PRIVACY-PRESERVING FEDERATED LEARNING MODEL FOR EMAIL SPAM DETECTION

A PROJECT REPORT

Submitted by

VATSAL GUPTA [17104060]
ANANYA SHARMA [17104038]
ARJAV JAIN [17104070]

Under the guidance of

Mr. M. Gurve

(Assistant Professor, Department of Computer Science & IT)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Sector 62, Noida, Uttar Pradesh 201309

MAY 2020

ABSTRACT

Over the past few years, machine learning has revolutionized fields such as computer vision, natural language processing, speech recognition, and email spam filtering. Much of this success is based on collecting vast amounts of data, often in privacy-invasive ways. Federated Learning is a new subfield of machine learning that allows training models without collecting the data itself. Instead of sharing data, users collaboratively train a model by only sending weight updates to a server. In this project, we have applied this technique to a very relevant domain - Email Spam Filtering. Even though several different techniques have been developed to filter spam, due to the spammer's rapid adoption of new spam detection techniques, we are still overwhelmed with spam emails. To this end, in this project, we perform a comparative analysis of several existing machine learning algorithms in the domain of email spam filtering. Following this, we use the two best algorithms as the underlying architecture of our Federated Learning model.

PREFACE

This project aims to employ the federated learning technique to classify emails as spam or ham (not spam) while ensuring user privacy and data confidentiality. The model manifested in our project can help people recognize the nature of mails received without sharing their private information. Doing this project has helped us to enhance our knowledge of federated learning and its emerging applications. Furthermore, the application of several neural networks and other machine learning algorithms as the underlying architecture of our model has also enabled us to gain a better understanding of the machine learning domain. Since the model has been implemented mainly in python, this project has contributed to increased fluency in the python environment and its libraries.

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to our mentor, Mr. Mahendra Gurve, for his valuable guidance, consistent encouragement, timely help, and providing us with an excellent atmosphere for doing our project. During the entire duration of our dissertation work, despite his busy schedule, he has extended cheerful and cordial support to us for completing this project work. We would also like to convey our sincere regards to all other faculty members of the department of CSE & IT, JIIT, who have bestowed their great effort and guidance at appropriate times. Our thanks and appreciations also go to people who have willingly helped in developing this project.

Contents

ABSTRACT i

PREFACE ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

ABBREVIATIONS x

1 INTRODUCTION 1

1.1 Problem Statement 1

1.2 Research Motivation 1

1.3 Research objectives 3

2 LITERATURE SURVEY 4

2.1 Email Spam Filtering using Traditional Machine Learning Approaches . . . 4

2.2 Spam Filtering using Neural Networks 6

2.3 Federated Learning 7

3 DATASET DESCRIPTION 8

3.1 Summary 8

4 RESEARCH METHODOLOGY & DESIGN 10

4.1 Data Acquisition 10

4.2 Exploratory Data Analysis (EDA) 10

4.3 Data Pre-processing 13

4.4 Traditional Machine Learning Algorithms 15

4.4.1 Logistic Regression . . . 15
4.4.2 Naive Bayes . . . 16
4.4.3 Support Vector Machine Classifier . . . 16
4.4.4 Decision Tree Classifier . . . 17
4.4.5 Random Forest Classifier . . . 18
4.4.6 Comparison . . . 18
4.5 Neural Networks . . . 19
4.5.1 Simple RNN . . . 19
4.5.2 Long Short-Term Memory (LSTM) . . . 20
4.5.3 Gated Recurrent Unit (GRU) . . . 21
4.5.4 Comparison . . . 23
4.6 Federated Learning . . . 23
4.7 Homomorphic Encryption . . . 25

5 PROJECT SPECIFICATIONS 27
5.1 Hardware Specifications . . . 27
5.1.1 Recommended Configurations . . . 27
5.1.2 Minimum Configurations . . . 27
5.2 Software Specifications & Requirements . . . 27
5.2.1 Project Compatibility . . . 28
5.2.2 Python Packages . . . 28

6 EVALUATION METRICS USED 30
6.1 F1 Score . . . 30
6.1.1 Precision & Recall . . . 30
6.2 Loss & Accuracy . . . 32
6.3 Area Under the Curve (AUC) Score . . . 32

7 ANALYSIS AND RESULTS 33
7.1 Analysis of Neural Networks . . . 33
7.1.1 Simulation Settings . . . 33
7.1.2 Evaluation . . . 34
7.2 Analysis of Traditional Machine Learning Algorithms . . . 36

7.2.1 Evaluation of our Federated Learning Framework 37

8 CONCLUSION AND FUTURE DIRECTION **39**

8.1 Conclusion 39

8.2 Future Direction 39

List of Tables

7.1 Simulation Settings 33
7.2 Comparison of Neural Networks 34
7.3 Logistic Regression with and without Federated Learning (FL) 38

List of Figures

3.1 **CSV Dataset Snapshot** 9

4.1 **Number of Spam/Ham Emails in Training Set** 11
4.2 **Number of Spam/Ham Emails in Test Set** 11
4.3 **Top 20 Most Frequent Words in Spam Emails** 12
4.4 **Top 20 Most Frequent Words in Ham Emails** 12
4.5 **Distribution of Length of Emails** 13
4.6 **Research Framework of our Project** 14
4.7 **Logistic Regression Classifier** 15
4.8 **Support Vector Machine Email Spam Classifier** 17
4.9 **Email Spam Classification using Decision Tree Classifier** 17
4.10 **A Random Forest Classifier** 18
4.11 **Simple RNN** 19
4.12 **LSTM** 21
4.13 **Sigmoid Function** 22
4.14 **GRU** 22
4.15 **Our Federated Learning Framework** 23
4.16 **Federated Learning - Steps Involved** 24
4.17 **Homomorphic Encryption** 25

6.1 **Precision and Recall** 31
6.2 **AUC** 32

7.1 **Training Accuracy Comparison of Different Neural Networks on our Dataset** 34
7.2 **Validation Accuracy Comparison of Different Neural Networks on our Dataset** 35
7.3 **Loss Comparison of Different Neural Networks on our Dataset** 35
7.4 **Performance of Traditional Machine Learning Algorithms on Enron Email Dataset** 36

7.5 **Model Comparison Based on Area Under the Curve** 36
7.6 **Evaluationg of GRU-enabled Federated Learning Model** 37

ABBREVIATIONS

HDD	Hard Disk Drive
SSD	Solid State Drive
EDA	Exploratory Data Analysis
AI	Artificial Intelligence
AUC	Area Under the Curve
NB	Naive Bayes
SVM	Support Vector Machine
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
RNN	Recurrent Neural Network

Chapter 1

INTRODUCTION

1.1 Problem Statement

In the world of technological advances and unprecedented digitalization, data confidentiality and user privacy have been of primary concern. The Artificial Intelligence (AI) market is dominated by tech giants such as Google, Amazon, and Microsoft, offering cloud-based AI solutions and APIs. In traditional AI methods, sensitive user data are sent to the servers where models are trained. This compromises the confidentiality and integrity of users' data. In many cases, this data is exploited by organizations for their own benefits. Towards this end, there is a need for a privacy-preserving technique in several existing domains.

One such domain is email spam filtering. Currently, existing email spam detection models rely on cloud servers to run the appropriate machine learning algorithms. Although such methods have proved to be highly accurate in terms of classifying emails as spam or not, they have one notable shortcoming. These models are privy to all conversations taking place between people in every corner of the world. A lot of the times, these conversations are of private nature. A blatant breach of such a conversation can have a potentially devastating impact on the individuals that are part of the dialogue.

1.2 Research Motivation

Many problems in computer science are tremendously challenging to solve by handcrafting algorithms. Spam detection systems are a prominent example of this. Machine learning provides an alternative approach to solving such problems. It is often referred to as the field of learning from example data. By collecting data and then applying statistical techniques, patterns can be found in an automated way. In the example of email spam detection, one would collect a large number of emails and then apply machine learning algorithms to find patterns in the data

that help with interpreting the email content. In the past few years, this idea has been successfully applied to many different areas. Like spam filtering models, most speech recognition and recommender systems are also based on machine learning nowadays. The areas of computer vision and natural language processing also increasingly rely on data-driven approaches. For example, most state-of-the-art solutions in object detection and machine translation are based on learning from data. Many learning algorithms that are now hugely successful have been known for many years. For instance, the backpropagation algorithm, which most of the deep learning is based on, was described as early as in 1970.

To explain why these ideas are only now being used successfully, three reasons are generally given. First, there have been some fundamental improvements to the algorithms that had a significant effect. To give just a single example, Adam has dramatically reduced the amount of tuning that is required to make gradient descent work well. A second reason for the recent success has been the increase in computational resources that are available. Processing power has repeatedly doubled over the years, and specialized hardware for linear algebra and machine learning has been released. However, the third reason, and often considered the most central one, is that more data to train on is available. Having more example data means that the algorithms get more information to decide what patterns are truly essential. Consequently, it is less likely that example data points are just memorized or that random patterns are detected as a useful signal.

The fact that having a lot of data is extremely important in building good machine learning models has been widely described and discussed. The predominant way of using machine learning nowadays involves collecting all this data in a data center. The model is then trained on powerful servers. However, this data collection process is often privacy-invasive [1]. Many users do not want to share private data with companies, making it difficult to use machine learning. Even when privacy is not a concern, having to collect data can be infeasible. For example, self-driving cars generate too much data to be able to send all of it to a server. Federated Learning is an alternative approach to machine learning, where data is not collected [2]. In a nutshell, the parts of the algorithms that touch the data are moved to the users' computers. Users collaboratively help to train a model by using their locally available data to compute model improvements. Instead of sharing their data, users then send only these abstract improvements back to the server. This approach is much more privacy-friendly and flexible. Applications on

mobile phones provide examples where this is especially evident. Users generate vast amounts of data through interaction with the device. This data is often deeply private and should not be entirely shared with a server. Federated Learning still allows training a common model using all this data, without necessarily sacrificing computational power or missing out on smarter algorithms [3]. In the future, Federated Learning approaches can even lead to better models than conventional techniques since more data is available.

1.3 Research objectives

The primary objective of our work is to analyze the performance of existing machine learning algorithms in the domain of email spam filtering. We aim to evaluate several neural network techniques like simple Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) along with traditional machine learning algorithms such as Logistic Regression, Decision Tree Classifier, Naive Bayes (NB), Support Vector Machine (SVM) classifier, and Random Forest Classifier. Among these, we choose the algorithms that yield the best result in email spam detection and then apply them in a federated environment. The main objectives of our project work are summarised as follows:

1. Applying several machine learning algorithms for the classification of emails as ham or spam.
2. Providing a comparative analysis of the performance of these approaches.
3. Applying two such approaches (with the best results) in a federated environment.
4. Performing homomorphic encryption in the federated environment to ensure an even higher level of data security.
5. Comparing the performance of machine learning algorithms when used with and without Federated Learning.

Chapter 2

LITERATURE SURVEY

With the advancing technology and digitalization, today's world works in the digital era. While digitalization is creating new opportunities for us, it is also creating a greater risk of data vul-nerability and privacy invasion. The digital world is vulnerable to hackers and spammers. To avoid such vulnerabilities much research is being done in both industry and academia to come up with new privacy-preserving methods so that people can advance technology while keeping users' privacy intact. In this section, we discuss the existing literature in the direction of spam filtering and federated learning. For better readability, we have divided this section into three subsections, namely:

- Email Spam Filtering using Traditional Machine Learning Approaches
- Spam Filtering Using Neural Networks
- Federated Learning Applications

2.1 Email Spam Filtering using Traditional Machine Learning Approaches

Many researchers and academicians have proposed different email spam classification techniques that have been successfully used to classify data. These methods include decision tree, random forests, SVM), Naive Bayes, and logistic regression [4]. It is clear from the literature that it is possible to use these classification methods for spam mail filtering by using a content-based filtering technique to identify certain features (usually keywords frequently utilized 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 [5]. SVM algorithms are very potent for pattern recognition and data classification [6]. According to several researchers, SVM algorithms can be easily trained and can outperform many of the other popular email

spam classification methods [7]. However, for high dimension data, the strength and efficacy of SVM diminish over time due to the computational complexities of the processed data [[8, 9].

Logistic regression is another statistical method that can be utilized for spam filtering. Spam emails typically share a specific type of characteristics. The logistic regression classifier takes advantage of this to classify emails as spam or not. Words that recurrently show up in spam emails can be used as predictor variables in the logistic regression model [10].

Another machine learning algorithm that has been successfully applied to email spam filtering is the decision tree classifier. Compared to other approaches, decision tree classifiers need little effort from users during the training of datasets. Decision tree classifiers thoroughly perform variable analysis or feature selection of the email corpus during data training. The performance of a tree does not depend on the relationships among parameters. A significant advantage of a decision tree classifier 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 classifiers, it still has some drawbacks, for example, unless there is appropriate pruning, it can be challenging to control tree growth. The decision tree classifier is a nonparametric machine learning algorithm that is very vulnerable to overfitting of training data. This renders them inaccurate to a large extent.

To overcome the limitations faced by decision trees, many researchers have put forth the use of random forests, which take a large number of relatively uncorrelated trees operating as a committee for classification of an example. Random forest outperforms any of the individual constituent models.

Naive Bayes is another excellent machine learning algorithm that has been applied in email spam filtering. A Naive Bayes (NB) classifier applies 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. Some of its advantages include, 1) Need for lesser training data, 2) Scalability, 3) No bottleneck creation by an increase in the number of predictors and discrete units of information [11].

2.2 Spam Filtering using Neural Networks

In recent times, many researchers have shifted their focus from traditional machine learning algorithms to artificial neural networks for spam filtering. In [12], the authors have proposed a novel SMS spam detection based on the case study of the SMS spams in the English language using Natural Language Process and Deep Learning techniques. Following data preprocessing, the authors have used the data to develop a spam filtering model based on Long Short-Term Memory and Gated Recurrent Unit algorithms. The performance of the proposed models is compared to the models based on machine learning algorithms like Support Vector Machine and Naive Bayes. Their experimental results demonstrated that the model built from the Long Short-Term Memory technique provides the best overall accuracy as high as 98.18%. On accurately screening spam messages, this model shows the ability that it can detect spam messages with the 90.96% accuracy rate, while the error percentage that it misclassifies a non-spam message as a spam message is only 0.74%.

In [13], the authors have proposed a novel deep learning architecture based on Convolutional Neural Network (CNN)and Long Short-Term Neural Network (LSTM) for spam detection. The authors of this paper also proposed to use WordNet and ConceptNet to improve the coverage in Embedding space and provide better initialization for further deep learning architecture. Experimental results showed that the proposed approach outperformed all other approaches on the two datasets that were used i.e., SMS Dataset and Twitter Dataset.

In [14], the authors have empirically explored a gated recurrent neural network model for deceptive opinion spam detection. Their experimental results showed that the neural model with attention mechanism outperformed the model without attention mechanism. For cross-domain experiments, the results showed that the neural model gave a higher performance than the discrete model. This showed that the neural model has a stronger generalization ability compared with the discrete model. Further experiments showed that the accuracies could be improved by integrating discrete and neural features.

2.3 Federated Learning

Although Federated Learning is a relatively new domain, much work is being done in academia to adopt its use in several fields [15, 16, 17, 18]. The authors of [19] have trained a recurrent neural network language model using a federated learning framework for the purpose of next-word prediction in a virtual keyboard for smartphones. Their work compares the server-based training using stochastic gradient descent with training on client devices using the Federated Averaging algorithm. Their work demonstrates the feasibility and benefit of training language models on client devices without exporting sensitive user data to servers. In [20], the authors have proposed a novel joint transmit power and resource allocation approach for enabling ultra-reliable low-latency communication (URLLC) in vehicular networks that leverages principles from Federated Learning.

Despite the growing research in the domain of Federated Learning, to the best of our knowledge, no existing literature has applied Federated Learning in the domain of spam filtering.

Chapter 3

DATASET DESCRIPTION

The dataset used for this work has been retrieved from a website owned and operated by the Carnegie Mellon University [21]. This dataset was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). It contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages. This data was originally made public and posted to the web by the Federal Energy Regulatory Commission during its investigation.

Since this dataset has a large number of emails (as .txt files) classified as ham and spam, it is the most widely used dataset by academicians for use in the domain of email spam filtering. Owing to computational constraints, we haven't used the entirety of data present in the dataset (approx. 1.7 GB worth of emails). We have used 600 ham emails and 250 spam emails as part of the code. Although the same subset of the dataset has been used in all the codes (multiple machine learning models have been used), since different students have written the codes, there is a difference in how the dataset has been formatted (i.e., as csv or txt dataset). A snapshot of the csv dataset has been provided in figure 3.1. The dataset is straightforward to understand. It has only two attributes:

1. **Label:** The classification label of the email. Possible Values - 1) Ham (Not Spam) and 2) Spam.
2. **Email:** The original email.

3.1 Summary

1. **Dataset Name:** Enron Email Dataset
2. **Original Format:** .txt email files
3. **Number of Rows in Original Dataset (when converted to .csv):** Approximately 500,000
4. **Number of Rows (used in our project):** 850 (600 Ham, 250 Spam)
5. **Number of Columns (when converted to .csv):** 2

6. **Dataset Source:** Machine Learning Department, Carnegie Mellon University - `https://www.cs.cmu.edu/~enron/`
7. **Dataset Avalaibility** - Publicly available for free on multiple websites

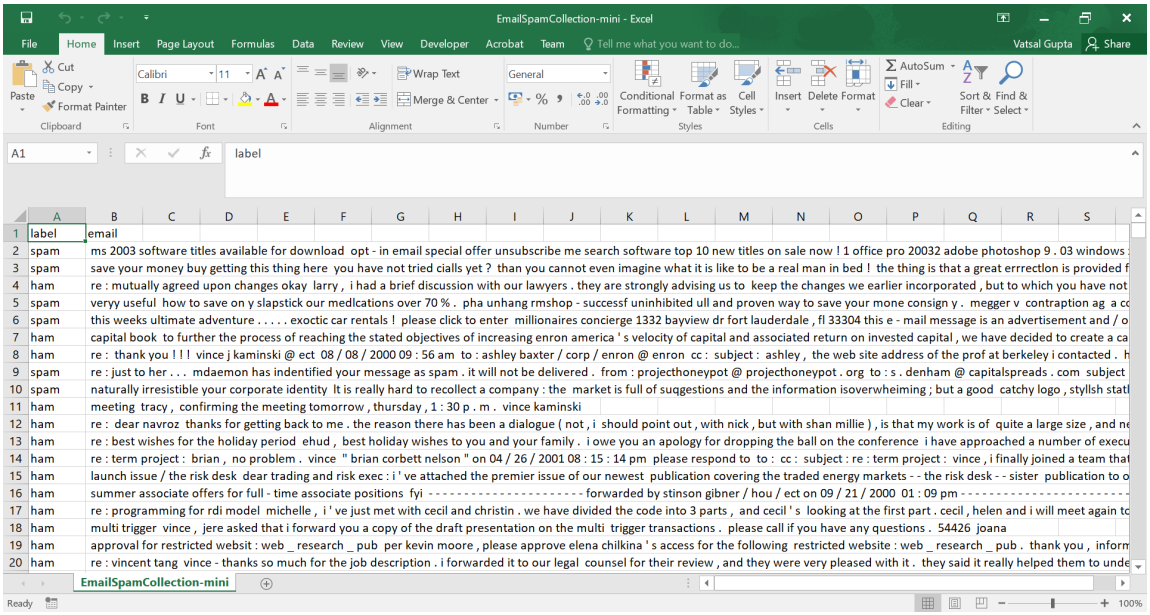

Figure 3.1: CSV Dataset Snapshot

Chapter 4

RESEARCH METHODOLOGY & DESIGN

4.1 Data Acquisition

The dataset under scrutiny has been acquired from a website operated by the Machine Learning Department (MLD) of the Carnegie Mellon University. The dataset contains around 500,000 spam and ham emails. A detailed description of the dataset has already been presented in Chapter 3.

4.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a systematic way of visualisation and transformation to explore and summarise the main characteristics of the available data. The main objectives of an EDA include:

- Generating questions about the available data.
- Searching for answers by vizualising, transforming, and modeling the data.
- Using the findings to generate new questions.

In our project, the EDA was carried out a subset of the enron email dataset consisting of 850 emails. The following criteria were considered while performing the EDA:

- Number of spam/ham emails in the training set. (Figure 4.1)
- Number of spam/ham emails in the test set. (Figure 4.2)
- Top 20 most frequent words in spam emails. (Figure 4.3)
- Top 20 most frequent words in ham emails. (Figure 4.4)
- Distribution of length of spam/ham emails. (Figure 4.5)

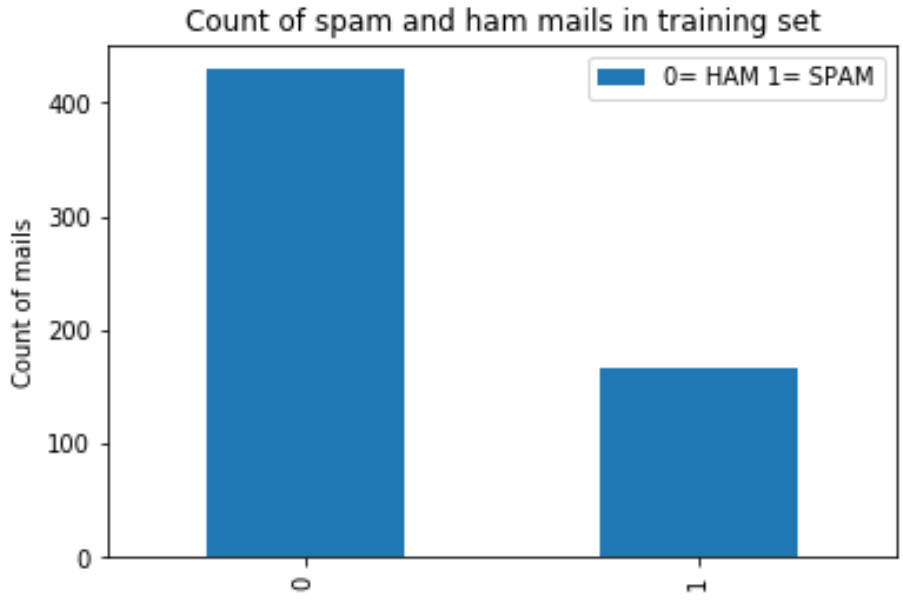

Figure 4.1: Number of Spam/Ham Emails in Training Set

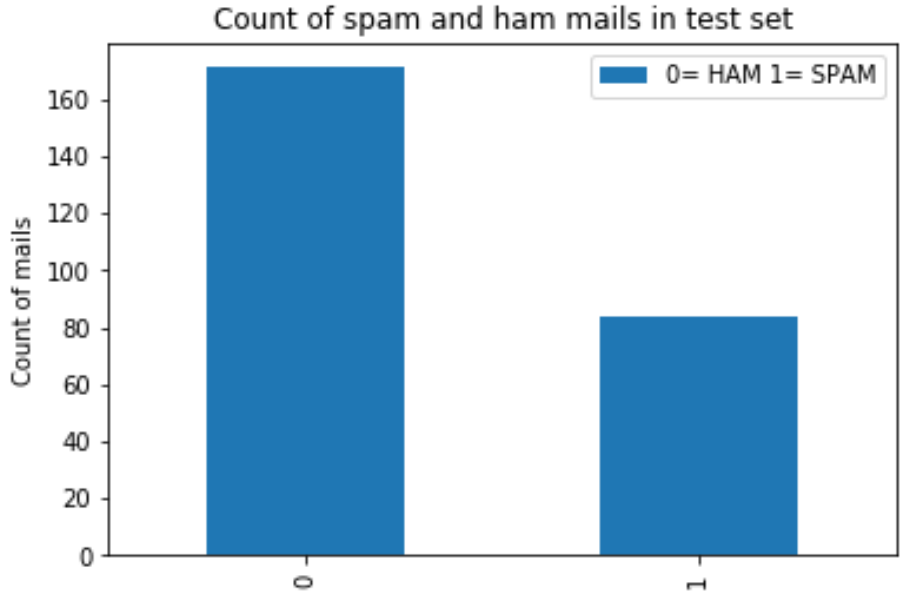

Figure 4.2: Number of Spam/Ham Emails in Test Set

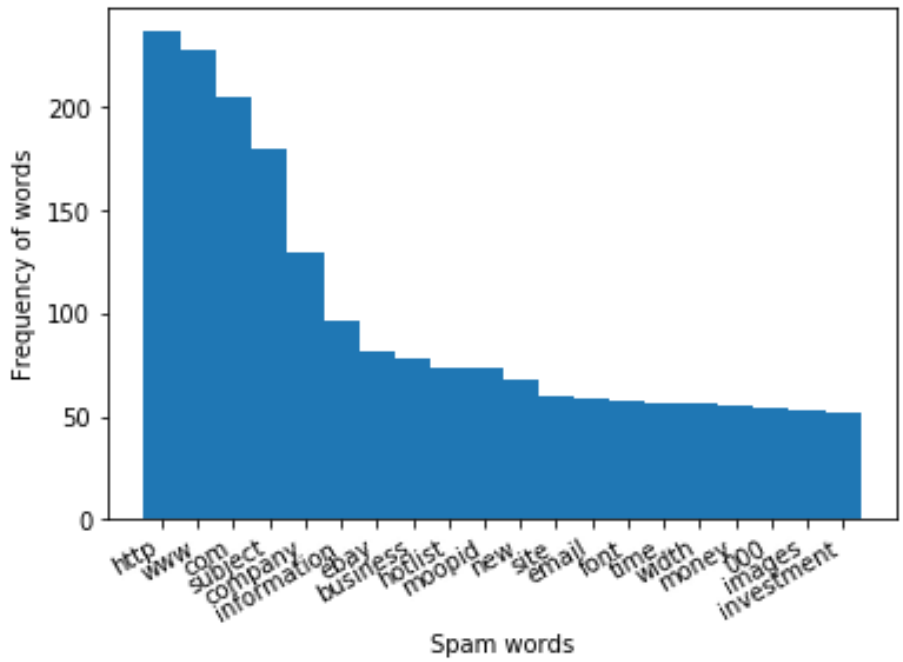

Figure 4.3: Top 20 Most Frequent Words in Spam Emails

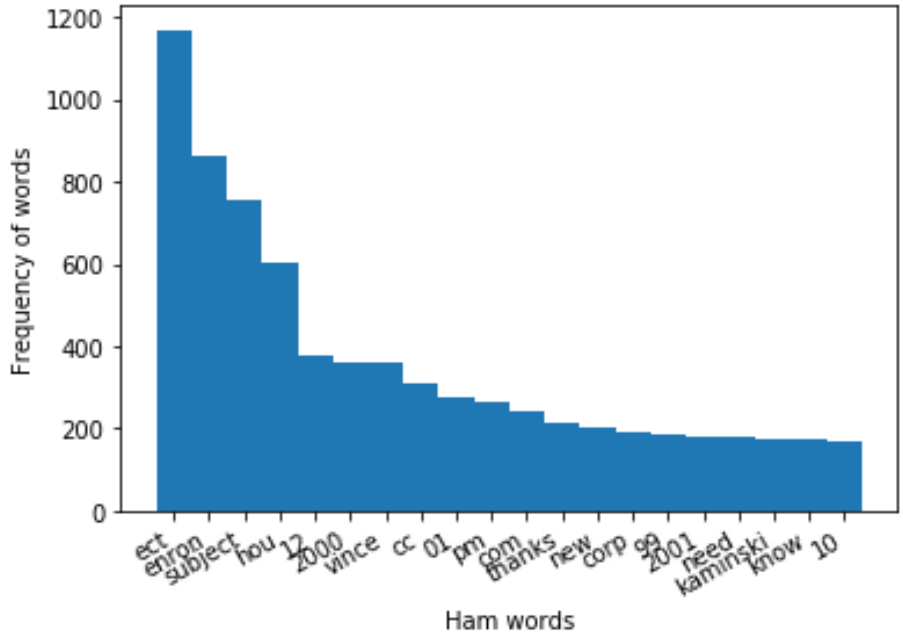

Figure 4.4: Top 20 Most Frequent Words in Ham Emails

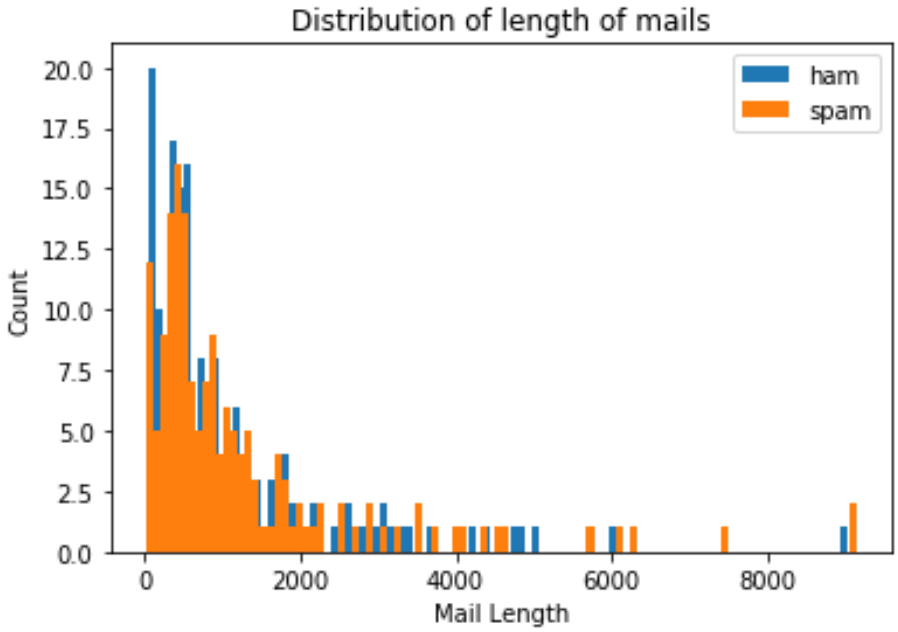

Figure 4.5: Distribution of Length of Emails

4.3 Data Pre-processing

Data pre-processing is a data mining procedure that involves converting raw data into an understandable format. Real-world data is often incomplete, inconsistent, and lacking in certain behaviors or trends and is likely to contain several errors. Data pre-processing is a proven method of addressing these issues. Data Pre-processing steps involved in our project:

1. **Padding and Truncating:** If the specified row length is longer than the input row length, the rows are padded. If the specified row length is shorter than the input row length, the rows are truncated.
2. **Word Tokenization:** Word tokenization is the process of splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis like classifying and counting them for a particular sentiment etc.
3. **Stop Word Removal:** Words such as articles and some verbs are usually considered stop words because they don't help us to find the context or the true meaning of a sentence. These are words that can be removed without any negative consequences to the final model to be trained.

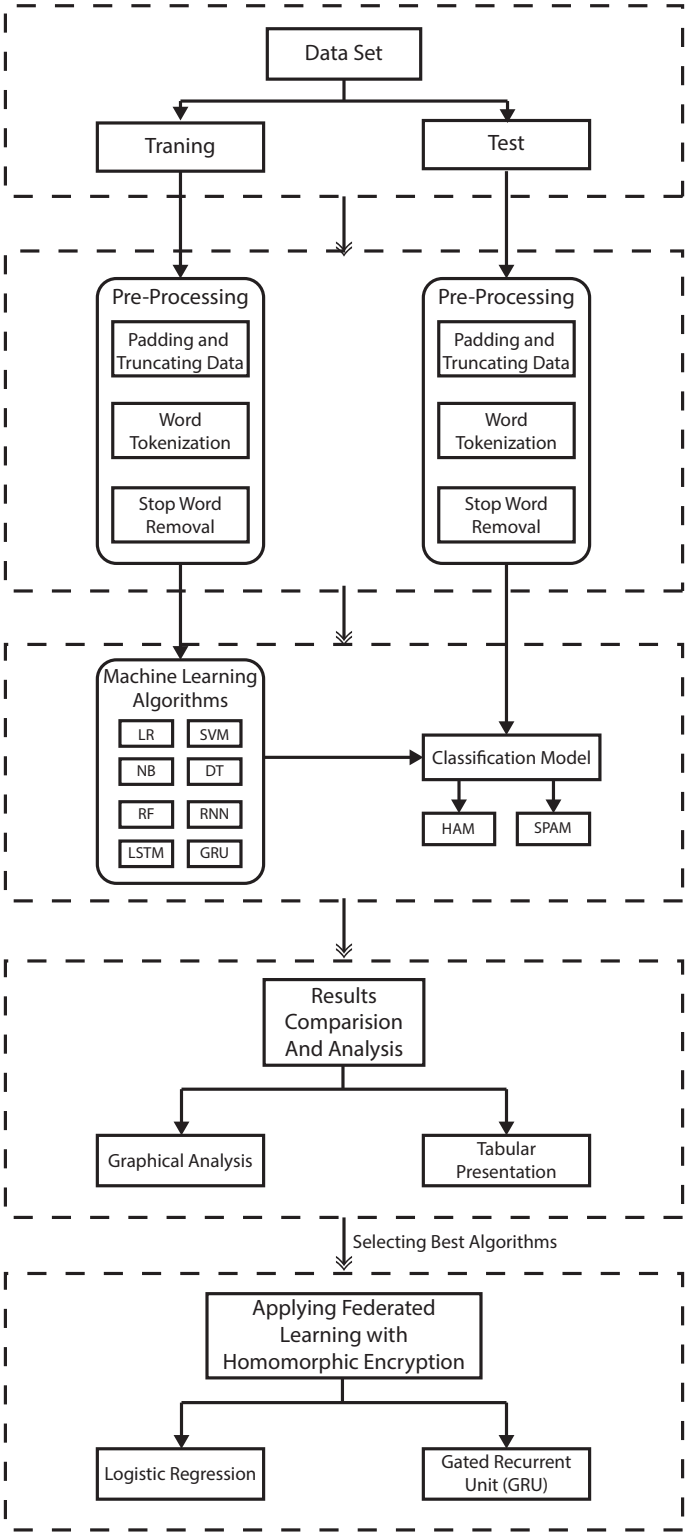

Figure 4.6: Research Framework of our Project

4.4 Traditional Machine Learning Algorithms

4.4.1 Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for various classification problems. It is a predictive analysis algorithm and based on the concept of probability. Logistic Regression uses a complex cost function, known as the 'Sigmoid function' or the 'logistic function.' The sigmoid function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

How does classification take place?

We expect our logistic regression classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1. For example, we have two classes - spam and ham(Class 1 — spam, Class 2 — ham). We set a threshold value above which we classify values into Class 1, and of the value goes below the threshold, then we classify it in Class 2.

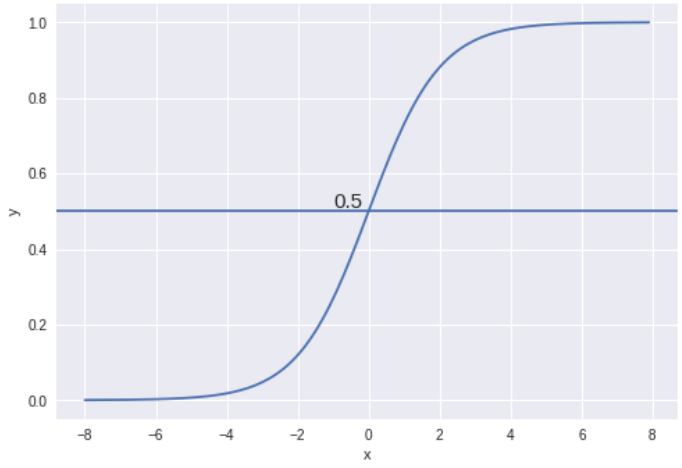

Figure 4.7: Logistic Regression Classifier

4.4.2 Naive Bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x), and P(x|c). Look at the equation below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{4.2}$$

where,

- P(c|x) is the posterior probability of class c (target) given predictor x (attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They are fast and easy to implement but their biggest disadvantage is that they require the predictors to be independent. In most of the real-life cases, the predictors are dependent, which hinders the performance of the Naive Bayes classifier.

4.4.3 Support Vector Machine Classifier

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

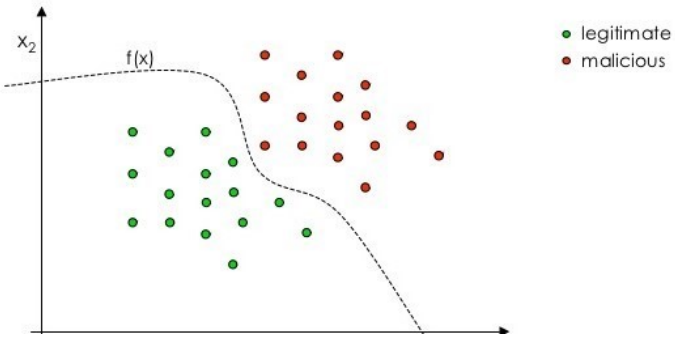

Figure 4.8: Support Vector Machine Email Spam Classifier

4.4.4 Decision Tree Classifier

The general motive of using Decision Tree is to create a training model that can use to predict the class or value of target variables by learning decision rules inferred from prior data (training data). In decision tree classifiers, we start from the root of the tree for predicting a class label for a record. We compare the values of the root attribute with the record's attribute. Based on the comparison, we follow the branch corresponding to that value and jump to the next node. We continue comparing our record's attribute values with other internal nodes of the tree until we reach a leaf node with predicted class value. One drawback of decision tree classifiers is that they are highly prone to overfitting. More often than not, this results in the classifier's low prediction accuracy as compared to other machine learning algorithms.

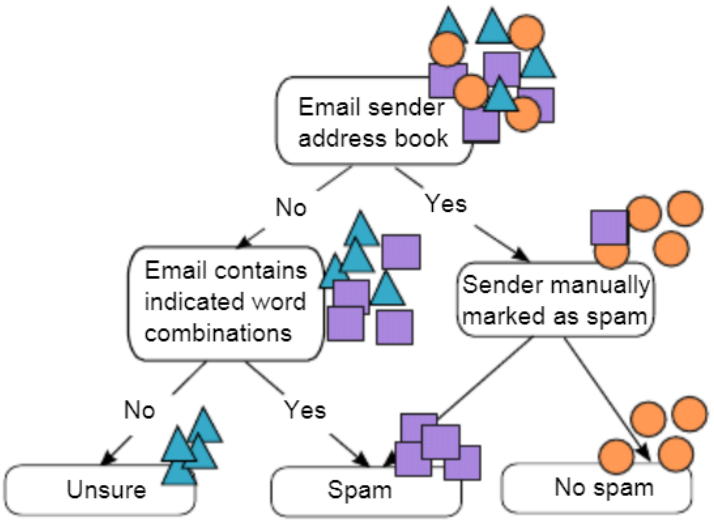

Figure 4.9: Email Spam Classification using Decision Tree Classifier

4.4.5 Random Forest Classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each tree in the random forest spits out a class prediction, and the class with the most votes becomes our model's prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. The low correlation between models is the key. The reason for their incredible results is that the trees protect each other from their individual errors (as long as they do not always all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group, the trees can move in the correct direction. However, random forest classifiers often suffer from the same problem as that of decision tree classifiers – overfitting.

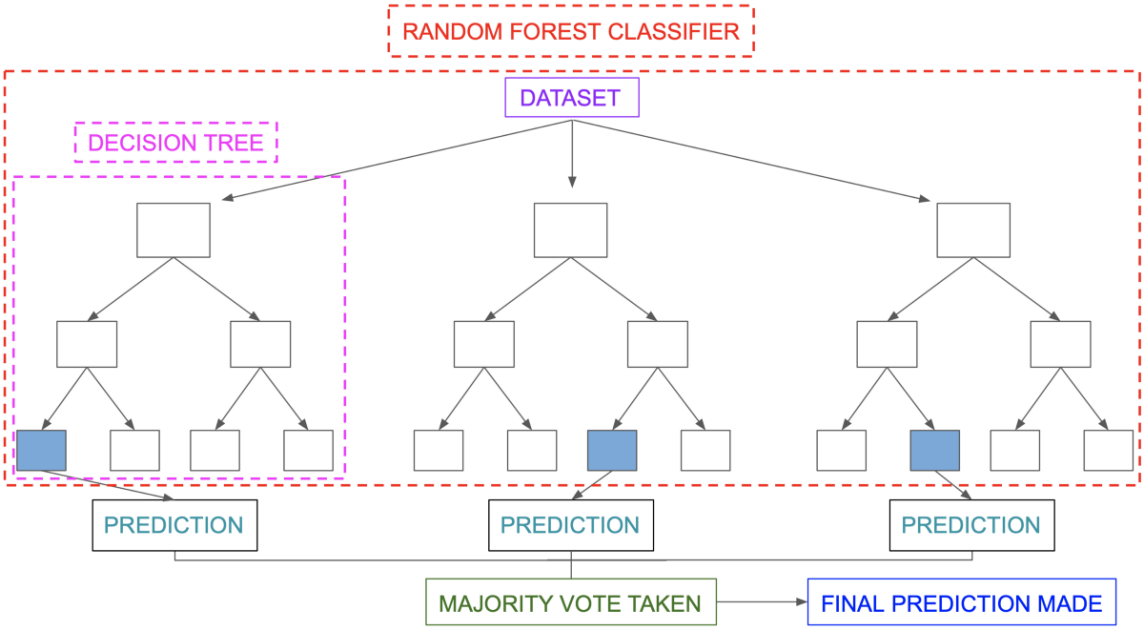

Figure 4.10: A Random Forest Classifier

4.4.6 Comparison

In the comparative analysis (given in section 7.2) of the aforementioned classification algorithms, it was found that Logistic Regression and SVM substantially outperformed other classification algorithms. Their Area Under the Curve (AUC) and F1 score were pretty close; however, the Logistic Regression classifier had slightly better metrics compared to the SVM

classifier.

4.5 Neural Networks

4.5.1 Simple RNN

Recurrent Neural Networks (RNNs) are a popular kind of neural networks used extensively in use cases that deal with sequential or contextual data. RNNs can deal with sequences of variable length (unlike feedforward networks) by defining a recurrence relation over multiple timesteps.

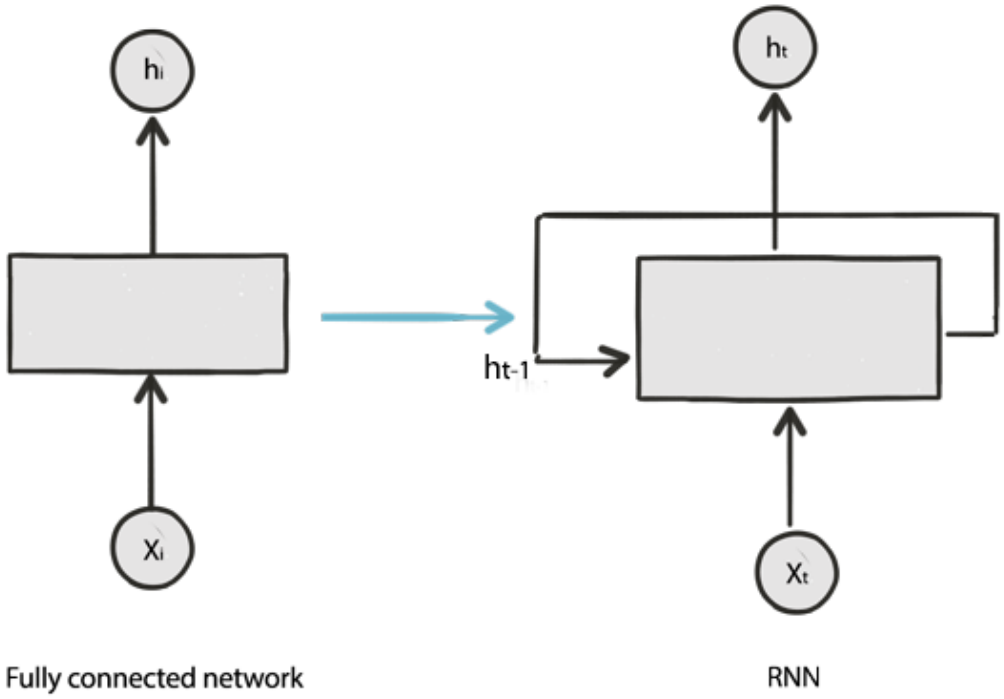

Figure 4.11: Simple RNN

The RNN can be viewed as a state model with a feedback loop. The state evolves due to the recurrence relation, and the feedback is fed back into the state with a delay of one timestep. This delayed feedback loop gives the model memory because it can remember information between timesteps in the states. The final output of the network Y_k at a certain

timestep k is typically computed using the current input X_k and one or more previous states $\{S_{k-i}, S_{k-i+1}, ..., S_{k-1}\}$.

There are some problems with the simple implementation of RNNs. They learn through backpropagation over time. This could lead to vanishing gradient or exploding gradient problems if we ask them to learn from long term dependencies. To overcome these problems, we use LSTM (long short-term memory), a special kind of recurrent network, and GRU (Gated Recurrent Unit), a slightly modified version of an LSTM.

4.5.2 Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Like simple RNNs, an LSTM also has feedback connections. It can process not only single data points (such as images) but also entire sequences of data (such as speech or video). An LSTM unit mainly consists of a cell state (current information flow of the unit) and three Gates - forget gate, input gate, and output gate. Some variations of the LSTM unit do not have one or more of these gates or maybe have other gates. For example, Gated Recurrent Units (GRUs) do not have an output gate.

The cell state is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell, and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The equations for the gates in LSTM are:

$$\begin{aligned} i_t &= \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right) \\ f_t &= \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right) \\ o_t &= \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right) \end{aligned} \tag{4.3}$$

where,

- $i_t \rightarrow$ represents input gate.
- $f_t \rightarrow$ represents forget gate.
- $o_t \rightarrow$ represents output gate.

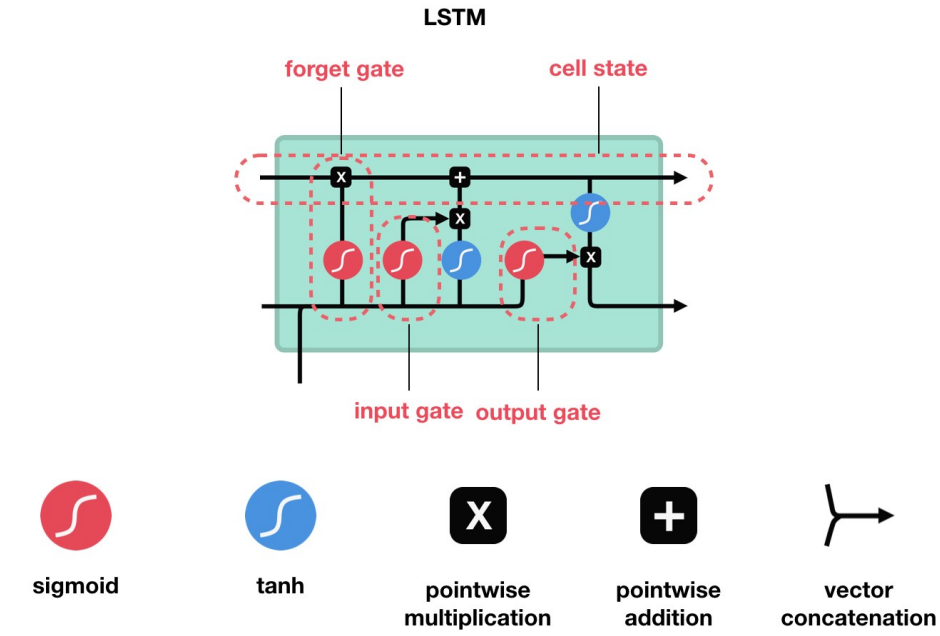

Figure 4.12: LSTM

- $\sigma \rightarrow$ represents the sigmoid function.
- $w_x \rightarrow$ weight for the respective gate(x) neurons.
- $b_x \rightarrow$ biases for the respective gates(x).
- $h_{t-1} \rightarrow$ output of the previous LSTM block (at timestamps t-1).
- $x_t \rightarrow$ input at current timestamp.

Gates in LSTM use the sigmoid activation functions, which output a value between 0 or 1. A sigmoid activation function is used since we want a gate to give only positive values and should be able to give us a clear-cut answer whether we need to keep a particular feature, or we need to discard that feature.

- "0" means the gates are blocking everything.
- "1" means gates are allowing everything to pass through it.

4.5.3 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a popular variant of the LSTM network. The main difference between LSTM and GRU is that a GRU only has only two gates (update and reset gate). GRU

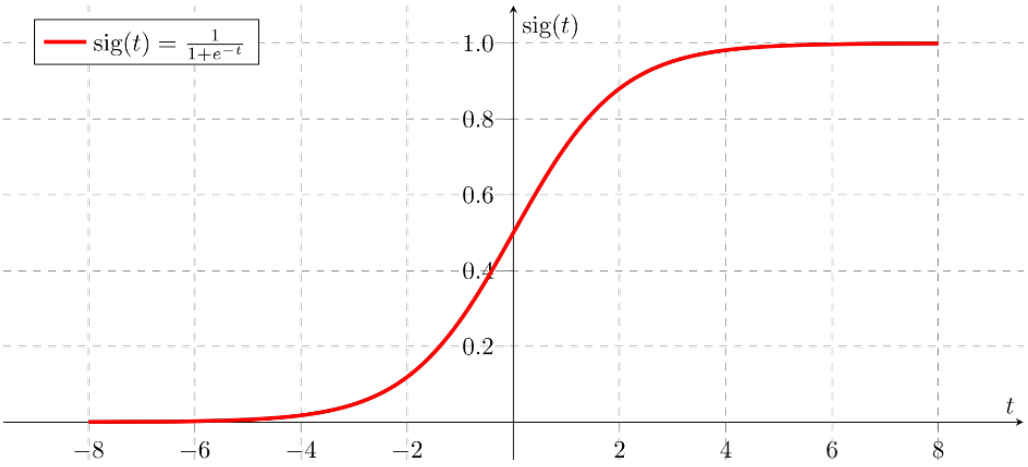

Figure 4.13: Sigmoid Function

has a reduced number of gates; hence it has a reduced number of tensor operations compared to an LSTM. This makes GRUs computationally cheaper and faster than LSTMs. GRUs and LSTMs both overcome the vanishing gradient problem, and are, therefore, better alternatives to simple RNNs.

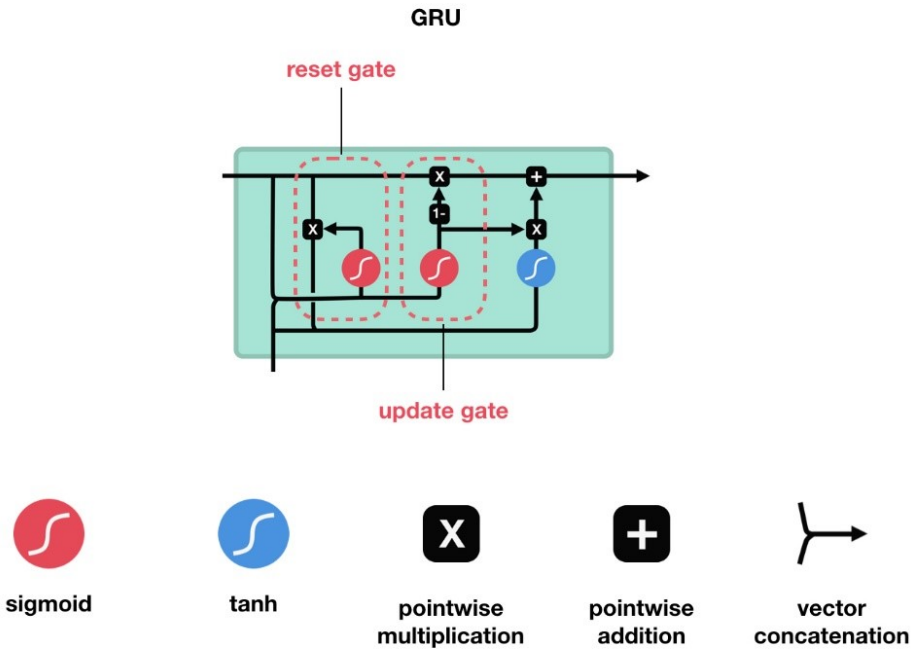

Figure 4.14: GRU

4.5.4 Comparison

The detailed comparison between the three neural networks has been provided in section 7.1. It was found that GRU and LSTM outperform simple RNN. Furthermore, GRU yielded slightly better results in comparison to LSTM.

4.6 Federated Learning

Federated Learning (FL) is a distributed machine learning approach that enables training on a large corpus of decentralized data residing on devices like mobile phones. FL is one instance of the more general approach of "bringing the code to the data, instead of the data to the code" and addresses the fundamental problems of privacy, ownership, and locality of data.

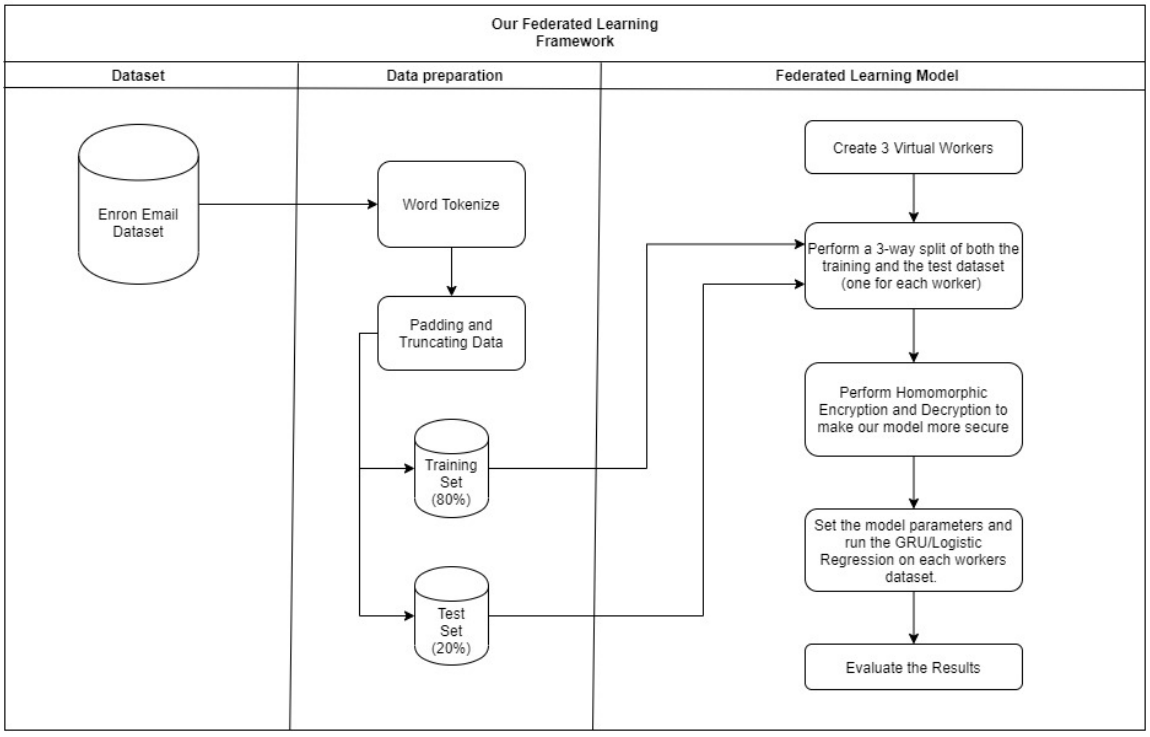

Figure 4.15: Our Federated Learning Framework

Federated learning algorithms may use a central server that orchestrates the different steps of the algorithm and acts as a reference clock, or they may be peer-to-peer, where no such central server exists.

To ensure reliable task performance of a final, central machine learning model, federated

learning relies on an iterative process broken up into an atomic set of client-server interactions known as a federated learning round. Each round of this process consists in transmitting the current global model state to participating nodes, training local models on these local nodes to produce a set of potential model updates at each node, and then aggregating and processing these local updates into a single global update and applying it to the global model. In the methodology below, we use a central server for this aggregation, while local nodes perform local training depending on the central server's orders. Initially, a statistical model is chosen to be trained on local nodes and initialized. Nodes are activated and wait for the central server to give calculation tasks. For multiple iterations of federated learning rounds, the following steps are performed:

1. Selection

A fraction of local nodes is selected to start training on local data. They all acquire the same current statistical model from the central server. Other nodes wait for the next federated round.

2. Configuration

- The central server orders selected nodes to undergo training of the model on their local data in a pre-specified fashion (e.g., for some batch updates of gradient descent).

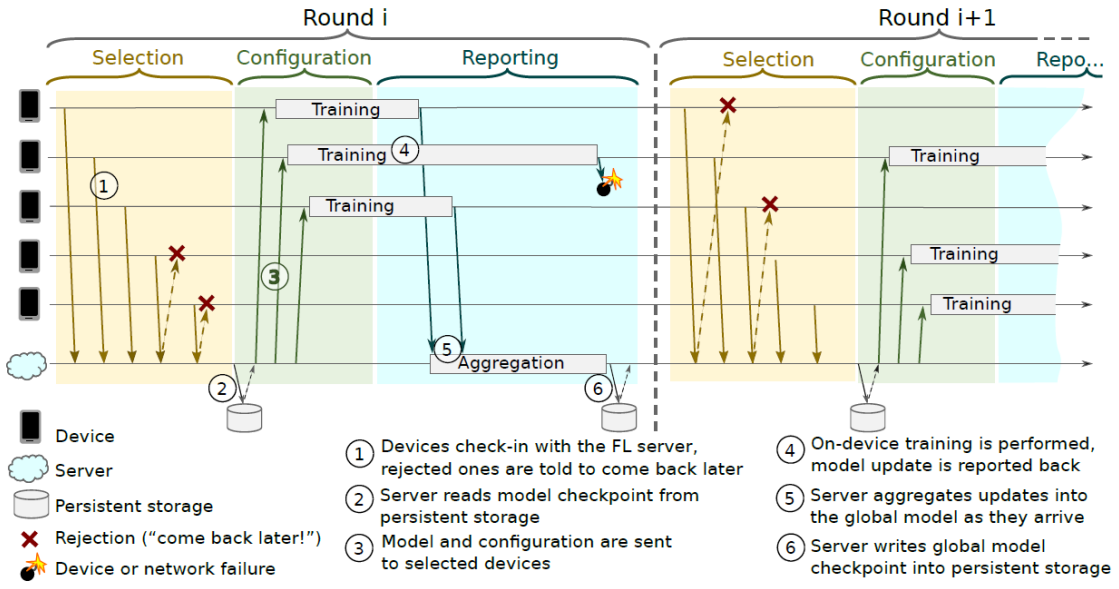

Figure 4.16: Federated Learning - Steps Involved

3. Reporting

Each node returns the locally learned incremental model updates to the central server. The central server aggregates all results and stores the new model. It also handles failures (e.g., connection lost with a node while training). The system returns to the selection phase.

4. Termination

When a pre-specified termination criterion (e.g., the maximal number of rounds or local accuracies higher than some target) has been met, the central server orders the end of the iterative training process. The central server contains a robust model that was trained on multiple heterogeneous data sources.

4.7 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. Homomorphic encryption is frequently used for privacy-preserving outsourced storage and computation.

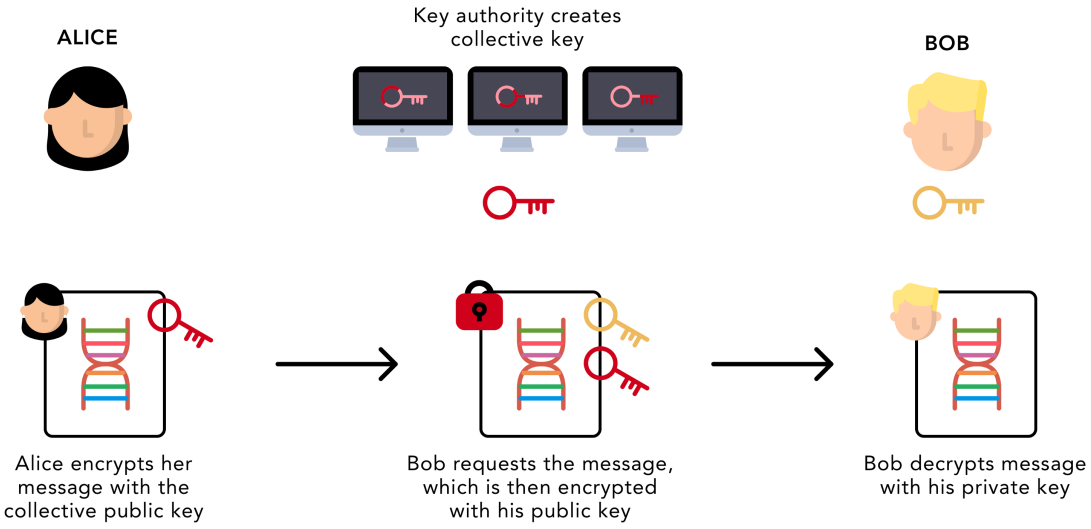

Figure 4.17: Homomorphic Encryption

Applying standard encryption methods presents a dilemma: if the data is stored unencrypted, it can reveal sensitive information to the storage/database service provider. On the other hand, if it is encrypted, it is impossible for the provider to operate on it. If data is encrypted, then answering even a simple counting query (for example, the number of records or files that contain a particular keyword) would typically require downloading and decrypting the entire database content. Homomorphic encryption allows a user to modify data without needing to decrypt it first.

Chapter 5

PROJECT SPECIFICATIONS

5.1 Hardware Specifications

5.1.1 Recommended Configurations

- **CPU Speed:** 1.4 GHz or higher recommended (per core)
- **Processor:** Intel Core i5-8100 (8th Gen) Dual Core or above
- **Memory/ RAM:** 8GB or higher
- **Storage:** 300 GB Hard Disk Drive (HDD)/ 128 GB Solid State Drive (SSD) or above

5.1.2 Minimum Configurations

- **CPU Speed:** 1.2 GHz (per core)
- **Processor:** Intel Core i3-6100 (6th Gen) Dual Core
- **Memory/ RAM:** 4GB
- **Storage:** 100 GB HDD/ 64 GB SSD

5.2 Software Specifications & Requirements

- **Operating System Used:** Microsoft Windows 10 Version 10.0.17763 64 bit
- **Programming Language Used:** Python 3.7.3 64 bit
- **IDEs/ Environments Used:**
  Spyder 3.3.6
  VS Code 1.44.0
  Jupyter Notebook 6.0.3

5.2.1 Project Compatibility

- **Operating Systems:**
  Microsoft Windows 7 or above
  Mac OS El Capitan or above
  Linux Flavours - Debian/Ubuntu, Fedora, Mandriva, Slackware, ArchLinux, Flatpak, openSUSE, RHEL
- **Python Version:** Python 3.x.x
- **IDEs:** Compatible with all Python IDEs provided that all the required packages and modules are installed.
- **Recommended IDEs:** Spyder 3.3.x and VS Code 1.35.1 or above

5.2.2 Python Packages

- pandas
- numpy
- matplotlib
- sklearn
- pickle
- string
- nltk
- sacremoses
- torch
- re
- phe
- keras
- os
- glob
- syft

How to install the packages in Python?

- In case of standalone python IDE:
  - Using pip3 installer in a terminal.
- In case of anaconda environment:
  - Using conda install or pip3 installer in anaconda prompt.

Chapter 6

EVALUATION METRICS USED

Evaluation metrics are used to verify a model's performance. It is crucial to choose proper metrics in order to evaluate how well an algorithm is performing. The evaluation metrics used in our project for comparing different machine learning algorithms are given below:

6.1 F1 Score

In the statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score.

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.1}$$

6.1.1 Precision & Recall

In pattern recognition, information retrieval, and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are, therefore, based on an understanding and measure of relevance. If TP, FP, and FN denote true positives, false positives and false negatives respectively, then:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

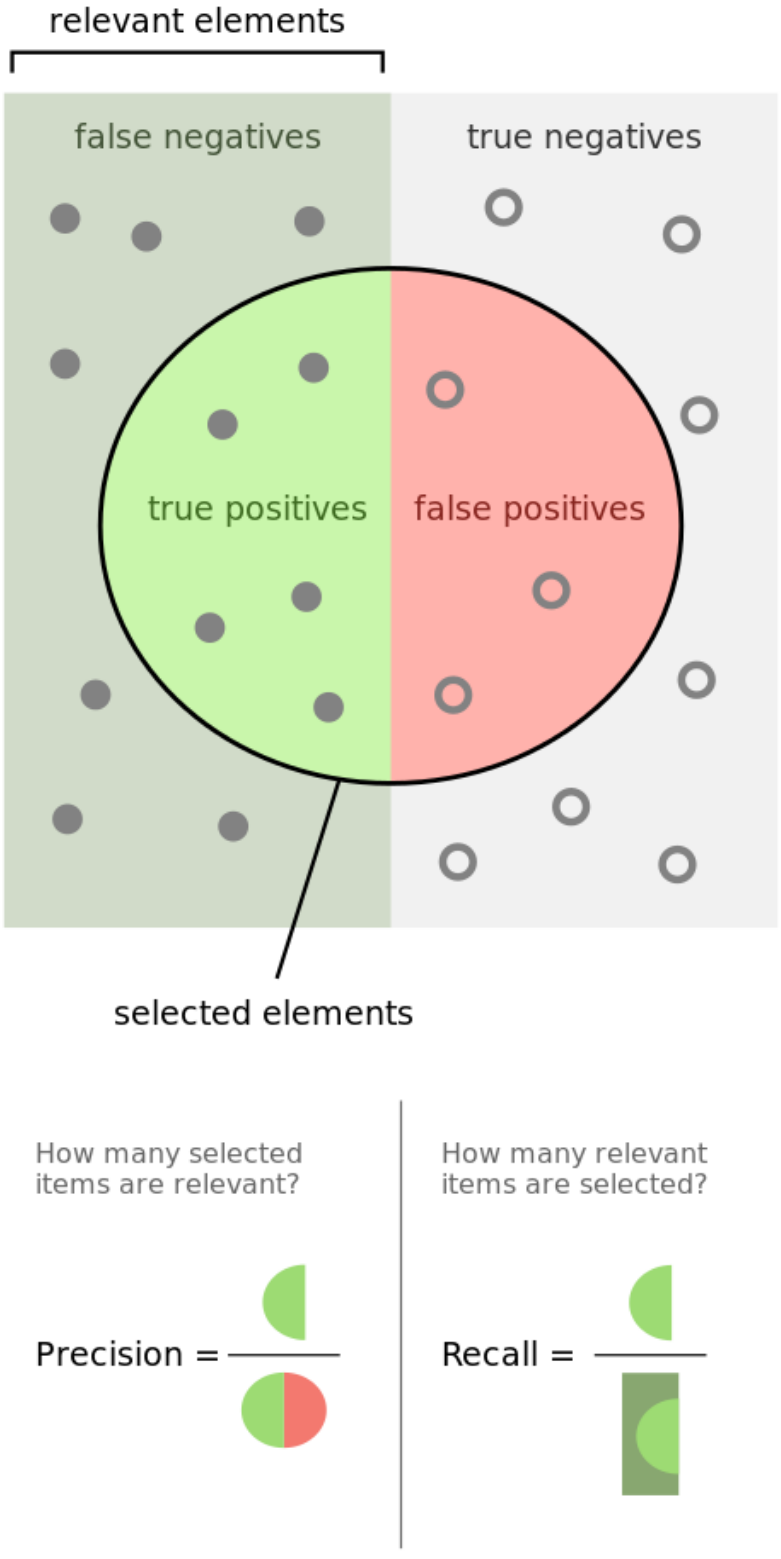

Figure 6.1: Precision and Recall

6.2 Loss & Accuracy

A loss function is used to optimize a machine learning algorithm. The loss is calculated on training and validation, and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or validation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization.

An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage. It is the measure of how accurate your model's prediction is compared to the true data.

6.3 Area Under the Curve (AUC) Score

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

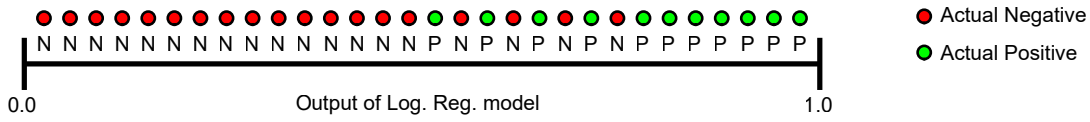

Figure 6.2: AUC

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0. AUC is desirable for the following two reasons:

- AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values.
- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

Chapter 7

ANALYSIS AND RESULTS

7.1 Analysis of Neural Networks

7.1.1 Simulation Settings

To fairly evaluate the performance of all three neural networks, the simulation settings were kept the same for all of them. All of them were trained for a duration of 10 epochs using the adam optimizer. Adam is an extension of stochastic/mini-batch gradient descent and provides the benefits of both AdaGrad and RMSprop optimization functions.

Table 7.1: Simulation Settings

Parameter	Value/Type
Deep Learning Framework	Keras on top of TensorFlow (Python)
Optimizer	Adam
Batch Size	60
Epochs	10
Loss Function	Binary Cross Entropy (Log Loss)

The cost function used in our model is the binary cross-entropy (log loss) function. If $p(y_i)$ denotes the probability outcome of an email being spam as given by our prediction model, then the log loss (also known as binary cross-entropy) cost function can be calculated as:

$$J(\theta) = -\frac{1}{\mathcal{S}} \sum_{i=1}^{\mathcal{S}} y_i * \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) * \log\left(1 - p\left(y_i\right)\right) \tag{7.1}$$

where,

$$0 \leq p(y_i) \leq 1 \tag{7.2}$$

Since a cost function should penalize wrong predictions, the function established in the equation (7.1) works very well for our model. For each example with y_i = 1, it adds the negative log probability of the classification of an email being spam to the function, i.e., if our model

outputs a low probability of an email being spam, then the negative log probability will be high, which will increase the value given by cost function and vice versa.

7.1.2 Evaluation

Table 7.2: Comparison of Neural Networks

Evaluation Metric (Final Value)	Simple RNN	GRU	LSTM
Training Accuracy	0.9985	0.9985	0.9956
Validation Accuracy	0.9408	0.9882	0.9882
Training Loss	0.01	0.012	0.05
Validation Loss	0.11	0.05	0.06

The training and validation accuracies of all three neural networks over ten epochs are given in the figure 7.1 and 7.2, respectively. The training and validation accuracy for the same networks during the same epochs is given in figure 7.3. Since GRU has the highest validation accuracy and the lowest validation loss, we can safely assume that GRU had the best performance on the Enron Email Dataset. Table 7.2 provides the final values of the four metrics used to determine the best algorithm for our model.

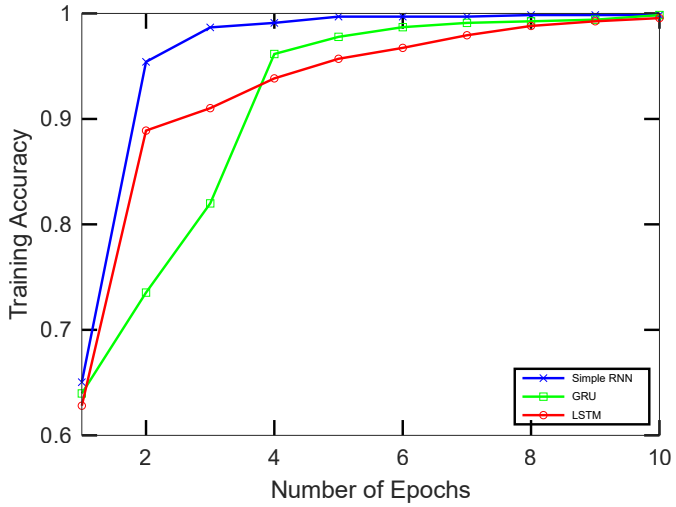

Figure 7.1: Training Accuracy Comparison of Different Neural Networks on our Dataset

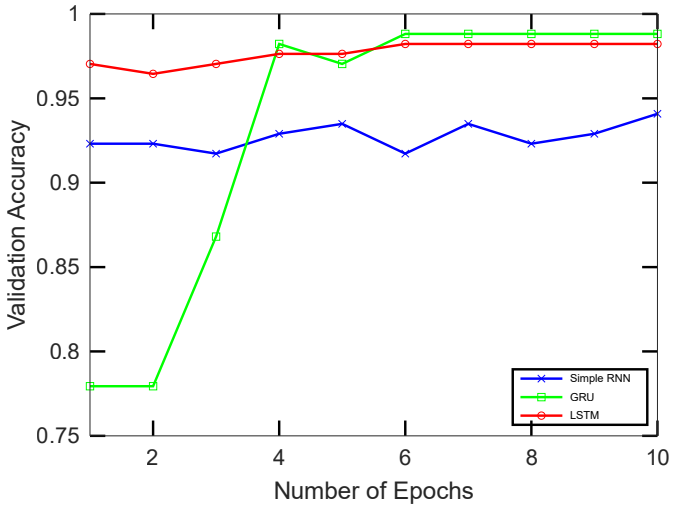

Figure 7.2: Validation Accuracy Comparison of Different Neural Networks on our Dataset

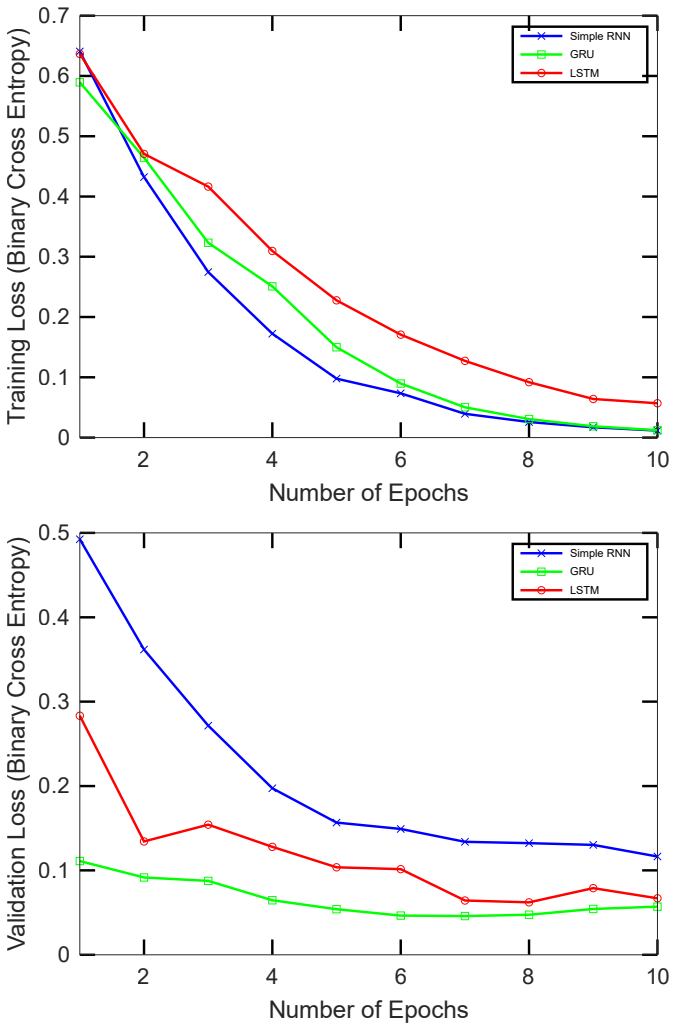

Figure 7.3: Loss Comparison of Different Neural Networks on our Dataset

7.2 Analysis of Traditional Machine Learning Algorithms

The performance of different machine learning algorithms that were implemented, namely, Logistic Regression, Naive Bayes, SVM classifier, Decision Tree Classifier, and Random Forest Classifier, has been evaluated in figures 7.4 and 7.5.

```
train preprocessing is complete
test preprocessing is complete
                    ModelName   F1Score       AUC
0         Logistic Regression  0.900901  0.918264
1                 Naive Bayes  0.338028  0.601695
2  Support Vector Classifier   0.878505  0.895573
3   Decision Tree Classifier   0.727273  0.838474
4   Random Forest Classifier   0.827586  0.882189
```

Figure 7.4: Performance of Traditional Machine Learning Algorithms on Enron Email Dataset

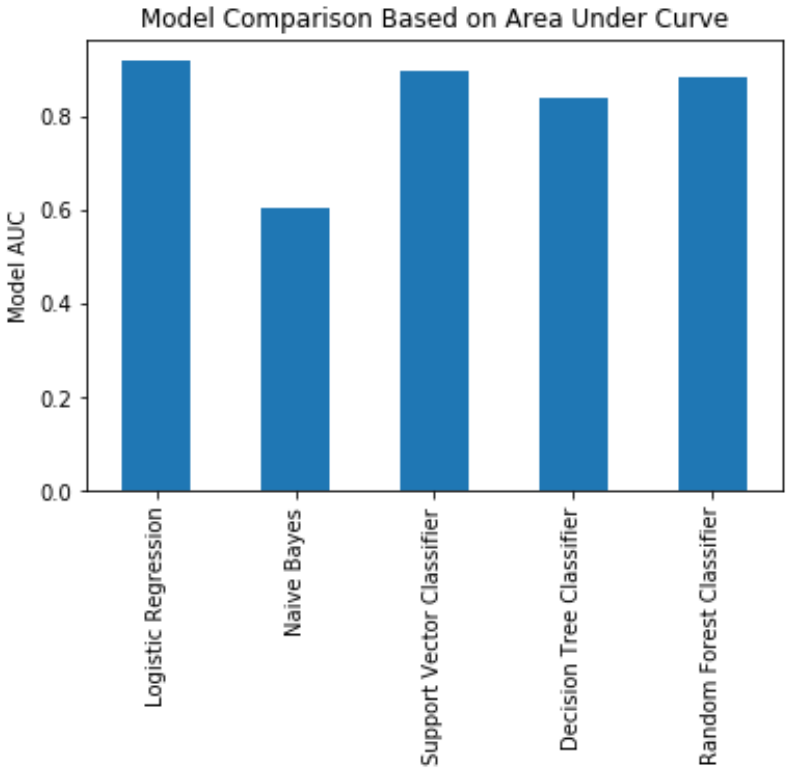

Figure 7.5: Model Comparison Based on Area Under the Curve

It is clear from the figures that Logistic Regression has had the best performance among all the email spam classifiers on our dataset. Support Vector Classifier also yields an excellent

F1 score. Although Decision Tree Classifier and Random Forest Classifier yield a good AUC score, they suffer from a low F1 score. Of all the algorithms, Naive Bayes has proved to have the worst performance, with an F1 score of mere 0.33 and AUC of 0.60. Keeping in line with these results, we have implemented the Logistic Regression algorithm with our Federated Learning framework.

7.2.1 Evaluation of our Federated Learning Framework

GRU-enabled Framework

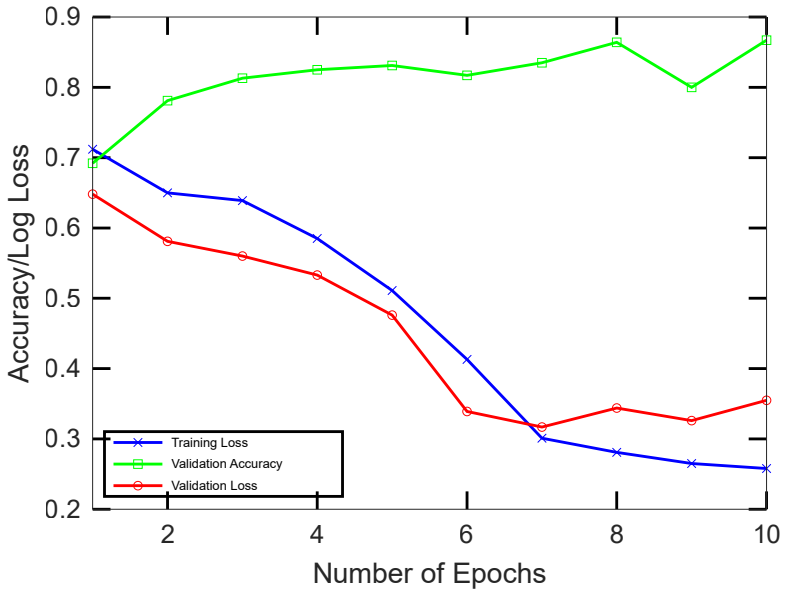

Figure 7.6: Evaluation of GRU-enabled Federated Learning Model

It can be seen from figure 7.6 that our Federated Learning (FL) model powered by GRU manages to get a good level of accuracy (almost 87%). Although it is lesser than the accuracy achieved by the GRU model without Federated Learning (almost 98%), it is important to note that:

- In this project, the Non-Federated Learning GRU model was implemented using an inbuilt python library. Compared to our self-made GRU model that has been used in the Feder-ated Learning framework, the python GRU model is more optimized. Some difference in accuracy between FL and non-FL models is attributed to this reason.
- More importantly, Federated Learning models do tend to have lower accuracy than Non-Federated models since they are not trained on entire datasets at once, but rather, on various smaller datasets.

Logistic Regression Enabled Federated Learning

In table 7.3, it can be seen that the Logistic Regression enabled email spam classifier yields reliable results with and without the use of Federated Learning. As expected, the results of the Non-Federated model are better than those of the Federated model, but only marginally. Considering the fact that Federated Learning ensures that users do not have to compromise on their privacy for availing email spam filtering services, this marginal difference in accuracy can be o verlooked. However, it is important to note that the same cannot be said for services which are very sensitive to even the slickest accuracy changes, such as medical services. Much research is being done in academia to overcome this limitation and make Federated Learning models even more accurate than Non-Federated Learning models.

Table 7.3: Logistic Regression with and without Federated Learning (FL)

Evaluation Metric	**with FL**	**without FL**
F1 Score	0.822	0.900
AUC Score	0.861	0.918

Chapter 8

CONCLUSION AND FUTURE DIRECTION

8.1 Conclusion

This dissertation identified and analyzed an emerging domain in the field of Computer Science, named Federated Learning. Privacy concerns are the primary motivation behind this approach to machine learning. A lot of data on consumer devices is considered private and should not be sent to a server. Federated Learning allows training a model on this data by first processing it locally. Only weight updates derived using the data are sent to a server. Furthermore, the use of homomorphic encryption technique can ensure that the server can only read updates from users once a certain number of them are received. However, it can take a long time until individual iterations are completed since the server needs to wait until users can respond with updates. Furthermore, federated Learning models are often associated with high computational costs and low accuracy. It is notable, however, that the benefits of Federated Learning far outweigh its drawbacks, which are bound to be mastered sooner or later.

8.2 Future Direction

Taking a close look at the problems discussed, we can say that our model is pretty much expandable to other domains as well. However, there exist certain limitations in the proposed model. One limitation of our model is its sensitive nature to the quality of input data that may be inaccurate or have missing information. Another limitation of the proposed model is the decreased accuracy in comparison to Non-Federated models. Therefore, future works could be aimed at finding a way to make federated models more accurate. Moreover, owing to computational constraints, our model hasn't been trained and tested on large datasets.

For future work, it still remains to be shown that more advanced techniques such as differentially-private Federated Learning and the personalization approaches can work in practice. Literature about training more complex models outside of simulations is still missing. While

differentially-private Federated Learning has theoretical guarantees for the level of privacy, the additional computational cost is huge. Future research could focus on making it more feasible to use these techniques.

Bibliography

[1] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[2] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] N. Bouguila and O. Amayri, "A discrete mixture-based kernel for svms: application to spam and image categorization," *Information processing & management*, vol. 45, no. 6, pp. 631–642, 2009.

[5] S. Mason, "New law designed to limit amount of spam in e-mail," 2003.

[6] Z. S. Torabi, M. H. Nadimi-Shahraki, and A. Nabiollahi, "Efficient support vector machines for spam detection: a survey," *International Journal of Computer Science and Information Security*, vol. 13, no. 1, p. 11, 2015.

[7] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[8] B. Yu and Z.-b. Xu, "A comparative study for content-based dynamic spam classification using four machine learning algorithms," *Knowledge-Based Systems*, vol. 21, no. 4, pp. 355–362, 2008.

[9] Y. Feng and H. Zhou, "An effective and efficient two-stage dimensionality reduction algorithm for content-based spam filtering," *J. Comput. Inf. Syst*, vol. 9, no. 4, pp. 1407–1420, 2013.

[10] N. Englesson, "Logistic regression for spam filtering," *Mathematical Statistics Stockholm University Bachelor Thesis*, vol. 9, 2016.

[11] N. F. Rusland, N. Wahid, S. Kasim, and H. Hafit, "Analysis of naïve bayes algorithm for email spam filtering across multiple datasets," in *IOP Conference Series: Materials Science and Engineering*, vol. 226, p. 012091, IOP Publishing, 2017.

[12] P. Poomka, W. Pongsena, N. Kerdprasop, and K. Kerdprasop, "Sms spam detection based on long short-term memory and gated recurrent unit," *International Journal of Future Computer and Communication*, vol. 8, no. 1, 2019.

[13] G. Jain, M. Sharma, and B. Agarwal, "Spam detection in social media using convolutional and long short term memory neural network," *Annals of Mathematics and Artificial Intelligence*, vol. 85, no. 1, pp. 21–44, 2019.

[14] Y. Ren and D. Ji, "Neural networks for deceptive opinion spam detection: An empirical study," *Information Sciences*, vol. 385, pp. 213–224, 2017.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," 2019.

[16] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.

[17] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," 2018.

[18] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, Jan. 2019.

[19] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018.

[20] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2018.

[21] W. W. Cohen, "Enron email dataset." https://www.cs.cmu.edu/ enron/. Accessed On: Feb. 21 ,2020.