PH1$H HOOK

An AI based Phishing (fish•uhng) detection using NER, Sequence Models and many more ...

By: TEAM 6

Name: Samarth Sharma

Deadline: 7th November 2024

Course: Natural Language Processing

TABLE OF CONTENTS

Chapter 1: Project Overview 03

Chapter 2: Introduction to Phishing and its impact on Cyber Crime 05

Chapter 3: ENRON Email Dataset 09

Chapter 4: Machine Learning and Deep Learning for Classification of Email Content and Results 13

Chapter 5: Website Display 21

Chapter 6: Future Developments 24

Chapter 7: References 27

CHAPTER 1: PROJECT OVERVIEW

The project was developed as a mandatory project for the course Natural Language Processing, an elective for the students of Bachelors of Technology 7th semester at the Indian Institute of Information Technology (IIIT) Lucknow, which presented five ideas to select from for the creation of the project.

The five options are mentioned as follows:

1. Implement the NIST (National Institute of Standards and Technology) Database with help of NLP model/methodology and extract information such as: type of environments/operating system, Attacks vectors cast-off, pre-requisites input and potential output etc. (https://www.nist.gov/) To extract information from the NIST database involves several steps such as, including data retrieval, preprocessing, NLP model, and information extraction.

2. Implement the CVE Database with help of NLP model/methodology and extract the needful information such as the goal is to automate the extraction of key information from CVE entries and make it accessible for further analysis or integration into security systems. The detailed plan that includes the model/methodology, steps, and examples of how to achieve this using NLP techniques and Python libraries like BeautifulSoup, requests, and transformers for more advanced models. (https://www.cve.org/)

3. Implement the MITRE ATT&CK Framework with help of NLP model/methodology and extract information such as: type of environments/operating system, Attacks vectors cast-off, pre-requisites input and potential output etc. Extract and classify tactics, techniques, and procedures (TTPs) from threat reports. (https://cve.mitre.org)
4. Implement the Phishing Email Detection Framework with help of NLP model/methodology and extract information. Detect and classify phishing emails using NLP. (https://www.phishtank.com)
5. Implement the Security Policy and Compliance Automation with help of NLP model/methodology and extract information. Automate the extraction and enforcement of security policies from regulatory documents.

The team deliberated over the presented ideas and concluded for the selection of Option 4. The project was worked on and developed by the following team members:

- Aditya Tripathi: LCB2021011
- Saumya: LIT2021058
- Samarth Sharma: LCI2021010
- Ashwani Dhiman: LCI2021046

The dataset ‘Enron Corpus’ or ‘Enron Email Dataset’ was utilised for the training of the Natural Language Processing (NLP) algorithm which takes this dataset and attempts to understand the mapping of the corpus to the corresponding prediction for detectiong phishing attempts (if any)

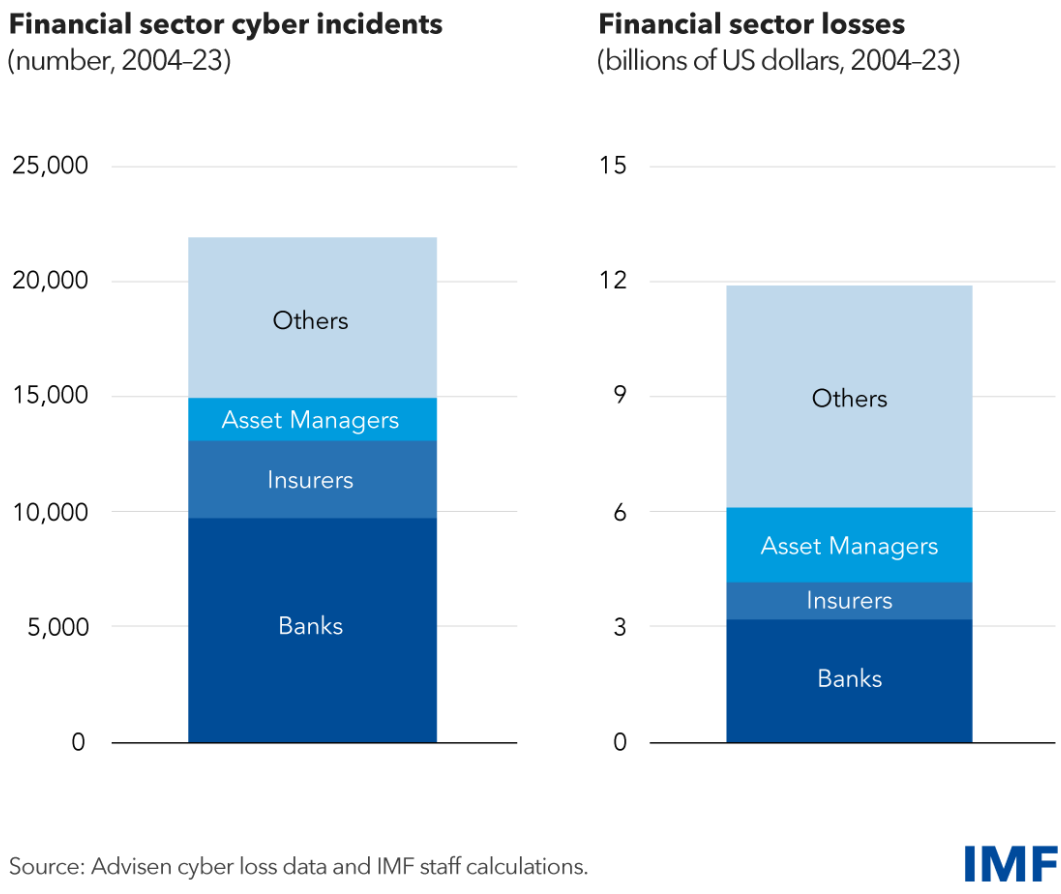
For creating the algorithm that trains on the above dataset, we are using Logistic Regression, LSTMs and Naive Bayes Algorithm.

Finally, to present the project overall, we are creating a website where we would deploy our trained model. The website utilised native small server side applications such as Flask and Raw codes in HTML, CSS and JS.

CHAPTER 2: INTRODUCTION TO PHISHING AND ITS IMPACT ON CYBER CRIME

In the 21st century, as the number of people connecting to the internet rises, so does the variety of attempts in the fields of cyber threats, hacking, and other attacks. These attacks come under the field of Cyber Crime, and their damaging impact has been increasing of the past years.

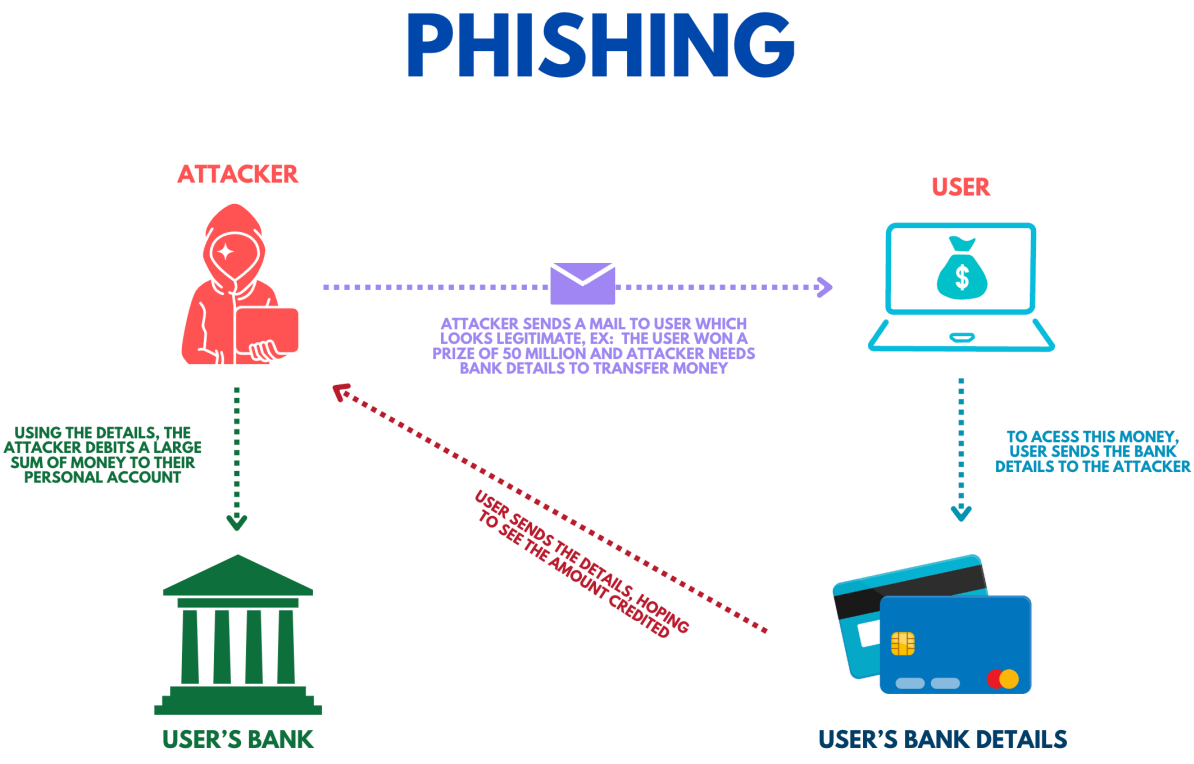
According to the International Monetary Fund (or IMF), the financial sector has suffered more than 20,000 cyberattacks, causing USD 12 Billion in losses over the past 20 years.



For instance, The Bangladesh Bank Cyber Heist, observed in February 2016 observed the alleged North Korean backed - Lazarus Group, the elite cyber-crime organisation, attempt to illegallly transfer US 1 Billion into accounts of four individuals (30 out of the 35 attempts were detected, resulting in a loss of USD101 Million), who later on whitewashed the money in casinos in Phillippines and finally escaping to Macau. It was also observed that a Sri Lankan NGO firm may also have been involved in money laundering.

An academic report[1], discussing the events of the heist mentions how it is suspected by the Federal Bereau of Investigation that the Lazarus Group utilised Spear Phishing, a form of Phishing to target specific individuals, groups or organisations (in this case Bangladesh Bank) to either gain access to a system or extract money from an individual. In this case, the hackers gained access to the banking's international transfer system and strategically attacked the organisation by taking advantage of series of holidays (Bangaldesh is closed on Fri-Sat, United States SWIFT is closed on Sat-Sun and the following Mon-Tue was holiday in Phillipines due to the Chinese New Year, giving enough time to launder and escape with the sum).

Phishing (pronounced fish•uhng) is a form of social engineering attack which involves tricking the victim into taking an action which might benefit the attacker. The wide variety of the complexity of the possible attempts range from a simple *Nigerian Prince Scam* to the elaborate scheme mentioned above.



Phishing has a devasting impact on many individuals and organisations. According to Egress, a security company, over 90% of organisations have fallen victims to the phishing attacks in the year 2023, with over 14 million attempts to cause cyber damage. According to a report [2] by an American-Japanese cyber security software company, in comparision to the previous year (2022), phishing attacks aimed towards data theft saw a growth of nearly 17% attempts, while unknown credential linking attacks rose by a whopping 29%.

There are various types of Phishing attacks. Some of them are mentioned below:

- Nigerian Prince: Known since late 1800s (then by name of Spanish Prisoner Scam), an alleged Nigerian prince in a desperate situation offers to give the victim a large sum of money for a small fee upfront. Unsurprisingly, when the fee is paid, no large sum of money ever arrives.

- Account Deactivation: Playing on a user's panic by creation of an urgency, the victim, who believes an important account is going to be deactivated, are tricked handing over important information such as login credentials to attackers. For instance, this could be triggered by a mail from the Bank

- Website Forgery: This attack is commonly paired with other scams such as the account deactivation scam. In this attack, the attacker creates a website that is virtually identical to the legitimate website of a business the victim uses, such as a bank.

[2] https://www.trendmicro.com/en_in/ciso/23/e/worldwide-email-phishing-stats-examples-2023.html#:~:text=Phishing%20attacks%20aimed%20at%20stealing,detections%20leaped%20a%20significant%2029%25.

- Spear Phishing: The most effective type of attack, accounting for more than 90% of the cases. For example, a mail targetted to a middle class family to get into a lucky jackpot for winning a car by entering an amount as a fee to be in the draw.

- Whaling: These type of attacks are typically targeted towards senior executives of multi-million or billion dollar firms with content likely to require the attention of the victim such as legal subpoenas or other executive issues. In 2016, Snapchat Inc fell into this attack, which did not lead to a monetory loss, but led to leakage and selling of data in the dark web.

The project aims to use Machine Learning algorithms along with Natural Language Processing for the development of a model which works towards determining and detecting Phishing Attempts.

CHAPTER 3: ENRON EMAIL DATASET

Enron Corporation was a Houston based energy, services and commodities firm which disbanded in December 2001, following the Enron Accounting Scandel, one of the largest corporate frauds in history to date.

Enron utilised various accounting loopholes and special-purpose entities to hide billions in debt and inflate profits, misleading investors and regulators. The company's executives, including CEO Jeffrey Skilling and CFO Andrew Fastow, engaged in off-the-books financing, manipulated financial statements, and used complex accounting tricks to make Enron appear financially robust while it was actually insolvent. Arthur Andersen, Enron's auditing firm, furthered the deception by approving these fraudulent financial practices.

The Enron Dataset is a database with 600,000 emails generated by 158 employees of the Enron Corporation, generated by the the Enron email servers by the Federal Energy Regulatory Commission (FERC) during their investigation into the scandal.

The dataset was downloaded from Kaggle[3] which is in it's unlabelled dataset which has only two columns i.e. "file" and "message".

In the 600,000 emails available in the dataset, a record in the form of a message looks as below:

```
Message-ID: <15464986.1075855378456.JavaMail.evans@thyme>
Date: Fri, 4 May 2001 13:51:00 -0700 (PDT)
From: phillip.allen@enron.com
To: john.lavorato@enron.com
Subject: Re:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Phillip K Allen
X-To: John J Lavorato <John J Lavorato/ENRON@enronXgate@ENRON>
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent Mail
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst

Traveling to have a business meeting takes the fun out of the
trip.  Especially if you have to prepare a presentation.  I would
suggest holding the business plan meetings here then take a trip
without any formal business meetings.  I would even try and get
some honest opinions on whether a trip is even desired or
necessary.

As far as the business meetings, I think it would be more
productive to try and stimulate discussions across the different
groups about what is working and what is not.  Too often the
presenter speaks and the others are quiet just waiting for their
turn.   The meetings might be better if held in a round table
discussion format.

My suggestion for where to go is Austin.  Play golf and rent a
ski boat and jet ski's.  Flying somewhere takes too much time.
```

The code below demonstrates how we use the python email library which extracts the message with the **`message_from_string()`** method to divide the meta-data into a pytree of tuples.

```
[ ] message = df.loc[1]['message']
    e = email.message_from_string(message)

    e.items()
```

```
[('Message-ID', '<15464986.1075855378456.JavaMail.evans@thyme>'),
 ('Date', 'Fri, 4 May 2001 13:51:00 -0700 (PDT)'),
 ('From', 'phillip.allen@enron.com'),
 ('To', 'john.lavorato@enron.com'),
 ('Subject', 'Re:'),
 ('Mime-Version', '1.0'),
 ('Content-Type', 'text/plain; charset=us-ascii'),
 ('Content-Transfer-Encoding', '7bit'),
 ('X-From', 'Phillip K Allen'),
 ('X-To', 'John J Lavorato <John J Lavorato/ENRON@enronXgate@ENRON>'),
 ('X-cc', ''),
 ('X-bcc', ''),
 ('X-Folder', "\\Phillip_Allen_Jan2002_1\\Allen, Phillip K.\\'Sent Mail"),
 ('X-Origin', 'Allen-P'),
 ('X-FileName', 'pallen (Non-Privileged).pst')]
```

[3] https://www.kaggle.com/datasets/wcukierski/enron-email-dataset

Similarly, the method get_payload() method to get the final string of the message. Along with this, we use a **`get_field()`** function as well as the **`body()`** function for the purpose of extracting a specified email field from a list of email messages and returns a list containing the values of that field and extraction of the main body of each email in a list of email messages, finally adding the extracted content as a new column.

```
def get_field(field: str, messages) -> list:
    column = []
    for message in messages:
        e = email.message_from_string(message)
        column.append(e.get(field))
    return column
```

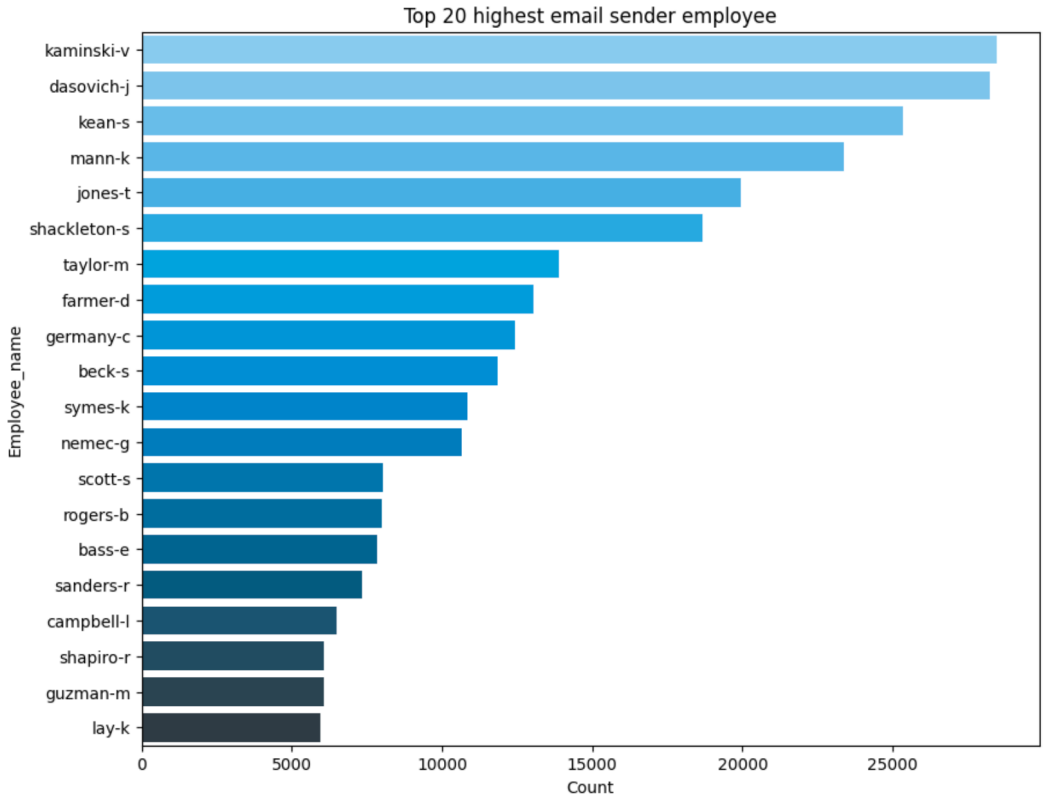
```
df['date'] = get_field("Date", df['message'])
df['subject'] = get_field("Subject", df['message'])
df['X-Folder'] = get_field("X-Folder", df['message'])
df['X-From'] = get_field("X-From", df['message'])
df['X-To'] = get_field("X-To", df['message'])
df.head(3)
```

	file	message	date	subject	X-Folder	X-From	X-To
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...	Mon, 14 May 2001 16:39:00 -0700 (PDT)		\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	Tim Belden <Tim Belden/Enron@EnronXGate>
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...	Fri, 4 May 2001 13:51:00 -0700 (PDT)	Re:	\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	John J Lavorato <John J Lavorato/ENRON@enronXg...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...	Wed, 18 Oct 2000 03:00:00 -0700 (PDT)	Re: test	\Phillip_Allen_Dec2000\Notes Folders\'sent mail	Phillip K Allen	Leah Van Arsdall

```
def body(messages):
    column = []
    for message in messages:
        e = email.message_from_string(message)
        column.append(e.get_payload())
    return column

df['body'] = body(df['message'])
df.head(3)
```

	file	message	date	subject	X-Folder	X-From	X-To	body
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...	Mon, 14 May 2001 16:39:00 -0700 (PDT)		\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	Tim Belden <Tim Belden/Enron@EnronXGate>	Here is our forecast\n\n
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...	Fri, 4 May 2001 13:51:00 -0700 (PDT)	Re:	\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Se...	Phillip K Allen	John J Lavorato <John J Lavorato/ENRON@enronXg...	Traveling to have a business meeting takes the...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...	Wed, 18 Oct 2000 03:00:00 -0700 (PDT)	Re: test	\Phillip_Allen_Dec2000\Notes Folders\'sent mail	Phillip K Allen	Leah Van Arsdall	test successful. way to go!!!

We have the following chart which shows the top email senders.



Eventually, we drop the unnecessary columns such as:

- file
- message
- date
- X-From
- X-To
- employee

To deal with missing values, since we have less than 1% of records missing some sort of a value, we decided to drop all of them. The final dataset only contains the content only. The below image demonstrates the same.

	content
1	Traveling to have a business meeting takes the...
2	test successful. way to go!!!
4	Let's shoot for Tuesday at 11:45.
5	Greg, How about either next Tuesday or Thurs...
7	any morning between 10 and 11:30
...	...
517396	This is a trade with OIL-SPEC-HEDGE-NG (John L...
517397	Some of my position is with the Alberta Term b...
517398	2 -----Original Message----- From: Doucet, ...
517399	Analyst Rank Stephane Brodeur 1 Chad Cl...
517400	i think the YMCA has a class that is for peopl...

489236 rows × 1 columns

For Natural Languange Processing, we use tokenise as well as stemming and lemmitisation to achieve the final dataset. Eventually we simply assign them the tag ‘spam’ for a suspicious file and ‘ham’ for an email which seems legit.

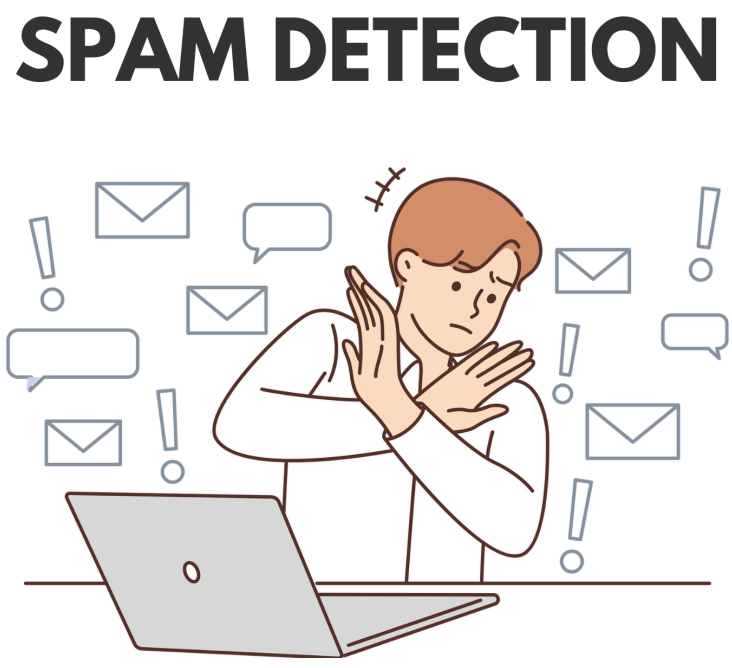
CHAPTER 4: MACHINE LEARNING AND DEEP LEARNING FOR CLASSIFICATION OF EMAIL CONTENT AND RESULTS

The project utilises two simple machine learning algorithms, along with a deep learning sequential model, called Long Short Term Memory (LSTMs).

In Natural Language Processing, the text (also known as corpus) is needed to be converted into a mathematical representation in order for the models to understand. To do this, we initially convert each word into single entities called tokens. To give tokens a mathematical meaning, we assign them vectors.

In Machine Learning, vectors are seen as an array of scalars and due to large amount of components, vectors are often seen as simple Euclidean point in a space where each component corresponds to an orthogonal axis (angle between which is 90°)

Vectors are used to represent text numerically and acts a huge role in the foundation to many machine learning techniques for Natural Language Processing. To understand the importance of vectors, we can observe the following scenario: Lets say in the era of 1990s, the concept of machine learning does not exist and we have to create a software to detect spam.

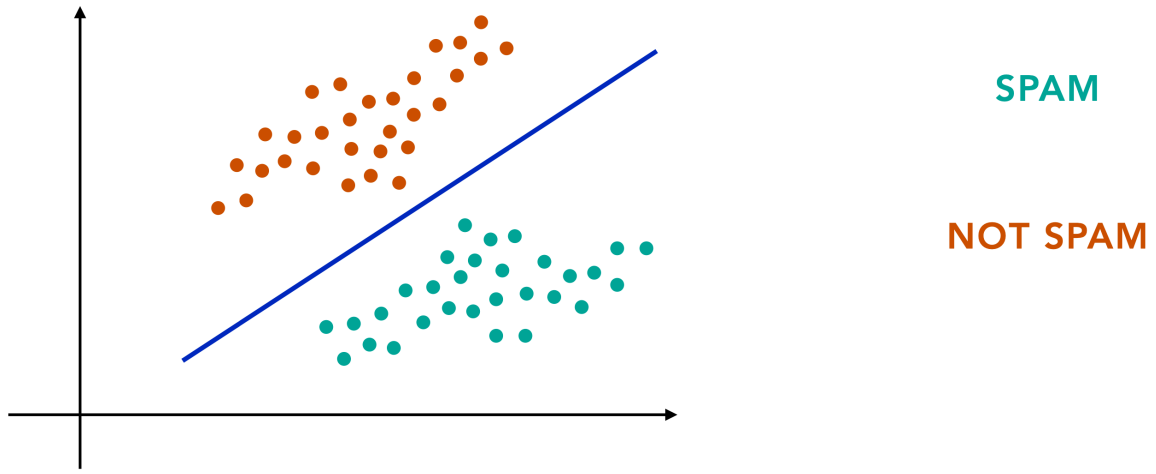


Without vectors, we can have two solutions:

- Solution 1: We target specific words like the ‘Nigerian Prince’ or ‘Credit’ and then raise a flag when these words come. However this solution has a problem, if a student gets a mail regarding his examination and credits to pass, the system would flag the mail as spam.

- Solution 2: We can declare certain emails from organisations such as Netflix or Facebook are safe while others are spam. Although this may sound logical, but hackers would know these emails are not considered as spam and thus they could create an exact replica of these websites and would pretend to be Netflix or Facebook, resulting in a large amount of Phishing.

Since neither solution sounds optimal, lets have a look at a situation where we utilise vectors: Lets say in a situation we can map an email to a spam or not spam and put all spams in a single cloud while not spam in other clouds. This way we can seperate both spam and not spam (also called ham) as below



To vectorise our tokens, there are multiple approaches to achieve this. These are as follows:

A) Bag of Words (BoW)

- Although text are sequential, many approaches does not consider word order in their solutions.
- Most of the vector models and classic machine learning approaches use the bag of words approach.

- The problem with Bag of Words is that often in real world, order matters. For example in the below example, we can see how BoW can cause confusion regarding the sentence

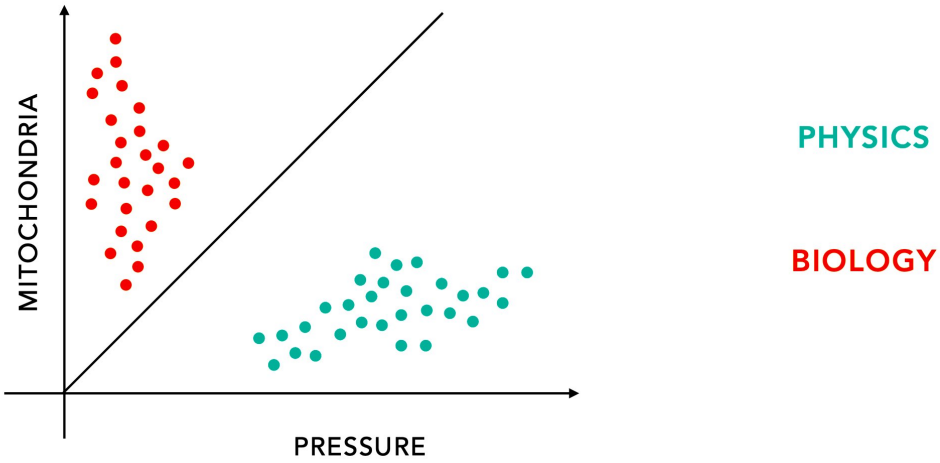
B) Count Vectorisation

- We would simply use the concept of counting to see which tokens appear in which corpus. It can also be called as an instance of Bag Of Words approach

- Lets say, a document has 3 sentences below, then we have their subsequent vector conversion on the principle of counting

ORIGINAL DOCUMENT		AND	LOVE	I	DOGS	DRUGS
I LOVE DOGS	→	0	1	1	1	0
I LOVE DRUGS	→	0	1	1	0	1
I LOVE DOGS AND DRUGS	→	1	1	1	1	1

- This approach can be beneficial when say, classifying documents in Physics and Biology. For this, we have counted mainly two keywords: Mitochondria -> Biology and Pressure -> Physics



- The problem with this approach is that, in a typical document, there can be many words and this still does not solve the issue of knowing which word corresponds to which position in the vector.

- Moreover in case of a specific document, for instance documents related to brain, the word 'brain' could act as a stopword as we know that the documents are related to brain, therefore we can drop those as well. To solve these issues, we use TF-IDF

C) Term Frequency – Inverse Document Frequency

- Since stop words are document specific and since their large occurences could overpower over rare words (which hold the most value in terms of variation and generalisation).

- Since stopwords have too much influence on the vector space and the same time are useless, it is important to drop them.

- The main idea behind TF-IDF is that we want to scale down the word counts of the words we want to ignore due to their appearance in multiple documents as they do not contribute in differentiating between documents

- We can define TF as count of how many times the word appears as well as IDF as the count of how many documents in which these words appear in

$$\text{TF\_IDF}(T, D) = \text{TF}(T, D) \times \text{IDF}(T)$$

- In the above formula, TF can have two arguments as the counts of the term can be different for each document and for IDF, we only have a single argument as for each document IDF of a particular term remains same
    - Term Frequency: Output using Count Vectoriser
    - IDF: Take Logarithm of inverse fraction and use with TF to find the TF_IDF. Note, we take the logarithmic due to it's montonic properties and thus would cause the value of TF_IDF to go down as T appears in more documents D

VARIATIONS TO TF

BINARY

1 : IF WORD APPEARS
0 : IF WORD NOT APPEARS

SIMILAR TO COUNT VECTORISATION

NORMALISE

DEFAULT APPROACH

$$TF = \frac{COUNT(T, D)}{\Sigma COUNT(T', D)}$$

THE LOG

REDUCE EFFECTS OF EXTREME VALUES

$$TF = LOG(1 + COUNT(T, D))$$

VARIATIONS TO IDF

SMOOTH

REMOVES THE POSSIBILITY OF GETTING A 0 AND THUS DIVIDING BY 0

$$IDF = LOG\left(\frac{N}{N(T) + 1}\right) + 1$$

PROBABILISTIC

GIVES LOG - ODDS OR MORE COMMONLY KNOWN AS LOGIC

$$IDF = LOG\left(\frac{N - N(T)}{N(T)}\right)$$

IDF MAX

USE MAX TERM COUNT FROM SAME DOCUMENT, MAKING RATIO RELEVANT TO DOCUMENT RATHER THAN DATASET

$$IDF = LOG\left(\frac{MAX\ N(T')}{N(T)}\right)$$

We utilise TF-IDF to vectorise our corpus and then use the following variety of Machine Learning models towards it.

A) Logistic Regression:

- Although the name indicates regression, we use LR as a classification algorithm. We can give the input tokens to Logisitic Regression to the model which then uses the

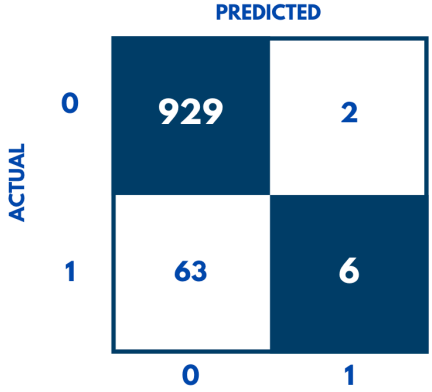
$$LN\left(\frac{P}{1-P}\right) = B_0 + B_1X_1$$

HERE

LN : NATURAL LOGARITHM
P : PROBABILITY OF ACCEPTED OFFER
B : COEFFICIENTS
X : FEATURES / INDEPENDENT VARIABLES

- From the above formula, we can see that the result is continuous in nature, due to which we call it regression. However, we round off the output value to get the prediction (0 for Legitimate and 1 for Phishing Mail)

LOGISTIC REGRESSION
CONFUSION MATRIX



CLASSIFICATION REPORT:

	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.94	1.00	0.97	931
1	0.75	0.09	0.16	69
ACCURACY			0.94	1000
MACRO AVG	0.84	0.54	0.56	1000
WEIGHTED AVG	0.92	0.94	0.91	100

B) Multi-Nomial Naive Bayes

- Naive Bayes uses the concept of Bayes Theorem to find the most likely output type and rounds it off to the nearest prediction

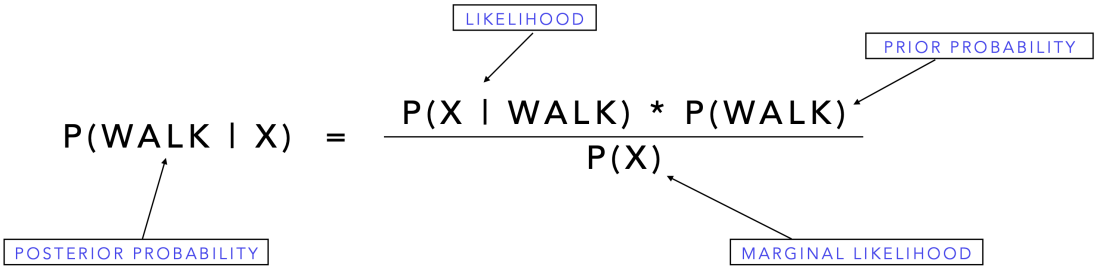
APPROACH

STEP 1 : WE FIND THE PRIOR PROBABILITY

STEP 2 : WE FIND THE MARGINAL LIKELIHOOD

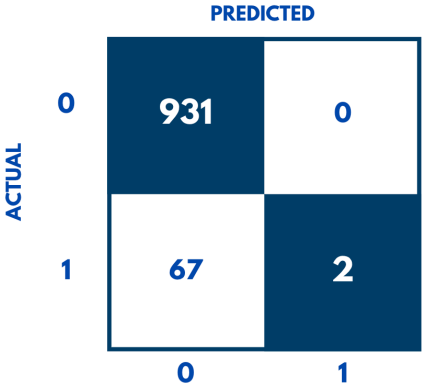
STEP 3 : WE FIND THE LIKELIHOOD

STEP 4 : WE USE THE FINDINGS OF STEPS 1,2 AND 3 TO FIND POSTERIOR PROBABILITY



- The results of Multi-nomial Naive Bayes is as follows

MULTI NOMIAL NAIVE BAYES
CONFUSION MATRIX

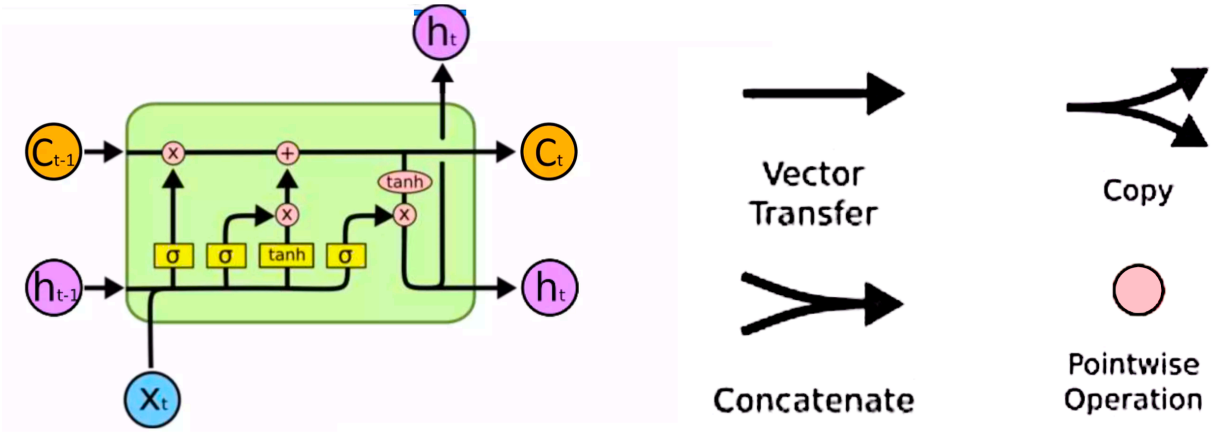


CLASSIFICATION REPORT:

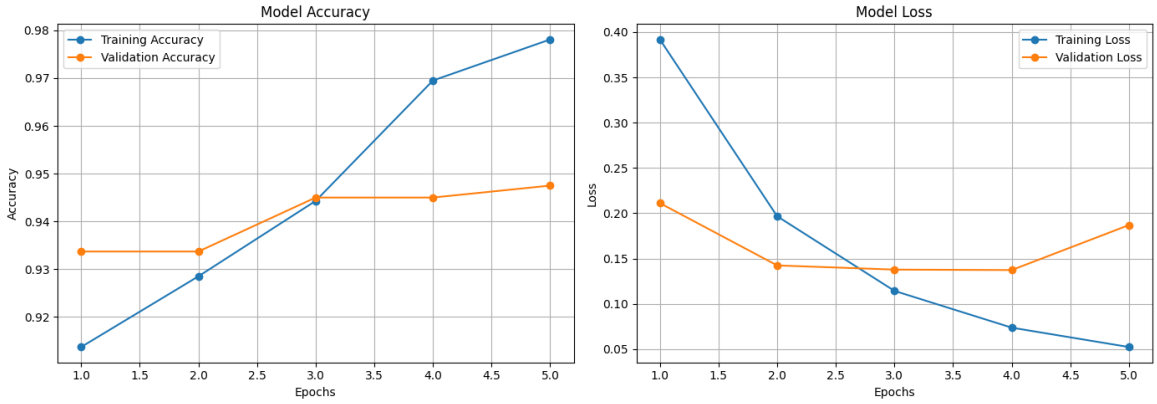
	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.93	1.00	0.97	931
1	1.00	0.03	0.06	69
ACCURACY			0.93	1000
MACRO AVG	0.97	0.51	0.51	1000
WEIGHTED AVG	0.94	0.93	0.90	100

C) Long Short Term Memory

- Long Short Term Memory was developed on the flaws of Vanishing and Exploding Gradients experienced by Recurrent Neural Networks

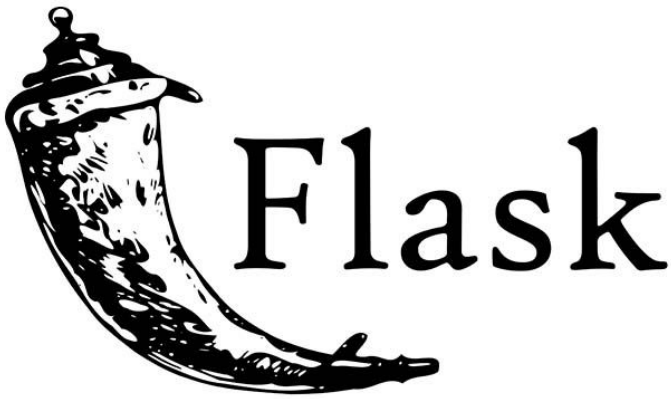


- LSTMs use 3 valves:
  - Forget Valve -> If Open, the same memory flows through and if closed, same memory does not flow through
  - Memory Valve -> If open, the new parts are added to the memory and if closed, no new parts are added to the memory
  - Output Valve -> If open, the new parts to the memory and if closed, no new new parts are added to the memory
- LSTM results are as follows



CHAPTER 5: WEBSITE DISPLAY

For the display of the results, we are developing a simple HTML-CSS website to deploy our models utilising the Python backend library Flask which is a lightweight, flexible web framework for Python, known for its simplicity and ease of use.



We are using Flask to building this small to medium-sized web applications and APIs, and since the website does not need extensive features, therefore to promote speed and simplicity, we used Flask over Django

Some of the key features of Flask include the following:

- Minimalist Core: Flask is designed with a "micro" philosophy, meaning it provides only the essentials for web development (like routing and request handling) and leaves everything else up to the developer to add via extensions.

- Routing: Flask allows developers to define URL routes easily. Each route corresponds to a specific function in your application, making it straightforward to control how users interact with your web app.

- Template Engine: Flask uses Jinja2, a powerful templating engine that allows developers to embed Python-like code within HTML. This is particularly useful for rendering dynamic content in web pages.

- Built-in Development Server: Flask comes with a simple development server, making it easy to test applications locally before deploying them to a production environment.

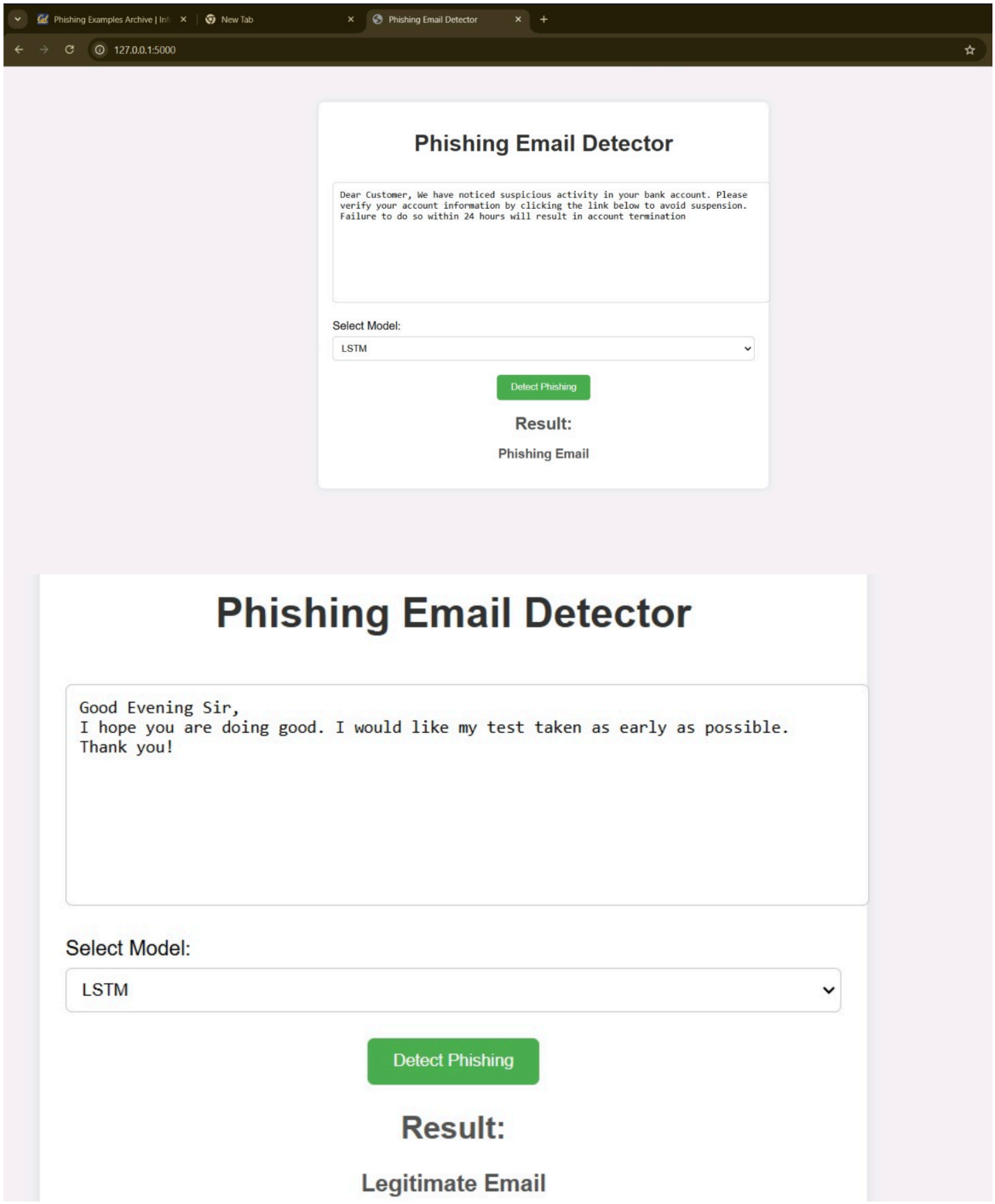
- Extensions: Flask has a rich ecosystem of extensions that add functionality like database integration, form handling, and authentication. These can be added as needed, keeping the core application lightweight.

```python
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)

# Load the vectorizer and the classifier model
with open("tfidf_vectorizer.pkl", "rb") as vectorizer_file:
    vectorizer = pickle.load(vectorizer_file)
with open("naive_bayes_model.pkl", "rb") as model_file:  # Replace with actual model file name
    model = pickle.load(model_file)

Codeium: Refactor | Explain | Generate Docstring | ×
@app.route('/')
def index():
    return render_template('index.html')

Codeium: Refactor | Explain | Generate Docstring | ×
@app.route('/predict', methods=['POST'])
def predict():
    # Get the text input from the form
    input_text = request.form.get("input_text")

    # Check if input_text is None
    if not input_text:
        return "Error: No input text provided. Please enter some text."

    # Transform the text using the loaded vectorizer
    transformed_text = vectorizer.transform([input_text.lower()])

    # Use the classifier model to predict
    prediction = model.predict(transformed_text)[0]

    # Map the prediction to a more readable format
    result = "Phishing Email" if prediction == 1 else "Not a Phishing Email"

    # Return the result
    return f"Prediction: {result}"
if __name__ == "__main__":
    app.run(debug=True)
```

The above code represents app.py, the python file which contains Flask and the integration of the model to the website to select the model, give an input prompt and then select the generate button to give out results

The website is currently locally hosted and is in development towards future developments (see Future Development)

The website functionality is as follows

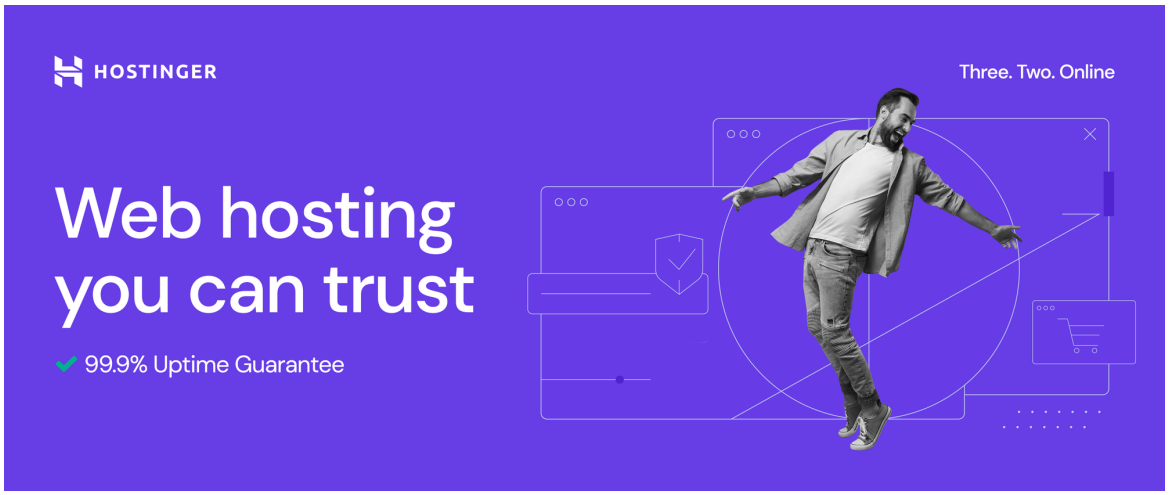


CHAPTER 6: FUTURE DEVELOPMENTS

The future developments we are aiming towards to improve the quality and understanding of this project is as follows:

A) Final Hosting

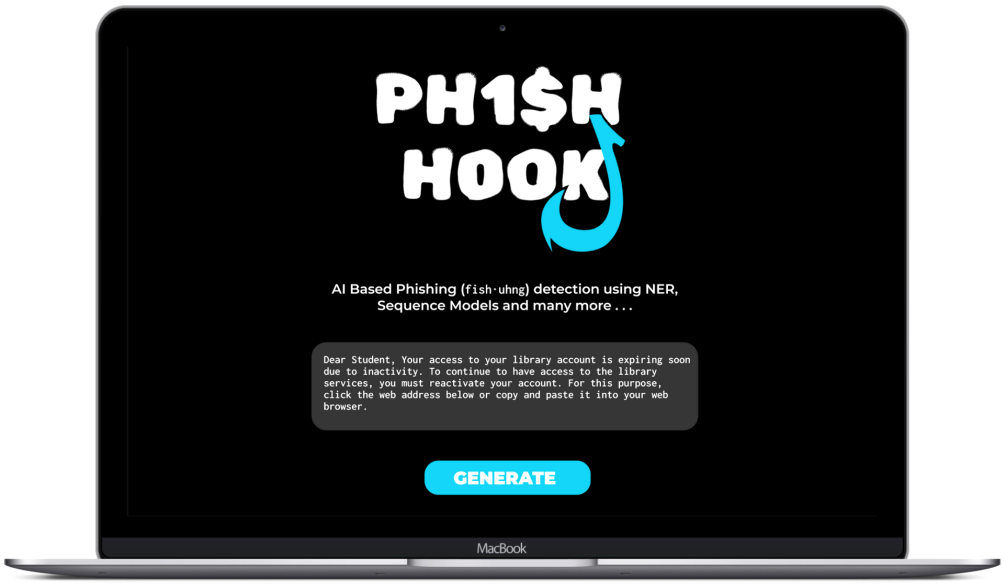
Currently the website is locally hosted on devices. For improving the accessibility of the project, the project is in works towards the global deployment through platforms such as Hostingers, Vercel and other platforms



B) FrontEnd Development

To improve the user experience, we are working towards shifting the techstack from simple HTML-CSS towards organised frameworks such as MERN stack, JAM Stack, MEAN Stack and other platforms to create the accessibility and usability of the project in the below prototype

TO



C) Improved Results

Currently we are simply predicting why an email content indicated an email to be legitimate or indicating towards a possible phishing attack. However, it is in the development to display reformed results, such as keywords or trigger words which resulted in the text to be classified as a certain value. The below prototype displays the aimed result

CHAPTER 7: REFERENCES

The project incorporates multiple references to act as a basis and inspiration to build our project. The references are as follows:

- https://www.imf.org/external/am/2009/phishing.htm
- https://www.bbc.com/news/stories-57520169
- https://www.trendmicro.com/en_in/ciso/23/e/worldwide-email-phishing-stats-examples-2023.html#:~:text=Phishing%20attacks%20aimed%20at%20stealing,detections%20leaped%20a%20significant%2029%25.
- https://repository.gatech.edu/server/api/core/bitstreams/df3d1c19-47be-4738-a855-9c049b709d52/content
- https://www.kaggle.com/datasets/wcukierski/enron-email-dataset
- F. -J. Yang, "An Implementation of Naive Bayes Classifier," 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2018, pp. 301-306, doi: 10.1109/CSCI46756.2018.00065. keywords: {Probabilistic logic;Bayes methods;Tools;Python;Machine learning;Data mining;Engines;Naive Bayes Classifier, Probabilistic Classification, Bayesian Theory},
- S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- https://www.analyticsvidhya.com/blog/2024/03/how-to-deploy-a-machine-learning-model-using-flask/