**Machine learning for email spam filtering: review, approaches and open research problems**

**1. INTRODUCTION**

In recent times, unwanted commercial bulk emails called spam has become a huge problem on the internet. The person sending the spam messages is referred to as the spammer. Such a person gathers email addresses from different websites, chatrooms, and viruses. Spam prevents the user from making full and good use of time, storage capacity and network bandwidth. The huge volume of spam mails flowing through the computer networks have destructive effects on the memory space of email servers, communication bandwidth, CPU power and user time. The menace of spam email is on the increase on yearly basis and is responsible for over 77% of the whole global email traffic. Users who receive spam emails that they did not request find it very irritating. It is also resulted to untold financial loss to many users who have fallen victim of internet scams and other fraudulent practices of spammers who send emails pretending to be from reputable companies with the intention to persuade individuals to disclose sensitive personal information like passwords, Bank Verification Number (BVN) and credit card numbers.

According to report from Kaspersky lab, in 2015, the volume of spam emails being sent reduced to a 12-year low. Spam email volume fell below 50% for the first time since 2003. In June 2015, the volume of spam emails went down to 49.7% and in July 2015 the figures was further reduced to 46.4% according to anti-virus software developer Symantec. This decline was attributed to reduction in the number of major botnets responsible for sending spam emails in billions. Malicious spam email volume was reported to be constant in 2015. The figure of spam mails detected by Kaspersky Lab in 2015 was between 3 million and 6 million. Conversely, as the year was about to end, spam email volume escalated. Further report from Kaspersky Lab indicated that spam email messages having pernicious attachments such as malware, ransomware, malicious macros, and JavaScript started to increase in December 2015. That drift was sustained in 2016 and by March of that year spam email volume had quadrupled with respect to that witnessed in 2015. In March 2016, the volume of spam emails discovered by Kaspersky Lab is 22,890,956. By that time the volume of spam emails had skyrocketed to an average of 56.92% for the first quarter of 2016. Latest statistics shows that spam messages accounted for 56.87% of e-mail traffic worldwide and the most familiar types of spam emails were healthcare and dating spam. Spam results into unproductive use of resources on Simple Mail Transfer Protocol (SMTP) servers since they have to process a substantial volume of unsolicited emails. The volume of spam emails containing malware and other malicious codes between the fourth quarter of 2016 and first quarter of 2018 is depicted in Fig. 1 below. To effectively handle the threat posed by email spams, leading email providers such as Gmail, Yahoo mail and Outlook have employed the combination of different machine learning (ML) techniques such as Neural Networks in its spam filters. These ML techniques have the capacity to learn and identify spam mails and phishing messages by analyzing loads of such messages throughout a vast collection of computers. Since machine learning have the capacity to adapt to varying conditions, Gmail and Yahoo mail spam filters do more than just checking junk emails using pre-existing rules. They generate new rules themselves based on what they have learnt as they continue in their spam filtering operation. The machine learning model used by Google have now advanced to the point that it can detect and filter out spam and phishing emails with about 99.9 percent accuracy. The implication of this is that one out of a thousand messages succeed in evading their email spam filter. Statistics from Google revealed that between 50-70 percent of emails that Gmail receives are unsolicited mail. Google's detection models have also incorporated tools called Google Safe Browsing for identifying websites that have malicious URLs. The phishing-detection performance of Google have been enhanced by introduction of a system that delay the delivery of some Gmail messages for a while to carry out additional comprehensive scrutiny of the phishing messages since they are easier to detect when they are analyzed collectively. The purpose of delaying the delivery of some of these suspicious emails is to conduct a deeper examination while more messages arrives in due course of time and the algorithms are updated in real time. Only about 0.05 percent of emails are affected by this deliberate delay. Though there are several email spam filtering methods in existence, the state-of-the-art approaches are discussed in this paper. We explained below the different categories of spam filtering techniques that have been widely applied to overcome the problem of email spam. Content Based Filtering Technique: Content based filtering is usually used to create automatic filtering rules and to classify emails using machine learning approaches, such as Naïve Bayesian classification, Support Vector Machine, K Nearest Neighbor, Neural Networks. This method normally analyses words, the occurrence, and distributions of words and phrases in the content of emails and used then use generated rules to filter the incoming email spams. Case Base Spam Filtering Method: Case base or sample base filtering is one of the popular spam filtering methods. Firstly, all emails both non-spam and spam emails are extracted from each user's email using collection model. Subsequently, pre-processing steps are carried out to transform the email using client interface, feature extraction, and selection, grouping of email data, and evaluating the process. The data is then classified into two vector sets. Lastly, the machine learning algorithm is used to train datasets and test them to decide whether the incoming mails are spam or non-spam. Heuristic or Rule Based Spam Filtering Technique: This approach uses already created rules or heuristics to assess a huge number of patterns which are usually regular expressions against a chosen message. Several similar patterns increase the score of a message. In contrast, it deducts from the score if any of the patterns did not correspond. Any message's score that surpasses a specific threshold is filtered as spam; else it is counted as valid. While some ranking rules do not change over time, other rules require constant updating to be able to cope effectively with the menace of spammers who continuously introduce new spam messages that can easily escape without been noticed from email filters. A good example of a rule based spam filter is SpamAssassin. Previous Likeness Based Spam Filtering Technique: This approach uses memory-based, or instance-based, machine learning methods to classify incoming emails based to their resemblance to stored examples (e.g. training emails). The attributes of the email are used to create a multi-dimensional space vector, which is used to plot new instances as points. The new instances are afterward allocated to the most popular class of its K-closest training instances. This approach uses the k-nearest neighbor (kNN) for filtering spam emails. Adaptive Spam Filtering Technique: The method detects and filters spam by grouping them into different classes. It divides an email corpus into various groups, each group has an emblematic text. A comparison is made between each incoming email and each group, and a percentage of similarity is produced to decide the probable group the email belongs. Many researchers and academicians have proposed different email spam classification techniques which have been successfully used to classify data into groups. These methods include probabilistic, decision tree, artificial immune system, support vector machine (SVM), artificial neural networks (ANN), and case-based technique. It have been shown in literature that it is possible to use these classification methods for spam mail filtering by using content-based filtering technique that will identify certain features (normally keywords frequently utilised in spam emails). The rate at which these features appear in emails ascertain the probabilities for each characteristic in the email, after which it is measured against the threshold value. Email messages that exceed the threshold value are classified as spam. ANN is a non-linear model that seeks to imitate the functions of biological neural networks. It is made up of simple processing components named neurons and carries out its computational operations by processing information. Several research work have employed neural network to classify unwanted emails as spam by applying content-based filtering. These techniques decide the properties by either computing the rate of occurrence of keywords or patterns in the email messages. Literatures show that Neural Network algorithms that are utilised in email filtering attain moderate classification performance. Some of the most popular spam email classification algorithms are Multilayer Perceptron Neural Networks (MLPNNs) and Radial Base Function Neural Networks (RBFNN). Researchers used MLPNN as a classifier for spam filtering but not many of them used RBFNN for classification. Support Vector Machines (SVM) has proved over the years to be one of the most powerful and efficient state-of-the-art classification techniques for solving the email spam problem. They are supervised learning models that analyze data and identify patterns used for categorisation and exploring the relationship between variables of interest. SVM algorithms are very potent for the identification of patterns and classifying them into a specific class or group. They can be easily trained and according to some researchers, they outperform many of the popular email spam classification methods. This is because during training, SVM use data from email corpus. However, for high dimension data, the strength and efficacy of SVM diminish over time due to computational complexities of the processed data. According, SVM is a good classifier due to its sparse data format and satisfactory recall and precision value. SVM has high classification accuracy. Moreover, SVM is considered a notable example of “kernel methods”, which is one of the central areas of machine learning. Decision tree is another machine learning algorithm that has been successfully applied to email spam filtering. Decision trees (DT) need comparatively minute effort from users during training of datasets. DT completely perform variable analysis or feature selection of the email corpus data training. The performance of a tree does not depend on the relationships among parameters. A great benefit of decision tree is its capacity to assign unambiguous values to problems, decisions, and results of every decision. This decreases vagueness in decision-making. Another huge advantage of the decision tree compared to other machine learning techniques is the fact that it makes open all the likely options and follows each option to its end in one view, giving room for straightforward evaluation among the different nodes of the tree. Despite the numerous advantages of Decision tree, it still has some drawbacks which are: unless there is appropriate pruning, controlling tree growth can be very difficult. Decision trees are a nonparametric machine learning algorithm that is incredibly adaptable and vulnerable to overfitting of training data. This makes them to some extent poor classifiers and limit their classification accuracy. The different types of Decision trees that have been applied to email spam filtering are NBTree Classifier, C4.5/J48 Decision Tree Algorithm and Logistic Model Tree Induction (LMT). Naïve Bayes is another wonderful machine learning algorithm that has been applied in email spam filtering. A Naive Bayes (NB) classifier simply apply Bayes' theorem on the context classification of each email, with a strong assumption that the words included in the email are independent of each other. NB is desirable for email spam filtering because of its simplicity, ease of implementation and quick convergence compared to conditional models such as logistic regression. It needs fewer training data. It is very scalable. No bottleneck is created by increase in the number of predictors and discrete unit of information. NB can be used to solve both classification problems involving two or more classes. It can be used to make forecasting that is subject to or involving probability variation. They can effectively manage continuous and discrete data. NB algorithms are not susceptible to irrelevant features. Naıve Bayes algorithm is predominantly famous in business-related and open-source spam filters. This is because apart from the advantages listed above, NB needs little training time or speedy assessment to detect and filter email spam. NB filters need training that can be offered by the earlier set of non-spam and spam messages. It keeps the record of the changes that take place in each word that occurs in legitimate, illegitimate messages, and in both. NB can be applied to spam messages in diverse datasets having different features and attribute. Stochastic optimization techniques such as evolutionary algorithms (EAs) have also been applied to spam filtering. This is because they do not have any sophisticated mathematical computation. Also, they can handle the solutions generated, they seek to recognise individuals that have the optimal solutions for the problem. Several earlier works exist that integrated Genetic Algorithms with Neural Networks to enhance the performance of neural network algorithms. A related approach of evolutionary computation methods such as Genetic Algorithms (GAs) is Particle Swarm Optimization (PSO), which is a technique that can be used for optimizing many continuous nonlinear functions and classification techniques. PSO is inspired by the social behaviour of animals such as flocks of bird and shoal of fishes. It has been applied in many areas of human endeavour such as neural network, swarm robotics, telecommunications, signal processing, data mining, and several other applications. PSO algorithm operates on a population (swarm) of particles, with the characteristic of no crossover and mutation calculation as found in genetic algorithm. Every particle have a position and velocity. Each of the particle is a potential solution in the swarm. This makes it easy to implement. What appears to be the most efficient spam filtering approach now is the automatic email filtering which have successfully been used for frustrating the malicious intentions of spammers. Some years back, the largest part of the spam email can be efficiently addressed by stopping emails originating from specified addresses or remove messages with specific subject lines. More deceitful and sophisticated techniques such as utilising arbitrary sender addresses and/or inserting haphazard characters to the beginning or the end of the message subject line are now been used by spammers to surmount the hurdle posed by the filtering methods. Owing to the fact that a good number of real-world filters make use of the amalgamation of ML and application-specific knowledge in the form of hand-coded rules, comprehending the revolutionising attributes of spam is also germane, and many studies have been done on this subject. However, in spite of the increasing research efforts on spam filtering, the growth of spam emails is still on alarming rate. This is evident with spammers devising more sophisticated methods for dodging detection, a very good example are emails with stego images (i.e. images with information hidden inside). The two common approaches used for filtering spam mails are knowledge engineering and machine learning. Emails are classified as either spam or ham using a set of rules in knowledge engineering. The person using the filter, or the software company that stipulates a specific rule-based spam-filtering tool must create a set of rules. Using this method does not guarantee efficient result since there is need to continually update the rules. This can lead to time wastage and it is not suitable especially for naive users. Machine learning approach have proved to be more efficient than knowledge engineering approach. No rule is required to be specified, rather a set of training samples which are pre-classified email messages are provided. A particular machine learning algorithm is then used to learn the classification rules from these email messages. Several studies have been carried out on machine learning techniques and many of these algorithms are being applied in the field of email spam filtering. Examples of such algorithms include Deep Learning, Naïve Bayes, Support Vector Machines, Neural Networks, K-Nearest Neighbour, Rough sets, and Random Forests.

**1.1 Objective of the Project**

In recent times, unwanted commercial bulk emails called spam has become a huge problem on the internet. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches. Our review covers survey of the important concepts, attempts, efficiency, and the research trend in spam filtering. The preliminary discussion in the study background examines the applications of machine learning techniques to the email spam filtering process of the leading internet service providers (ISPs) like Gmail, Yahoo and Outlook emails spam filters. Discussion on general email spam filtering process, and the various efforts by different researchers in combating spam through the use machine learning techniques was done. Our review compares the strengths and drawbacks of existing machine learning approaches and the open research problems in spam filtering.

**2. LITERATURE SURVEY**

**Measuring, Characterizing, and Avoiding Spam Traffic Costs**

Spam messages propagate malware, disseminate phishing exploits, and advertise illegal products. Those messages generate costs for users and network operators, but it's difficult to measure the costs associated with spam traffic and determine who actually pays for it. Here, the authors provide a method to quantify the transit costs of spam traffic, identifying the routes traversed by spam messages collected at five honeypots. Combining the volume of spam traffic with traceroute measurements and a database of internetwork business relationships, they show that stub networks are systematically subject to high spam traffic costs. They also show that some networks profit from spam traffic and might not be interested in filtering it. Finally, a simple-but-effective algorithm is presented to identify the networks that would benefit from cooperating to filter spam traffic at the origin, to reduce transit costs.

**An E-mail Filtering Approach Using Classification Techniques**

E-mail is one of the most popular ways of communication due to its accessibility, low sending cost and fast message transfer. However, Spam emails appear as a severe problem affecting this application of today’s Internet. Filtering is an important approach to isolate those spam emails. In this paper, an approach for filtering spam email is proposed, which is based on classification techniques. The approach analyses the body of Email messages and assigns weights to terms (features) that can help identifying spam and clean (ham) emails. An adaptation is proposed that tries to reduce the dimensionality of the extracted features, in which only determined (meaningful) terms are regarded by consulting a dictionary. A thorough comparative study has been studied among different classification algorithms that prove the efficiency of the filtering approach proposed. The approach has been evaluated using Enron dataset.

**A Neural Network Classifier for Junk E-Mail**

Most e-mail readers spend a non-trivial amount of time regularly deleting junk e-mail (spam) messages, even as an expanding volume of such e-mail occupies server storage space and consumes network bandwidth. An ongoing challenge, therefore, rests within the development and refinement of automatic classifiers that can distinguish legitimate e-mail from spam. A few published studies have examined spam detectors using Naïve Bayesian approaches and large feature sets of binary attributes that determine the existence of common keywords in spam, and many commercial applications also use Naïve Bayesian techniques. Spammers recognize these attempts to thwart their messages and have developed tactics to circumvent these filters, but these evasive tactics are themselves patterns that human readers can often identify quickly. Therefore, in contrast to earlier approaches, our feature set uses descriptive characteristics of words and messages similar to those that a human reader would use to identify spam. This preliminary study tests this alternative approach using a neural network (NN) classifier on a corpus of e-mail messages from one user. The results of this study are compared to previous spam detectors that have used Naïve Bayesian classifiers. Also, it appears that commercial spam detectors are now beginning to use descriptive features as proposed here.

**A review of machine learning approaches to Spam filtering**

In this paper, we present a comprehensive review of recent developments in the application of machine learning algorithms to Spam filtering, focusing on both textual- and image-based approaches. Instead of considering Spam filtering as a standard classification problem, we highlight the importance of considering specific characteristics of the problem, especially concept drift, in designing new filters. Two particularly important aspects not widely recognized in the literature are discussed: the difficulties in updating a classifier based on the bag-of-words representation and a major difference between two early naive Bayes models. Overall, we conclude that while important advancements have been made in the last years, several aspects remain to be explored, especially under more realistic evaluation settings.

**Spam Filtering and Email-Mediated Applications**

This chapter reviews and examines two important research topics related to intelligent email processing, namely, email filtering and email-mediated applications. We present a framework to show a full process of email filtering. Within the framework, we suggest a new method of combining multiple filters and propose a novel filtering model based on ensemble learning. For email-mediated applications, we introduce the concept of operable email (OE). It is argued that operable email will play a fundamental role in future email systems, in order to meet the need of the World Wide Wisdom Web (W4). We demonstrate the use of OE in implementing an email assistant and other intelligent applications on the World Social Email Network (WSEN).

**Image spam hunter**

Spammers are constantly creating sophisticated new weapons in their arms race with anti-spam technology, the latest of which is image-based spam. The newest image-based spam uses simple image processing technologies to vary the content of individual messages, e.g. by changing foreground colors, backgrounds, font types, or even rotating and adding artifacts to the images. Thus, they pose great challenges to conventional spam filters. In this paper, we propose a system using a probabilistic boosting tree to determine whether an incoming image is a spam or not based on global image features, i.e. color and gradient orientation histograms. The system identifies spam without the need for OCR and is robust in the face of the kinds of variation found in current spam images. Evaluation results show the system correctly classifies 90% of spam images while mislabeling only 0.86% of non-spam images as spam.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

The person sending the spam messages is referred to as the spammer. Such a person gathers email addresses from different websites, chatrooms, and viruses. Spam prevents the user from making full and good use of time, storage capacity and network bandwidth. The huge volume of spam mails flowing through the computer networks have destructive effects on the memory space of email servers, communication bandwidth, CPU power and user time. In all existing systems doesnot find the spam mails effectively. It is also resulted to untold financial loss to many users.

**Disadvantages of Existing System:**

1. Less Security.

**3.2 Proposed System**

In this project we are giving brief review on various machine learning algorithms such as SVM, Random Forest, Decision Tree, CNN, KNN, MLP and many more to predict spam emails. We should done experiments with above algorithms by using various SPAM datasets such as SPAM ARCHIVE, SPAMBASE, LINGSPAM, PU1 and many more but we are using SPAMBASE dataset to evaluate performance of above algorithms in terms of accuracy, precision and recall.

**Advantages of Proposed System:**

1. Security is more compared to existing systems.
2. Accuracy is more.

**Modules:**

1. Upload SpamBase Dataset

2.Preprocess Dataset

3. Run KNN, Naive Bayes & Multilayer Perceptron Algorithms

4.Run SVM, Decision Tree & AdaBoost Algorithms’

5.Run Random Forest & CNN Algorithm

6. Accuracy Comparison Graph

7.Recall Comparison Graph’

8.Precision Comparison Graph

1. Upload SpamBase Dataset

The selecting and uploading ‘spambase.data’ dataset and then click on ‘Open’ button to load dataset. Then the dataset may loaded.

2.Preprocess Dataset

Preprocessing is the second module in our project. To read all values from dataset and then split data into train and test part where application used 80% dataset for training and 20% dataset for testing.

3. Run KNN, Naive Bayes & Multilayer Perceptron Algorithms

We have to run all 3 algorithms and get there prediction metrics, we got evaluation metrics such as accuracy, recall and precision for all 3 algorithms.

4.Run SVM, Decision Tree & AdaBoost Algorithms

First we have to run Run SVM, Decision Tree & AdaBoost Algorithms. Then we got metrics for SVM, decision tree and AdaBoost algorithms.

5.Run Random Forest & CNN Algorithm

We should run Random Forest & CNN Algorithm, then we got accuracy for CNN and random forest algorithms.

6. Accuracy Comparison Graph

In graph x-axis represents algorithm name and y-axis represents accuracy of all those algorithms and from above graph we can conclude that MLP neural network give better prediction accuracy compare to all other algorithms.

7.Recall Comparison Graph

In graph x-axis represents algorithm name and y-axis represents Recall values of all those algorithms.

8.Precision Comparison Graph

In graph x-axis represents algorithm name and y-axis represents Precision values of all those algorithms.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**Operating Environment**

Windows XP.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows7/8
* Programming Language - Python

**4. SYSTEM DESIGN**

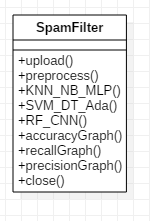
**UML Diagram:**

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic 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 objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**Class Diagram:**



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**Use case Diagram:**

Preprocess Dataset

Run KNN, Naive Bayes & Multilayer Perceptron Algorithms

Run SVM, Decision Tree & AdaBoost Algorithms

Run Random Forest & CNN Algorithm

Accuracy Comparison Graph

User

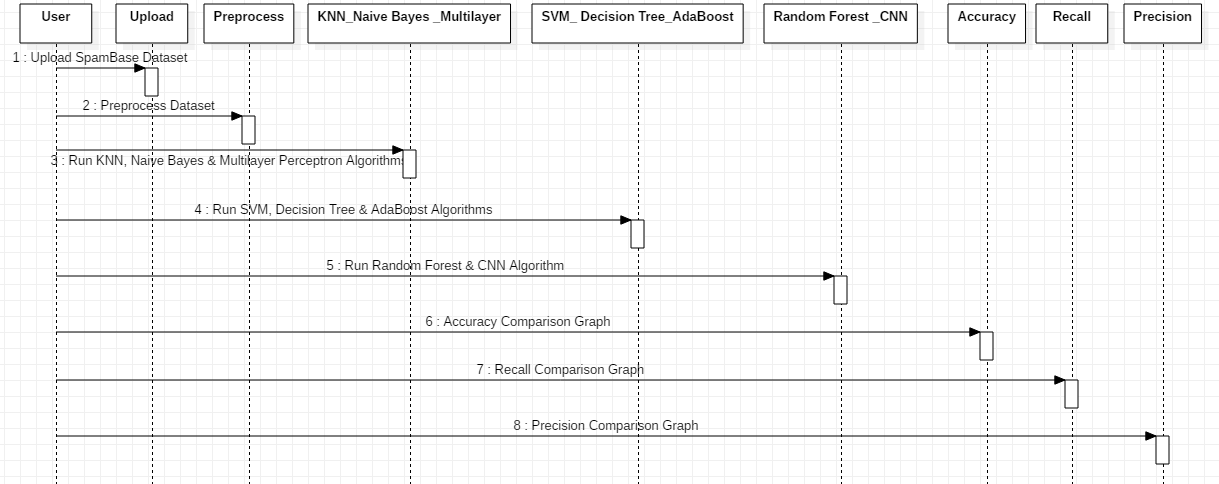
Upload SpamBase Dataset

Recall Comparison Graph

Precision Comparison Graph

**Sequence diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**Component Diagram:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

User

Preprocess Dataset

KNN, Naive Bayes & Multilayer Perceptron Algorithms

SVM, Decision Tree &

AdaBoost Algorithms

Random Forest & CNN Algorithms

Accuracy Comparison

Graph

Upload SpamBase

Dataset

Recall Comparison Graph

Precision Comparison Graph

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

User

Preprocess Dataset

KNN, Naive Bayes & Multilayer Perceptron

SVM, Decision Tree & AdaBoost

Random

Forest & CNN

Accuracy Comparison Graph

Upload SpamBase Detection Dataset

Recall Comparison Graph

Precision Comparison Graph

**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Upload SpamBase Dataset

Preprocess Dataset

Run KNN, Naive Bayes & Multilayer Perceptron Algorithms

Run SVM, Decision Tree & AdaBoost Algorithms

Run Random Forest & CNN Algorithm

Accuracy Comparison Graph

Recall Comparison Graph

Precision Comparison Graph

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

2. Dataset loaded

4. Preprocess completed

User

6. metrics displayed

8. results displayed

1. Upload SpamBase Dataset 10. Results displayed

3. Preprocess Dataset 12. Graph displayed

5. Run KNN, Naive Bayes & Multilayer Perceptron Algorithms

7. Run SVM, Decision Tree & AdaBoost Algorithms 14. Graph displayed

9. Run Random Forest & CNN Algorithm 16. Graph displayed

11. Accuracy Comparison Graph

13. Recall Comparison Graph

15. Precision Comparison Graph

**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**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 softwares 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 Linux.

**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 MySQLdb library using import MySQLdb .

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 OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code;**

**SpamFilter.py:**

from tkinter import \*

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from tkinter import simpledialog

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Activation

from keras.models import model\_from\_json

import pickle

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

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.naive\_bayes import BernoulliNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import svm

from sklearn.neural\_network import MLPClassifier

from sklearn.ensemble import AdaBoostClassifier

main = tkinter.Tk()

main.title("Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems")

main.geometry("1000x650")

global filename

global classifier

global X, Y

global dataset

accuracy = []

recall = []

precision = []

global X\_train, X\_test, y\_train, y\_test

def upload():

global filename

global dataset

filename = filedialog.askopenfilename(initialdir = "spambase")

text.delete('1.0', END)

text.insert(END,filename+' Loaded\n\n')

dataset = pd.read\_csv(filename)

text.insert(END,str(dataset.head))

def preprocess():

global X\_train, X\_test, y\_train, y\_test

text.delete('1.0', END)

global X, Y

global dataset

dataset = dataset.values

cols = dataset.shape[1]-1

X = dataset[:,0:cols]

Y = dataset[:,cols]

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

text.insert(END,"Total spam & non spam records found in dataset is : "+str(len(X))+"\n")

text.insert(END,"Splitting dataset into train & test. ML will user 80% dataset size for training and 20% for testing\n")

text.insert(END,"Splitted training records = "+str(len(X\_train))+"\n")

text.insert(END,"Splitted testing records = "+str(len(X\_test))+"\n")

def KNN\_NB\_MLP():

text.delete('1.0', END)

accuracy.clear()

recall.clear()

precision.clear()

knn = KNeighborsClassifier(n\_neighbors = 2)

knn.fit(X\_train, y\_train)

predict = knn.predict(X\_test)

knn\_precision = precision\_score(y\_test, predict,average='macro') \* 100

knn\_recall = recall\_score(y\_test, predict,average='macro') \* 100

knn\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"KNN Precision : "+str(knn\_precision)+"\n")

text.insert(END,"KNN Recall : "+str(knn\_recall)+"\n")

text.insert(END,"KNN Accuracy : "+str(knn\_acc)+"\n")

accuracy.append(knn\_acc)

recall.append(knn\_recall)

precision.append(knn\_precision)

nb = BernoulliNB(binarize=0.0)

nb.fit(X\_train, y\_train)

predict = nb.predict(X\_test)

nb\_precision = precision\_score(y\_test, predict,average='macro') \* 100

nb\_recall = recall\_score(y\_test, predict,average='macro') \* 100

nb\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"Naive Bayes Precision : "+str(nb\_precision)+"\n")

text.insert(END,"Naive Bayes Recall : "+str(nb\_recall)+"\n")

text.insert(END,"Naive Bayes Accuracy : "+str(nb\_acc)+"\n")

accuracy.append(nb\_acc)

recall.append(nb\_recall)

precision.append(nb\_precision)

mlp = MLPClassifier(random\_state=1, max\_iter=300)

mlp.fit(X\_train, y\_train)

predict = mlp.predict(X\_test)

mlp\_precision = precision\_score(y\_test, predict,average='macro') \* 100

mlp\_recall = recall\_score(y\_test, predict,average='macro') \* 100

mlp\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"MLP Precision : "+str(mlp\_precision)+"\n")

text.insert(END,"MLP Recall : "+str(mlp\_recall)+"\n")

text.insert(END,"MLP Accuracy : "+str(mlp\_acc)+"\n\n")

accuracy.append(mlp\_acc)

recall.append(mlp\_recall)

precision.append(mlp\_precision)

def SVM\_DT\_Ada():

sv = svm.SVC()

sv.fit(X\_train, y\_train)

predict = sv.predict(X\_test)

svm\_precision = precision\_score(y\_test, predict,average='macro') \* 100

svm\_recall = recall\_score(y\_test, predict,average='macro') \* 100

svm\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"SVM Precision : "+str(svm\_precision)+"\n")

text.insert(END,"SVM Recall : "+str(svm\_recall)+"\n")

text.insert(END,"SVM Accuracy : "+str(svm\_acc)+"\n")

accuracy.append(svm\_acc)

recall.append(svm\_recall)

precision.append(svm\_precision)

dt = DecisionTreeClassifier()

dt.fit(X\_train, y\_train)

predict = dt.predict(X\_test)

dt\_precision = precision\_score(y\_test, predict,average='macro') \* 100

dt\_recall = recall\_score(y\_test, predict,average='macro') \* 100

dt\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"Decision Tree Precision : "+str(dt\_precision)+"\n")

text.insert(END,"Decision Tree Recall : "+str(dt\_recall)+"\n")

text.insert(END,"Decision Tree Accuracy : "+str(dt\_acc)+"\n")

accuracy.append(dt\_acc)

recall.append(dt\_recall)

precision.append(dt\_precision)

ada = AdaBoostClassifier()

ada.fit(X\_train, y\_train)

predict = ada.predict(X\_test)

ada\_precision = precision\_score(y\_test, predict,average='macro') \* 100

ada\_recall = recall\_score(y\_test, predict,average='macro') \* 100

ada\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"AdaBoost Precision : "+str(ada\_precision)+"\n")

text.insert(END,"AdaBoost Recall : "+str(ada\_recall)+"\n")

text.insert(END,"AdaBoost Accuracy : "+str(ada\_acc)+"\n\n")

accuracy.append(ada\_acc)

recall.append(ada\_recall)

precision.append(ada\_precision)

def RF\_CNN():

rf = svm.SVC()

rf.fit(X\_train, y\_train)

predict = rf.predict(X\_test)

rf\_precision = precision\_score(y\_test, predict,average='macro') \* 100

rf\_recall = recall\_score(y\_test, predict,average='macro') \* 100

rf\_acc = accuracy\_score(y\_test,predict)\*100

text.insert(END,"Random Forest Precision : "+str(rf\_precision)+"\n")

text.insert(END,"Random Forest Recall : "+str(rf\_recall)+"\n")

text.insert(END,"Random Forest Accuracy : "+str(rf\_acc)+"\n")

accuracy.append(rf\_acc)

recall.append(rf\_recall)

precision.append(rf\_precision)

Y1 = to\_categorical(Y)

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X, Y1, test\_size=0.2)

cnn\_model = Sequential()

cnn\_model.add(Dense(512, input\_shape=(X\_train1.shape[1],)))

cnn\_model.add(Activation('relu'))

cnn\_model.add(Dropout(0.3))

cnn\_model.add(Dense(512))

cnn\_model.add(Activation('relu'))

cnn\_model.add(Dropout(0.3))

cnn\_model.add(Dense(2))

cnn\_model.add(Activation('softmax'))

cnn\_model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

print(cnn\_model.summary())

acc\_history = cnn\_model.fit(X, Y1, epochs=10, validation\_data=(X\_test1, y\_test1))

acc\_history = acc\_history.history

acc\_history = acc\_history['accuracy']

accuracy.append(acc\_history[9]\*100)

predict = cnn\_model.predict(X\_test1)

predict = np.argmax(predict, axis=1)

testY = np.argmax(y\_test1, axis=1)

cnn\_precision = precision\_score(testY, predict,average='macro') \* 100

cnn\_recall = recall\_score(testY, predict,average='macro') \* 100

recall.append(cnn\_recall)

precision.append(cnn\_precision)

text.insert(END,"CNN Precision : "+str(cnn\_precision)+"\n")

text.insert(END,"CNN Recall : "+str(cnn\_recall)+"\n")

text.insert(END,"CNN Accuracy : "+str(acc\_history[9]\*100)+"\n")

def accuracyGraph():

height = [accuracy[0],accuracy[1],accuracy[2],accuracy[3],accuracy[4],accuracy[5],accuracy[6],accuracy[7]]

bars = ('KNN ACC', 'NB ACC','MLP ACC','SVM ACC','Decision Tree ACC','AdaBoost ACC','Random Forest ACC','CNN ACC')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def recallGraph():

height = [recall[0],recall[1],recall[2],recall[3],recall[4],recall[5],recall[6],recall[7]]

bars = ('KNN Recall', 'NB Recall','MLP Recall','SVM Recall','Decision Tree Recall','AdaBoost Recall','Random Forest Recall','CNN Recall')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def precisionGraph():

height = [precision[0],precision[1],precision[2],precision[3],precision[4],precision[5],precision[6],precision[7]]

bars = ('KNN Precision', 'NB Precision','MLP Precision','SVM Precision','Decision Tree Precision','AdaBoost Precision','Random Forest Precision','CNN Precision')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def close():

main.destroy()

font = ('times', 15, 'bold')

title = Label(main, text='Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems ', justify=LEFT)

title.config(bg='lavender blush', fg='DarkOrchid1')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=100,y=5)

title.pack()

font1 = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload SpamBase Dataset", command=upload)

uploadButton.place(x=10,y=100)

uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset", command=preprocess)

preprocessButton.place(x=300,y=100)

preprocessButton.config(font=font1)

firstButton = Button(main, text="Run KNN, Naive Bayes & Multilayer Perceptron Algorithms", command=KNN\_NB\_MLP)

firstButton.place(x=480,y=100)

firstButton.config(font=font1)

secondButton = Button(main, text="Run SVM, Decision Tree & AdaBoost Algorithms", command=SVM\_DT\_Ada)

secondButton.place(x=940,y=100)

secondButton.config(font=font1)

thirdButton = Button(main, text="Run Random Forest & CNN Algorithm", command=RF\_CNN)

thirdButton.place(x=10,y=150)

thirdButton.config(font=font1)

accButton = Button(main, text="Accuracy Comparison Graph", command=accuracyGraph)

accButton.place(x=300,y=150)

accButton.config(font=font1)

recallButton = Button(main, text="Recall Comparison Graph", command=recallGraph)

recallButton.place(x=560,y=150)

recallButton.config(font=font1)

precisionButton = Button(main, text="Precision Comparison Graph", command=precisionGraph)

precisionButton.place(x=840,y=150)

precisionButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=160)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='light coral')

main.mainloop()

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

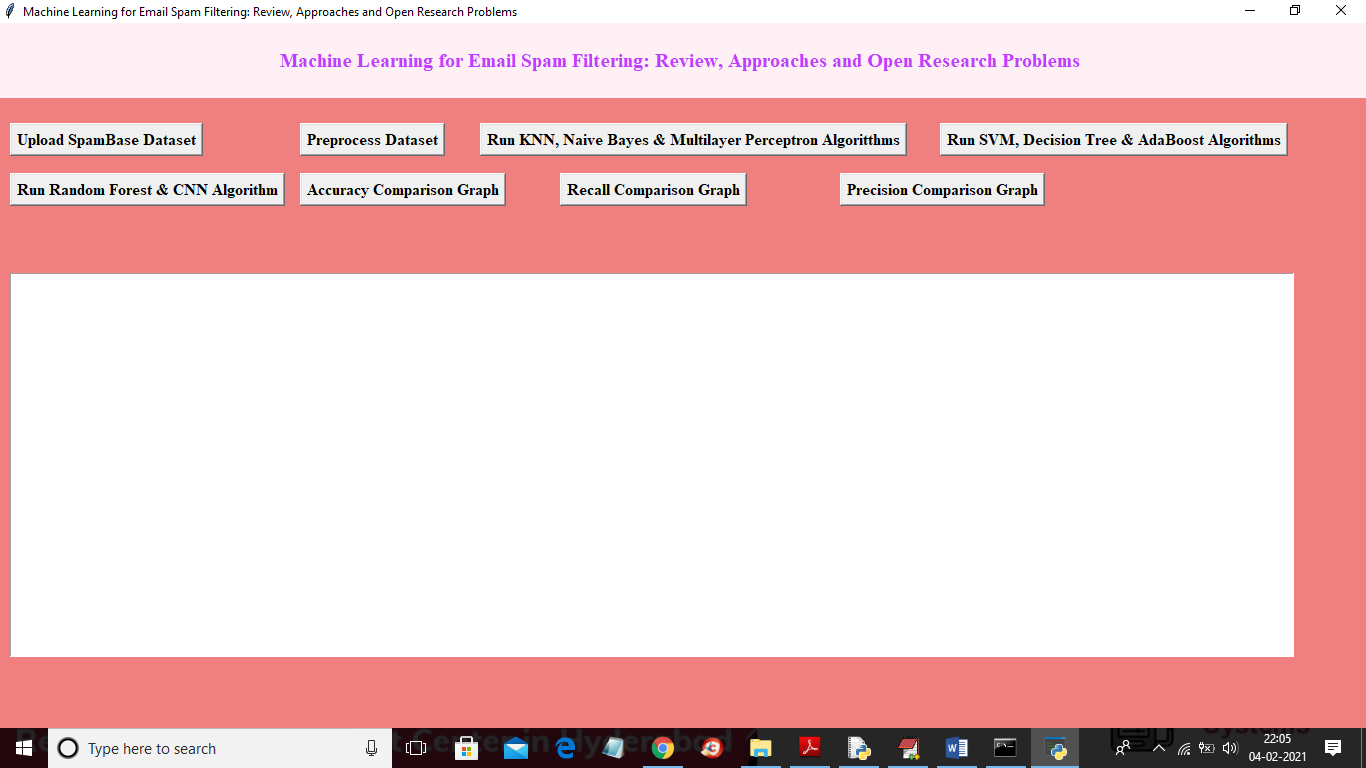
After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

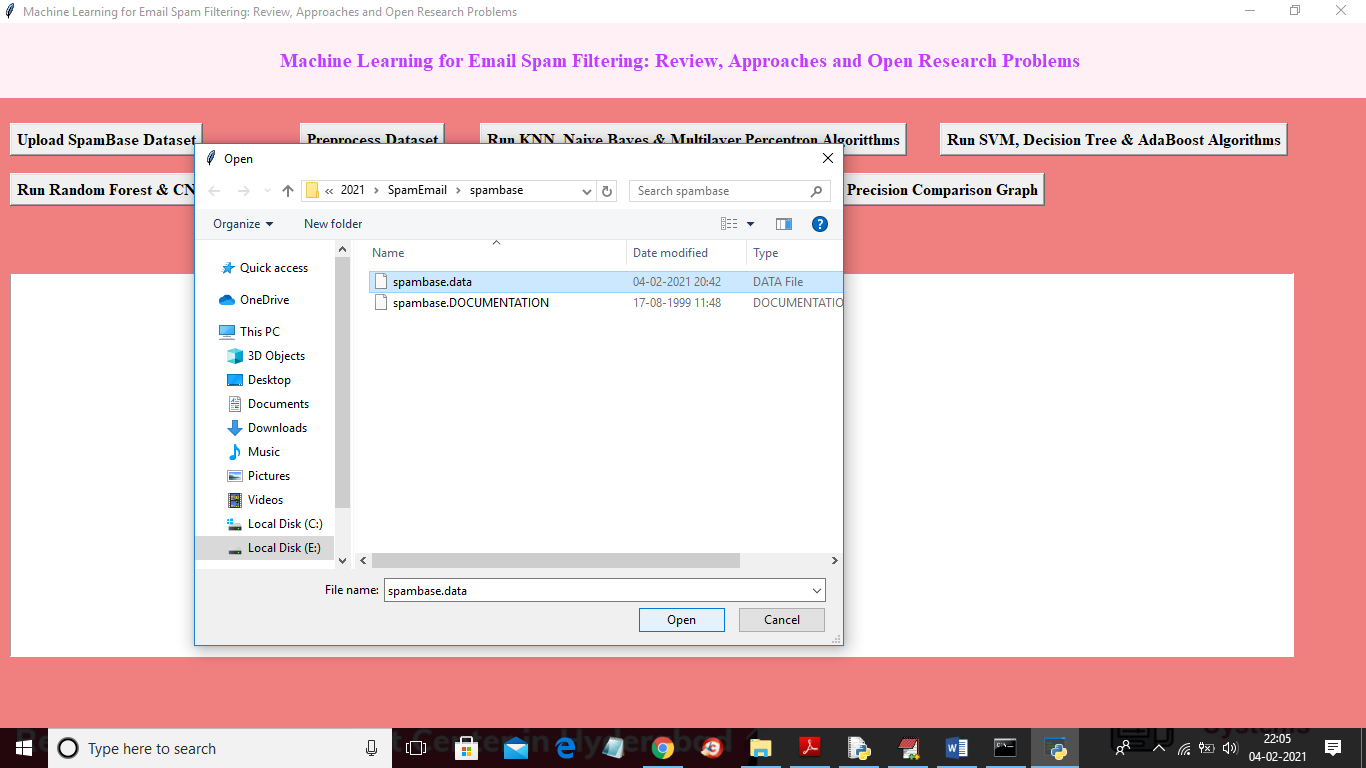
When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload SpamBase Dataset | Test whether the SpamBase Dataset uploaded or not | If the dataset may not uploaded | We cannot do further process | The SpamBase Dataset uploaded successfully | High | High |
| 02 | Preprocess Dataset | Verify either the dataset may preprocess or not | Without loading the dataset | We cannot do preprocess | Preprocess completed successfully | High | High |
| 03 | Run KNN, Naive Bayes & Multilayer Perceptron Algorithms | Verify either KNN, Naive Bayes & Multilayer Perceptron Algorithms  Performed or not | Without preprocess | We cannot run properly KNN, Naive Bayes & Multilayer Perceptron Algorithms | KNN, Naive Bayes & Multilayer Perceptron Algorithms  Are processed successfully | High | High |
| 04 | Run SVM, Decision Tree & AdaBoost Algorithms | Verify either KNN, Naive Bayes & Multilayer Perceptron Algorithms  Performed or not | Without preprocess | We cannot run SVM, Decision Tree & AdaBoost Algorithms | SVM, Decision Tree & AdaBoost Algorithms  processed successfully | High | High |
| 05 | Run Random Forest & CNN Algorithm | Verify either Random Forest & CNN Algorithm  Performed or not | Without preprocess | We cannot run Random Forest & CNN Algorithm | Random Forest & CNN Algorithms  Are processed successfully | High | High |
| 06 | Accuracy Comparison Graph | Verify either the Accuracy Comparison Graph displayed or not | Without saving the accuracy values | The graph cannot displayed succesdfully | The Accuracy Comparison Graph displayed successfully | High | high |
| 07 | Recall Comparison Graph | Verify either the Recall Comparison Graph displayed or not | Without saving the Recall values | The Recall graph cannot displayed succesdfully | The Recall Comparison Graph displayed successfully | High | high |
| 09 | Precision Comparison Graph | Verify either the Precision Comparison Graph displayed or not | Without saving the Precision values | The graph cannot displayed succesdfully | The Precision Comparison Graph displayed successfully | High | high |

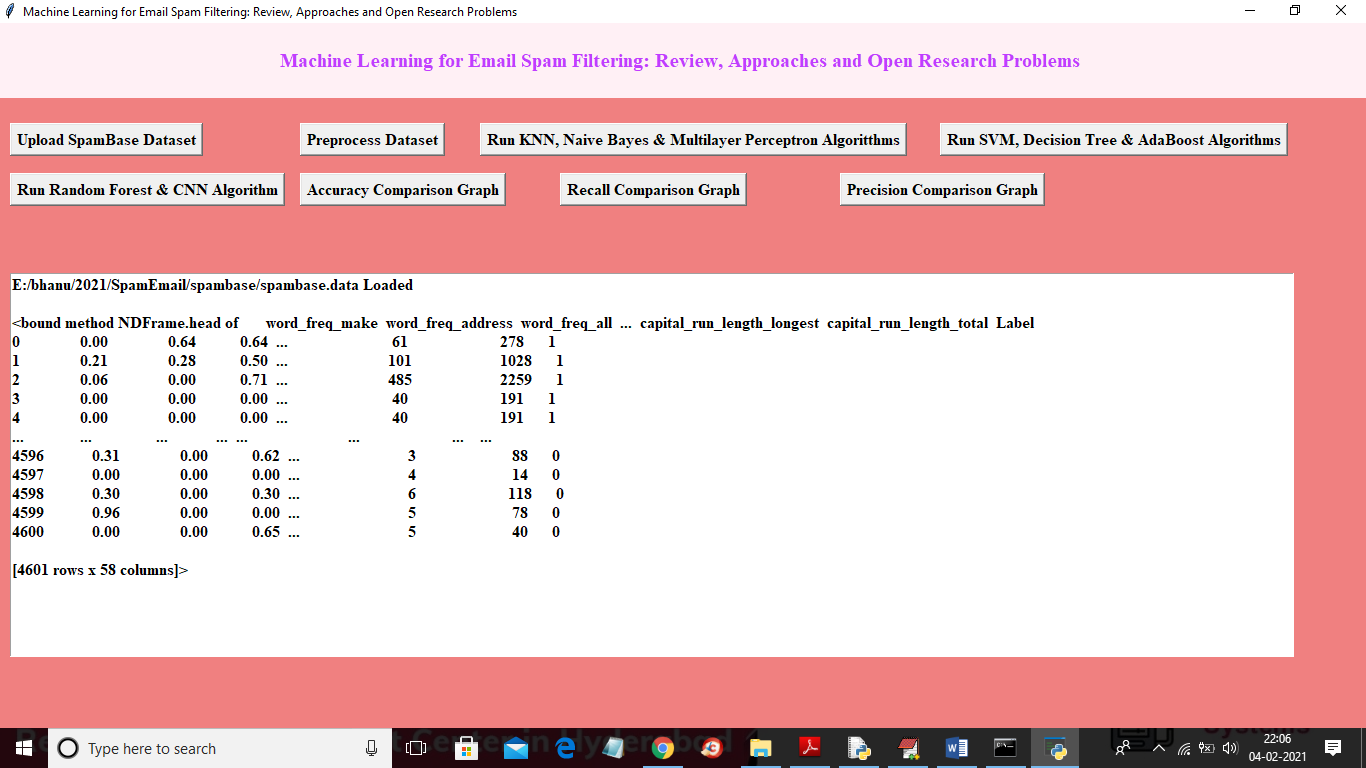
**7. SCREENSHOTS**



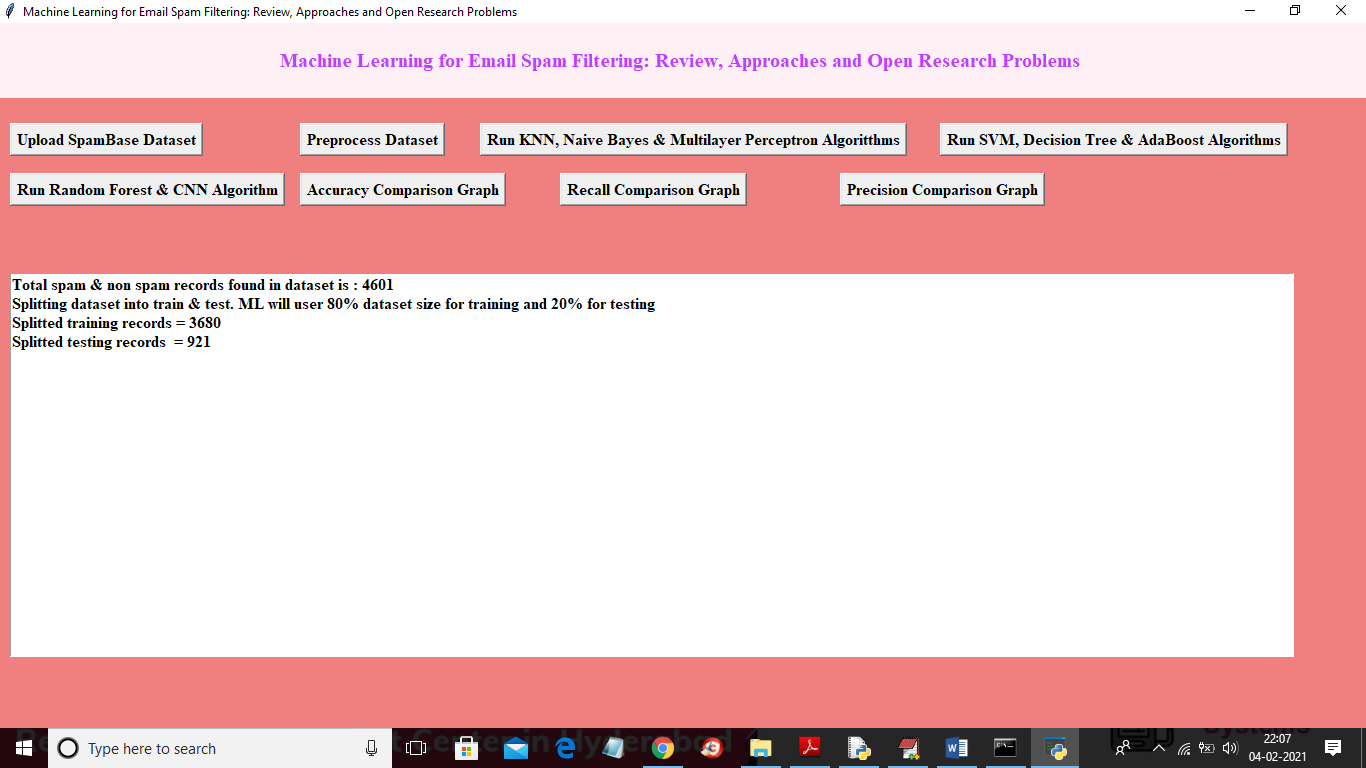
In above screen click on ‘Upload SpamBase Dataset’ and upload dataset



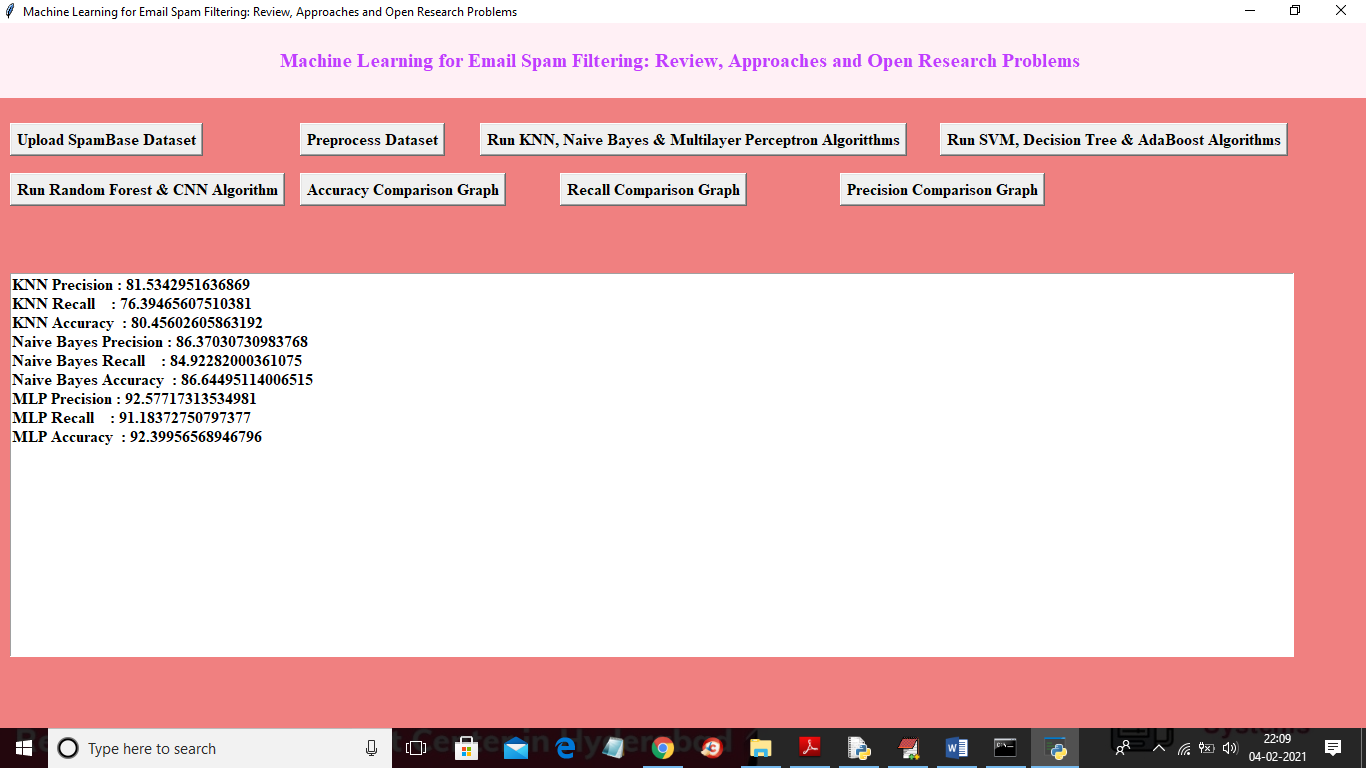
In above screen selecting and uploading ‘spambase.data’ dataset and then click on ‘Open’ button to load dataset and to get below screen



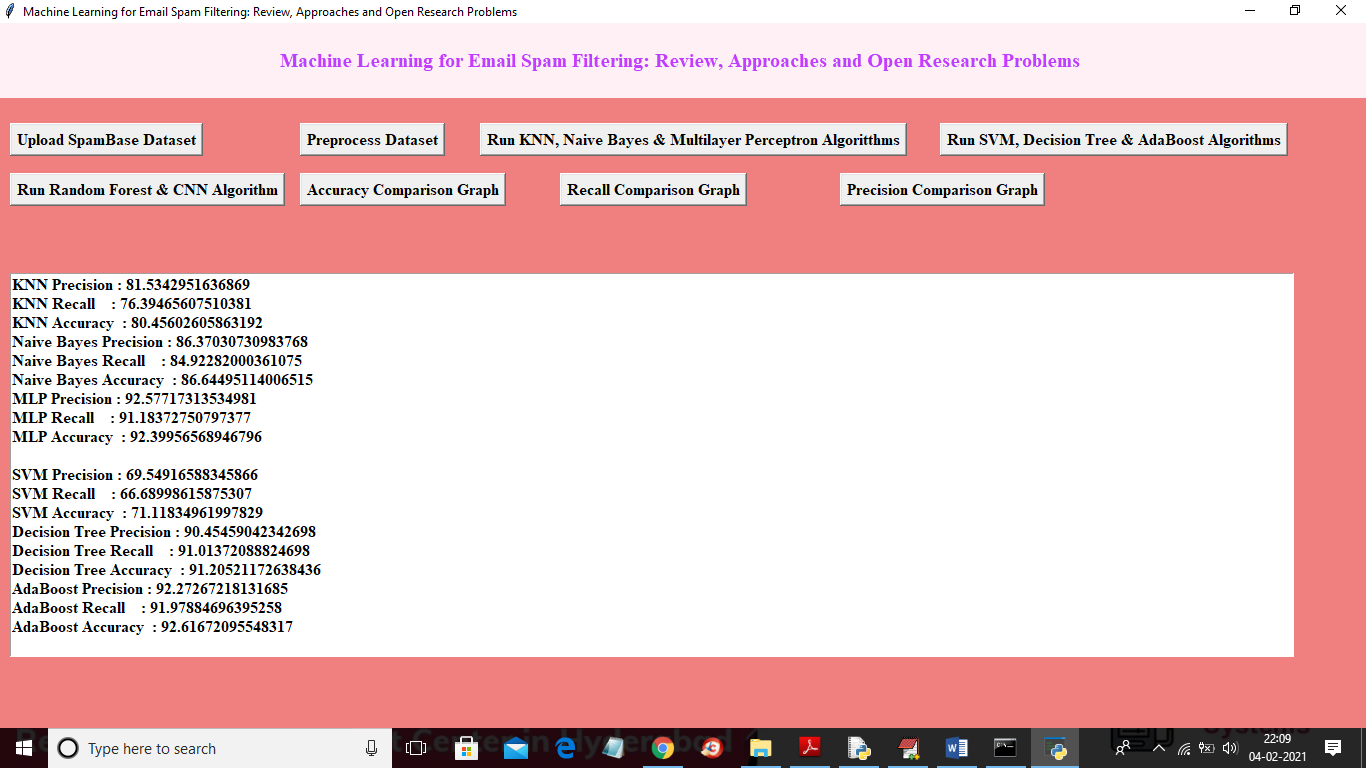
In above screen dataset loaded and we can see some records from the dataset and now click on ‘Preprocess Dataset’ button to read all values from dataset and then split data into train and test part where application used 80% dataset for training and 20% dataset for testing



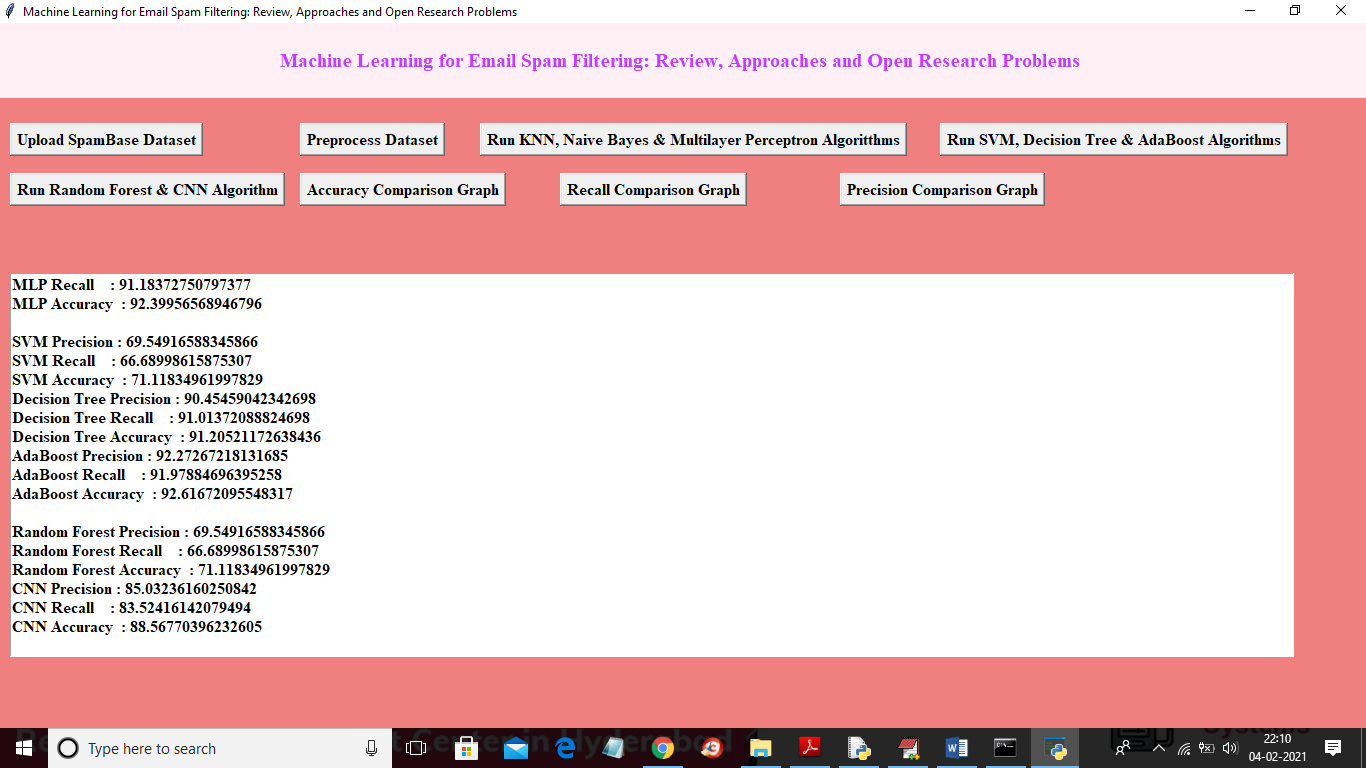
In above screen we can see dataset contains 4601 records and application using 3680 for training and 921 records for testing and now dataset is ready and now click on ‘Run KNN, Naive Bayes & Multilayer Perceptron Algorithms’ button to run all 3 algorithms and get there prediction metrics



In above screen we got evaluation metrics such as accuracy, recall and precision for all 3 algorithms and now click on ‘Run SVM, Decision Tree & AdaBoost Algorithms’ button to run this 3 algorithms also



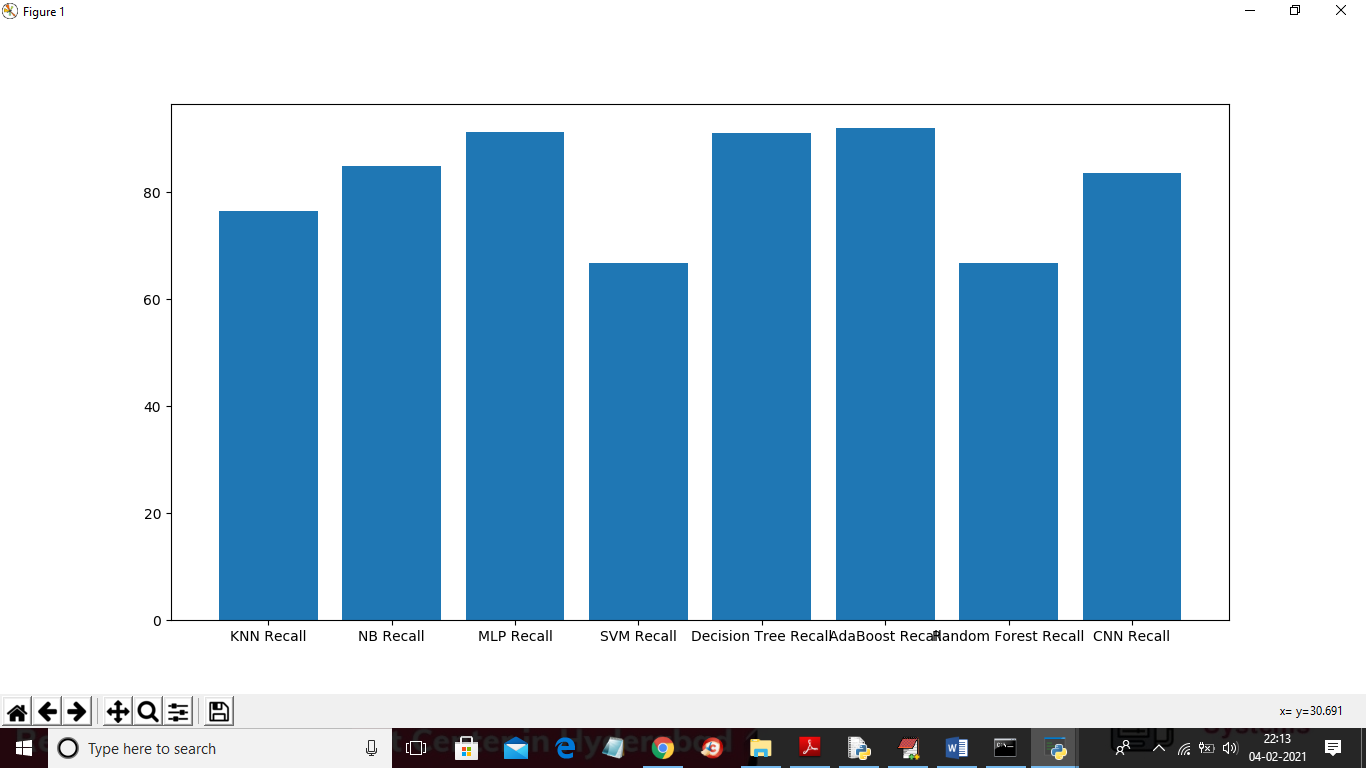
In above screen we got metrics for SVM, decision tree and AdaBoost algorithms and now click on ‘Run Random Forest & CNN Algorithm’ button to run both algorithms and then get below result



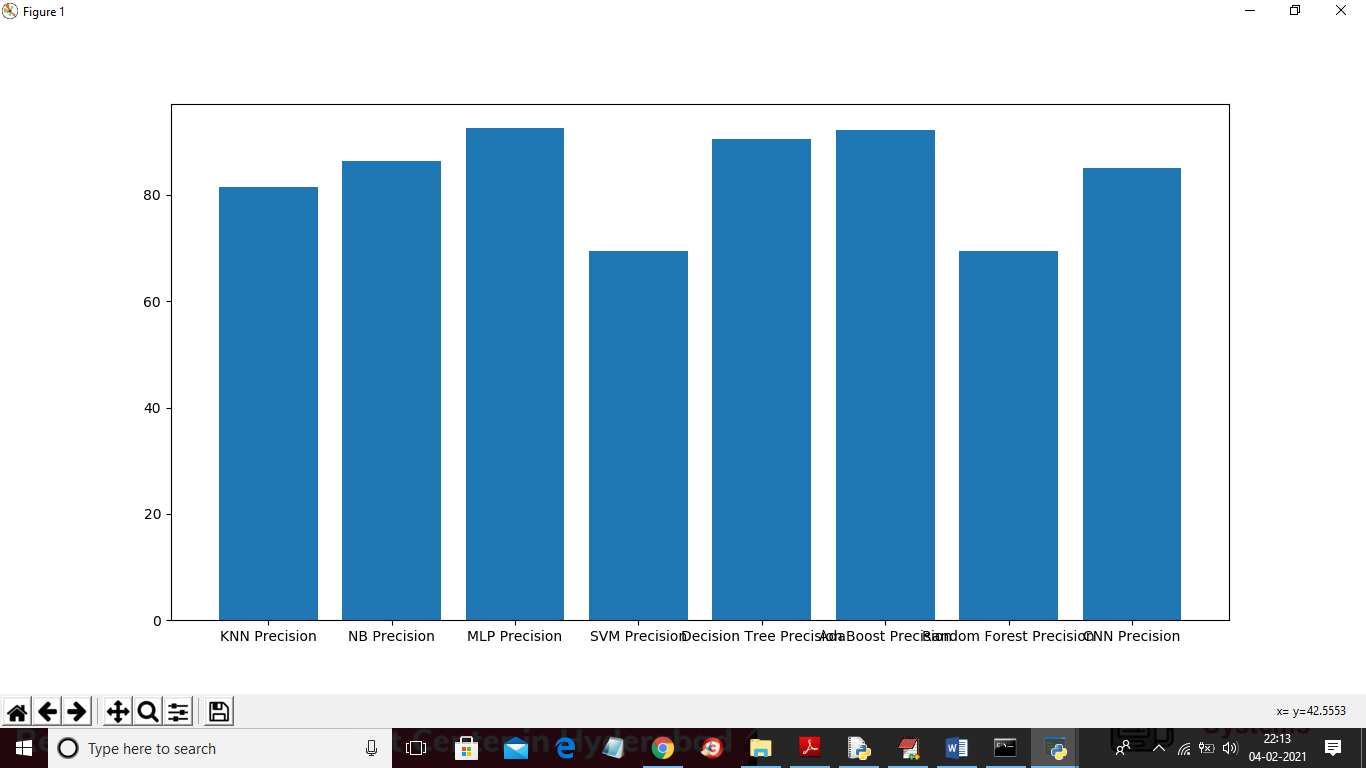
In above screen we got accuracy for CNN and random forest algorithms also and now click on ‘Accuracy Comparison Graph’ button to get below accuracy comparison between all algorithms



In above screen x-axis represents algorithm name and y-axis represents accuracy of all those algorithms and from above graph we can conclude that MLP neural network give better prediction accuracy compare to all other algorithm. Now click on ‘Recall Comparison Graph’ button to get below recall graph



Now click on ‘Precision Comparison Graph’ button to get below precision graph.



In all above 3 graph MLP give better accuracy, precision and recall.

**8. CONCLUSION**

In this study, we reviewed machine learning approaches and their application to the field of spam filtering. A review of the state of the art algorithms been applied for classification of messages as either spam or ham is provided. The attempts made by different researchers to solving the problem of spam through the use of machine learning classifiers was discussed. The evolution of spam messages over the years to evade filters was examined. The basic architecture of email spam filter and the processes involved in filtering spam emails were looked into. The paper surveyed some of the publicly available datasets and performance metrics that can be used to measure the effectiveness of any spam filter. The challenges of the machine learning algorithms in efficiently handling the menace of spam was pointed out and comparative studies of the machine learning technics available in literature was done. We also revealed some open research problems associated with spam filters. In general, the figure and volume of literature we reviewed shows that significant progress have been made and will still be made in this field. Having discussed the open problems in spam filtering, further research to enhance the effectiveness of spam filters need to be done. This will make the development of spam filters to continue to be an active research field for academician and industry practitioners researching machine learning techniques for effective spam filtering. Our hope is that research students will use this paper as a spring board for doing qualitative research in spam filtering using machine learning, deep leaning and deep adversarial learning algorithms.

**REFERENCES**

[1] M. Awad, M. Foqaha, Email spam classification using hybrid approach of RBF neural network and particle swarm optimization, Int. J. Netw. Secur. Appl. 8 (4) (2016).

[2] D.M. Fonseca, O.H. Fazzion, E. Cunha, I. Las-Casas, P.D. Guedes, W. Meira, M. Chaves, Measuring characterizing, and avoiding spam traffic costs, IEEE Int. Comp. 99 (2016).

[3] Visited on May 15, 2017, Kaspersky Lab Spam Report, 2017, 2012, https://www. securelist.com/en/analysis/204792230/Spam\_Report\_April\_2012.

[4] E.M. Bahgat, S. Rady, W. Gad, An e-mail filtering approach using classification techniques, in: The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Springer International Publishing, BeniSuef, Egypt, 2016, pp. 321–331.

[5] N. Bouguila, O. Amayri, A discrete mixture-based kernel for SVMs: application to spam and image categorization, Inf. Process. Manag. 45 (6) (2009) 631–642.

[6] Y. Cao, X. Liao, Y. Li, An e-mail filtering approach using neural network, in: International Symposium on Neural Networks, Springer Berlin Heidelberg, 2004, pp. 688–694.

[7] F. Fdez-Riverola, E.L. Iglesias, F. Diaz, J.R. Mendez, J.M. Corchado, SpamHunting: an instance-based reasoning system for spam labelling and filtering, Decis. Support Syst. 43 (3) (2007) 722–736.

[8] S. Mason, New Law Designed to Limit Amount of Spam in E-Mail, 2003. <http://www.wral.com/technolog>.

[9] I. Stuart, S.H. Cha, C. Tappert, A neural network classifier for junk e-mail, in: Document Analysis Systems VI, Springer Berlin Heidelberg, 2004, pp. 442–450.

[10] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Elsevier, 2011.

[11] S.N. Qasem, S.M. Shamsuddin, A.M. Zain, Multi-objective hybrid algorithms for radial basis function neural network design, Knowl. Based Syst. 27 (2012) 475–497.

[12] J.D. Schaffer, D. Whitley, L. Eshelman, Combinations of genetic algorithms and neural networks: a survey of the state of the art, Combinations of Genetic Algorithms and Neural Networks, 1992, pp. 1–37.

[13] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based optimization algorithms, Adv. Eng. Inf. 19 (2005) 43–53.

[14] L.H. Gomes, C. Cazita, J.M. Almeida, V. Almeida, W.J. Meira, Workload models of spam and legitimate e-mails, Perform. Eval 64 (7–8) (2007) 690–714.

[15] C.C. Wang, S.Y. Chen, Using header session messages to anti-spamming, Comput. Secur. 26 (5) (2007) 381–390.

[16] T.S. Guzella, W.M. Caminhas, A review of machine learning approaches to spam filtering, Expert Syst. Appl. 36 (7) (2009) 10206–10222.

[17] C.P. Lueg, From spam filtering to information retrieval and back: seeking conceptual foundations for spam filtering, Proc. Assoc. Inf. Sci. Technol. 42 (1) (2005).

[18] X.L. Wang, Learning to classify email: a survey, in: 2005 International Conference on Machine Learning and Cybernetics (Vol. 9, pp. 5716-5719), IEEE, Aug 2005.

[19] W. Li, N. Zhong, Y. Yao, J. Liu, C. Liu, Spam filtering and email-mediated applications, in: Paper presented at the International Workshop on Web Intelligence Meets Brain Informatics, 2006.

[20] G.V. Cormack, Email spam filtering: a systematic review, Found. Trends Inf. Retr. 1 (4) (2008) 335–455.

[21] E.P. Sanz, J.M.G. Hidalgo, J.C.C. Perez, Email spam filtering, Adv. Comput. 74 (2008) 45–114.

[22] S. Dhanaraj, V. Karthikeyani, A study on e-mail image spam filtering techniques, in: Paper presented at the International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013.

[23] A. Bhowmick, S.M. Hazarika, Machine Learning for E-Mail Spam Filtering: Review, Techniques and Trends, arXiv:1606.01042v1 [cs.LG] 3 Jun 2016, 2016, pp. 1–27. [24] C. Laorden, X. Ugarte-Pedrero, I. Santos, B. Sanz, J. Nieves, P.G. Bringas, Study on the effectiveness of anomaly detection for spam filtering, Inf. Sci. 277 (2014) 421–444. [25] I. Katakis, G. Tsoumakas, I. Vlahavas, Email mining : emerging techniques for email management, in: A. Vakali, G. Pallis (Eds.), Web Data Management Practices: Emerging Techniques and Technologies, Idea Group Publishing, USA, 2007 chap 10.