Project Report

On

Hindi Spam Mail Detection: Safeguarding Against Unwanted Communication.

Submitted
In partial fulfillment
For the award of the Degree of

PG-Diploma in Artificial Intelligence

(C-DAC, ACTS (Pune))

Guided By:

Dr. Krishnanjan B.

Submitted By:

Dhanashri Kolekar (230340128010)

Sohan Gaikwad (230340128011)

Parth Sevak (230340128017)

Rushikesh Kumbhakarn (230340128021)

Samruddhi Ubhe (230340128028)

Centre for Development of Advanced Computing (C-DAC), ACTS (Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Dr. Krishnanjan B.** C-DAC ACTS, Pune for her constant guidance and helpful suggestion for preparing this project **Hindi Spam Mail Detection: Safeguarding against Unwanted Communication model.** We express our deep gratitude towards her for inspiration, personal involvement, constructive criticism that she provided us along with technical guidance during the course of this project.

We take this opportunity to thank Head of the department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mrs. Namrata Ailawar**, Process Owner, for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P R (Program Head)** and **Dr**. (**Mrs.) Priyanka Ranade** (Course Coordinator, PG-DAI) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS Pune**, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Dhanashri Kolekar (230340128010)
Sohan Gaikwad (230340128011)
Parth Sevak (230340128017)
Rushikesh Kumbhakarn (230340128021)
Samruddhi Ubhe (230340128028)

ABSTRACT

In today's globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been increase in the SPAM emails. SPAM emails, also known as junk email involves nearly identical messages sent to numerous recipients by email. With many languages being digitized for the electronic world, the use of English is still abundant. However, various native languages of different regions are emerging gradually. The Hindi language is getting its pace as a medium for communications used in social media platforms, websites, and emails. With the increased usage of emails, Hindi's number and variety of spam content also increase. Spam emails are inappropriate and unwanted messages usually sent to breach security. These spam emails include phishing URLs, advertisements, commercial segments, and a large number of indiscriminate recipients. Thus, such content is always a hazard for the user, and many studies have taken place to detect such spam content. However, there is a dire need to detect spam emails, which have content written in Hindi language.

Amidst the abundance of digital content, maintaining the integrity of online communication becomes crucial. The "Hindi Spam Mail Detection: Safeguard Against Unwanted Communication" project steps forward with a clear mission: to distinguish between legitimate messages and unwanted spam, particularly in Hindi language. This mission harnesses the combined power of machine learning, deep learning, and natural language processing to build robust models for effective message classification.

. In this project, we use text mining to perform automatic spam filtering to use emails effectively. We try to identify patterns using classification algorithms to enable us classify the emails as HAM or SPAM.

The project adopts a comprehensive strategy, encompassing data preparation, TF-IDF feature extraction, and a diverse range of classification algorithms from conventional methods like Naive Bayes and SVM to sophisticated machine learning model.

Table of Contents

Chapter 1
Introduction

1.1 Introduction

The Internet has become an inseparable part of human lives, where more than four and half billion Internet users find it a convenient to use it for their facilitation. Moreover, emails are considered as a reliable form of communication by the Internet users .Over the decades, e-mail services have been evolved into a powerful tool for the exchange of different kind of information. The increased use of the e-mail also entails more spam attacks for the Internet users. Spam can be sent from anywhere on the planet from users having deceptive intentions that has access to the Internet. Spams are unsolicited and unwanted emails sent to recipients who do not want or need them.

This initiative revolves around the central goal of identifying and differentiating spam messages from authentic ones. While this task may seem daunting due to the nuanced nature of human communication, it is fortified by the fusion of machine learning, deep learning, and natural language processing techniques. By leveraging these cutting-edge methodologies, the project seeks to create models capable of effectively classifying messages in Hindi.

Despite the advancement of spam filtering applications and services, there is no definitive way to distinguish between legitimate and malicious emails because of the ever-changing content of such emails. Spams have been sent for over three or four decades now, and with the availability of various antispam services, even today, nonexpert end-users get trapped into such hideous pitfall. In e-mail managers, spam filters detect spam and forward it to a dedicated space, spam folder, allowing the user to choose whether or not to access them. Spam filtering tools such as corporate e-mail systems, e-mail filtering gateways, contracted antispam services, and end-user training can deal with spam emails in English or any other language.

This document delves into the intricacies of the project's implementation, showcasing how advanced technologies can be harnessed to safeguard digital communication spaces. Through precise classification and spam mails detection, the project emerges as a vital step toward ensuring that individuals engaging in digital conversations can do so with authenticity, trust, and confidence.

1.2 Objective

The objectives of the project work are as -

- To test a system capable of recognizing Hindi Email content.
- To have better understanding of different classification algorithm.
- To study identify spam detection using machine learning and deep learning algorithms as Spam fills inbox with number of ridiculous emails.
- The number of spam emails is rapidly increasing in marketing, chain communications, stock market tips, politics, and education.

The study will put emphasis on the testing of the CRNN software using computer printed and handwritten English alphabets, as the system is capable of learning and recognizing a single character at a time. The duration of training the system will, therefore, be long because the handwritten characters have more complex factors to be considered such as alignment and different writing styles.

Chapter 2

LITERATURE REVIEW

In the era of information technology, information sharing has become very easy and fast. Users can exchange information on a variety of platforms with people all around the world. Email is the most straightforward, affordable, and quick method of disseminating information on a global scale. Emails are also susceptible to a variety of attacks due to their simplicity, with spam being the most prevalent and destructive. Nobody wants to receive emails that are not relevant to them because doing so wastes their time and resources. Additionally, these emails may contain malicious material concealed as attachments or URLs that could compromise the host system's security. Spam is any irrelevant and unwanted message or email sent by the attacker to a significant number of recipients by using emails or any other medium of information sharing. As a result, there is a huge need for email system security. Spam emails could contain Trojans, rats, and viruses. Attackers primarily employ this strategy to entice people to use internet services.

Kriti Agarwal, Tarun Kumar. [1] In this paper, an integrated approach of machine learning based Naïve Bayes (NB) approach and computational intelligence based is considered for the email spam detection. NB is based on the Bayes theorem having strong independence and probability distribution property. For experimentation, above 16617 emails from the spam dataset are considered. Experimental results are evaluated for NB and integrated approach of NB in terms of precision, recall, f-measure and accuracy.

Mohammed Reza Parsei, Mohammed Salehi. [2] In this paper a new approach based on the strategy that how frequently words are repeated was used. The key sentences, those with the keywords, of the incoming emails have to be tagged and thereafter the grammatical roles of the entire words in the sentence need to be determined, finally

they will be put together in a vector in order to take the similarity between received emails. K-Mean algorithm is used to classify the received e-mail.

J. Devlin, M.W. Chang, K. Lee, and K. Toutanova.[3] In this research paper Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks.

Kingshuk Debnath; Nirmalya Kar.[4] In this paper The motivation of this research is to build email spam detection models by using machine learning and deep learning techniques so that spam emails can be distinguished from legitimate emails with high accuracy. The Enron email dataset has been used and deep learning models are developed to detect and classify new email spam using LSTM and CNN. NLP approach was applied to analyze and perform data preprocessing of the text of the email. The results are compared to the previous models in email spam detection. The proposed deep learning approach obtained the highest accuracy.

Isra'a AbdulNabi, Qussai Yaseen.[5] In this paper, state-of-the-art models were experiment against the task of detecting spam emails. Bert contextual word embedding improves the capability of detecting spam emails compared to Keras word embedding that represents each word by a unique integer The results against unseen data reflect the persistence and robustness of the models that were perfectly fit. For future work, results can be improved even higher by taking a larger input sequence. Also, the SPAM detection task can be applied to another text language for e.g: Hindi.

Chapter 3
Methodology and Techniques

3.1 Methodology:

The In the following sections, creating of dataset, training of learning models, and data preprocessing are explained.

3.1.1 Dataset

For this study, the raw data collected is obtained from the online resource kaggle, which will be used to train machine learning models. The data was originally available in English language, and it was obtained in the comma separated values (CSV) format [7]. Further, the dataset obtained was translated using the Googletrans python library in Hindi, which uses the Google Translate Ajax API. After this, a manual correction of the translated data was performed by the authors. We have used our own Hindi translated Hindi scripted dataset, which includes 16617 spam and ham emails. We created our own dataset, because Hindi is written using English alphabets, and Hindi script is based on English alphabets. we obtained a total of 16617 emails, listed in two columns. The first column, labeled as ‘label’ having the two possible values as spam or ham, was meant to be used to classify emails. The second column is Hindi ‘e-mail Text’ and contained a variety of e-mail content. It was decided that up to 80% of the emails will be used to train the models (approximately 12000emails), whereas the remaining 20% will be used to test models individually (4000 emails). emails will have no difficulty doing so now because the dataset is published publicly.

Here is a link to the GitHub repository:
https://github.com/sohangkwd/DAI-Project-Hindi-Spam-Mail-Detection-Safeguarding-Against-Unwanted-Communication..git

भाषानेट

एक बहुभाषी इंटरनेट और वैश्विक स्वीकार्यता पहल नागरिकों को भाषा की रुकावट के बिना डिजिटल माध्यम में आसानी से सूचना बनाने, संवाद करने, आदान-प्रदान करने, संसाधित करने और जानकारी प्राप्त करने में सक्षम बनाने के लिए एक पारिस्थितिकी तंत्र प्रदान करता है।

3.1.2 Data Preprocessing

In machine learning (ML), the preprocessing phrase refers to organizing and managing of raw data before using it to train and test different learning models. In simplistic words, preprocessing is a ML data mining approach that turns raw data into a usable and resourceful structure.

The very first step in the construction of a ML model is preprocessing, in which data from the actual world, typically incomplete, imprecise, and inaccurate owing to flaws and deficient, is morphed into a precise, accurate, and usable input variables and trends.

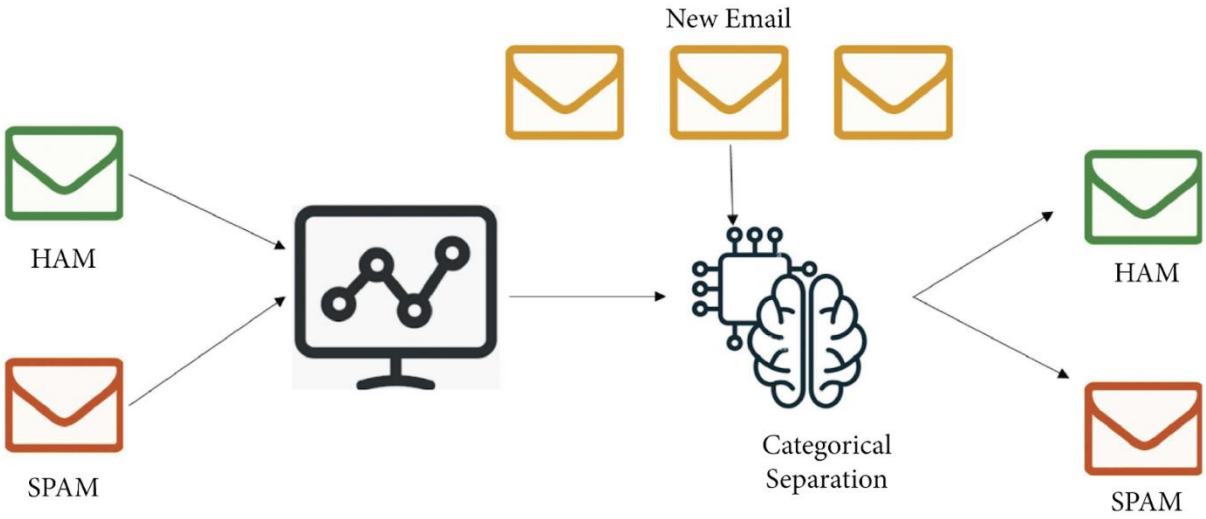


Fig.1 Proposed Work

3.1.3 Import Data

The first stage is to import the dataset, which is downloaded from 'Kaggle' and then converted to CSV format in Hindi [7]. The dataset containing 16617 emails were already classified as spam and ham. The data was obtained while being written in the English Language. As explained before, unlike the usual approach, we have translated the data set into Hindi, to achieve our goal of Hindi spam e-mail detection.

The dataset contains 2 columns "Email Text" and "Label Target".

1. The "Email Text" column contains the email header, subject, and content.
2. The "Target" variable 0 represent "not spam" and 1 represents "spam". The ratio of ham to spam class is 67:33 this is an imbalanced dataset.

3.1.4 Tokenization

Being a critical phase of preprocessing, in this step, all the words from emails are gathered, and the number of times each word appears and location of appearance are counted. With the aid of Count Vectorizer, we were able to find the repetition of words in our dataset. Each word is given a unique number, and hence, they are called tokens, also depicting their occurrences and quantity of occurrences. The token includes one of a kind feature values that will later help in the creation of feature

vectors. In a tokenization phase, every word is assigned a unique token. tokens and unique numbers allotted to every token in a dataset. It is a screenshot of tokens taken after tokenization of dataset with the help of tokenizer.

3.1.5 Stop Word Removal

Once the dataset has been transformed into unique tokens, the next step is to delete every unnecessary word with no significance, e.g., white spaces, commas, full stops, colons, semicolons, and punctuation marks. Stop words elimination is the name given to the method of eliminating unnecessary words. Python has built-in library known as natural language toolkit (NLTK), which has been widely used in language processing. Here, we used NLTK toolkit for stop words removal process for elimination of unnecessary words and spaces.

3.1.6 Stemming

After the tokens have been created, the next step is to stem them. Stemming is the method of converting the dataset's derived terms back to their original forms. First, the base terms are stripped of prefixes and suffixes. Next, both modified and misspelled words are changed into their base or stem words using the stemming algorithm. For this step as well, we used NLTK python library to perform a perfect stemming process. After stemming of the content emails, spam words can be easily identified.

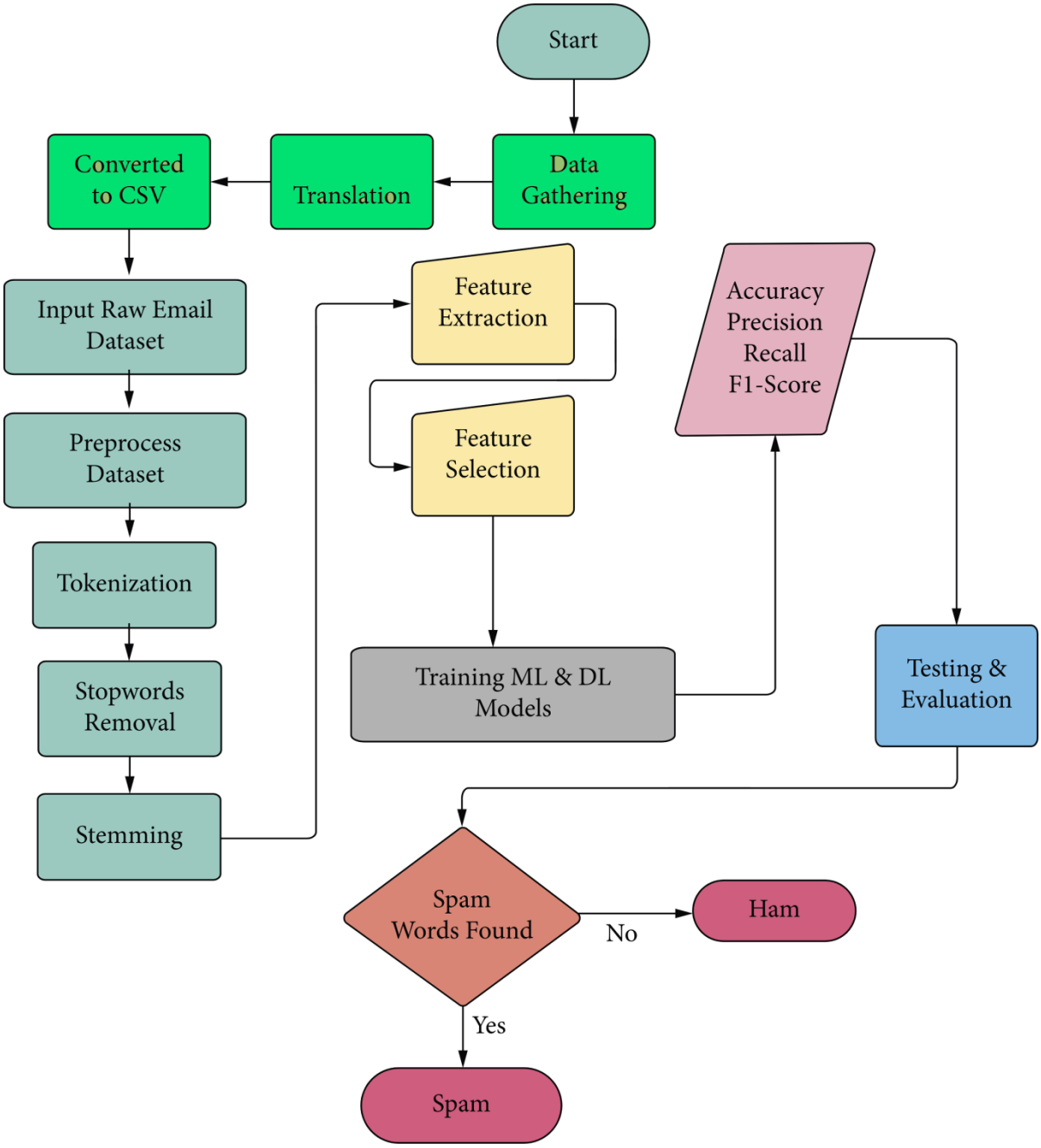


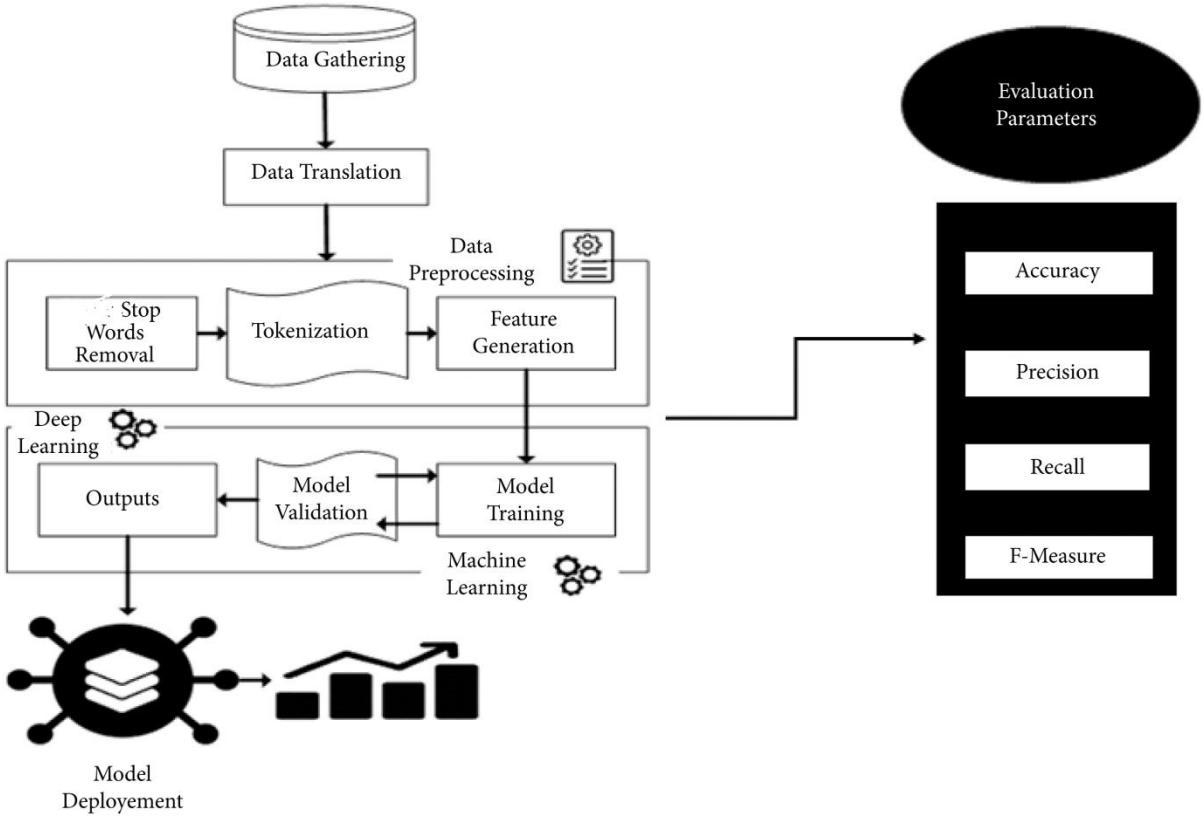
Fig.2 preprocessing diagram

3.2 Model Description

3.2.1 Feature Extraction and Selection

Feature extraction is the process of converting a large raw dataset into a more manageable format. Any variable, attribute, or class can be extracted from the dataset during this step, depending on the original dataset.

Feature extraction is a crucial step in training of the model, which helps in producing more reliable and accurate results. During the feature extraction process, out of the possible many attributes, the method of selecting some key variables that properly characterize data is called feature selection.



3.2.2 Model Training

We considered using convolutional layers for feature extraction, but initial experiments showed that it provided only minor improvement of accuracy and considerably increased the training time. We also tried simple image enhancement techniques (Gaussian blur for noise reduction and binarization) as additional preprocessing steps, which provided small increase of accuracy for Seq2Seq approach

3.2.3 Testing Model

In this module, we test an AI machine planned utilizing research information Quality protection is needed to make the product framework work appropriately. All chance settled upon? Does the program fill in true to form? All program testing standards should be remembered for the specialized detail. What's more, programming testing can uncover every one of the defects and shortcomings that have happened during improvement. Once the application is delivered, you don't need your clients to come to your home together. Various kinds of tests just take into account recognition of blunders during activity.

3.2.1 Support Vector Machine

The Support Vector Machine (SVM) is another supervised machine learning algorithm. It only works for datasets that have been classified. For training purposes, SVM often uses both positive and negative datasets. Negative datasets are not used in any other machine learning model’s preparation. SVM is the most commonly used classification and regression model. For the classification of data, it is more reliable than any other model. SVM is the fastest and most reliable classification model when we only have a small amount of labelled data. The SVM model employs a hyperplane to separate positive and negative values (spam and ham) from the dataset.

3.3 Deep Learning Algorithm

Neural networks are used in deep learning, which is a new technology. The DL models have a collection of hidden layers with weights that can be modified. The DL models are given an input, which is then processed inside hidden layers to make a prediction using adjustable weights.

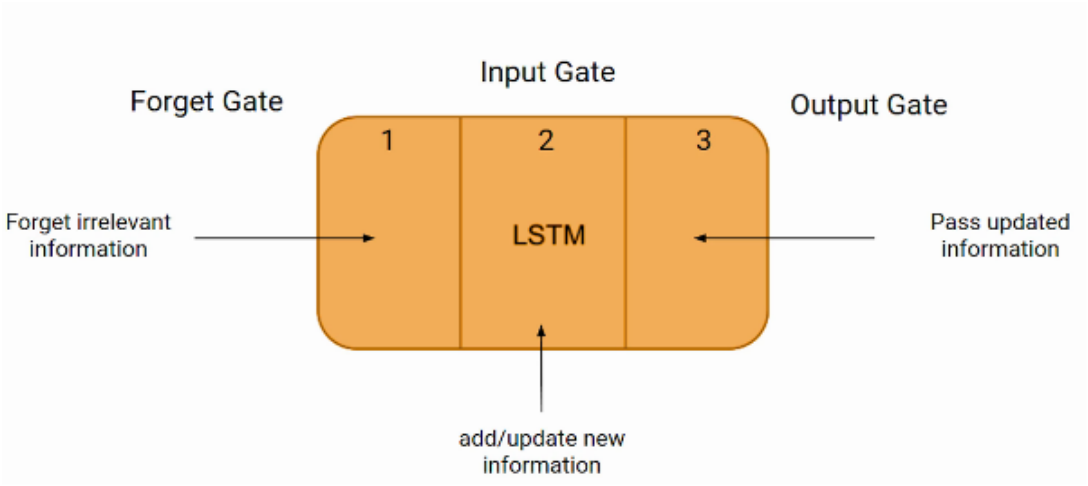
3.3.1 Long Short Term Memory(LSTM)

LSTMs are specially constructed RNN nodes to preserve long lasting dependencies. They consist of self-connected memory cell that can be compared to the classical RNN node and three gates that control output and input of the node. Each gate is in fact a sigmoid function of the input to the LSTM node. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting.

The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell.

The first gate is an input gate which controls whether new input is available for the node. The second gate is a forget gate which makes possible for the node to reset activation values of the memory cell. Here last gate is an output gate controlling which parts of the cell output are available to the next nodes.

Further improvement to the RNN models based on LSTM is achieved by the use of opposite two directional layers or so-called Bidirectional Long Short-Term Memory (BLSTM). The goal of the forward layer is to learn context of the input by processing the sequence from the beginning to the end, while the backwards layer performs the opposite operation by processing the sequence from the end to the beginning.



3.3.2 Gated Recurrent Unit(GRU)

To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

Now, you can see how GRUs are able to store and filter the information using their update and reset gates. That eliminates the vanishing gradient problem since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network.

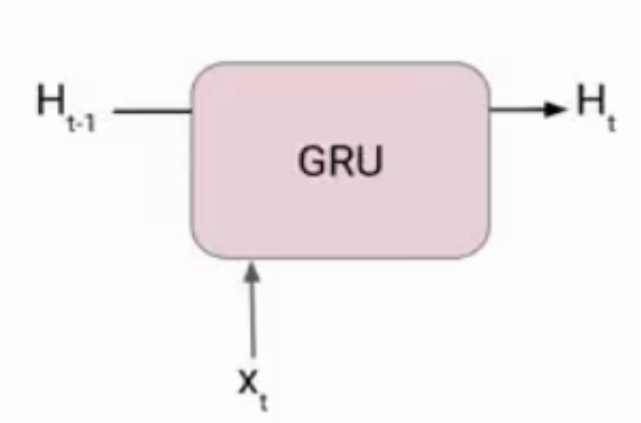


Fig. GRU Architecture

Chapter 4

Implementation

1. Use of Python Platform for writing the code with **Keras**, **TensorFlow**, **OpenCV**
2. Hardware and Software Configuration:

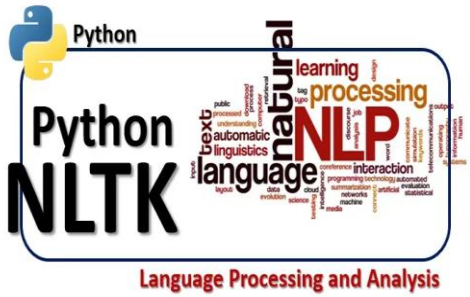
Hardware Configuration:

- CPU: 16 GB RAM, Quad core processor
- GPU: 16GB RAM **Nvidia's GTX 1080Ti**

Software Required:

- **Anaconda**: It is a package management software with free and open-source distribution of the Python and R programming language for scientific computations (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify deployment.
- **Jupyter Notebook**:
  Jupyter is a web-based interactive development environment for Jupyter notebooks, code, and data.
  Jupyter is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.
  Jupyter is extensible and modular: write plugins that add new components and integrate with existing ones.

- **Google Colab**: Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

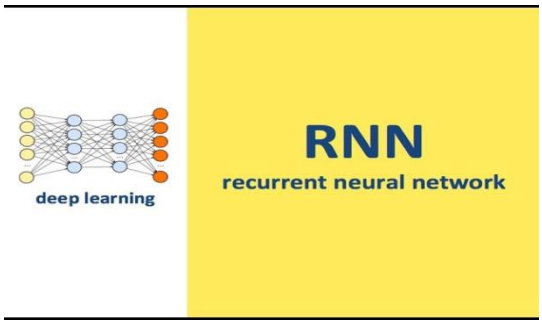


- **NLTK**:The Natural Language Toolkit (NLTK) is a popular open-source library for natural language processing (NLP) in Python. It provides a wide range of tools and resources for working with human language data, including data loading, preprocessing, tokenization, stemming, tagging, parsing, and classification. NLTK is widely used in academia and industry for research, development, and educational purposes.

- **Machine Learning:**

  Machine Learning is an application of Artificial Intelligence (AI) which enables a program(software) to learn from the experiences and improve itself at a task without being explicitly programmed. For example, how would you

write a program that can identify fruits based on their various properties, such as color, shape, size, or any other Machine Learning today has all the attention it needs. Machine Learning can automate many tasks.



- **Deep Learning:**
  Deep learning is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions.
  Recurrent Neural Network (RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.

Model Summary (Seq) -

```
model_rnn = Sequential([
    Embedding(vocab_size, embedding_dim,
                          input_length=max_length),
    SimpleRNN(32),
    Dense(10, activation='relu'),
    Dense(1, activation='sigmoid')
])
model_rnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 32258, 100)        7139300

 simple_rnn (SimpleRNN)      (None, 32)                4256

 dense (Dense)               (None, 10)                330

 dense_1 (Dense)             (None, 1)                 11

=================================================================
Total params: 7,143,897
Trainable params: 7,143,897
Non-trainable params: 0
_________________________________________________________________
```

Model Epochs –

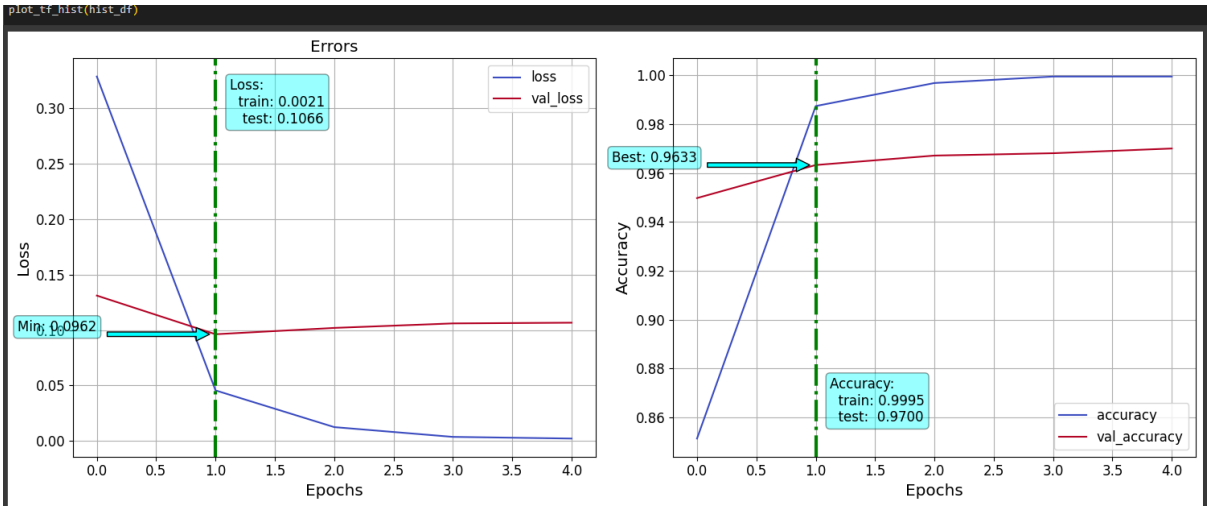
```
model_lstm.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
history = model_lstm.fit(trn_pad, y_train, epochs=5,
                 validation_data = (tst_pad, y_test),verbose=1)
```

```
Epoch 1/5
130/130 [==============================] - 214s 2s/step - loss: 0.3286 - accuracy: 0.8513 - val_loss: 0.1311 - val_accuracy: 0.9498
Epoch 2/5
130/130 [==============================] - 207s 2s/step - loss: 0.0455 - accuracy: 0.9874 - val_loss: 0.0962 - val_accuracy: 0.9633
Epoch 3/5
130/130 [==============================] - 204s 2s/step - loss: 0.0124 - accuracy: 0.9969 - val_loss: 0.1019 - val_accuracy: 0.9671
Epoch 4/5
130/130 [==============================] - 205s 2s/step - loss: 0.0036 - accuracy: 0.9995 - val_loss: 0.1059 - val_accuracy: 0.9681
Epoch 5/5
130/130 [==============================] - 208s 2s/step - loss: 0.0021 - accuracy: 0.9995 - val_loss: 0.1066 - val_accuracy: 0.9700
```

Loss Curve –

```
plot_tf_hist(hist_df)
```




```
hist_df.describe()
```

	loss	accuracy	val_loss	val_accuracy
count	5.000000	5.000000	5.000000	5.000000
mean	0.078463	0.966925	0.108334	0.963671
std	0.140914	0.064824	0.013360	0.008158
min	0.002129	0.851306	0.096191	0.949758
25%	0.003623	0.987427	0.101883	0.963285
50%	0.012442	0.996857	0.105949	0.967150
75%	0.045544	0.999516	0.106595	0.968116
max	0.328576	0.999516	0.131054	0.970048

Classification Using Deep Learning Model –

Model Summary(GRU)-

In [36]:

```
model_gru.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model_gru.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 416, 100)          897300

 bidirectional (Bidirectiona  (None, 64)               25728
 l)

 dense_4 (Dense)             (None, 10)                650

 dense_5 (Dense)             (None, 1)                 11

=================================================================
Total params: 923,689
Trainable params: 923,689
Non-trainable params: 0
_________________________________________________________________
```

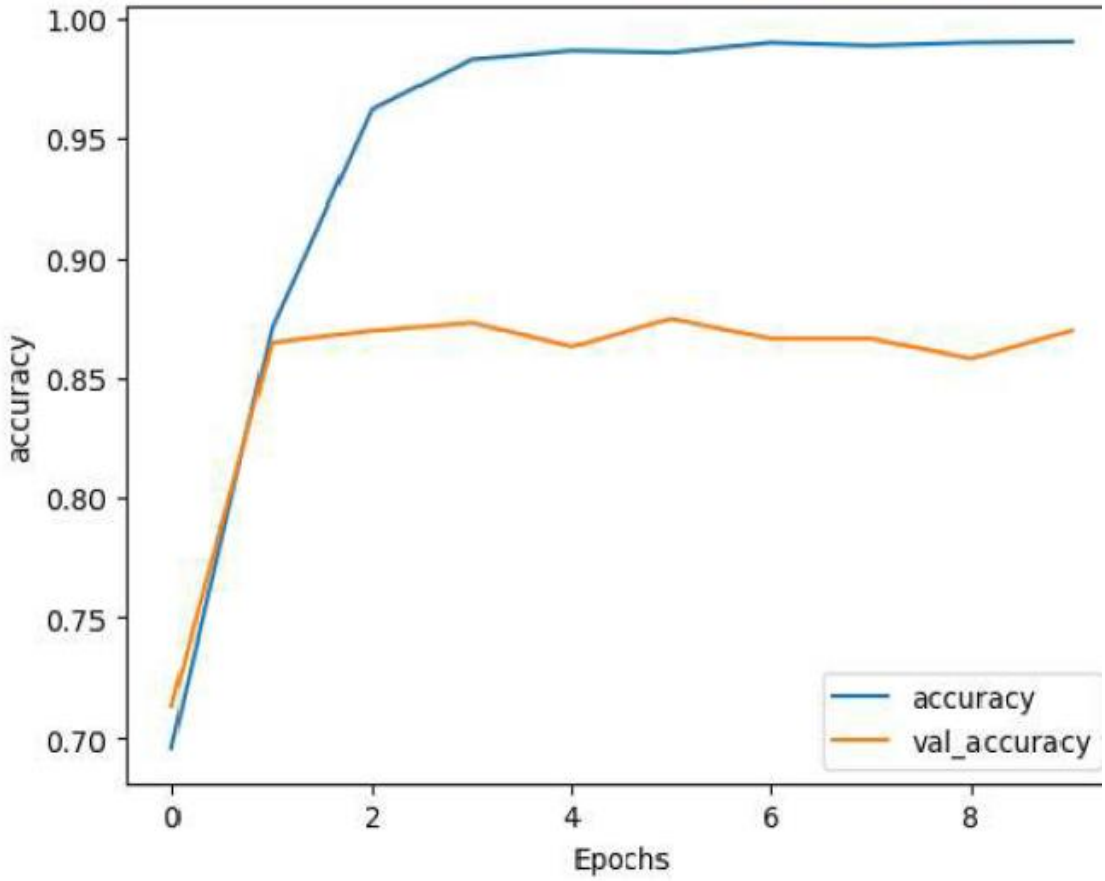
Model Epochs –

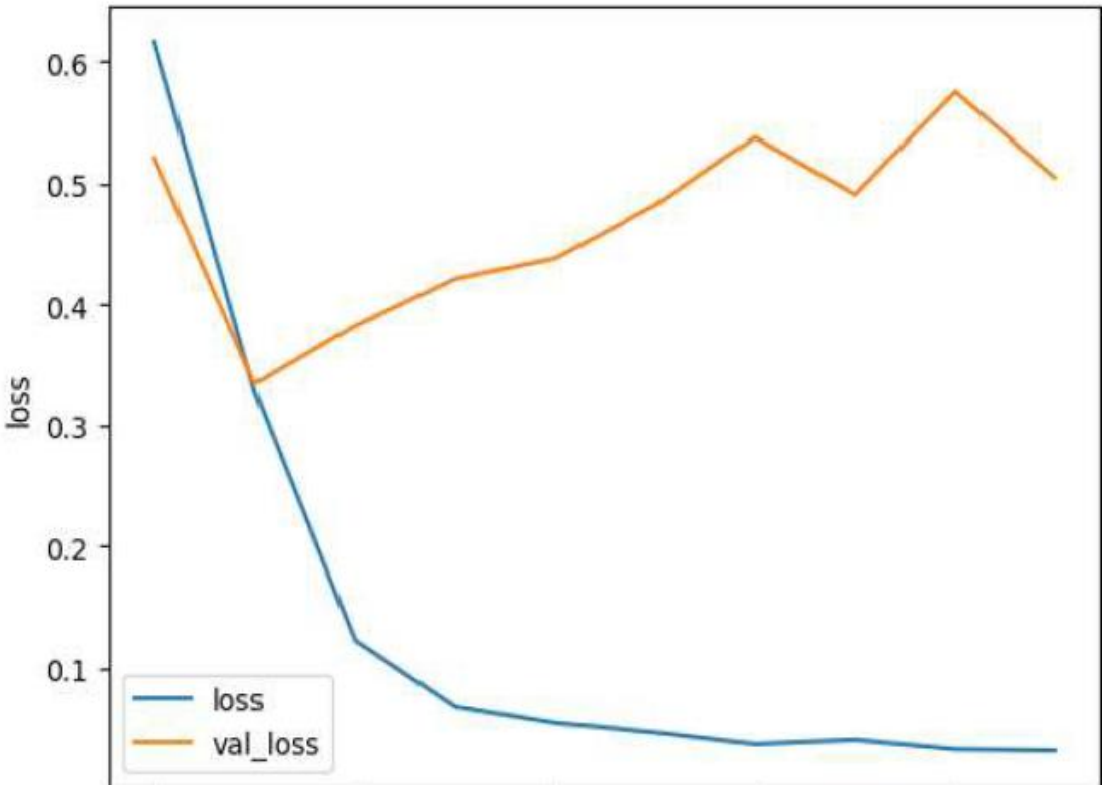
In [37]:

```
history = model_gru.fit(trn_pad, y_train, epochs=10,
                    validation_data = (tst_pad, y_test),verbose=1)
```

```
Epoch 1/10
75/75 [==============================] - 17s 187ms/step - loss: 0.6163 - a
ccuracy: 0.6958 - val_loss: 0.5212 - val_accuracy: 0.7133
Epoch 2/10
75/75 [==============================] - 13s 172ms/step - loss: 0.3282 - a
ccuracy: 0.8712 - val_loss: 0.3347 - val_accuracy: 0.8650
Epoch 3/10
75/75 [==============================] - 13s 173ms/step - loss: 0.1226 - a
ccuracy: 0.9621 - val_loss: 0.3823 - val_accuracy: 0.8700
Epoch 4/10
75/75 [==============================] - 13s 170ms/step - loss: 0.0679 - a
ccuracy: 0.9829 - val_loss: 0.4213 - val_accuracy: 0.8733
Epoch 5/10
75/75 [==============================] - 13s 169ms/step - loss: 0.0545 - a
ccuracy: 0.9867 - val_loss: 0.4384 - val_accuracy: 0.8633
Epoch 6/10
75/75 [==============================] - 13s 168ms/step - loss: 0.0468 - a
ccuracy: 0.9858 - val_loss: 0.4818 - val_accuracy: 0.8750
Epoch 7/10
75/75 [==============================] - 12s 166ms/step - loss: 0.0370 - a
ccuracy: 0.9900 - val_loss: 0.5378 - val_accuracy: 0.8667
Epoch 8/10
75/75 [==============================] - 12s 166ms/step - loss: 0.0405 - a
ccuracy: 0.9887 - val_loss: 0.4911 - val_accuracy: 0.8667
Epoch 9/10
75/75 [==============================] - 12s 166ms/step - loss: 0.0327 - a
ccuracy: 0.9900 - val_loss: 0.5752 - val_accuracy: 0.8583
Epoch 10/10
75/75 [==============================] - 12s 164ms/step - loss: 0.0318 - a
ccuracy: 0.9904 - val_loss: 0.5049 - val_accuracy: 0.8700
```

Loss Curve -


1.00
0.95
0.90
0.85
0.80
0.75
0.70
accuracy
0
2
4
6
8
Epochs
accuracy
val_accuracy


0.6
0.5
0.4
0.3
0.2
0.1
loss
loss
val_loss

Model Summary (LSTM)-

In [39]:

```
from keras.layers import LSTM
```

In [40]:

```
model_lstm = Sequential([
    Embedding(vocab_size, embedding_dim,
                              input_length=max_length),
    Bidirectional(LSTM(32)),
    Dense(10, activation='relu'),
    Dense(1, activation='sigmoid')
])
model_lstm.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 416, 100)          897300

 bidirectional_1 (Bidirectio  (None, 64)               34048
 nal)

 dense_6 (Dense)             (None, 10)                650

 dense_7 (Dense)             (None, 1)                 11

=================================================================
Total params: 932,009
Trainable params: 932,009
Non-trainable params: 0
_________________________________________________________________
```

In [41]:

```
model_lstm.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

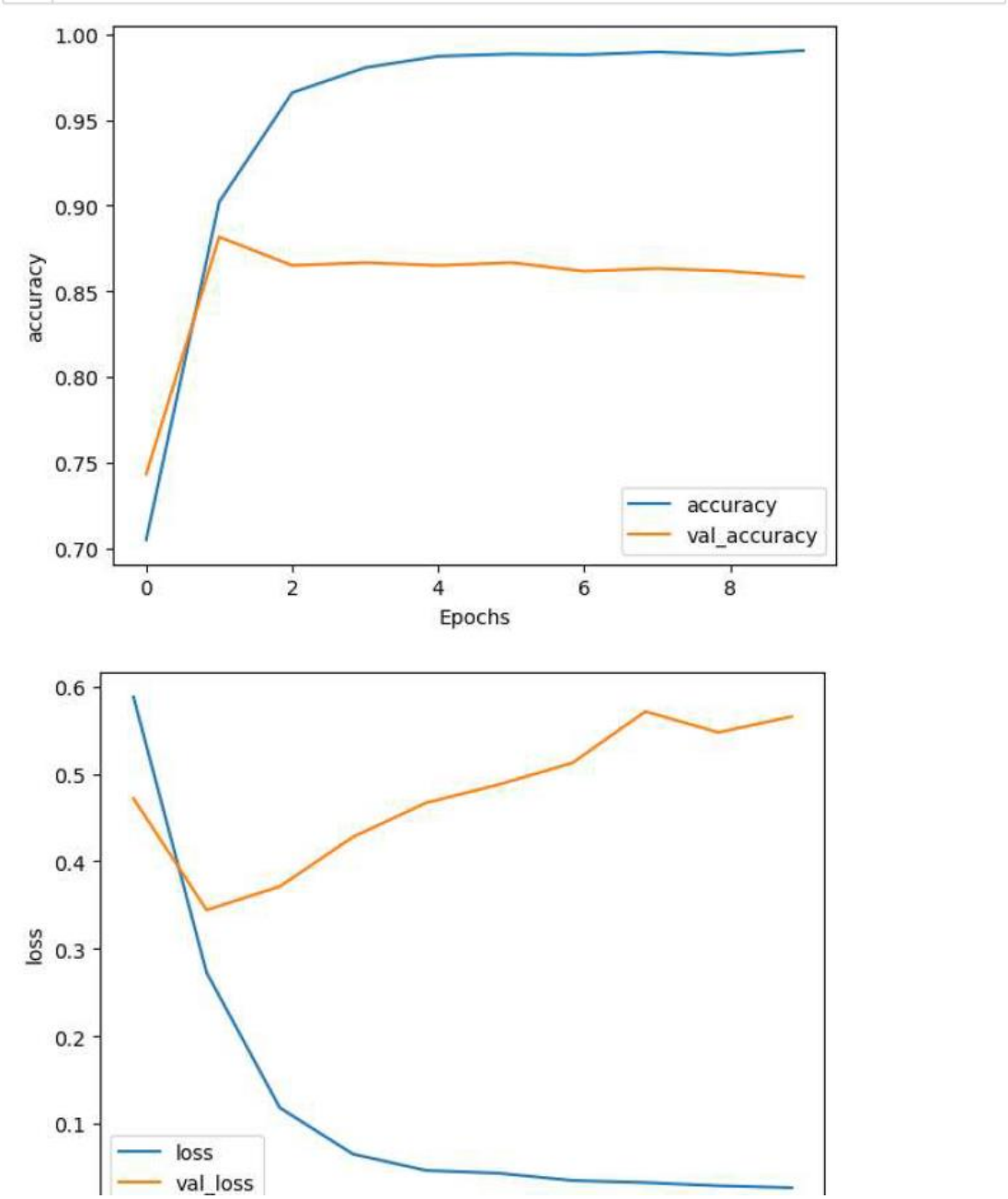
Model Epochs –

In [42]:

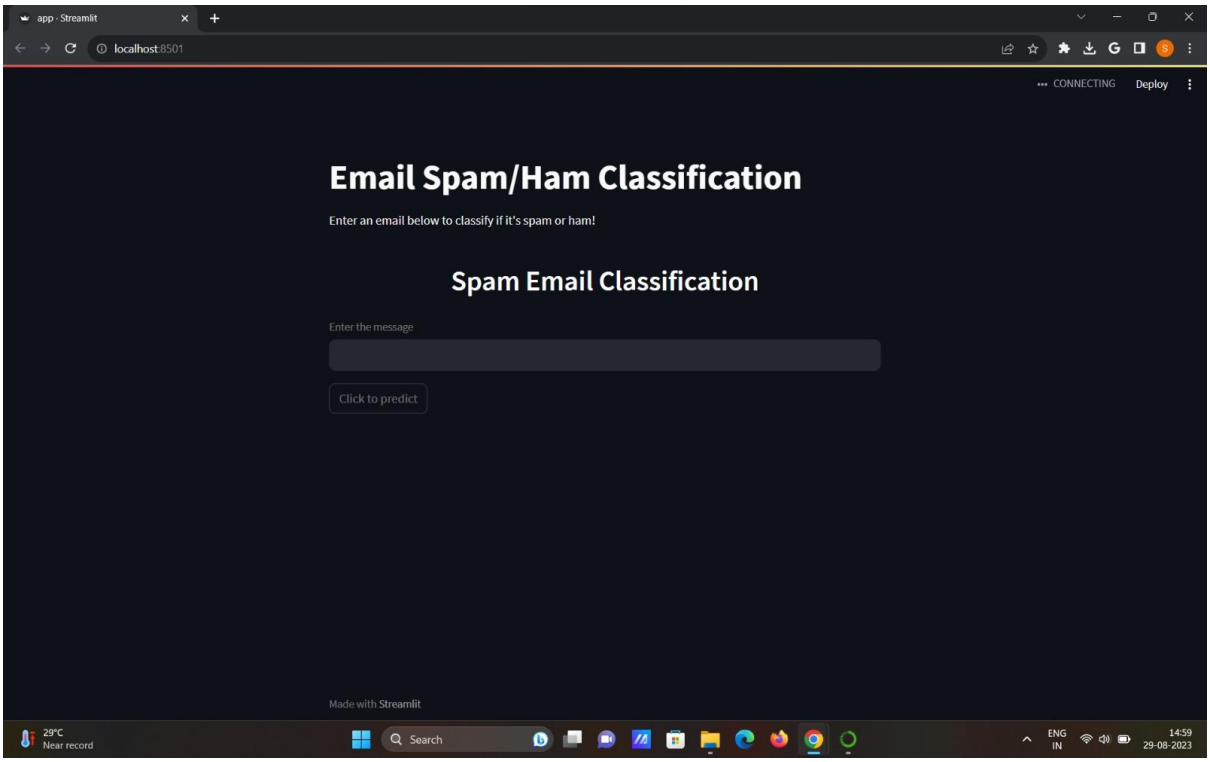
```
history = model_lstm.fit(trn_pad, y_train, epochs=10,
                  validation_data = (tst_pad, y_test),verbose=1)
```

```
Epoch 1/10
75/75 [==============================] - 19s 211ms/step - loss: 0.5880 - a
ccuracy: 0.7050 - val_loss: 0.4719 - val_accuracy: 0.7433
Epoch 2/10
75/75 [==============================] - 15s 198ms/step - loss: 0.2728 - a
ccuracy: 0.9021 - val_loss: 0.3443 - val_accuracy: 0.8817
Epoch 3/10
75/75 [==============================] - 15s 199ms/step - loss: 0.1181 - a
ccuracy: 0.9658 - val_loss: 0.3712 - val_accuracy: 0.8650
Epoch 4/10
75/75 [==============================] - 15s 203ms/step - loss: 0.0652 - a
ccuracy: 0.9804 - val_loss: 0.4278 - val_accuracy: 0.8667
Epoch 5/10
75/75 [==============================] - 15s 200ms/step - loss: 0.0464 - a
ccuracy: 0.9871 - val_loss: 0.4669 - val_accuracy: 0.8650
Epoch 6/10
75/75 [==============================] - 15s 199ms/step - loss: 0.0432 - a
ccuracy: 0.9883 - val_loss: 0.4879 - val_accuracy: 0.8667
Epoch 7/10
75/75 [==============================] - 15s 202ms/step - loss: 0.0349 - a
ccuracy: 0.9879 - val_loss: 0.5126 - val_accuracy: 0.8617
Epoch 8/10
75/75 [==============================] - 15s 199ms/step - loss: 0.0325 - a
ccuracy: 0.9896 - val_loss: 0.5715 - val_accuracy: 0.8633
Epoch 9/10
75/75 [==============================] - 15s 200ms/step - loss: 0.0289 - a
ccuracy: 0.9879 - val_loss: 0.5475 - val_accuracy: 0.8617
Epoch 10/10
75/75 [==============================] - 15s 206ms/step - loss: 0.0266 - a
ccuracy: 0.9904 - val_loss: 0.5656 - val_accuracy: 0.8583
```

Loss Curve –


1.00
0.95
0.90
0.85
0.80
0.75
0.70
accuracy
0
2
4
6
8
Epochs
accuracy
val_accuracy
0.6
0.5
0.4
0.3
0.2
0.1
loss
loss
val_loss

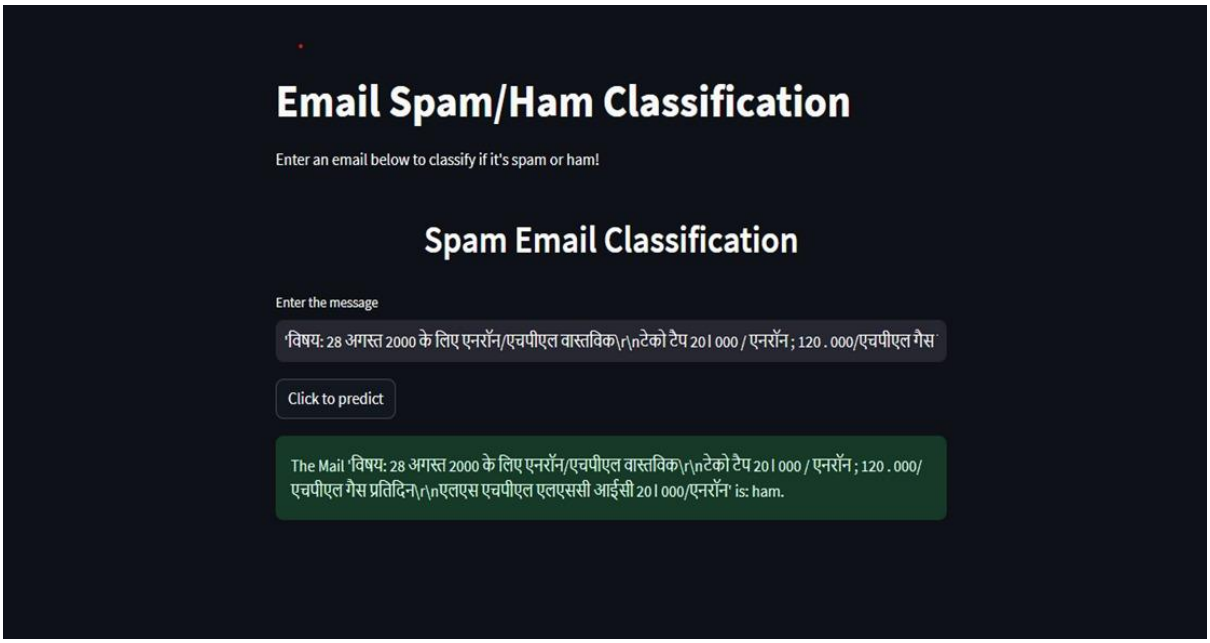
User Interface –



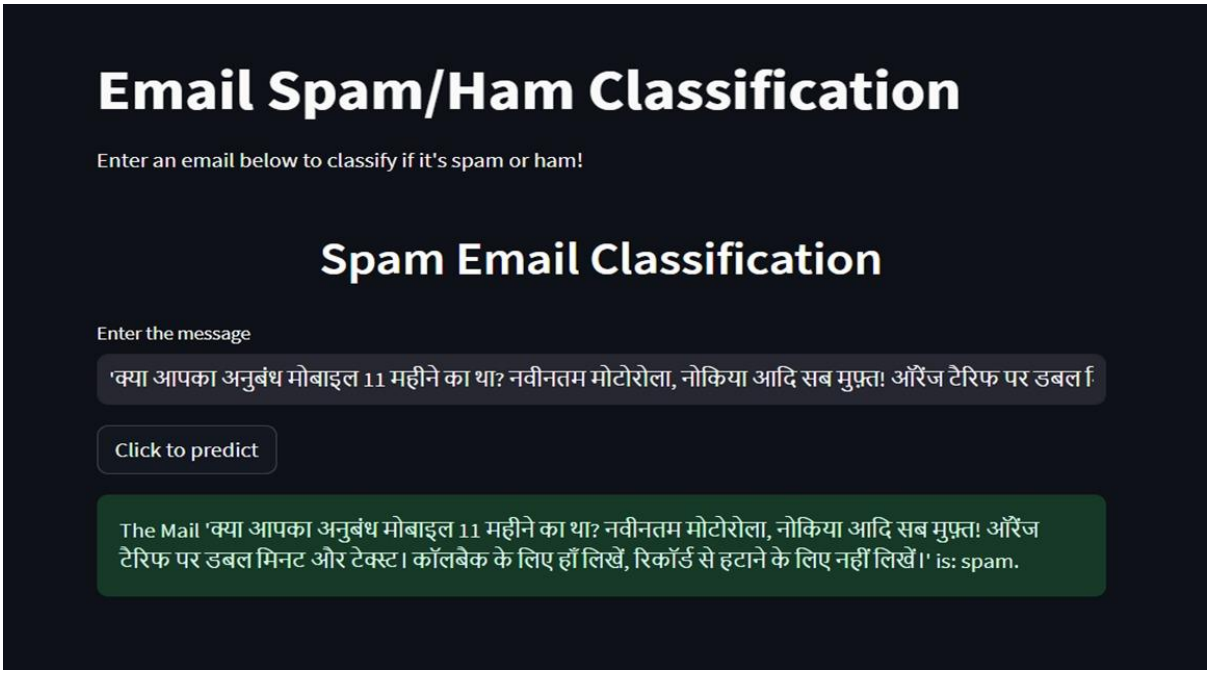
Chapter 5

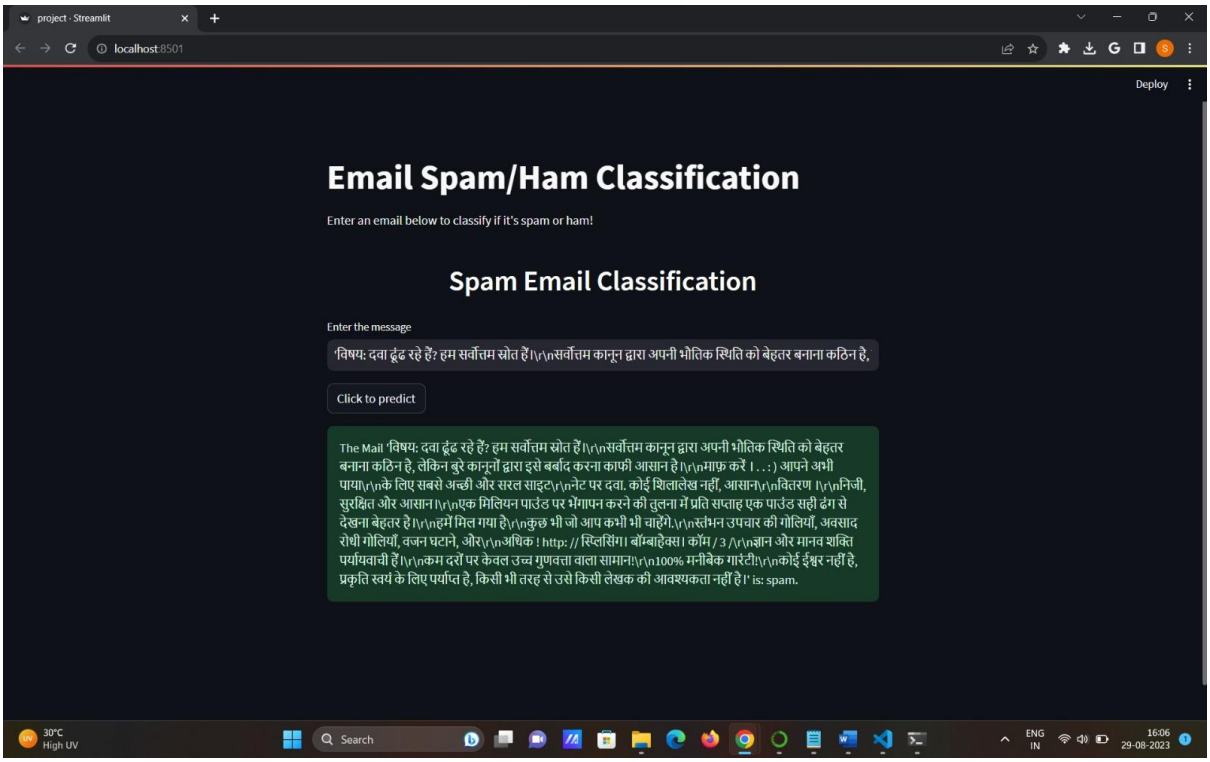
Results

Ham Email Predicted Results–



Spam Email Classification –





Accuracy Score –

Model Evaluation

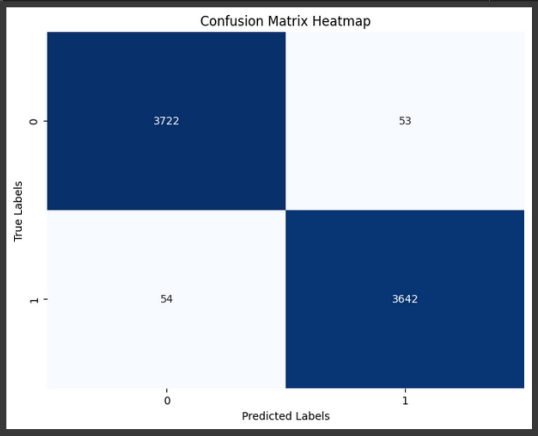
```
[ ]  1 from sklearn.metrics import classification_report
     2 print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.99	0.97	0.98	2424
1	0.97	0.99	0.98	2441
accuracy			0.98	4865
macro avg	0.98	0.98	0.98	4865
weighted avg	0.98	0.98	0.98	4865

Confusion Matrix

```
[70] import matplotlib.pyplot as plt
     import seaborn as sns
```

```
confusion_matrix_plot = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_plot, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



SVM

SMOTE

```
[53] svm = SVC(kernel='linear',random_state=0)
```

```
[54] svm.fit(x_train,y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
[55] y_pred = svm.predict(x_test)
```

```
[56] accuracy_score(y_test,y_pred)
```

```
0.9856779547583991
```

```
[57] f1_score(y_test,y_pred)
```

```
0.9855229332972535
```

Classification Report

```
from sklearn.metrics import accuracy_score,classification_report
```

```
[59] classification_report_hindidata = classification_report(y_test, y_pred)
```

```
print("Classification Report:")
print(classification_report_hindidata)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3775
           1       0.99      0.99      0.99      3696

    accuracy                           0.99      7471
   macro avg       0.99      0.99      0.99      7471
weighted avg       0.99      0.99      0.99      7471
```

Chapter 6

Conclusion

6.1 Conclusion

- In conclusion, the "Hindi Spam Mail Detection" project exemplifies the potential of advanced technologies in safeguarding digital communication realms. By empowering users to interact confidently and authentically online, the project pioneers a safer and more reliable digital experience for Hindi-speaking individuals.

The model can be adapted for scalability, better convergence, and better accuracy.

6.2 Future Enhancement –

- Reduce the time of processing.
- Extension for various Document Classification applications.
- Scope to integrate various contextual features.
- Multi-category Classification.
- Inbox Classification as well as Email Filteration.

Chapter 7

References

[1] Kriti Agarwal, Tarun Kumar “Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization”, Proceedings of
the Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018.

[2] Mohammed Reza Parsei, Mohammed Salehi E-Mail Spam Detection Based on Part of Speech Tagging 2nd International Conference on Knowledge
Based Engineering and Innovation (KBEI), 2015.

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[4] K. Debnath and N. Kar, "Email Spam Detection using Deep Learning Approach," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), Faridabad, India, 2022, pp. 37-41, doi: 10.1109/COM-IT-CON54601.2022.9850588.

[5] AbdulNabi, I., Yaseen, Q. “Spam email detection using deep learning techniques” 12th International Conference on Ambient Systems, Networks and Technologies, ANT 2021 / 4th International Conference on Emerging Data and Industry 4.0, EDI40 2021.

[7] https://github.com/ryanhsin98/SpamEmailPrediction/tree/main

[8]https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[9] https://github.com/sks5495/Spam-Filter-for-Hindi-Emails

[10] https://www.kaggle.com/datasets/rajnathpatel/multilingual-spam-data