

# **FEDCULA - Fraudulent Email Detection using CNN and URL Lexical Analysis**

**A Project Report**

*Submitted by*

**HITESH N. SONEJI  
ANIRUDDH SOMAN  
ANIRUDDH VYAS**

*Under the Guidance of*

**DR. SHUBHA PUTHRAN**  
*in partial fulfillment for the award of the  
degree of*

**BACHELORS OF TECHNOLOGY  
COMPUTER ENGINEERING**

**At**



**MUKESH PATEL SCHOOL OF TECHNOLOGY  
MANAGEMENT AND ENGINEERING**

**APRIL 2021**

## DECLARATION

We, Hitesh N. Soneji, Aniruddh Soman, and Aniruddh Vyas Roll No. E041, E040 and E058 B.Tech (Computer Engineering), VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature:

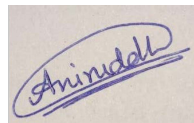


Name: Hitesh N. Soneji

Roll No. E041

Place: Mumbai

Date: 13/04/2021



Aniruddh Soman

E040



Aniruddh Vyas

E058

# **CERTIFICATE**

This is to certify that the project entitled “Email Fraud Detection” is the bonafide work carried out by Hitesh N. Soneji, Aniruddh Soman and Aniruddh Vyas of B.Tech (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIII semester of the academic year 2020-2021, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

---

Dr. Shubha Puthran

Internal Mentor

---

Examiner 1

---

Examiner 2

---

Dean

# Table of contents

LIST OF FIGURES.....	I
LIST OF TABLES.....	II
ABBREVIATIONS.....	III
ABSTRACT .....	IV
1. INTRODUCTION .....	1
1.1 ABOUT EMAIL FRAUD .....	1
1.2 TYPES OF FRAUDULENT EMAILS.....	1
1.3 PROJECT AIM.....	2
1.4 PROBLEM DEFINITION.....	2
1.5 SYSTEM SPECIFICATIONS .....	2
1.5.1 <i>Software Specifications</i> .....	2
1.5.2 <i>Python Libraries Used</i> .....	3
1.5.3 <i>Hardware Specifications</i> .....	3
1.6 REPORT STRUCTURE .....	3
2. REVIEW OF LITERATURE.....	4
3. ANALYSIS AND DESIGN .....	9
4. METHODS IMPLEMENTED .....	13
4.1 DATA PRE-PROCESSING .....	14
4.2 CORRELATION .....	15
4.2.1 <i>Cramer's V Correlation:</i> .....	16
4.2.2 <i>Pearson's Correlation:</i> .....	17
4.2.3 <i>Information Gain:</i> .....	18
4.3 WORD EMBEDDINGS.....	19
4.4 URL CLASSIFICATION.....	19
4.5 DECISION TREE.....	21
4.6 RANDOM FOREST.....	22
4.7 NAÏVE BAYES .....	23
4.8 SUPPORT VECTOR MACHINE .....	24
4.9 K-NEAREST NEIGHBORS .....	25
4.10 CONVOLUTIONAL NEURAL NETWORK .....	26
4.11 GENERAL ARCHITECTURE AND FLOW OF CLASSIFICATION.....	27
4.12 IMPLEMENTATION SCREENSHOTS .....	29
5. RESULTS AND DISCUSSION .....	31
5.1 DATA PRE-PROCESSING.....	31
5.2 CORRELATION .....	31
5.3 RANDOM FOREST ON EVERY DATASET .....	32
5.4 URL CLASSIFIER.....	32
5.5 RESULTS OF ALL TECHNIQUES ON SINGLE DATASET .....	33
5.6 RESULTS OF ALL TECHNIQUES ON MIXED DATASET .....	34
5.7 RESULT ANALYSIS OF CNN.....	34
5.8 COMPARISON OF ACCURACIES AND DISCUSSION .....	35
6. CONCLUSION AND FUTURE SCOPE .....	37
SOCIETAL BENEFITS.....	38
REFERENCES .....	39

<b>PUBLICATIONS.....</b>	<b>42</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>43</b>

## **List of Figures**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
3	ANALYSIS AND DESIGN	
	Fig. 3.1: Architecture of the model	9
	Fig. 3.2: Use Case Diagram	10
	Fig. 3.3: Sequence Diagram	11
	Fig. 3.4: Class Diagram	12
4	METHODS IMPLEMENTED	
	Fig. 4.1: Data Pre-Processing	14
	Fig. 4.2: Correlation Techniques	16
	Fig. 4.3: Lexical Analysis of URLs	20
	Fig. 4.4: Decision Tree	22
	Fig. 4.5: Random Forest	23
	Fig. 4.6: Bayes Theorem	23
	Fig. 4.7: Support Vector Machine	24
	Fig. 4.8: K – Nearest Neighbor	25
	Fig. 4.9: Convolutional Neural Networks	26
	Fig. 4.10: General Architecture	27
	Fig. 4.11: Flow of Classification	28
	Fig. 4.12: Website Homepage	29
	Fig. 4.13: Results page	30
5	RESULTS AND DISCUSSION	
	Fig. 5.1: Dataset after Pre-Processing	31
	Fig. 5.2: URL Classifier Output	32
	Fig. 5.3: CNN Confusion Matrix	35

## **List of Tables**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2	REVIEW OF LITERATURE	
	Table 2.1: Strengths and Weaknesses	6-8
5	RESULTS AND DISCUSSION	
	Table 5.1: Random Forest Results	32
	Table 5.2: Accuracy results on single datasets	33
	Table 5.3: Time taken on single dataset	33
	Table 5.4: Accuracy results on complex datasets	34
	Table 5.5: Result analysis of CNN	35

# Abbreviations

Abbreviation	Description
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Networks
RF	Random Forest
NB	Naive Bayes
DT	Decision Trees
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative



## **Abstract**

Emails have become an integral part of our lives, be it personal or professional, emails are used everywhere. It is one of the most popular mode of communication. Everyday millions of the spam mails are sent over internet to targeted population to advertise services, products and dangerous software etc. Most websites require the customers to enter their email ID, increasing the risk of exposure to spammers who attack them by sending spam messages. There are many techniques that can be used to detect fraudulent emails. The aim of this project is to create a model that works in real time using machine learning and deep learning techniques to detect fraudulent emails and displaying them to the user. This model will first fetch the emails from the user's inbox and pass the emails to the classifier. After pre-processing the emails, machine learning and deep learning techniques are applied to the emails to classify it into fraudulent and non-fraudulent emails.

# Chapter 1

## Introduction

### **1.1 About Email Fraud**

Electronic mail or email is a method of exchanging messages between people. It first saw usage in the 1960s, but gained popularity in the 1970s. Back then, the person communicating via email needed to be online to receive an email, but that is no longer the case, one can receive emails while offline as well. Emails are one of, if not the most common methods of communication and as such, they are used by everyone, from the common man to the CEOs of MNCs. Like any other form of communication, emails are prone to attacks from hackers and phishers.

The major email protocols in use, SMTP and POP are easily accessible to everyone, making them vulnerable to attacks. Email fraud is the intentional deception made for personal gain or to damage another individual through email. Email fraud started happening around the time email became a mainstream method of communication.

There are various types of email fraud, some of the most prominent ones are Spoofing, Phishing for Data and Bogus Offers. The general public is not aware of such methods of fraud and often fall prey to it. As per a recent statistic in [1], more than a hundred million phishing/fake emails related to coronavirus are sent to people. The spammer sends an email claiming to collect donations for relief efforts in relation to COVID-19, and obtain the credit card credentials, bank details and money from good Samaritans. These are a small subset of all spam emails, relating only to COVID-19. Considering all spam emails related to banking, marketing, advertising, anti-virus spam, etc. the number would be much higher.

### **1.2 Types of Fraudulent Emails**

Fraudulent Emails are of many types, some of the most common ones are:

1. Spam – the emails sent by spammers to sell fake products or commit fraud,
2. Phishing – these emails are used to trick unsuspecting users into divulging confidential information such as usernames, passwords and bank account details. They often mimic official emails from companies such as PayPal, eBay, Citibank, etc.,
3. CEO Fraud – this fraud involves sending

emails pretending to be the CEO of the company and asking for money transfers from the finance department. This type of fraud is becoming more commonplace, 4. Spoofing – spoofing emails trick users into visiting malicious sites or download software which then steal the users' sensitive information.

Distinguishing between legitimate and fraudulent emails is not an easy task for the layman. With the increase in reach of malicious emails and poor spam filters, the common person often falls prey to scams and ends up losing money or their personal data. Cybercrime via emails has become a common problem. According to [2], even big company executives fall victim to such problems and lose millions of dollars in these scams. The necessity of a smart email spam filter has become the urge of the society, with people of all ages getting caught in scams. Many spam filters become obsolete as soon as the spammers find a way to bypass such filters.

### **1.3 Project Aim**

The aim of the project is to create an email fraud detection system that is capable of filtering out fraudulent emails from the legitimate ones. The model is built using Python and makes use of techniques such as Machine Learning and Deep Learning to detect fraudulent emails. The model is trained using popular datasets such as SpamAssassin and SpamBase.

### **1.4 Problem Definition**

Email fraud detection model which can help people easily identify and discard fraudulent emails using Machine Learning and Deep Learning Techniques, as well as the improvement of existing techniques.

### **1.5 System Specifications**

#### **1.5.1 Software Specifications**

Python Interpreter: 3.7.0 or higher.

Operating System: Windows, Linux or Mac

### **1.5.2 Python Libraries Used**

Django: for front end implementation

pandas: for pre-processing and reading the data

gensim: to load the Google Word2Vec model and read it.

Sklearn: to implement various metric analysis and Machine Learning algorithms

numpy: for simplifying mathematical formula implementation

nltk: to load and use various Natural Language Processing tools like stopwords.

Keras: for implementation of the CNN model

### **1.5.3 Hardware Specifications**

Processor: Intel core i7

RAM: 16GB

Hard Disk: 50GB

## **1.6 Report Structure**

The report is divided into 6 chapters. The first chapter presents the introduction to the project. The second chapter contains the literature survey conducted by us on the topic which consists of 31 papers. The third chapter contains the design and analysis of the model. The fourth chapter contains the methods implemented by us in the model. The fifth chapter contains the results obtained by us. The sixth chapter contains the conclusion and our future scope.

## **Chapter 2**

### **Review of Literature**

For conducting our literature survey, we have framed some questions which will help us in the development of our model. Through these questions we have studied various email fraud detection techniques and the datasets used by them. The questions are:

1. Which data pre-processing techniques have been implemented for preparing the data?
2. Which ML and DL techniques have been used for classification of the emails?
3. What datasets have been utilized across the literature?
4. What Evaluation metrics have been most commonly used in the literature?

To answer these questions, we have studied 31 papers which use different techniques to detect fraudulent emails. Using these papers, we were able to answer the above proposed questions. Some of the common pre-processing techniques used are stop word removal, word tokenization, tf-idf (term frequency inverse document frequency) and Word2Vec (Word Embeddings). Stop Word Removal is the one of the most basic processing techniques in which stop words like we, us, and, or, if etc. are removed because they appear very frequently and can create problems with classifications. Word Tokenization is the process by which the given text is converted into a list of individual words. This helps us to perform natural language processing (NLP) easily. The words can be easily captured and analyzed upon using this technique. Term Frequency Inverse Document Frequency (tf-idf) is a statistical measure that tells how relevant a word is to a document in a collection of documents. Using tf-idf we can find out how important is a particular word in the document. These techniques help us to shape the datasets to our requirements.

Many machine learning and deep learning techniques were used to classify the emails. Some of the prominently used techniques are Support Vector Machine (SVM), Naïve Bayes, Random Forest, Convolutional Neural Networks (CNN) and K-Nearest Neighbors (KNN). Support Vector Machine is one of the most commonly used classification techniques. Various iterations of this techniques are used to classify the emails. In Naive Bayes classification (based on Bayes' theorem), every object has certain features which form the characteristics of said object. This method determines the probability of the features independently of each other, before classifying an object. A random forest is a classification technique comprising of many decision trees each of which make a prediction, classifying an object and passing their vote. The forest then tallies the votes and the majority vote is chosen as the class for the object. Convolutional neural network

(CNN) is a deep learning neural network designed for processing structured arrays of data such as images. CNN is widely used in computer vision and has become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

Regarding the datasets, we found the use of nine public and thirteen private datasets. ‘UCI’, ‘SpamBase’, ‘PhishTank’, ‘SpamAssasin’, ‘Ling Spam’, ‘SpamCorpus’, ‘PhishCorpus’, ‘Jose Nazario’ and ‘Enron’ are part of the public datasets in the literature, while Universities, Private and Others are part of the Self-Created Datasets which are not open-source. SpamAssasin is the most widely used dataset in the literature, followed by Enron. These datasets are easily accessible on the web and contain a large amount of emails.

The commonly used evaluation metrics in the papers are Confusion Matrix, Accuracy, Recall, Precision, F1 Score, Specificity and False Positive Rate. Confusion Matrix is made up of four metrics: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). Using these four metrics we can calculate metrics such as Accuracy, Recall, Precision, F1 Score etc. Accuracy tells us how often the model makes correct predictions. It is calculated by dividing the total number of correct predictions by the total number of predictions. Recall is the ratio of correctly predicted True values to the total number of True values. Essentially, it tells us how often a True value is predicted as True. Precision is the fraction of relevant instances among the retrieved instances. F1 score is a metric that combines recall and precision by taking their harmonic mean.

We found that some researchers reported their results across a single evaluation metric only (e.g., accuracy), which is not an effective strategy when considering imbalanced datasets. Additionally, very few researchers considered computational effort as an evaluation metric for their systems. Furthermore, one of the findings that there is a lack of datasets built on new data, all the datasets used were old. Hence, to tackle this issue effectively and by at par with the current techniques of spammers to bypass current systems, updated datasets are required in this field. Finally, the implementation of traditional ML techniques has become very common and newer techniques like hybrid models, more use of DL techniques needs to be implemented.

We surveyed 31 papers and prepared a table that displays the methods used. It lists their strengths and weakness.

Methods Used	Strengths	Weaknesses
<b>Model named THEMIS based on RCNN [3]</b>	THEMIS model gives a high accuracy of 99.848%. It has a recall of 99% and precision of 99.664%. It performs better than CNN and LSTM. It has a low FPR which reduces the risk of filtering out the legitimate email to a greater extent.	THEMIS cannot detect phishing emails with no email header and only an email body.
<b>SVM and ELM [4]</b>	The proposed SVM spam detector yielded the best results. The difference in their accuracies is low but the difference in the speeds is very high. The results of these two machines were much better than any other classifier that was used on the same dataset.	ELMs essentially make use of Neural Networks. So it will have the same drawbacks as Neural Networks one of them being that Neural Networks need a large dataset to achieve a better accuracy.
<b>Incremental SVM [5]</b>	The retraining in this model helps to keep the model up to date and is ready to train it with new spam words.	The re-training could make the model take more time to run in the longer period.
<b>Clustering, J48, Naive Bayes, SVM, ANN, Decision tree [6]</b>	Naïve Bayes technique gives faster results and better accuracy over other techniques	SVM gives better accuracy than Naïve Bayes but takes more time to build a model.
<b>Polynomial filter + RF [7]</b>	Out of all algorithms used, Random Forest gave the highest accuracy and shows promising results.	The accuracy reduces when duplicate emails appear.
<b>LR, DT, NB, KNN and SVM [8]</b>	RF performs the best followed by KNN which takes less time to train the model.	RF takes much more time to train the model but gives higher accuracy and KNN gives slight less accuracy while training is fast. It is a trade-off between time and accuracy
<b>Random Forest [9]</b>	<ul style="list-style-type: none"> <li>a) It can handle a high number of features</li> <li>b) It can handle a mixture of binary, numeric and categorical features</li> <li>c) It can estimate which features are more important than others</li> </ul>	<ul style="list-style-type: none"> <li>a) Random Forest creates a lot of complexity as 100s of trees are created and their outputs are combined to give the required output.</li> <li>b) It requires a huge amount of time to train as it generates lots of trees and makes decision on the majority of votes.</li> </ul>
<b>KNN for data expansion + LSTM for classification [10]</b>	LSTM captures the induced statements in the message better. This method performs better than existing phishing email detection method as it improves accuracy, reduces false negative rate and false positive rate.	
<b>KNN, DT and NB algorithm. [11]</b>	The experimental results show high values of TPR (true positive rate) and precision and the false positive rate is controlled within an acceptable range when a suitable classifier is used. Best performance was of NB.	The model will need to be updated regularly to keep it up to date with the changes made by phishers.
<b>Hybrid DT + LR with FN Threshold [12]</b>	The proposed method shows high accuracy in detecting spam email compared to other methods and it outperforms prior research. The proposed method has less complexity.	As it considers all features of email, accuracy decreases due to irrelevant features. DT's performance can also improve as it is low.

<b>NB + Particle Swarm Optimization [13]</b>	The proposed method produces much better results than what Naïve Bayes would show on the same dataset. The PSO helps to optimize the Naive Bayes parameters which helps to improve the accuracy.	Naïve Bayes can be swapped out and be replaced by any other classifier which would help to improve the accuracy even more.
<b>Multinomial NB [14]</b>	Processing time of NB is in small training sets, hence suitable for real-time filtering.	Accuracy achieved is lower due to small dataset used in comparison to SVM and ANN.
<b>Fuzzy-C Means [15]</b>	Only Paper to propose a different model like Fuzzy-C M. Gives best result for overlapped data set and comparatively better than k-means algorithm.	
<b>Feature extraction using NLP and ANN for classification [16]</b>	The accuracy with which the mails were detected are 92.2%. ANN gives high accuracy for the given dataset.	ANN model requires large amount of data for training, such accuracy cannot be achieved with small datasets.
<b>ANN, Naïve Bayes, SVM, Decision tree and Random Forest [17]</b>	Neural Network Model provides a better result than other classifiers on the given dataset. Neural network model worked as the dataset is complex with high dimensionality, and missing data. SVM gave the second best accuracy	Interpretability of neural network is low compared to the other models. Due to correlation between variables all other algorithms except neural network and SVM performed poor. DT performed the worst.
<b>J48, SVM, and Naïve Bayes. [18]</b>	The accuracy of the model is good irrespective of which classifier is applied on it. Frequency based features attain high accuracy.	This model deals with only the body of the mail. It does not consider the header of the mail.
<b>NB, LR DT, KNN, RF and SVM. [19]</b>	SVM gives the highest accuracy when used with Doc2Vec for mails with and without header. Doc2Vec models performed better compared to tf-idf models in both header and without header.	Huge variations in experimentation implemented but only a small dataset was used, using different datasets would give a better comparison and result analysis.
<b>A CNN based model called CNNPD [20]</b>	This model achieves high accuracy, precision, recall and f-measure.	This model is currently designed to handle small datasets. Large datasets would not produce same type of results. The model cannot detect spam using fake URL.
<b>Genetic Programming (GP) + RF [21]</b>	GP is an evolutionary model that evolves classification models that are represented as trees. It is adaptable and its generated models are interpretable. SMOTE oversampling along with GP gives a better overall performance of accuracy.	GP when used by itself shows competitive results but it's not enough. Its accuracy, recall for spam and g-mean is worse than other algorithms.
<b>Data Insertion + RF classifier [22]</b>	This method is effective in increasing the number of phishing samples without changing the malicious attributes. It also solves the problem of spatial bias during model training and reduces difference in statistical characteristics between benign and malicious samples.	The constructed samples are missing some text features or samples and there's a certain randomization in the data inserted in the sample which changes the sample's natural language processing result. The model space bias can be worked on further.
<b>NB [23]</b>	-	The paper does not implement the model, which does not help to understand how the model performed on datasets.



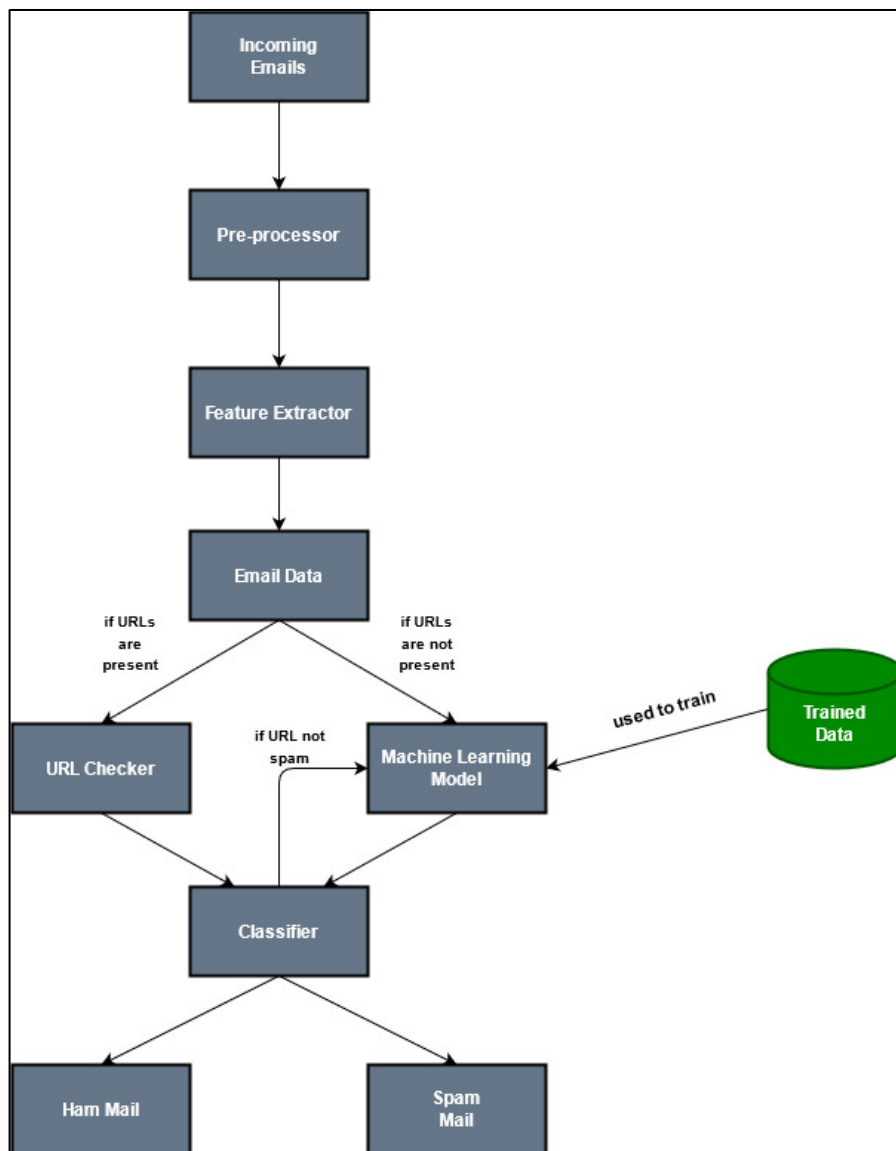
<b>K-nearest community classifier is used. [24]</b>	This model works well when all features are considered and the LCDBoD is used without closure.	Not as good as other classifiers like Bayesian or Vector Space.
<b>NB, SVM, DT, LSTM and ANN [25]</b>	DL models out-performed the ML models in classification	One hot encoding was used, if other methods would have been used then ML models would have performed better.
<b>Proposed a classification model using ML [26]</b>	The proposed model is a self-learning system that is customizable to each user and based on their dataset. The accuracy will only increase depending on the size of the dataset.	The model's accuracy depends on the size of the dataset rather than on the algorithm. There are algorithms which give better accuracy even if the dataset is small.
<b>Cuckoo-search SVM (CS-SVM) [27]</b>	CS-SVM performs better compared to traditional SVM model.	The model cannot work on a distributed system.
<b>NB and SVM used for classification [28]</b>	SVM and NB as per the paper provide better accuracies compared to other models in their literature.	This is just a model. There are no experimental results to show that this method would work.
<b>Proposed an header based system [29]</b>	The paper suggests that only looking at the email body is not enough and hence header based approach can be used to get potential features.	-
<b>Classification based on Geographical Location of sender [30]</b>	-	Phishers could evade this method by posing as small banks. Another way they could evade is by compromising a bank's system by send phishing messages or hosting a fake site.
<b>URL based feature extraction + NB classification [31]</b>	It yielded a probabilistic phishing detection framework that can quickly adapt to new attacks with reasonably good true positive rates and close to zero false positive rates. The system exploits the lexical and host-based features of URLs.	This method would not work for mails that do not contain URLs.
<b>URL based feature extraction and classification [32]</b>	Lexical and textual analysis of email helps to effectively classify email. FPR rate is reduced.	Accuracy of this method is lower than compared to other ML techniques.
<b>Weighted SVM [33]</b>	Improvised WSVM exhibits lower misclassification rate than SVM. In Improvised WSVM, when weights are produced by KFCM algorithm, the result is more promising.	System performance can still be improved in terms of accuracy and precision. There may be inconsistency in case of larger number of iterations compared with the average number of data.

**Table 2.1: Strengths and Weaknesses**

## Chapter 3

### Analysis and Design

Based on the conducted literature survey, we decided to create an email fraud detection model which uses ML and DL techniques to successfully categorize an email as fraud or legitimate. The first step was to create an architecture for the model as shown below:



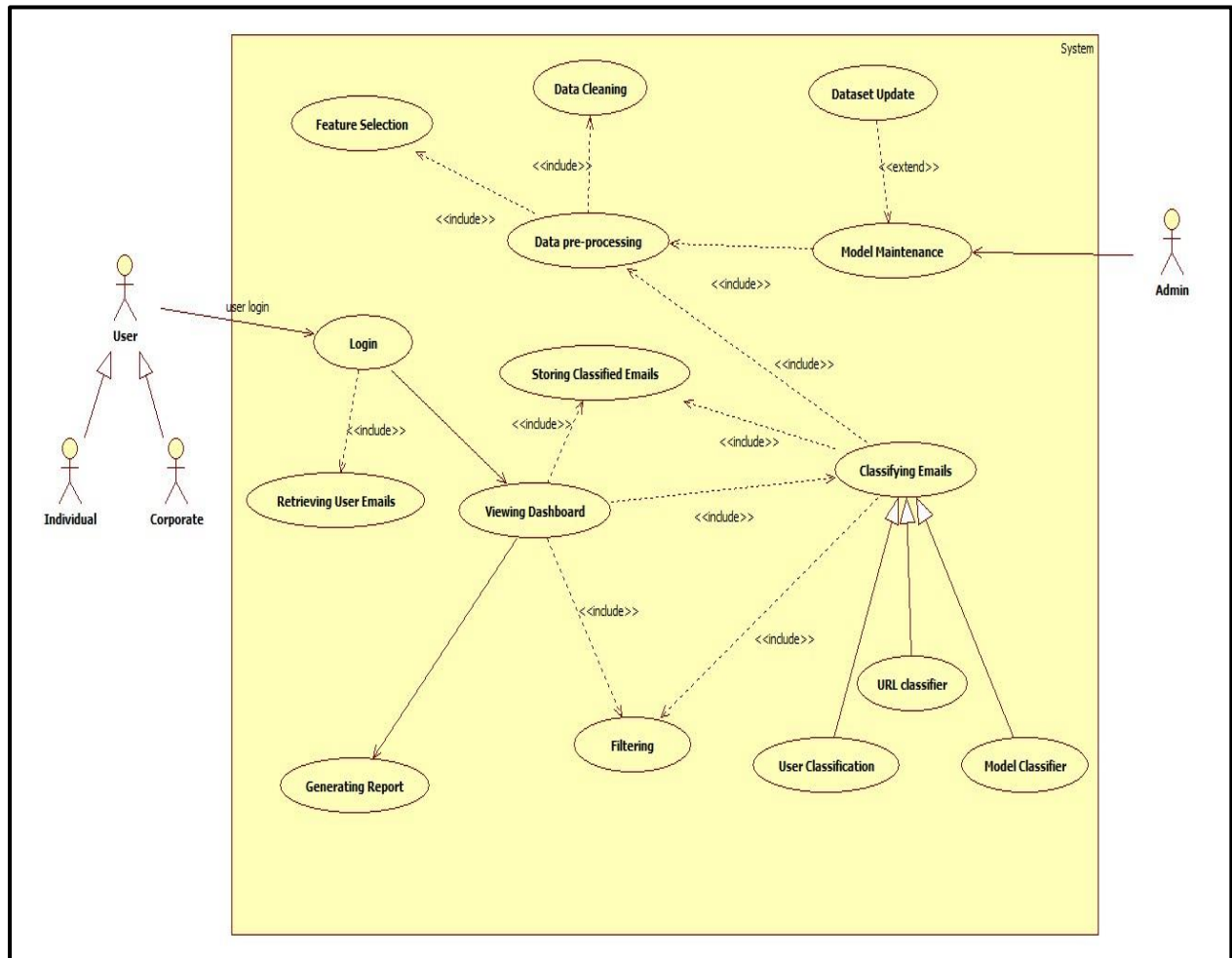
**Figure 3.1: Architecture of the model**

The model receives incoming emails and pre-processes them using techniques such as Word Tokenization and Stop Word Removal. Once the pre-processing is done, the features are extracted and stored separately as header and body. Using the header of the email we check if the sender is a serial spammer or not. If the sender's domain is legitimate, then we proceed to examine the body of the email. First, we check whether URLs are present and if so, whether they are black-listed or not. If there are no URLs, then we apply a machine learning model on the contents of the body.

This model helps us to determine whether the email is fraudulent or not. After the checking of both header and body is complete, we categorize the email as legitimate or fraud.

Following are the UML diagrams for our model:

## 1. Use Case Diagram

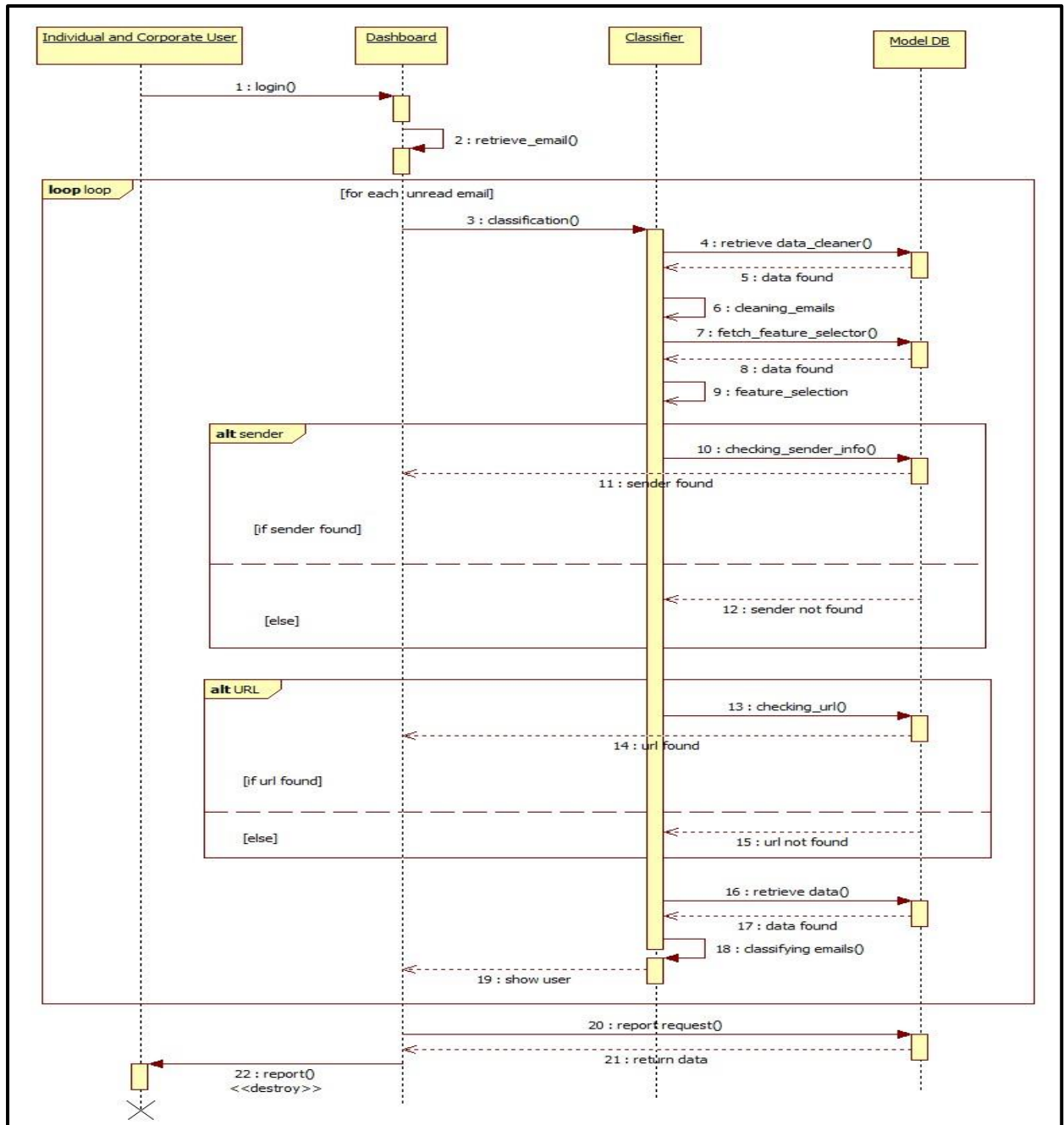


**Figure 3.2: Use Case Diagram**

Figure 3.2 shows the use case diagram for the project. A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped.

For our project, the users can be the general public, as well corporate, and the admin will be the one maintaining the project. The use case diagram shows the flow, from a when a user login till the whole process is done, and the final results are displayed. The user only needs to worry about providing the right credentials, and the other process takes place on it's own.

## 2. Sequence Diagram



**Figure 3.3: Sequence Diagram**

Figure 3.3 shows the Sequence diagram for this project. UML Sequence diagrams, are used for to explain the flow of the whole process. Each step, is shown under the sequence diagram. In our sequence diagram, we have 4 components, namely the users, Dashboard, Classifier and the Database. Moreover, the flow starts when the user inputs his/her credentials and emails are fetched from their email account, the dashboard has the user interface, from which the user interacts with the model. The model, then fetches, pre-processes, etc. the fetched emails and classifies them, with the helped of the saved model. Finally, the user get's the output in the dashboard.

### 3. Class Diagram

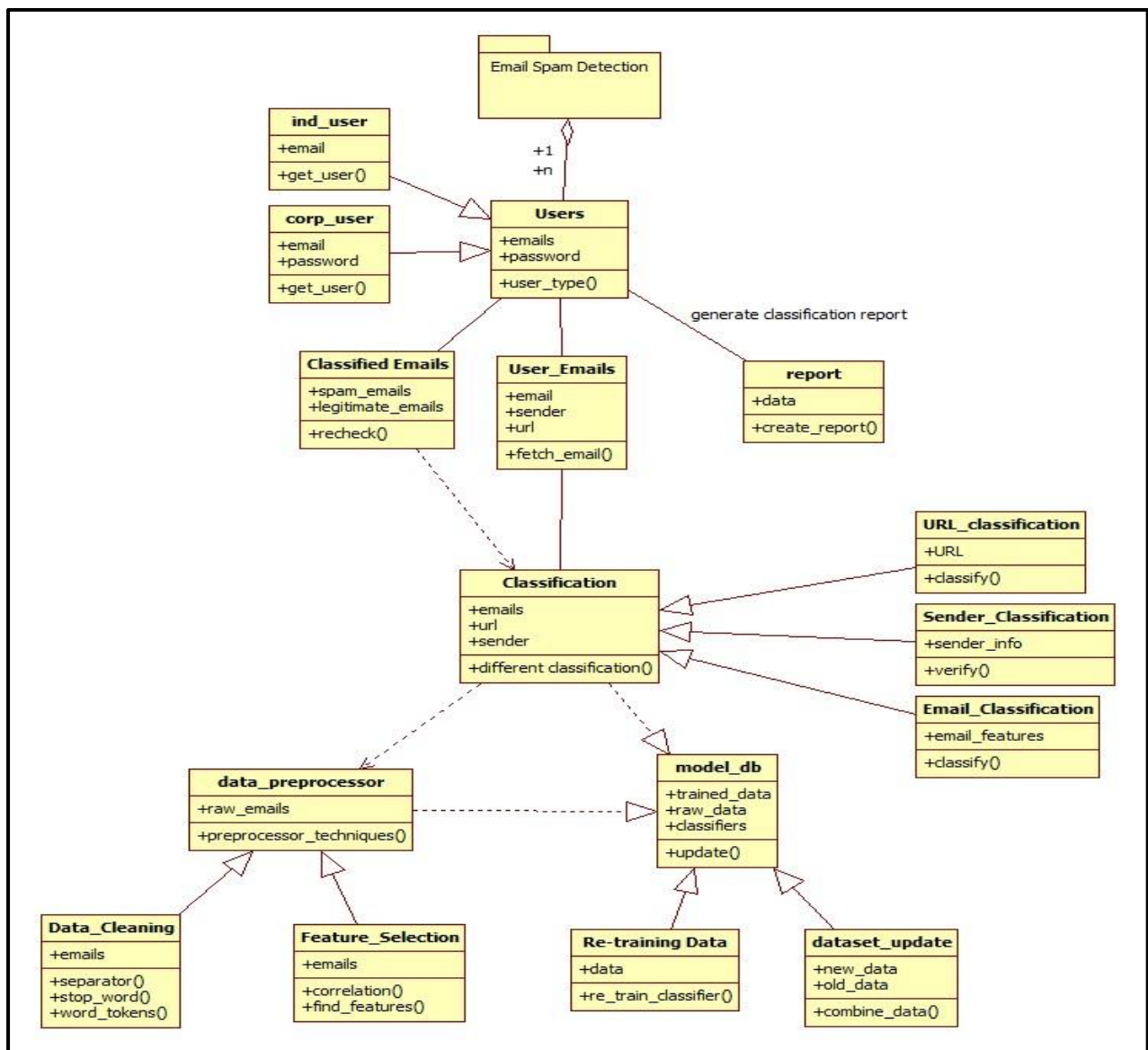


Figure 3.4: Class Diagram

The UML Class diagram for our project is displayed in figure 3.4. Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The project class diagram consists for a class for each different process. The main class being the classification. This diagram, explains the relationship between the different processes, users, etc.

## **Chapter 4**

### **Methods Implemented**

The methods that we have implemented till now are:

1. Data Pre-processing
2. Correlation
3. Word Embeddings
4. URL Classification
5. Decision Tree
6. Random Forest
7. Naïve Bayes
8. Support Vector Machine
9. K-Nearest Neighbors
10. Convolutional Neural Network
11. General Architecture

## 4.1 Data pre-processing

Data pre-processing is an important step to convert the given data into the desired structure.

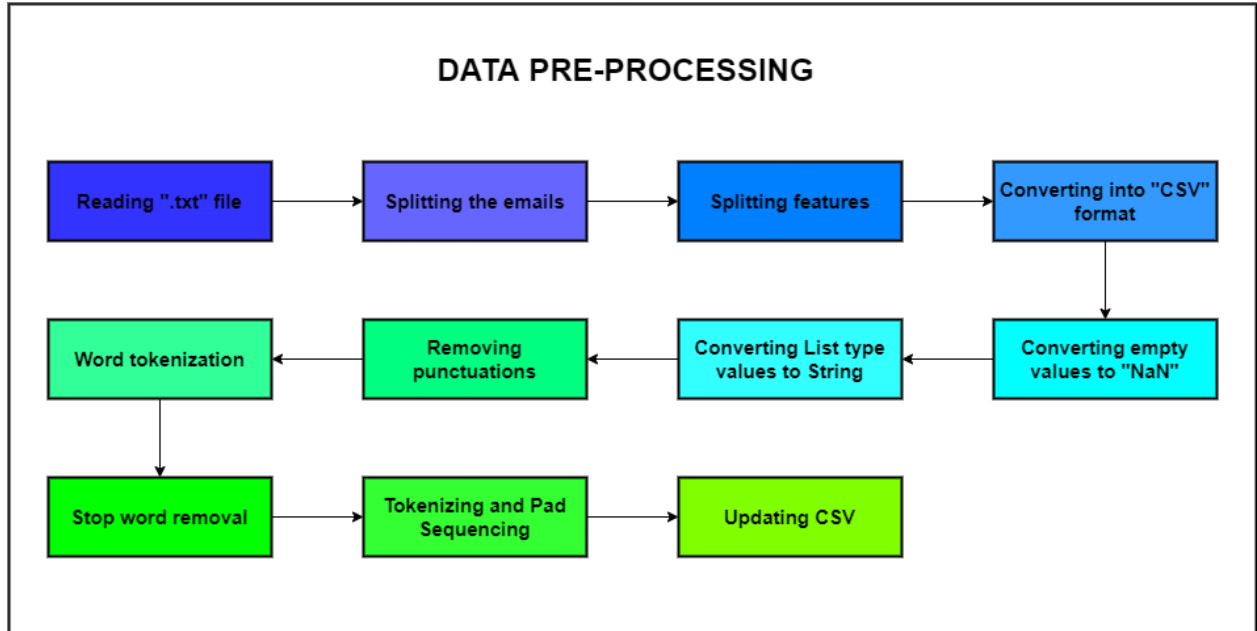


Figure 4.1: Data Pre-processing

Data pre-processing techniques applied on our dataset are as follows:

- **Reading .txt file:** The .txt files of the various datasets acquired were read using python libraries.
- **Splitting the Emails:** The emails present in these files were aggregated all together in one single file, or multiple files containing one email each, in this step the splitting of the took place and each email – as a whole was taken.
- **Splitting features:** After acquiring the individual emails the features of emails like header, body were each separated.
- **Converting to CSV format:** Under this step, each Email and its features were stored in the newly created .csv file for further analysis and processing.
- **Converting Empty values to NaN:** Not all emails have all features. So this empty values need to converted to NaN, as the pandas interpreter can't recognize such empty fields.
- **Converting List Type values to String:** The values of the features acquired were in python's list datatype, these values need to converted into strings for further processing.

- **Removing Punctuation:** Removal of punctuation marks (full stop, comma, question mark, etc.) in the given data.
- **Word Tokenization:** With the help of word tokenization, we can convert our text into a list of words. A 2D list is made, in which each sub-list constitutes one news article entry. We used the Natural Language Kit (NLTK) module of Python for Word Tokenization.
- **Stop Word Removal:** Stop word is a commonly used word such as ‘a’, ‘an’, ‘the’, ‘in’, ‘for’, ‘of’, etc. which need to be ignored for getting better accuracies and making the training and testing data consistent. We have used NLTK for stop word removal.
- **Tokenizing and Pad Sequencing:** Finally, we tokenize our text corpus into a vector of integers based on the embeddings, and then padding is done to have an equivalent size vector. Padding size is determined from the maximum sentence length found above.
- **Updating CSV:** After completing all the above steps the csv file is updated with the new cleaned values of the features.

These steps mentioned, are not only data acquiring and storing steps, but also the pre-processing steps which are required for the classification step.

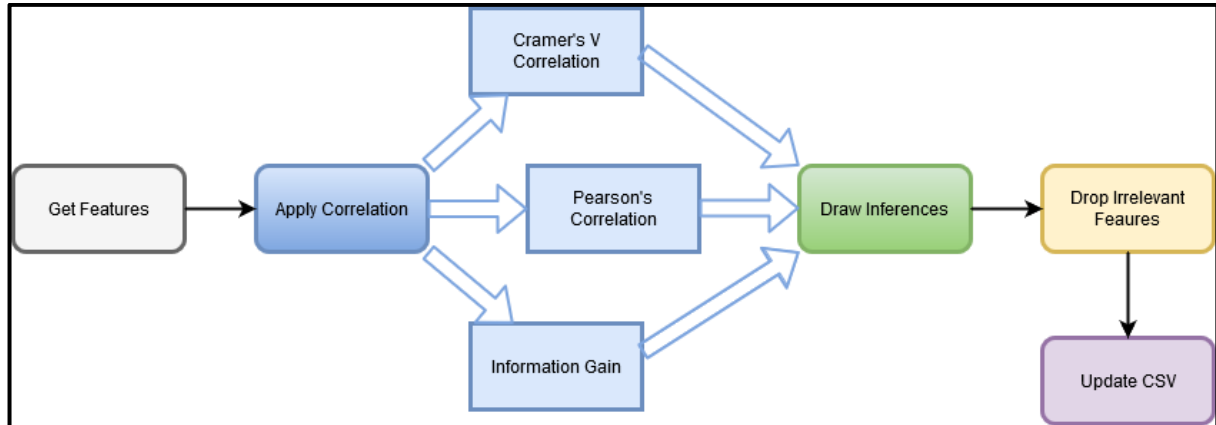
## **4.2 Correlation**

Correlation is used to test relationships between quantitative variables or categorical variables. In other words, it's a measure of how things are related. The study of how variables are correlated is called **correlation analysis**.

Correlations are useful because if we can find out what relationship variables have, we can make **predictions about future behavior**.

The figure below shows how we applied correlation to our model.





**Figure 4.2: Correlation Techniques**

After extracting the features, we applied three correlation techniques to determine which features would be useful for us. They are:

1. Cramer's V Correlation
2. Pearson's Correlation
3. Information Gain

#### **4.2.1 Cramer's V Correlation:**

Cramer's V works on the principle of Chi-square Test of Independence.

The Chi-square test of independence determines whether there is a statistically significant relationship between categorical variables. It is a hypothesis test that answers the question Do the values of one categorical variable depend on the value of other categorical variables? The Chi-square test of association evaluates relationships between categorical variables. Like any statistical hypothesis test, the Chi-square test has both a null hypothesis and an alternative hypothesis.

**Null hypothesis:** There are no relationships between the categorical variables. If you know the value of one variable, it does not help you predict the value of another variable.

**Alternative hypothesis:** There are relationships between the categorical variables. Knowing the value of one variable does help you predict the value of another variable.

Chi squared can be calculated using the below formula:

$$\chi^2 = \sum (O - E)^2 / E$$

where  $O$  represents the observed frequency.  $E$  is the expected frequency under the null hypothesis and computed by:

$$E = \frac{\text{row total} \times \text{column total}}{\text{sample size}}$$

Cramer's V is calculated using the below formula:

$$\phi_c = \sqrt{\frac{\chi^2}{N(k-1)}}$$

$\phi_c$  denotes Cramér's V;

$\chi^2$  is the Pearson chi-square statistic from the aforementioned test;

$N$  is the sample size involved in the test and

$k$  is the lesser number of categories of either variable.

#### **4.2.2 Pearson's Correlation:**

Pearson's correlation coefficient ( $r$ ) is a **measure of the strength of the association** between the two variables.

The first step in studying the relationship between two continuous variables is to draw a scatter plot of the variables to check for linearity. The correlation coefficient should not be calculated if the relationship is not linear. For correlation only purposes, it does not really matter on which axis the variables are plotted. However, conventionally, the independent (or explanatory) variable is plotted on the x-axis (horizontally) and the dependent (or response) variable is plotted on the y-axis (vertically).

Positive correlation indicates that both variables increase or decrease together, whereas negative correlation indicates that as one variable increases, so the other decreases, and vice versa.

Pearson's correlation is calculated using the below formula:

$$r = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}}$$

Where:

$N$  = the number of pairs of scores

$\Sigma xy$  = the sum of the products of paired scores

$\Sigma x$  = the sum of  $x$  scores

$\Sigma y$  = the sum of  $y$  scores

$\Sigma x^2$  = the sum of squared  $x$  scores

$\Sigma y^2$  = the sum of squared  $y$  scores

#### **4.2.3 Information Gain:**

Information gain works on the principle of entropy.

Information Gain, or IG for short, measures the reduction in entropy or surprise by splitting a dataset according to a given value of a random variable. A larger information gain suggests a lower entropy group or groups of samples, and hence less surprise

Entropy comes from information theory. The higher the entropy the more the information content.

$$\text{Entropy} = \sum_i -p_i \log_2 p_i$$

Information gain tells us how important a given attribute of the feature vectors is.

Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree.

Entropy lies between 0 and 1 – Where 0 indicates no surprises and 1 indicates surprises.

Information gain also lies between 0 and 1.

The information gain is calculated for each variable in the dataset. The variable that has the largest information gain is selected to split the dataset. Generally, a larger gain indicates a smaller entropy or less surprise.

$$\text{Information Gain} = \text{entropy}(\text{parent}) - [\text{average entropy}(\text{children})]$$

### **4.3 Word Embeddings**

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

The challenge comes in understanding the context, and whether two words like good and great are similar or not.

The word embeddings work on the principal of cosine similarity, and the objective is that for words which have similar context, their vectors should have an angle close to zero, or cosine angle close to one.

We have use the Word2Vec model for implementing Word Embeddings,

Word2Vec is a statistical method for learning a word embedding from a text corpus. It is provided by Google, and is trained on over 3 million words from Google News data. The vectors in this model have dimensions of 300, and the large quantity of words means it is easier to understand the context between words.

For our project we have used the GoogleNews Word2Vec model, and implemented as follows:

Building Vocabulary and finding maximum sentence length: A vocabulary is built for the training and testing data, with the help of the tokens created in the word tokenization step, this vocabulary serves as a corpus for holding the words. Greater the vocabulary size better would be the results. The maximum sentence length helps us to assign the appropriate embedding values to the CNN model.

Once the vocabulary is built it is then used further to feed the Random Forest classifier.

### **4.4 URL Classification**

Spamming through emails has become very common and so has the increase of using fake or malicious URLs in those emails to redirect the user and steal his/her personal information. Keeping this in mind we have included a URL classifier in our model.

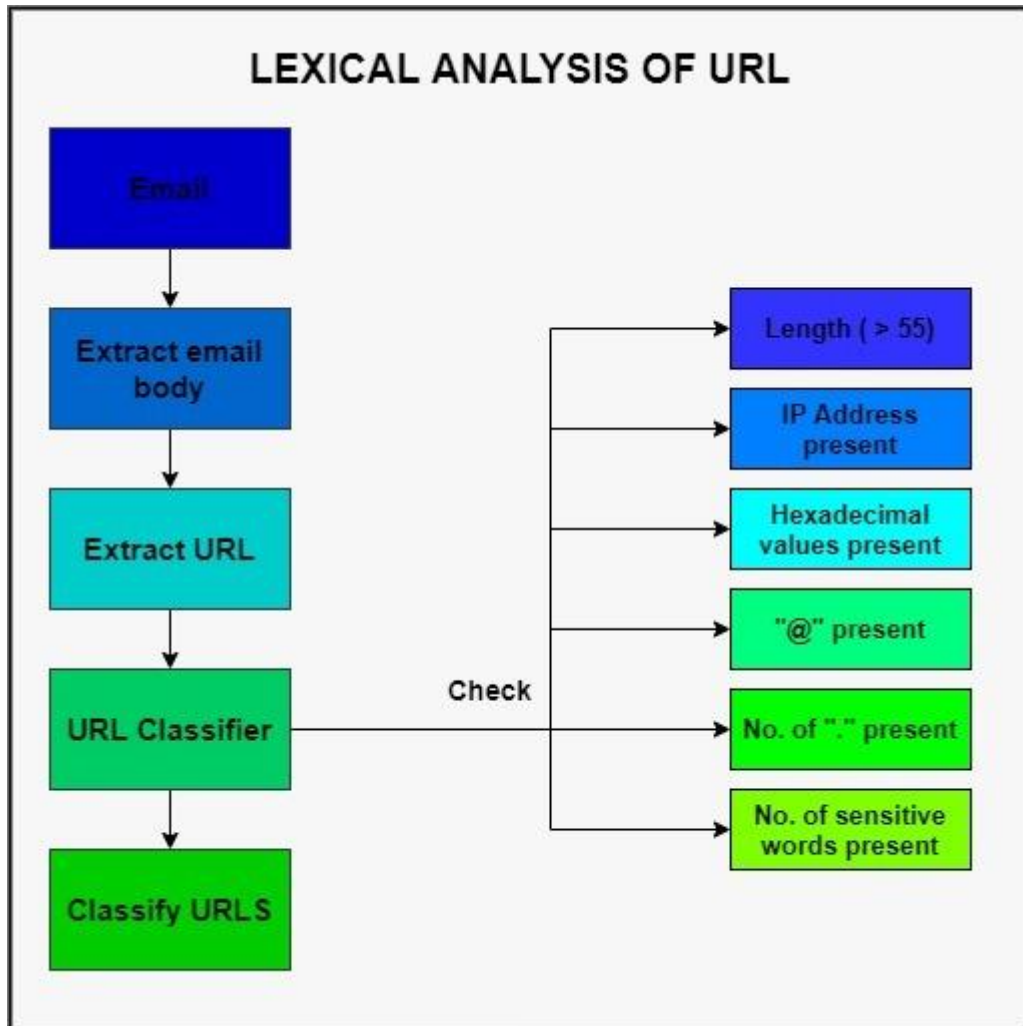


Figure 4.3: Lexical Analysis of URLs

This module does lexical analysis of the URLs. Lexical analysis of URLs checks the lexical features of the URL like URL length, finding certain characters, etc.

We first check the body of the email and extract any URLs present in it and pass it through the URL classifier.

In the URL classifier, we have 6 parts to check the URL. They are as follows:

1. **Length > 55:** If length of the URL is greater than 55 characters, 1 would be assigned in the list, as it is suspicious.
2. **IP address:** If IP address is found in the URL it can be suspicious and hence a 1 is assigned in the list.
3. **Hexadecimal Values present:** If hexadecimal values are present in the URL then it is suspicious and hence a 1 is assigned in the list.
4. **'@' present:** If the '@' symbol is present in the URL then that URL can be considered suspicious and we assign a 1 in the list.

5. **No. of ‘.’ Present:** If more than 5 ‘.’ Are present in the URL then it can considered suspicious and a 1 is assigned in the list.

6. **No. of sensitive words:** If more than 1 sensitive list from ('secure', 'account', 'update', 'login', 'signin', 'sign-in', 'banking', 'confirm', 'verify', 'suspend', 'username', 'password') are present in the given URL then a 1 will be assigned in the list.

Once the URL is checked through this a list is created having 0s and 1s for each property checked. Whenever we find more than three 1's in the list out of 6 values. Hence threshold set is greater than three 1's.

If the number of 1s found in the list is greater than this threshold then we classify the URL as a spam URL and hence the whole email is classified as spam.

If the URL is not classified as spam then the email would be further analysed and classification techniques would be applied.

#### **4.5 Decision Tree**

Decision trees, one of the simplest and yet most useful Machine Learning structures. Decision trees belong to a class of supervised machine learning algorithms, which are used in both classification and regression predictive modeling.

The goal of the algorithm is to predict a target variable from a set of input variables and their attributes. The approach builds a tree structure through a series of binary splits (yes/no) from the root node via branches passing several decision nodes (internal nodes), until we come to leaf nodes. It is here that the prediction is made. Each split partitions the input variables into feature regions, which are used for lower splits.

The Decision Tree model that we chose for implementation is ID3. ID3 uses entropy and Information Gain for predicting the parent node and the subsequent leaf nodes at each step.

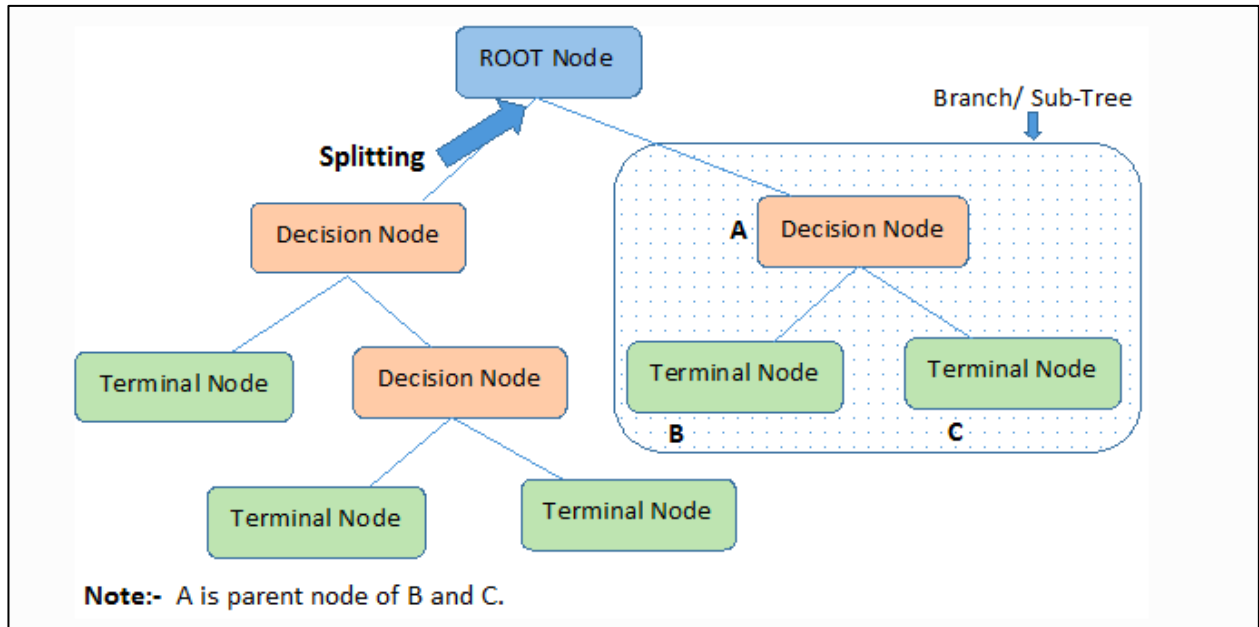


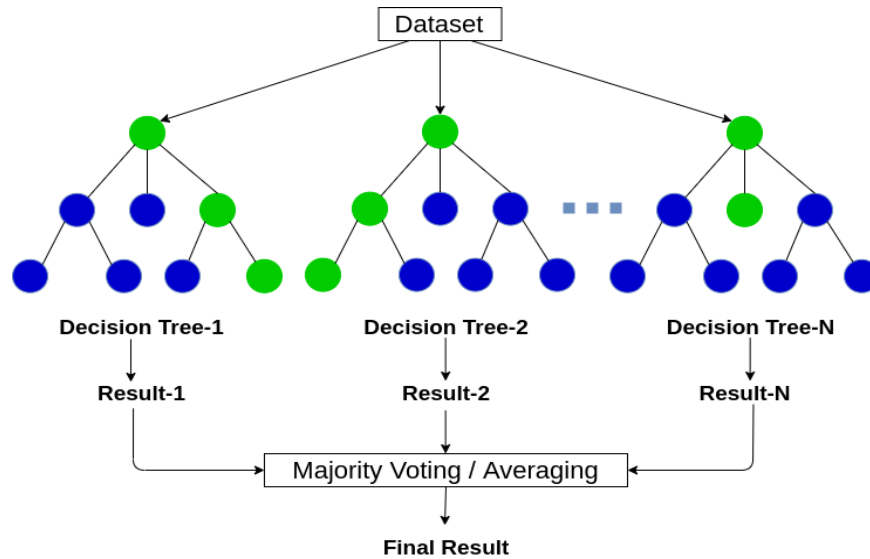
Figure 4.4: Decision Tree

## 4.6 Random Forest

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/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

It uses Decision tree as base for classification.

We have used the sklearn module of python to implement, we can specify the split method in the 'criterion' parameter.



**Figure 4.5: Random Forest**

As we have studied about ID3, we felt more comfortable with going ahead and implementing the entropy method (information gain). ID3 uses information gain as its attribute selection measure.

Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

## **4.7 Naïve Bayes**

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.

Bayes Theorem is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula is given as:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

The diagram shows the formula for Bayes Theorem with labels for each term:

- $P(A|B)$ : Probability of A occurring given evidence B has already occurred
- $P(B|A)$ : Probability of B occurring given evidence A has already occurred
- $P(A)$ : Probability of A occurring
- $P(B)$ : Probability of B occurring

**Figure 4.6: Bayes Theorem.**



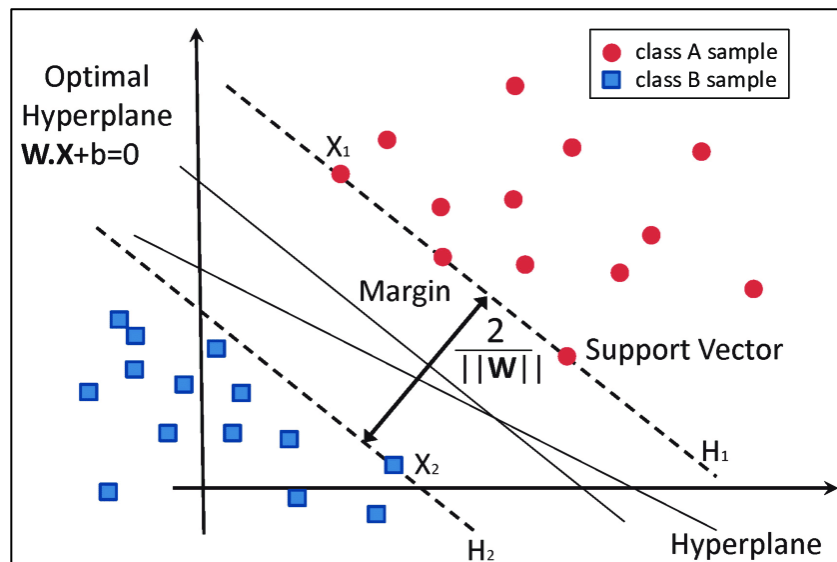
As shown, in figure 4.6, naïve bayes works on the formula of bayes theorem, where we calculate the probability of an even A occurring in terms of event B.

Naïve Bayes has many different version, the one implemented by us in this project is the Gaussian Naïve Bayes, it works on the principle, on Gaussian Distribution. Gaussian Distribution is also known as Normal Distribution. This method gives is useful for continuous and binary data classification. Instead of getting an absolute result as a 1 or 0, we get a probability value ranging from 0 to 1, and using another function we determine which probability is higher and assign that class label to the particular input.

#### **4.8 Support Vector Machine**

Support vector machine is a supervised learning system and used for classification and regression problems. Support vector machine is extremely favored by many as it produces notable correctness with less computation power. It is mostly used in classification problems.

In SVM, the data points are plotted based on their attributes, and then the hyperplane, which is at the maximum distance from the different sets of the data points, is created. The points on opposing sides of the hyperplane are classified into different categories. The points closest to the hyperplane are classified into different categories. The points closest to the hyperplane on each side are known as the support vectors.



**Figure 4.7: Support Vector Machine**

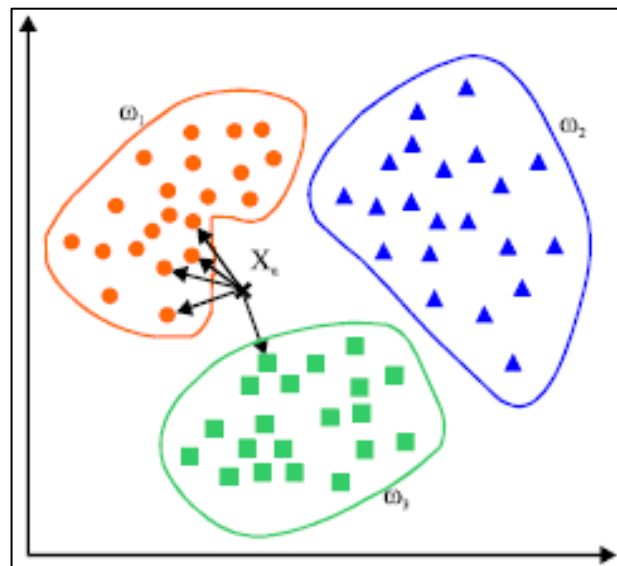
## **4.9 K-Nearest Neighbors**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

First select the number K of the neighbors, and K random data points are selected as data cluster centers. All other data points are classified into one of these clusters based on their distance from the centers. Once the clusters have been created, the centers are re-calculated, and if necessary any data points are reassigned. This process is continued till stable clusters are obtained, at which point any new data points can be easily classified.

As shown, in figure 4.8, the value of k selected in this example is 3, hence we get 3 different clusters of data, each corresponding to a different class label.

For our project, we first implemented KNN with a wide variety of K value, and found the best possible K value, which gave the highest accuracy, in our case the value of K selected is 5.



**Figure 4.8: K-Nearest Neighbours**

## 4.10 Convolutional Neural Network

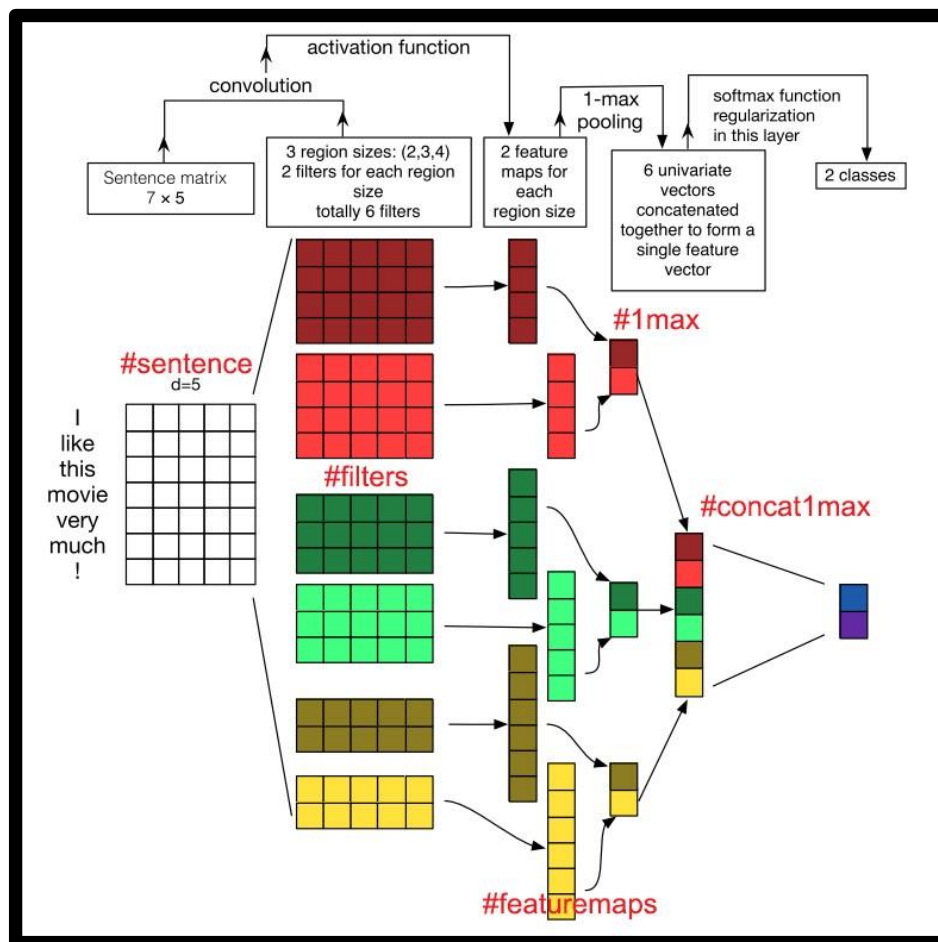


Figure 4.9: Convolutional Neural Network

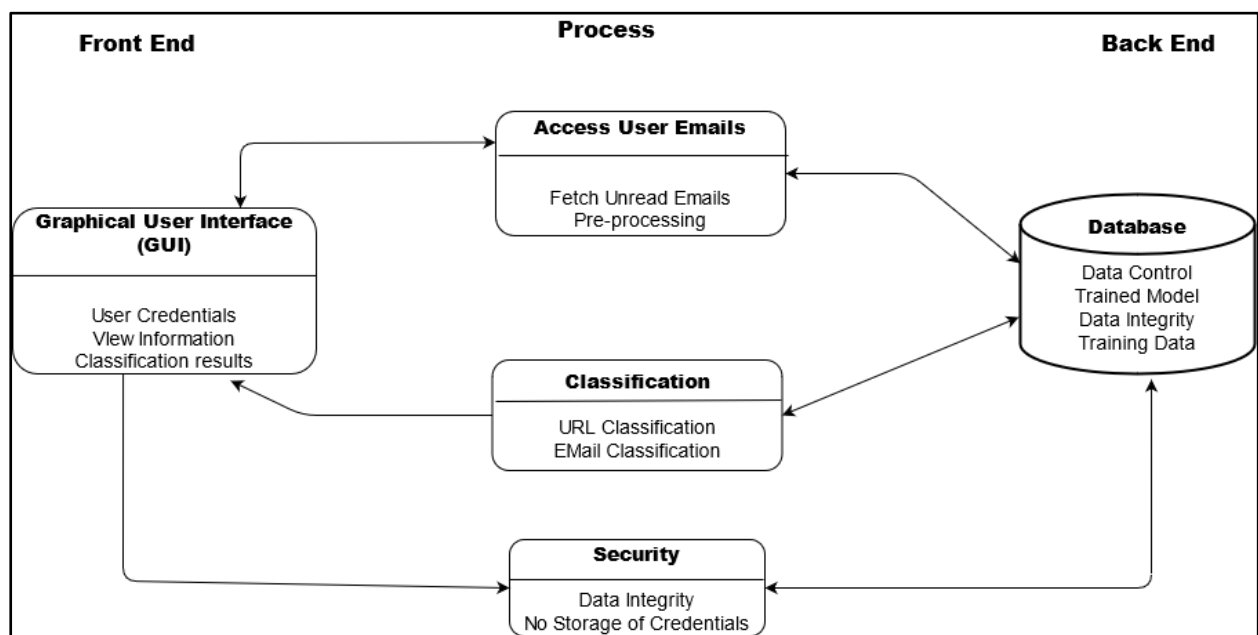
Convolutional Neural Network (CNN), is a type of Deep Learning algorithm, it was firstly introduced to deal with Image data, and its various tasks, like image processing, etc. Due to recent advancements in technology CNN can now also be used for other tasks and, text classification is where it is creating a lot of hype, due to it's higher accuracies compared to it's Machine Learning algorithm counterparts.

CNN has 3 layers, unlike any other Neural Network model, the input layer, the hidden layer, and the output layer. Now, as the name suggests, the hidden layer consist of convolutional layers, where as shown in figure 4.9, different filters and filter sizes are used to convolve, the text matrix, into smaller sizes of relevant information. At each convolutional layer, different features are extracted which makes the algorithm, get the context in different types. Now once, convolution is applied, an activation function is applied, which, makes the output an absolute value, either a 0 or 1.

After this done, the data is passed through a max-pooling layer, where the matrix is again made more smaller by just accepting, a value which is the highest of the remaining. Finally the data is passed through a flatten layer, where the layer converts a three-dimensional layer in the network into a one-dimensional vector to fit the input of a fully-connected layer for classification. This process is repeated multiple times, to get higher accuracies as each new time the data is passed through the filters, more and more complex features are extracted out of it.

In our project, we have implemented CNN, we have used 5 filter sizes, and 5 epochs for the processing. This have us the some better results compared to other configurations that we tried. CNN, model took the longest to train, but is also worth saying that it gives the best results and by a huge difference.

#### **4.11 General Architecture and Flow of Classification**

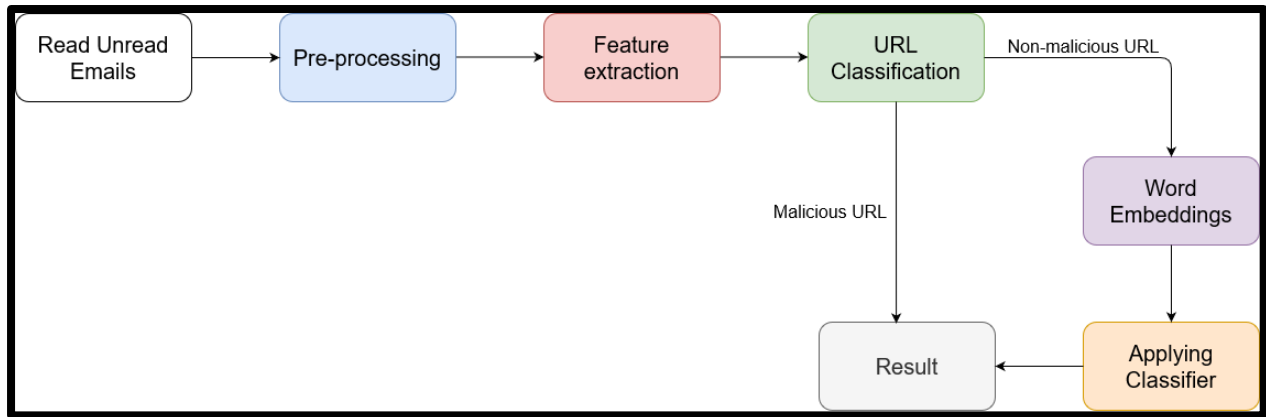


**Figure 4.10: General Architecture**

Figure 4.10 shows the general architecture, in how the data flows through multiple processes and finally, we get our result as spam or ham.

In the process, first the unread user emails are extracted from his email account and stored in temporary files, in the local storage, then the data is pre-processed, where first the emails are separated into different text files, and then each email is read and other techniques are used to clean the data, like stopwords removal, word tokenization, etc. After this, we extract the features,

and then pass it through our URL classifier, where a value if there are malicious emails detected in the email body, then the email is directly classified as spam, or else the email is then passed to the classifier, which in our model is CNN, where it is classified and finally we get our classified as spam or ham.

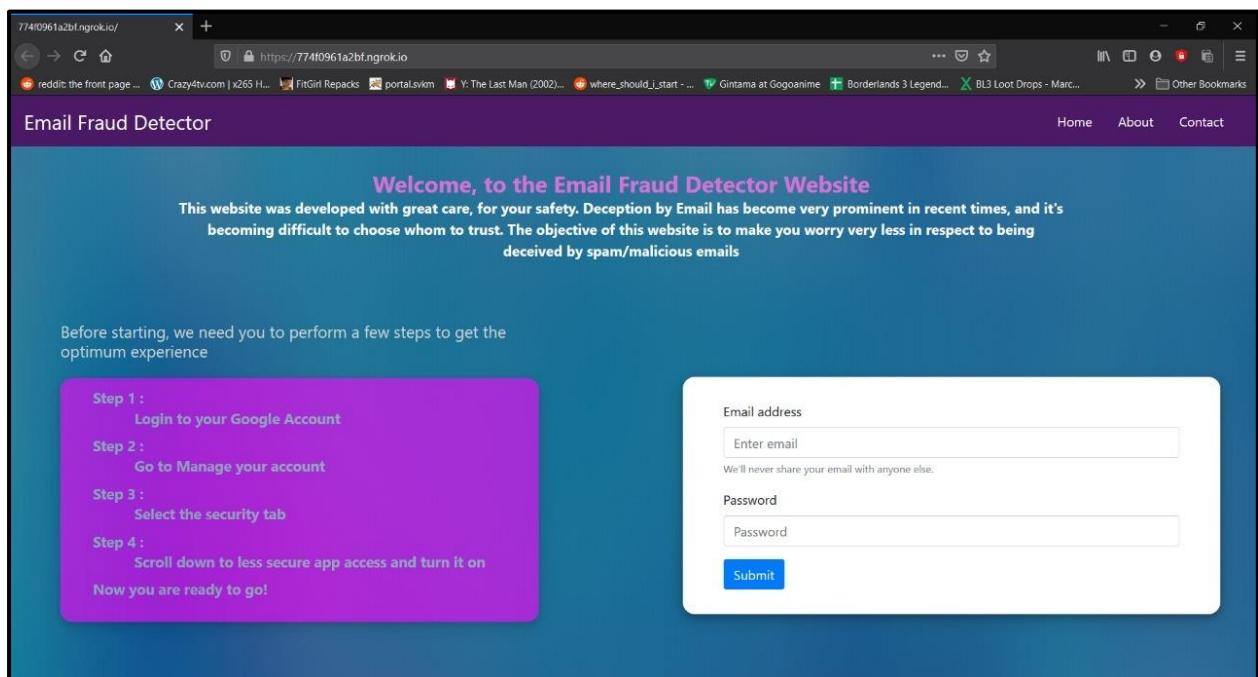


**Figure 4.11: Flow of Classification**

Figure 4.11 shows the general flow of classification implemented in the project.

## 4.12 Implementation Screenshots

We have implemented the front end of our project as a website, which can be accessed by anyone on the web. For implementing it we have used Django as our front end. Django is a python framework for creating and deploying websites with python backend. As discussed, in the previous chapters, we have used python as our backend, and hence we are using Django for robust implementation.



**Figure 4.12: Website Homepage**

Fig 4.12 shows the home page of our website. The aesthetics are kept elegant yet simplistic. A navigation bar, for quick traversal through the website has been done, we can also see, that a few instructions are listed on the left pane of the website. As, we are using the user's email account, due to Gmail's privacy policy, access to less secure app access needs to be turned on, to experience the power of our website. Moreover, the right pane, has the email address box, and password field, where the user's needs to enter their email address, which of course we won't be storing, and with the help of our backend we will enter their account, access the unread emails and after our classification process is complete, we finally present them to the user. No data is stored in the backend or any other database, the credentials entered are one time use only, and cannot be fetched again.

Index	Subject	Label
1	RE: Extra Tutorial scheduled of Subject Maths II (DIV E) E1 Batch on Monday, 12-03-2018	1
2	FW: Blazer alteration distribution	1
3	RE: Blazer alteration distribution	1
4	Extra lecture scheduled of Subject Engineering Mechanics II (DIV E) on Wednesday 14-03-2018	0
5	FW: Cultural Night	0
6	Change in class room on Saturday, 16-03-2018	1
7	RE: Change in class room on Saturday, 16-03-2018	0
8	Change in class room on Friday, 16-03-2018	0
9	Change in class room on Saturday, 24-03-2018	0
10	Class room for M2 Test of Subject CP II Tomorrow, 20-03-2018	0

**Figure 4.13: Results Page**

The next figure is the results page of our website, where the user, will be shown all the unread emails and the subject of those emails, to identify those emails, and a little tag, of 0 or 1, where 0 indicates, a safe/ham email, and a 1 indicates a spam/malicious email. For a better simplicity and less confusion, we have also indicated the spam emails, in red color and the ham emails in green color.

For security reasons, we have neither created a database nor storing any user information on our end. Every time, a user wants to use of our services, they will need to enter their credentials, at all times.

## Chapter 5

### Results and Discussion

We found and selected 6 datasets which contain a large number of emails, both fraudulent and legitimate. The selected datasets are: IWSPA, SpamAssassin, SpamBase, PhishTank, LingSpam and TREC 2007. We decided to convert these datasets into a standard format which could be used for applying the various methods.

#### 5.1 Data Pre-processing

Initially, the emails were stored as separate text files. These text files contained the header and body of the email together. Upon pre-processing, we managed to store all the emails in a single dataset and split the parts of the email into separate features such as From, To, Received, CC, Subject, Body etc. The figure below shows how the dataset looks after pre-processing.

	From	To	Date	Message ID	CC	X Mailer	Content Type	Subject	MIME Ver	Received	Body	Attachment	Label
0	nannan	To nannan	"', #####	<WYADCK	nan	Microsoft	multipart	generic cia	1	nan{'from'	html body	nan	1
1	nannan	Ya nannan	"', #####	<46191DC	nan	nan	text/plain;	typo debia	1	nan{'from'	hi ive upda	nan	0
2	nannan	Sh nannan	"', #####	<2007040	nan	dvgvhnrv,	multipart	authentic v	1	nan{'from'	mega auth	nan	1
3	nannan	St nannan	"', #####	<2001901	nan	Mutt 1.0.1	text/plain;	nannice ta	1	nan{'from'	hey billy re	nan	1
4	nannan	Ch nannan	"', #####	<001c01c7	nan	Microsoft	multipart	trembling s	1	nan{'from'	system ho	nan{'filena	1
5	nannan	Bo nannan	"', #####	<000f01c7	nan	Microsoft	multipart	duty	1	nan{'from'	program c	nan{'filena	1
6	nannan	B. nannan	"', #####	<01c77a0	nan	Microsoft	multipart	theorize	1	nan{'from'	glad see y	nan	1
7	nannan	Es nannan	"', #####	<6526097	nan	nan	Multipart	theorize g	1	nan{'from'	html body	nan{'filena	1
8	nannan	Re nannan	"', #####	<e8f601c7	nan	Microsoft	multipart	losing weig	1	nan{'from'	hoodialife	nan	1
9	nannan	Jo nannan	"', #####	<9894014	nan	nan	text/plain;	nanrnan co	1	nan{'from'	hi use r fin	nan	0
10	nannan	Bill nannan	"', #####	<01c77a0	nan	Microsoft	multipart	smile	1	nan{'from'	good dayv	nan	1

Figure 5.1: Dataset after pre-processing

#### 5.2 Correlation

After pre-processing, we obtained 12 features for each email. Now we have to apply correlation techniques to the dataset so that we can shortlist the features that are useful to our model. We applied 3 correlation techniques: Cramer's V, Pearson's and Information Gain. From the results of correlation, we found out that 'From', 'To', 'Date', 'Content Type', 'Message ID', 'Received', 'Body' and 'Subject' gave high correlation values whereas 'CC', 'X-Mailer', 'MIME' and 'Attachment' gave low correlation values. So, we do not take these features into consideration for our model.



### **5.3 Random Forest on every dataset**

Then we applied our first classification technique, Random Forest on the datasets. The table below shows the results obtained for each dataset:

Dataset	Accuracy obtained (%)
IWSPA – full Header	91.8
SpamAssassin	87.245
LingSpam	86.35
IWSPA No Header	91.09
SpamBase	95

**Table 5.1: Random Forest results**

We obtained the highest accuracy for SpamBase dataset, with all datasets having accuracy above 85%. PhishTank dataset contains URLs instead of emails, hence we did not apply Random Forest on that dataset. TREC 2007 dataset gave us a memory error.

### **5.4 URL Classifier**

We applied URL Classifier on the dataset and we found a list of fake URLs. One such URL is shown below:

[http://www.paypal.com.fr.cgi.bin.webscr.cmd.flow.session.ycwnzphbcryskgq9zzetmxi8zg51bfu4zhgnlq2wntjgx8jlj7gdispatch.c70bbe4152786147242b0tb71efa252acz.waterdamagerestorationhouston.com/update/Tmpjd09UUXdOVGcxT1RBPQ%3D%3D/erreur.htm?cmd=\\_error\\_login-run&](http://www.paypal.com.fr.cgi.bin.webscr.cmd.flow.session.ycwnzphbcryskgq9zzetmxi8zg51bfu4zhgnlq2wntjgx8jlj7gdispatch.c70bbe4152786147242b0tb71efa252acz.waterdamagerestorationhouston.com/update/Tmpjd09UUXdOVGcxT1RBPQ%3D%3D/erreur.htm?cmd=_error_login-run&)

```
http://www.paypal.com.fr.cgi.bin.webscr.cmd.flow.session.yc  
[1, 0, 1, 0, 1, 1]
```

**Figure 5.2: URL Classifier Output**

For this URL, the classifier flagged the following conditions: Length of URL greater than 55, Presence of hexadecimal characters, Large number of period marks and Large number of sensitive words.

As the classifier caught more than 3 violations, this URL was classified as spam.

### **5.5 Results of all techniques on single dataset**

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

We applied CNN, Random Forest, Decision Tree, SVM, KNN, Naïve Bayes techniques on the IWSPA dataset and compared them on the basis of two metrics – accuracy and time taken. Here are the results obtained: -

Technique	Accuracy obtained (%)
CNN	98.13
Random Forest	91.20
SVM	89.92
KNN	87.88
Decision Tree	85.32
Naïve Bayes	24.46

**Table 5.2: Accuracy results on single dataset**

As we can see, on the basis of accuracy, CNN technique gives us the best accuracy, above 98%. This is followed by Random Forest at 91%, SVM at 89.9% and then KNN, Decision Tree and Naïve Bayes

Technique	Time Taken
CNN	173.1482
Random Forest	1.5892
SVM	12.5132
KNN	9.9718
Decision Tree	0.5245
Naïve Bayes	0.0787

**Table 5.3: Time taken results on single dataset**

In terms of how long each technique takes to run, CNN takes by far the most time, nearly 3 minutes. All the other techniques take between one-tenth of a second (Naïve Bayes) to 12 seconds (SVM)

## **5.6 Results of all techniques on mixed dataset**

We then applied the same techniques on a combination of IWSPA, TREC 2007 and Lingspam dataset to once again check the accuracies. Here are the results obtained: -

Technique	Accuracy obtained (%)
CNN	97.20
Random Forest	82.98
SVM	70.98
KNN	66.53
Decision Tree	79.40
Naïve Bayes	60.03

**Table 5.4: Accuracy results on complex dataset**

Here we see that most techniques experience a drastic decrease in their accuracy, with Random Forest falling by 8%, SVM by 19% and so on. The exception to this is CNN, which experiences only a slight decrease of less than 1%.

## **5.7 Result Analysis of CNN**

Recall is the ratio of correctly predicted True values to the total number of True values. Essentially, it tells us how often a True value is predicted as True. The result of recall varies between zero and one, with zero meaning there were no correctly predicted true values, and one meaning all the true values were correctly predicted.

Precision is the fraction of relevant instances among the retrieved instances.

F1 score is a metric that combines recall and precision by taking their harmonic mean. F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost.

A Confusion matrix is an  $N \times N$  matrix used for evaluating the performance of a classification model, where  $N$  is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

Metrics	Values
Accuracy	97.20
Precision	0.9835
Recall	0.9166
F1 Score	0.9489

**Table 5.5: Result Analysis of CNN**

<b>True Positive</b> <b>3763</b>	<b>False Positive</b> <b>23</b>
<b>False Negative</b> <b>125</b>	<b>True Negative</b> <b>1375</b>

**Figure 5.3: CNN Confusion Matrix**

Here we have the result analysis on the CNN model. We can see the values for accuracy, precision, recall and F1 score which are 97.20, 0.9835, 0.9166 and 0.9489 respectively. There is also a confusion matrix, which shows the number of true positive, true negative, false positive and false negative decisions obtained by the CNN model.

### **5.8 Comparison of Accuracies and Discussion**

The algorithms, were applied on 2 different set of datasets, the first dataset consisted of only data from a single dataset namely IWSPA with headers. The second dataset we used was a mixture of 4 different datasets namely IWSPA no header, IWSPA with header, LingSpam and TREC. Moreover, the dataset sizes for the first one was 5000 emails, and the size for the second was 85000.

As we increased the data complexity, the time to run the algorithms, increased drastically, and we also saw a change in the accuracies. For all the Machine Learning Algorithms we saw, that the accuracy has decreased drastically except for Naïve Bayes classifier. We can see, that for Naïve Bayes, the accuracy has increased from a mere 30 % to a 69%, one of the reasons that we understood for this to happen, is as the data size increases the complexity increases and hence the naïve bayes model is more accurately being able to find probabilistic values.

Now, another question that arises is why does accuracy decrease in with increase in dataset. In, theory, as the data increases the model, should start performing better and hence give more accuracy, but in our case, the complexity of data is increasing with increase in data, we are not increasing the data from the same source but rather, from different datasets, which increases the complexity, and hence we can say that the accuracy decreases because of this reason. Now, in case of CNN, we can see that even when the dataset changes, the accuracy does not change much.

Less change in accuracy, and higher accuracies compared to all other algorithms applied, is one of the primary reasons for us choosing CNN as our model for classification. Higher training time, can be considered as a problem, but in our case once the model is trained, it just needs to be saved and used directly for classification and hence, the training occurs once. Subsequent, training would take place, which would be done, when more data is accumulated.

## Chapter 6

### Conclusion and Future Scope

For this project, we started out by learning what is email fraud and conducted a literature survey of 31 research and review papers, which detailed various techniques to detect fraudulent emails. After completion of the literature survey, we designed a theoretical architecture and prepared UML diagrams for our model. We then began the implementation of our model which included data pre-processing, correlation, classification etc. on six datasets. After standardizing the format, we applied correlation techniques to the datasets to select the relevant features needed for our model. We implemented an URL classification technique that detects phishing URLs in the body of the email. For email classification, we implemented various machine learning and deep learning techniques and used the results obtained to decide upon the method we were going to implement in our model. We implemented Decision Tree, Random Forest, SVM, KNN, Naïve Bayes and CNN. After intensive result analysis by using different datasets which had different complexities, the results showed optimum accuracy in terms of complexity of dataset for CNN, while accuracy dropped significantly in terms of our classifier algorithms. Hence, we chose CNN as our email classifier. The model was then tested in real time by taking the user's email id and password as input. The deployment model used here is a website and classifies the emails and presents the results in a table. Our model provides an accuracy of 97.20% but the time taken to perform the task is around 75secs.

We have understood the current trends in spam detection, the basic principle of it and how it is implemented in a different way across the literature. For future work, we would further work upon our project, to not only detect a certain type of spam emails, but also being able to classify those emails into different categories, we would also work on reducing the overall current time of sixty to seventy-five seconds to completely the emails, by understanding how Word2Vec, can be implemented in a more robust way. Another, issue that we encountered during the project, is the lack of recent public email datasets. With more recent emails, the overall performance of the project could have been increased substantially, hence we intend on working and finding ways to get and create more public email datasets in this area. Finally, we would like to implement this project as a website plugin, where each time a user opens his/her browser, they can easily see, the classified unread emails.

## **Societal Benefits**

Email Fraud detector, as a project, has the sole purpose of educating the people and help them classify their emails safely, from the many malicious emails out there in the world. The society, benefits from higher security and ease of mind. The project, can be used by people of all age groups, the age group most affected by malicious emails are the elderly people, who unknowingly fall for this type of fraudulent emails and lose their life savings, bank accounts, and many times, all their personal information gets leaked, this has been and will always be a area of great research, as the spammers, keep on defying the spam detection algorithms out there, which become obsolete quite fast. Email Spam Detector, project, with its regular learning process, will be kept updated which, will benefit the society as a large.

## References

- [1] J. Tidy, "Google blocking 18m coronavirus scam emails every day," *BBC*, 2020.  
<https://www.bbc.com/news/technology-52319093> (accessed Jul. 12, 2020).
- [2] M. Keyworth and M. Wall, "The 'bogus boss' email scam costing firms millions," *BBC*, 2016.  
<https://www.bbc.com/news/business-35250678> (accessed Jul. 12, 2020).
- [3] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism," *IEEE Access*, vol. 7, pp. 56329–56340, 2019, doi: 10.1109/ACCESS.2019.2913705.
- [4] S. O. Olatunji, "Extreme Learning machines and Support Vector Machines models for email spam detection," *Can. Conf. Electr. Comput. Eng.*, pp. 1–6, 2017, doi: 10.1109/CCECE.2017.7946806.
- [5] C. Y. Tseng and M. S. Chen, "Incremental SVM model for spam detection on dynamic email social networks," *Proc. - 12th IEEE Int. Conf. Comput. Sci. Eng. CSE 2009*, vol. 4, pp. 128–135, 2009, doi: 10.1109/CSE.2009.260.
- [6] T. Vyas, P. Prajapati, and S. Gadhwai, "A survey and evaluation of supervised machine learning techniques for spam e-mail filtering," *Proc. 2015 IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2015*, 2015, doi: 10.1109/ICECCT.2015.7226077.
- [7] S. More and S. A. Kulkarni, "Data mining with machine learning applied for email deception," *2013 Int. Conf. Opt. Imaging Sens. Secur. ICOSS 2013*, pp. 1–4, 2013, doi: 10.1109/ICOISS.2013.6678403.
- [8] S. Nandhini and J. Marseline K.S., "Performance Evaluation of Machine Learning Algorithms for Email Spam Detection," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Feb. 2020, pp. 1–4, doi: 10.1109/ic-ETITE47903.2020.312.
- [9] P. Deshmukh, M. Shelar, and N. Kulkarni, "Detecting of targeted malicious email," *Proc. - 2014 IEEE Glob. Conf. Wirel. Comput. Networking, GCWCN 2014*, pp. 199–202, 2015, doi: 10.1109/GCWCN.2014.7030878.
- [10] Q. Li, M. Cheng, J. Wang, and B. Sun, "LSTM based Phishing Detection for Big Email Data," *IEEE Trans. Big Data*, vol. 7790, no. c, pp. 1–11, 2020, doi: 10.1109/TBDDATA.2020.2978915.
- [11] X. Li, D. Zhang, and B. Wu, "Detection method of phishing email based on persuasion principle," *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, no. Itnec, pp. 571–574, 2020, doi: 10.1109/ITNEC48623.2020.9084766.
- [12] A. Wijaya and A. Bisri, "Hybrid decision tree and logistic regression classifier for email spam detection," *Proc. 2016 8th Int. Conf. Inf. Technol. Electr. Eng. Empower. Technol. Better Futur. ICITEE 2016*, pp. 1–4, 2017, doi: 10.1109/ICITEED.2016.7863267.
- [13] K. Agarwal and T. Kumar, "Email Spam Detection Using Integrated Approach of Naïve Bayes and Particle Swarm Optimization," *2018 Second Int. Conf. Intell. Comput. Control Syst.*, no. Iccics, pp. 685–690, 2018.



- [14] W. Peng, L. Huang, J. Jia, and E. Ingram, "Enhancing the Naive Bayes Spam Filter Through Intelligent Text Modification Detection," *Proc. - 17th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. Trust. 2018*, pp. 849–854, 2018, doi: 10.1109/TrustCom/BigDataSE.2018.00122.
- [15] A. K. Singh, S. Bhushan, and S. Vij, "Filtering spam messages and mails using fuzzy C means algorithm," *Proc. - 2019 4th Int. Conf. Internet Things Smart Innov. Usages, IoT-SIU 2019*, pp. 1–5, 2019, doi: 10.1109/IoT-SIU.2019.8777483.
- [16] N. Moradpoor, W. Buchanan, and B. Clavie, "Employing Machine Learning Techniques for the Detection and Classification of Phishing Emails," *IEEE Comput. Conf.*, no. July, pp. 149–156, 2017.
- [17] P. K. Panigrahi, "A comparative study of supervised machine learning techniques for spam E-mail filtering," *Proc. - 4th Int. Conf. Comput. Intell. Commun. Networks, CICN 2012*, pp. 506–512, 2012, doi: 10.1109/CICN.2012.14.
- [18] S. Nizamani, N. Memon, M. Glasdam, and D. D. Nguyen, "Detection of fraudulent emails by employing advanced feature abundance," *Egypt. Informatics J.*, vol. 15, no. 3, pp. 169–174, 2014, doi: 10.1016/j.eij.2014.07.002.
- [19] N. A. Unnithan, N. B. Harikrishnan, R. Vinayakumar, K. P. Soman, and S. Sundarakrishna, "Detecting phishing E-mail using machine learning techniques CEN-SecureNLP," *CEUR Workshop Proc.*, vol. 2124, pp. 50–56, 2018.
- [20] R. Alotaibi, I. Al-Turaiki, and F. Alakeel, "Mitigating Email Phishing Attacks using Convolutional Neural Networks," *ICCAIS 2020 - 3rd Int. Conf. Comput. Appl. Inf. Secur.*, pp. 1–6, 2020, doi: 10.1109/ICCAIS48893.2020.9096821.
- [21] M. Habib, H. Faris, M. A. Hassonah, J. Alqatawna, A. F. Sheta, and A. M. Al-Zoubi, "Automatic Email Spam Detection using Genetic Programming with SMOTE," *ITT 2018 - Inf. Technol. Trends Emerg. Technol. Artif. Intell.*, no. Itt, pp. 185–190, 2019, doi: 10.1109/CTIT.2018.8649534.
- [22] G. Yu, W. Fan, W. Huang, and J. An, "An Explainable Method of Phishing Emails Generation and Its Application in Machine Learning," *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, no. Itnec, pp. 1279–1283, 2020, doi: 10.1109/ITNEC48623.2020.9085171.
- [23] R. Chen, C. Zhang, J. Guo, and X. Wang, "Application of Naive Bayesian Algorithms in E-mail Classification," in *2019 Chinese Automation Congress (CAC)*, Nov. 2019, pp. 3933–3938, doi: 10.1109/CAC48633.2019.8997251.
- [24] M. Prilepok and M. Kudelka, "Spam detection based on nearest community classifier," *Proc. - 2015 Int. Conf. Intell. Netw. Collab. Syst. IEEE INCoS 2015*, pp. 354–359, 2015, doi: 10.1109/INCoS.2015.75.
- [25] S. Bagui, D. Nandi, S. Bagui, and R. J. White, "Classifying phishing email using machine learning and deep learning," *2019 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2019*, no. M1, pp. 1–2, 2019, doi: 10.1109/CyberSecPODS.2019.8885143.

- [26] A. A. Alurkar *et al.*, “A proposed data science approach for email spam classification using machine learning techniques,” *Jt. 13th CTTE 10th C. Conf. Internet Things - Bus. Model. Users, Networks*, vol. 2018-Janua, pp. 1–5, 2017, doi: 10.1109/CTTE.2017.8260935.
- [27] W. Niu, X. Zhang, G. Yang, Z. Ma, and Z. Zhuo, “Phishing emails detection using CS-SVM,” *Proc. - 15th IEEE Int. Symp. Parallel Distrib. Process. with Appl. 16th IEEE Int. Conf. Ubiquitous Comput. Commun. ISPA/IUCC 2017*, pp. 1054–1059, 2018, doi: 10.1109/ISPA/IUCC.2017.00160.
- [28] A. Zaid, J. Alqatawna, and A. Huneiti, “A proposed model for malicious spam detection in email systems of educational institutes,” *Proc. - 2016 Cybersecurity Cyberforensics Conf. CCC 2016*, pp. 60–64, 2016, doi: 10.1109/CCC.2016.24.
- [29] S. Bin Abd Razak and A. F. Bin Mohamad, “Identification of spam email based on information from email header,” *Int. Conf. Intell. Syst. Des. Appl. ISDA*, pp. 347–353, 2014, doi: 10.1109/ISDA.2013.6920762.
- [30] F. Sanchez and Z. Duan, “A sender-centric approach to detecting phishing emails,” *Proc. 2012 ASE Int. Conf. Cyber Secur. CyberSecurity 2012*, no. SocialInformatics, pp. 32–39, 2012, doi: 10.1109/CyberSecurity.2012.11.
- [31] N. A. Azeez and A. Oluwatosin, “CyberProtector: Identifying Compromised URLs in Electronic Mails with Bayesian Classification,” *Proc. - 2016 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2016*, pp. 959–965, 2017, doi: 10.1109/CSCI.2016.0184.
- [32] J. Hajgude and L. Ragha, “Phish mail guard: Phishing mail detection technique by using textual and URL analysis,” *Proc. 2012 World Congr. Inf. Commun. Technol. WICT 2012*, pp. 297–302, 2012, doi: 10.1109/WICT.2012.6409092.
- [33] V. Vishagini and A. K. Rajan, “An Improved Spam Detection Method with Weighted Support Vector Machine,” *2018 Int. Conf. Data Sci. Eng. ICDSE 2018*, pp. 1–5, 2018, doi: 10.1109/ICDSE.2018.8527737.
- [34] N. Budanović, “What’s On the Other Side of Your Inbox – 20 SPAM Statistics for 2021,” *DATAPROT*, 2021. <https://dataprot.net/statistics/spam-statistics> (accessed Feb. 11, 2021).
- [35] A. Iyengar, G. Kalpana, S. Kalyankumar, and S. Gunanandhini, “Integrated SPAM detection for multilingual emails,” *2017 Int. Conf. Inf. Commun. Embed. Syst. ICICES 2017*, no. Icices, pp. 2–5, 2017, doi: 10.1109/ICICES.2017.8070784.

## **Publications**

A review article titled 'A Comprehensive Review of Fraudulent Email Detection Model' was accepted and successfully presented at the 7th International Conference on Mathematics and Computing (ICMC), 2021, held virtually from 2nd March to 5th March, 2021.

## **Acknowledgements**

We would like to thank our mentor Dr. Shubha Puthran for her continuous guidance, assistance and suggestions throughout the planning and development of this project. Furthermore, we would also like to express our gratitude towards Professor Prathamesh Churi, whose constructive suggestions helped us. Finally, we would also like to thank all the people who have directly or indirectly helped us in achieving the purpose of this project.