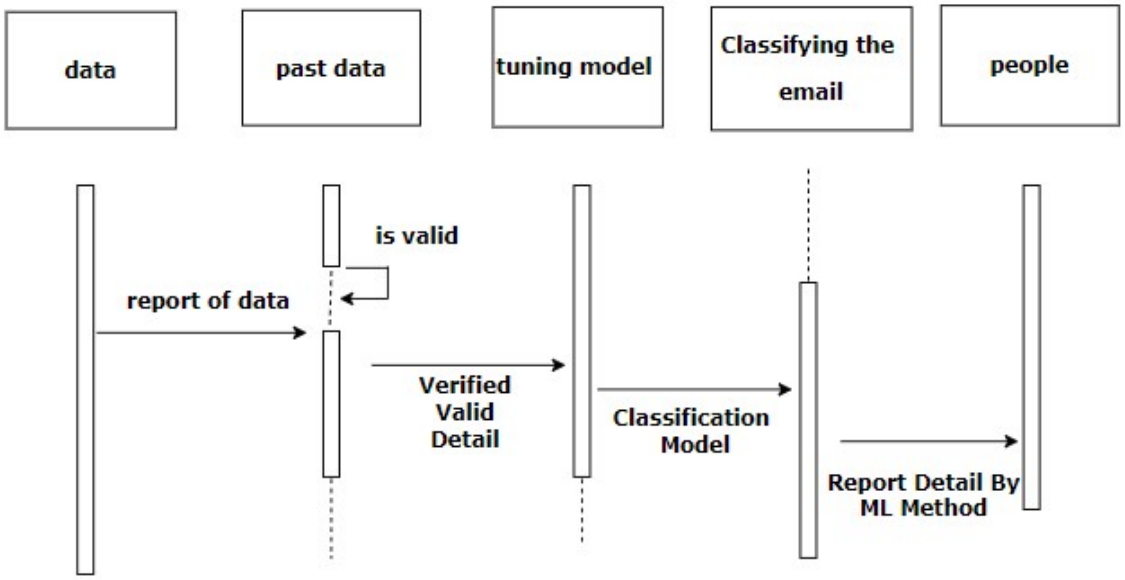
4.3.6 Entity Relationship Diagram (ERD)

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.


news Dataset
tuning algorithm
RF
nlp
Machine
Learning
LR
Input
Model
Output

4.3.7 SEQUENCE DIAGRAM

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.



CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 System Implementation

There are few modules which are involved in the implementation of this Project. The following are the modules of the project, which is planned in aid to Complete the project with respect to the proposed system, while overcoming Existing system and also providing the support for the future enhancement.

5.2 System Modules

A modular design reduces complexity, facilities change(a critical aspect of Software maintainability), and results in easier implementation by encouraging Parallel development of different part of system. Software with effective Modularity is easier to develop because function may be compartmentalized and Interface are simplified. Software architecture embodies modularity that is Software is divided into separately named and addressable components called Modules that are interfaced to satisfy problem requirements Our project contains the following modules :

1Data Pre-processing

2Data Analysis of Visualization

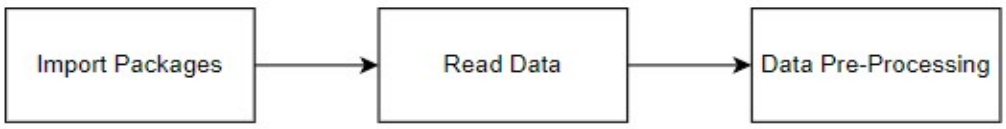
3Comparing Algorithm with prediction in the form of best accuracy result

5.3 Module Description

5.3.1 PREPROCESSING

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

MODULE DIAGRAM



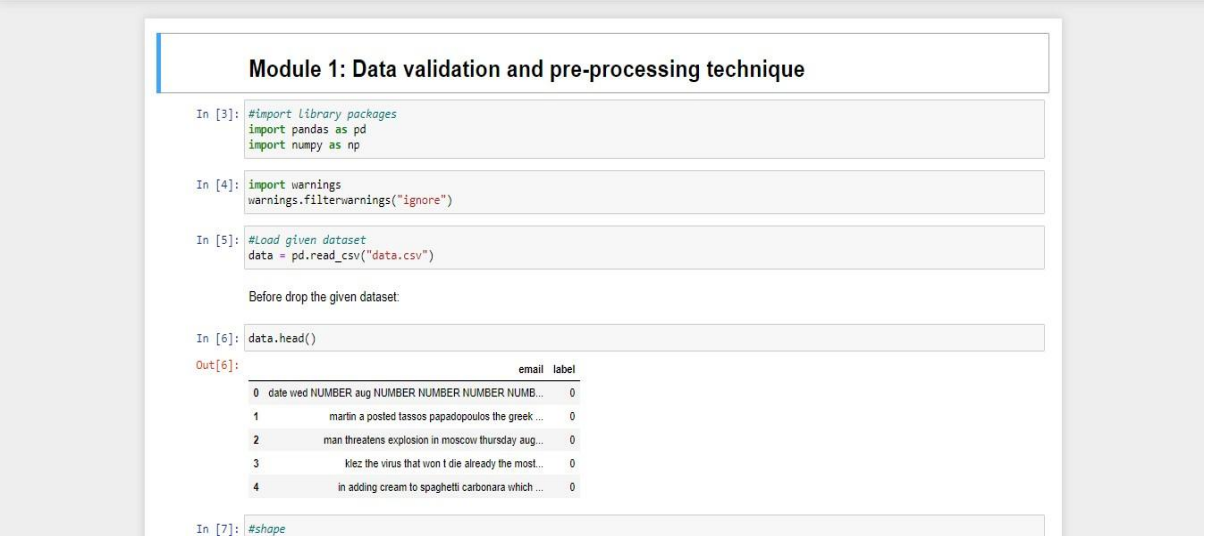
GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

5.3.2 DATA VALIDATION/ CLEANING/ PREPROCESSING PROCESS

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.



Module 1: Data validation and pre-processing technique

In [3]:
```
#import library packages
import pandas as pd
import numpy as np
```

In [4]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [5]:
```
#Load given dataset
data = pd.read_csv("data.csv")
```

Before drop the given dataset:

In [6]:
```
data.head()
```

Out[6]:

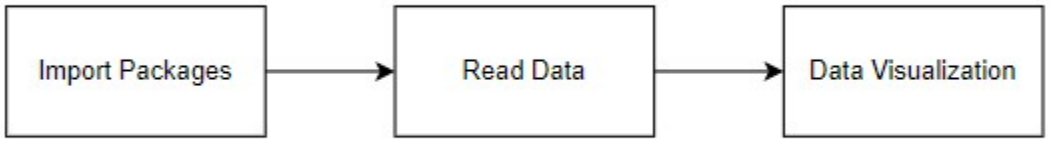
	email	label
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0
1	martin a posted tassos papadopoulos the greek ...	0
2	man threatens explosion in moscow thursday aug...	0
3	klez the virus that won t die already the most...	0
4	in adding cream to spaghetti carbonara which ...	0

In [7]:
```
#shape
```

5.3.3 EXPLORATION DATA ANALYSIS OF VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

MODULE DIAGRAM

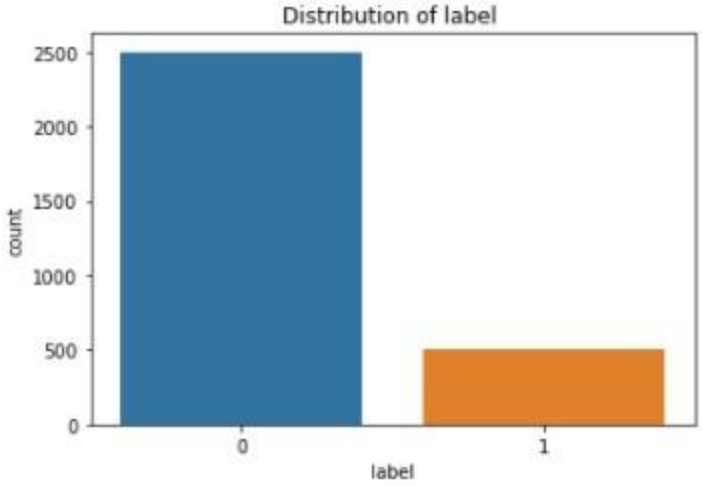


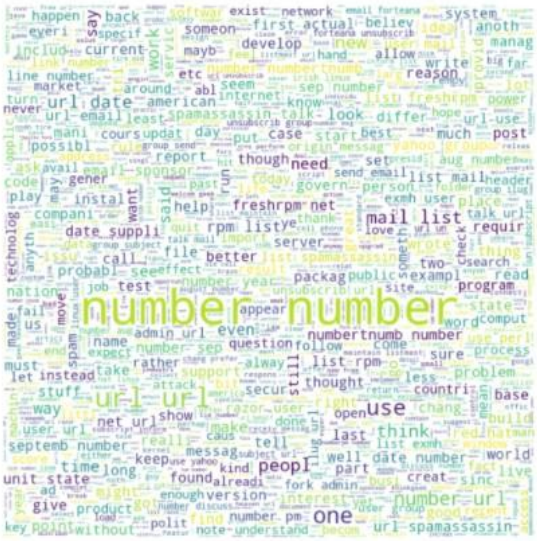
GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

```
Text(0.5, 1.0, 'Distribution of label ')
```





5.3.4 COMPUTER ALGORITHM WITH PREDICTION IN THE FORM OF BEST ACUURACY RESULT

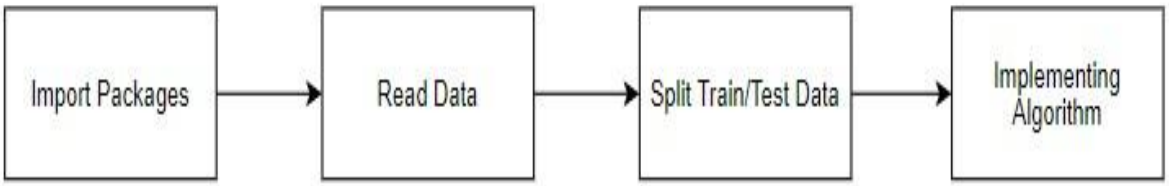
Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable

using a given set of independent variables.

- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

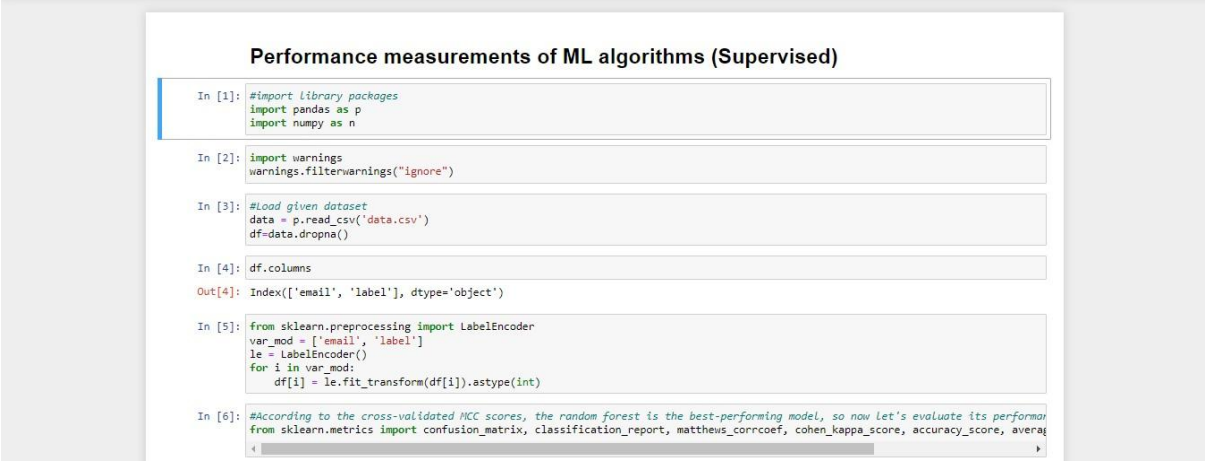
MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy



Performance measurements of ML algorithms (Supervised)

```
In [1]: #import library packages
        import pandas as p
        import numpy as n

In [2]: import warnings
        warnings.filterwarnings("ignore")

In [3]: #Load given dataset
        data = p.read_csv('data.csv')
        df=data.dropna()

In [4]: df.columns
Out[4]: Index(['email', 'label'], dtype='object')

In [5]: from sklearn.preprocessing import LabelEncoder
        var_mod = ['email', 'label']
        le = LabelEncoder()
        for i in var_mod:
            df[i] = le.fit_transform(df[i]).astype(int)

In [6]: #According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performa
        from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, averag
```

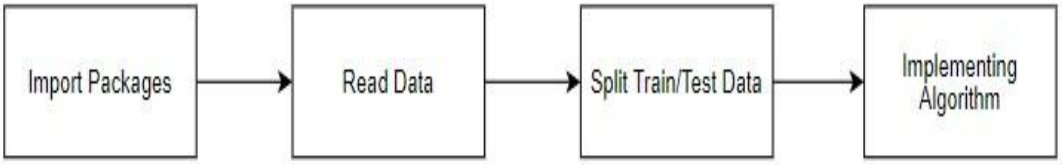
```
Accuracy result of Random Forest Classifier  is: 88.88888888888889

Classification report of Random Forest Classifier : Results:

              precision    recall  f1-score   support

           0       0.95      0.92      0.93       750
           1       0.65      0.74      0.69       150

    accuracy                           0.89       900
   macro avg       0.80      0.83      0.81       900
weighted avg       0.90      0.89      0.89       900

Confusion Matrix result of Random Forest Classifier : is:
 [[689  61]
 [ 39 111]]

Sensitivity :  0.9186666666666666

Specificity :  0.74
```

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

CHAPTER 6

SYSTEM TESTING

6.1 Testing Objectives

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides away to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 Categories of Software Testing

- Black Box Testing
- White Box Testing

Black Box Testing

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

White Box Testing

White Box Testing is the testing of a software solution's internal coding and infra structure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White bote

sting is also known as Clear Box testing, Open Box test in Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing.

White Box Testing Techniques:

- **Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage** - This technique is running a series of tests to ensure all branches are tested at least once.

Black Box Testing Techniques:

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or It is useful in reducing the number of test cases. It is mostly suitable for the systems where input is within certain ranges.
- **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is unique combination in each column.

6.3 Types of Testing

1. Unit Testing

2. Functional Testing

3. Performance Testing

4. Integration Testing

5. Objective

6. Integration Testing

7. Validation Testing

8. System Testing

9. Structure Testing

10. Output Testing

11. User Acceptance Testing

Unit testing

- Unit testing, also known as Module Testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.
- Unit Test comprises of the set of tests performed by an individual programmer before integration of the unit into the system.
- Unit test focuses on the smallest unit of software design- the software component or module.
- Unit test is white box oriented and the step can be conducted in parallel for multiple components
- Using component level design, important control paths are tested to uncover errors within the boundary of the module.

Functional Testing:

- Functional test cases involve exercising the code with normal input values for which the expected results are known, as well as the boundary values

- The objective is to take unit-tested modules and build a program structure that has been dictated by design.

Performance Testing:

- Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

Integration Testing:

- It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.
- It takes the unit tested modules and builds a program structure.
- All the modules are combined and tested as a whole.
- Integration of all the components to form the entire system and a over all testing is executed.

Validation Testing:

- Validation test succeeds when the software functions in a manner that can be reasonably expected by the client.
- Software validation is achieved through a series of black box testing which confirms to the requirements.
- Black box testing is conducted at the software interface.
- The test is designed to uncover interface errors, is also used to demonstrate that software functions are operational, input is properly accepted, output are produced and that the integrity of external information is maintained.

System Testing:

- System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specific requirement.
- It also tests to find the discrepancies between the system and its original objective, current specifications and system documentation.

Structure Testing

- It is concerned with exercising the internal logic of a program and traversing particular execution paths.
- This testing requires knowledge of the code so it is mostly done by the developers
- It is more concerned with the how system does it rather than the functionality of the system
- It is complementary to functional testing.

Output Testing

- Output of test cases compared with the expected results created during design of test cases.
- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs,
- The output comes out as the specified requirements as the user's hard copy.

User Acceptance Testing

- Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
- The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.

6.4 Test Cases

6.5 Test Case 1:

INPUT : Launching Anaconda Navigator

OUTPUT : Launch Successful

Test Case 2:

INPUT : Test data for Pre-Processing

OUTPUT : Pre-processed data

Test Case 3:

INPUT : Test data for data exploration

OUTPUT : Characteristics of test data

Test Case 4:

INPUT : Random Forest Algorithm

OUTPUT : Prediction for House Price.

Test Case 5:

INPUT : Change of Constrains and Predict the Different cases

OUTPUT : Prediction success full completes

Test Case 6:

INPUT : Predicted Data after processed

OUTPUT : Statical Analysis of the Input

Test Case 7:

INPUT : Statics Data

OUTPUT : Verification and Manipulation of Graph .

CHAPTER 7

EXPERIMENTAL RESULTS AND DISCUSSIONS

The below images show the result of the module implementation for Prediction of Ham or Spam using Machine Learning using Random Forest Algorithm.

```
In [1]: import numpy as np
        import pandas as pd

In [2]: data=pd.read_csv("data.csv")

In [3]: data.head()

Out[3]:
```

	email	label
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0
1	martin a posted tassos papadopoulos the greek ...	0
2	man threatens explosion in moscow thursday aug...	0
3	klez the virus that won t die already the most...	0
4	in adding cream to spaghetti carbonara which ...	0

```
In [21]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test,pre)*100)

         98.7878787878788

In [22]: input_word=input("ENTER THE SENTENCE:")

         ENTER THE SENTENCE:check the available reports you would like to receive keep on top of the latest news get great special deals
         now it is complimentary it costs nothing you can quit anytime financial stocks loans mortgage financial news stock market gover
         nment politics discussions credit cards mortgage refinancing loans health fitness holidays travel online pharmacies discounts s
         pecials general health fitness tips secrets alternative medicine health care under booked vacations special travel discounts ma
         ture intrests dating general interest adultwebmasters general adultwebmasters unrestricted sites adultwebmasters content buyers
         cassino s online gamblinng dating services personal ads send me NUMBER uncensored pictures daily this mail is never sent unsoli
         cited got it by error hyperlink click here to be removed from our subscribers list ndtxcpfjspwwtrkaxnxg

In [23]: data = cv.transform([input_word]).toarray()
         print(clf.predict(data))

         ['SPAM']
```

CHAPTER 8

CONCLUSION AND FUTURE ENHANCAMENT

Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This application can help to find the Prediction of email Spam or Ham.

Future Enhancament

However the decision making ability of Random forest algorithm is quite simple so it can only work for large number of data set . We try to remodify the way of decision making ability for small number of the data sets without reducing the accuracy of prediction. We try to reduce the training period of large data set by split tree selection to reduce the time taken by the algorithm. The Email Spam or Ham prediction to connect with cloud and to optimize the work in Artificial Intelligence environment.

CHAPTER 9

REFERENCE

1. B. Pang and L. Lee, "Opinion mining and sentiment analysis", Found. Trends Inf. Retr., vol. 2, no. 2, pp. 1-135, 2008.
2. B. Liu, "Sentiment analysis and opinion mining", Synthesis Lect. Hum. Lang. Technol., vol. 5, no. 1, pp. 1- 167, 2012
3. E. Fitzpatrick, J. Bachenko and T. Fornaciari, Automatic Detection of Verbal Deception, San Rafael, CA, USA:Morgan & Claypool, 2015.
4. [4] N. Jindal and B. Liu, "Analyzing and detecting review spam", Proc. IEEE Int. Conf. Data Mining, pp. 547- 552, Oct. 2007.
5. Tretyakov, K. (2004, May). Machine learning techniques in spam filtering. In Data Mining Problemoriented Seminar, MTAT (Vol.3, No. 177, pp. 60-79) .
6. N. Jindal and B. Liu, "Opinion spam and analysis", Proc. Int. Conf. Web Search Data Mining, pp. 219-230, 2008.
7. N. Jindal, B. Liu and E. P. Lim, "Finding unusual review patterns using unexpected rules", Proc. ACM Int. Conf. Inf. Knowl. Manage., pp. 1549-1552, 2010.
8. G. Wu, D. Greene, B. Smyth and P. Cunningham, "Distortion as a validation criterion in the identification of suspicious reviews", Proc. Workshop Social Media Anal., pp. 10-13, 2010.

CHAPTER 10

APPENDICES

Appendix A- Sample Coding

SAMPLE CODE:

Pre-Processing

```
#import library packages
import pandas as pd
import numpy as np
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
#Load given dataset
data = pd.read_csv("data.csv")
```

Before drop the given dataset:

In [ ]:

```
data.head()
```

In [ ]:

```
#shape
data.shape
```

After drop the given dataset:

In [ ]:

```
df = data.dropna()
```

In [ ]:

```
df.head()
```

In [ ]:

```
#shape
df.shape
```

In [ ]:

```
#columns
```

```
df.columns
```

In [ ]:

```
#To describe the dataframe
df.describe()
```

In [ ]:

```
#Checking datatype and information about dataset
df.info()
Checking duplicate values of dataframe
```

In [ ]:

```
#Checking for duplicate data
df.duplicated()
```

In [ ]:

```
sum(df.duplicated())
```

In [ ]:

```
#Checking sum of missing values
df.isnull().sum()
```

In [ ]:

```
df.label.unique()
```

In [ ]:

```
df.email.unique()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
var=['SFH', 'popUpWidnow', 'SSLfinal_State', 'Request_URL', 'URL_of_Anchor',
     'web_traffic', 'URL_Length', 'age_of_domain', 'having_IP_Address',
     'Result']
le=LabelEncoder()
for i in var:
   df[i]=le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.head()
```

Module – 2

Visualization

```
#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
#Load given dataset
data = p.read_csv('data.csv')
df=data.dropna()
```

In [ ]:

```
df
```

In [ ]:

```
df.columns
```

Spliting Train/Test:

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='label', axis=1)
#Response variable
y = df.loc[:,'label']
```

In [ ]:

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
```

In [ ]:

```
df.groupby('label').describe()
```

In [ ]:

```
#plotting graph for distribution
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = "label", data = df)
df.loc[:, 'label'].value_counts()
plt.title('Distribution of label ')
```

In [ ]:

```
df['label'].unique()
```

Training model:

In [ ]:

```
#!pip install nltk
```

In [ ]:

```
import nltk
```

```
nltk.download('stopwords')
```

In [ ]:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
import string
# remove whitespaces
df['email']=df['email'].str.strip()
# lowercase the text
df['email'] = df['email'].str.lower()
#remove punctuation
punc = string.punctuation
table = str.maketrans('','',punc)
df['email']=df['email'].apply(lambda x: x.translate(table))
# tokenizing each message
df['word_tokens']=df.apply(lambda x: x['email'].split(' '),axis=1)
# removing stopwords
df['cleaned_text'] = df.apply(lambda x: [word for word in x['word_tokens'] if word not in
stopwords.words('english')],axis=1)
# stemming
ps = PorterStemmer()
df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned_text']],axis=1)
# remove single letter words
df['final_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if len(word)>1]),axis=1)
```

In [ ]:

```
# divide the set in training and test
from sklearn.model_selection import train_test_split
X,X_test,y,y_test = train_test_split(df.loc[:,'email':],df['label'],test_size=0.2)
```

In [ ]:

```
# Now we'll create a vocabulary for the training set with word count
from collections import defaultdict
vocab=defaultdict(int)
for text in X['final_text'].values:
    for elem in text.split(' '):
        vocab[elem]+=1
```

In [ ]:

```
#!pip install wordcloud
```

In [ ]:

```
from wordcloud import WordCloud
```

```
ham=' '.join(X.loc[y==0,'final_text'].values)
ham_text = WordCloud(background_color='white',max_words=2000,width = 800, height =
800).generate(ham)

spam=' '.join(X.loc[y==1,'final_text'].values)
spam_text = WordCloud(background_color='black',max_words=2000,width = 800, height =
800).generate(spam)




plt.figure(figsize=[30,50])

plt.subplot(1,3,1)
plt.imshow(ham_text,interpolation='bilinear')
plt.title('')
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(spam_text, interpolation='bilinear')
plt.axis('off')
plt.title('')
```

Module – 3

```
#import library packages
import pandas as p
import numpy as n
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
#Load given dataset
data = p.read_csv('data.csv')
df=data.dropna()
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod = ['email', 'label']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

In [ ]:

```
X = df.drop(labels='label', axis=1)
#Response variable
y = df.loc[:,'label']
```

In [ ]:

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
RandomForestClassifier :
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()

rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Random Forest Classifier is:', x)
print("")

print("")
print('Classification report of Random Forest Classifier : Results:')
print("")

print(classification_report(y_test,predictR))
xd = (accuracy_score(y_test,predictR)*100)


cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest Classifier : is:\n', confusion_matrix(y_test,predictR))
print("")
```

```
sensitivity1  = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity  : ', sensitivity1  )
print("")
specificity1  = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity  : ', specificity1)
```

Logistic Regression:

In [ ]:

```
from sklearn.linear_model import LogisticRegression
logR= LogisticRegression()
logR.fit(X_train,y_train)
predictR = logR.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Logistic Regression is:', x)
print("")

print("")
print('Classification report of Logistic Regression : Results:')
print("")

print(classification_report(y_test,predictR))
xl = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression : is:\n', confusion_matrix(y_test,predictR))
print("")

sensitivity1  = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity  : ', sensitivity1  )
print("")
specificity1  = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity  : ', specificity1)
```

In [ ]:

```
def graph():
    import matplotlib.pyplot as plt
    data=[xd,xl]
    alg="DT","LR"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("black","gray"))
    plt.legend(b,data,fontsize=12)
    plt.savefig('comp.png')
```

In [ ]:

```
graph()
```

In [ ]:

```
import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='comp.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()
```

Module-4

```
import numpy as np
import pandas as pd
```

In [ ]:

```
data=pd.read_csv("data.csv")
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.shape
```

In [ ]:

```
data.columns
```

In [ ]:

```
data1=data.dropna()
data1.shape
```

In [ ]:

```
data1["label"] = data1["label"].map({0: "HAM",
                                     1: "SPAM"})
print(data1.head())
```

In [ ]:

```
data1.label.unique()
```

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
sns.countplot(x = "label", data=data1)
data1.loc[:, 'label'].value_counts()
plt.title('Distribution of label')
```

In [ ]:

```
data1.label.value_counts()
```

In [ ]:

```
import re
import string
import nltk
from nltk.corpus import stopwords
stopword=set(stopwords.words('english'))
stemmer = nltk.SnowballStemmer("english")
```

In [ ]:

```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["email"] = data["email"].apply(clean)
```

In [ ]:

```
x = np.array(data1["email"])
y = np.array(data1["label"])
```

In [ ]:

```
x
```

In [ ]:

```
y
```

In [ ]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

In [ ]:

```
cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train,y_train)
```

In [ ]:

```
pre=clf.predict(X_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,pre)*100)
```

In [ ]:

```
input_word=input("ENTER THE SENTENCE:")
```

In [ ]:

```
data = cv.transform([input_word]).toarray()
print(clf.predict(data))
```

Module-5

```
import numpy as np
import pandas as pd
```

In [ ]:

```
data=pd.read_csv("data.csv")
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.shape
```

In [ ]:

```
data.columns
```

In [ ]:

```
data1=data.dropna()
data1.shape
```

In [ ]:

```
data1["label"] = data1["label"].map({0: "HAM",
                                     1: "SPAM"})
print(data.head())
```

In [ ]:

```
data1.label.unique()
```

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
sns.countplot(x = "label", data=data1)
data1.loc[:, 'label'].value_counts()
plt.title('Distribution of label')
```

In [ ]:

```
data1.label.value_counts()
```

In [ ]:

```
import re
import string
import nltk
from nltk.corpus import stopwords
stopword=set(stopwords.words('english'))
stemmer = nltk.SnowballStemmer("english")
```

In [ ]:

```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["email"] = data["email"].apply(clean)
```

In [ ]:

```
x = np.array(data["email"])
```

```
y = np.array(data["label"])
```

In [ ]:

```
x
```

In [ ]:

```
y
```

In [ ]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

In [ ]:

```
cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train,y_train)
```

In [ ]:

```
pre=clf.predict(X_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,pre)*100)
```

In [ ]:

```
input_word=input("ENTER THE SENTENCE:")
```

In [ ]:

```
data = cv.transform([input_word]).toarray()
print(clf.predict(data))
```

\

Appendix B - Base Paper

Appendix C - Published Paper