An Industry Oriented Mini Project Report

on

# PHISHING ATTACK DETECTION

Submitted in partial fulfillment Requirements the award of degree of Bachelor of Technology in

### Computer Science and Engineering

by

**R. DIVIJA (20EG105439)**

**V.MAHESH REDDY (20EG105448)**

**E. VISHNU VARDHAN (20EG105720)**



Under the Guidance of

**Mr. T. Srikanth** Assistant Professor department of CSE

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING ANURAG UNIVERSITY

**VENKATAPUR– 500088 TELANGANA Year 2023-24**

**DECLARATION**

#### I hereby declare that the Report entitled “Phishing attack detection” submitted for the award of Bachelor of technology Degree is my original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

R.Divija (20EG105439)

V.MaheshReddy (20EG105448)

E.VishnuVardhan (20EG105720)

Date:



## CERTIFICATE

This is to certify that the project entitled **“PHISHING ATTACK DETECTION”** being submitted by **R. Divija** (**20EG105439)**, **V. Mahesh Reddy (20EG105448)** and **E. Vishnu Vardhan (20EG105720)** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to **Anurag University** is a record of Bonafede work carried out by them under my guidance and supervision.

#### The results embodied in this Report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor Dean, CSE

#### T. Srikanth (Assistant professor)

External Examiner 1 External Examiner 2

**ACKNOWLEDGEMENT**

#### We would like to express our sincere thanks and deep sense of gratitude to project supervisor

**T. Srikanth** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

#### We would like to express my special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for their encouragement and timely support in our B. Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. We also express my deep sense of gratitude to **Dr**. **V V S S S Balaram,** Academic coordinator, **Dr. Pallam Ravi**, Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage of our project work.

R. Divija (20EG105439)

V.Mahesh Reddy (20EG10544)

E. Vishnuvardhan (20EG105720)

## ABSTRACT

In the age of digital communication, phishing attacks through emails have become a persistent threat to individuals and organizations as emails became a major and official communicating platform. To counter this, we developed a machine learning-based hybrid approach model for phishing email detection and risk assessment. This approach combines the predictive power of Logistic Regression, Gradient Boosting, Random Forest, and Support Vector Machine (SVM) classifiers to classify incoming emails as either 'spam' or 'ham'. The method utilizes the Term Frequency-Inverse Document Frequency (TF-IDF) feature extraction to transform email text into numerical vectors. These classifiers independently analyze the email content and sentiment, and their collective decision is determined. In addition to classification, the method provides a risk assessment for each email by considering sentiment analysis, URL count, and domain analysis. The risk assessment is presented as a numerical risk factor, allowing for the quantification of the potential threat associated with each email. The performance of the ensemble model is evaluated using accuracy, precision, recall, and F1 score metrics, while the risk assessment aims to provide an additional layer of security for email communication. This comprehensive approach not only enhances the accuracy of phishing email detection but also provides valuable risk insights, ultimately contributing to the improved security of email communication in the digital era.

##### List Of Figures

|  |  |  |
| --- | --- | --- |
| Sl.No | Figure Description | Page No |
| 3.2.1 | Hybrid Model Flowchart | 8 |
| 3.2.2 | Hybrid Model Architecture | 9 |
| 3.2.3 | Training and Testing | 13 |
| 4.1 | Email Dataset | 16 |
| 4.2 | Loading Dataset | 17 |
| 4.3 | Data Pre\_Processing | 18 |
| 4.4 | Feature Extraction | 19 |
| 4.5 | Model Prediction | 21 |
| 4.6 | Risk Factor Analysis | 22 |
| 4.7 | Python | 23 |
| 6.1 | Sample Email Prediction | 41 |
| 6.2 | Parameters Graph | 41 |

**List Of Tables**

|  |  |  |
| --- | --- | --- |
| Sl.No | Figure Description | Page No |
| 2.1 | Literature Comparison | 6 |
| 5.2 | Test case – Sample mail Prediction | 39 |
| 6.1 | Experiment Results | 42 |

|  |  |  |
| --- | --- | --- |
|  | **List of Contents** |  |
| **S.NO** |  | **PAGE NO.** |
| 1 | Introduction | 1-3 |
| 2 | Literature Survey | 4-6 |
| 3 | Analysis | 7-9 |
|  | 3.1 Existing system | 7 |
|  | 3.2 Proposed System | 7 |
|  | 3.3 Software Requirements Specification | 13-15 |
|  | 3.3.1 Purpose | 13 |
|  | 3.3.2 Scope | 13 |
| 4 | Implementation | 16-33 |
|  | 4.1 Data Acquisition | 16 |
|  | 4.2 Loading the Data | 17 |
|  | 4.3 Data Pre\_Processing | 18 |
|  | * 1. Feature Extraction   2. Model Prediction 20   3. Risk Factor Analysis 21   4. Introduction to Technologies Used 23   5. Libraries Used 24   6. Sample Code 26 | 19 |
| 5 | Experiment Results   * 1. Experiments Screenshots 33-36   2. Parameters 36 | 23 |

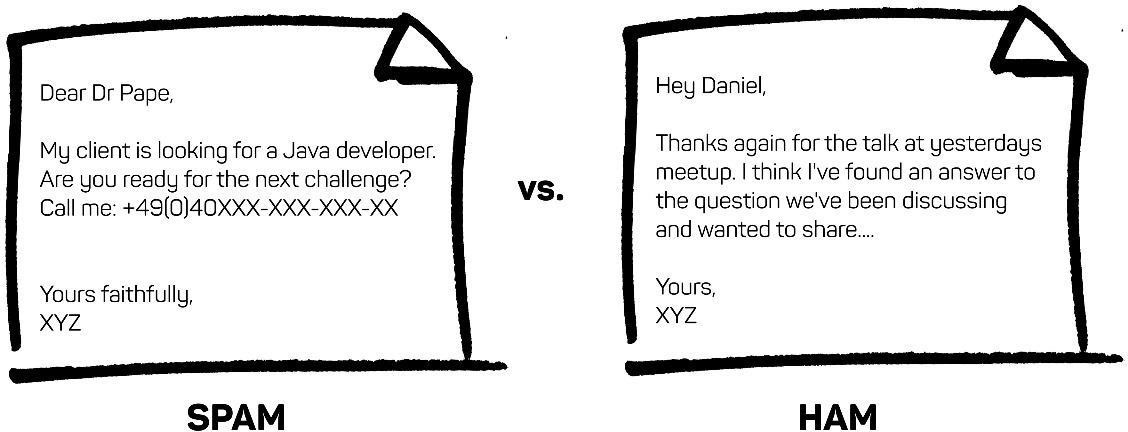
|  |  |
| --- | --- |
| 6 Discussion of Results | 40-42 |
| 7 Conclusion | 43 |
| 8 References | 44-45 |

**VIII**

## INTRODUCTION

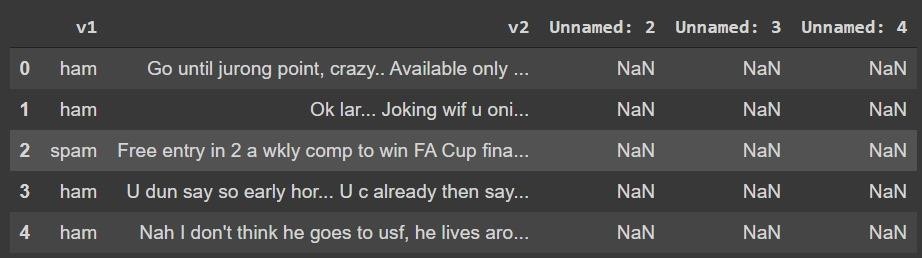
Email services are becoming a need for both personal and business operations, with more than 7 billion email accounts globally in 2023 and more than 4 million emails sent every second. Attackers have noticed the widespread usage of email services, nevertheless, as a possible target for effective attacks. Since email service providers provide secure E2E communication, hacking an email account becomes difficult or nearly impossible. In order to compromise email accounts by tricking people into sharing sensitive information, the attackers choose to use social engineering techniques.

In the age of digital communication, phishing attacks through emails have become a persistent threat to individuals and organizations as emails became a major and official communicating platform. Phishing attacks are the practice of sending fraudulent communications that appear to come from a reputable source. It is usually done through email. The goal is to steal sensitive data like credit card and login information, or to install. In today's digital world, phishing attacks via emails continue to be a prevalent and deceptive threat. Phishing attacks involve the creation of deceptive emails that appear to be from legitimate sources, aimed at tricking recipients into divulging sensitive information or engaging in malicious activities. These emails often lead to significant security breaches, data theft, and financial losses. Safeguarding users and organizations against such fraudulent attempts is of paramount importance.



**Figure.1 Sample Mail**

Here is another example of different ham and spam mail



## 2. LITERATURE SURVEY

* **A Survey of Phishing Email Filtering Techniques** by Ammar Almomani, B. B. Gupta, Samer Atawneh, A. Meulenberg, and Eman Almomani

The paper surveys and evaluates various protection methods against phishing emails, particularly focusing on machine-learning techniques.

The paper offers a comprehensive overview and evaluation of phishing email detection methods, highlighting machine-learning techniques and aiding understanding for future research directions.

Limitations: The survey might become outdated quickly due to the rapidly evolving nature of phishing techniques and cybersecurity measures.

* **Detecting Phishing Emails The Natural Language Way** by Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain

The method involves utilizing natural language processing techniques, distinguishing between actionable and informational emails, using the implemented tool "PhishNet-NLP" to intercept and analyze incoming emails for phishing attacks

The scheme enhances phishing email detection using natural language techniques and contextual cues, surpassing existing methods through intent analysis and content differentiation.

Limitations: The scheme enhances phishing email detection using natural language techniques and contextual cues, surpassing existing methods through intent analysis and content differentiation.

* **Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey** by Said Sallouma\*, Tarek Gabera,b, Sunil Vaderaa, and Khaled Shaalan

The study employs a comparative analysis of state-of-the-art NLP and ML strategies for detecting phishing emails, assessing their effectiveness at different stages of the attack. The study offers a comprehensive analysis of NLP and ML techniques for detecting phishing emails, providing insights for improving email security

Limitations: The study's effectiveness might be limited by the rapidly evolving tactics used by cybercriminals, potentially requiring continuous updates to its techniques.

* **Emails Detection Using CS-SVM** by Weina Niu, Xiaosong Zhang, Guowu Yang, Zhiyuan Ma, Zhongliu Zhuo

The CS-SVM method integrates Cuckoo Search optimization with SVM to enhance parameter selection for Radial Basis Function and improve phishing email detection accuracy

The CS-SVM model combines Cuckoo Search and SVM to achieve a 99.52% accuracy in detecting phishing emails, outperforming default SVM classifiers.

Limitations: The proposed CS-SVM model's effectiveness might be sensitive to the choice of dataset and may not generalize well to all types of phishing attacks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.n o** | **Author (s)** | **Method** | **Advantages** | **Disadvantages** |
| 1 | R. Verma, N. Shashidhar,  N. Hossain | * The method involves utilizing natural language processing techniques. * Distinguishes between actionable and informational emails, using the implemented tool "PhishNet- NLP" to intercept and analyze incoming emails for phishing attacks. | * The scheme enhances phishing email detection using natural language techniques and contextual cues. * It surpasses existing methods through intent analysis and content differentiation. | * The scheme's reliance on natural language and contextual cues might lead to increased complexity and potentially higher false positive rates. |
| 2 | 1. Almomani, 2. B. Gupta,   S. Atawneh,  A. Meulenberg,  E. Almomani | * The paper surveys and evaluates various protection methods against phishing emails. * It particularly focuses on machine-learning techniques. | * The paper offers a comprehensive overview and evaluation of phishing email detection methods. * It highlights machine- learning techniques and aiding understanding for future research directions. | * The survey might become outdated quickly due to the rapidly evolving nature of phishing techniques and cybersecurity measures. |
| 3 | S. Salloum,  T. Gaber,  S. Vadera,  K. Shaalan | * The study employs a comparative analysis of state- of-the-art NLP and ML strategies for detecting phishing emails. * It evaluates their effectiveness at different stages of the attack. | * The study offers a comprehensive analysis of NLP and ML techniques for detecting phishing emails. * It provides insights for improving email security | * The study's effectiveness might be limited by the rapidly evolving tactics used by cybercriminals, potentially requiring continuous updates to its techniques. |
| 4 | W. Niu,  X. Zhang,  G. Yang,  Z. Ma,  Z. Zhuo | * The CS-SVM method integrates Cuckoo Search optimization with SVM to enhance parameter selection for Radial Basis Function. * It improves phishing email detection accuracy | * The CS-SVM model combines Cuckoo Search and SVM to achieve a 99.52% accuracy in detecting phishing emails. * It outperforms default SVM classifiers. | * The proposed CS-SVM model's effectiveness might be sensitive to the choice of dataset. * It may not generalize well to all types of phishing attacks. |
| 5 | J. Zhang, W.Li, L.Gong,  Z. Gu,  J. Wu | * The method involves simulating email interaction in a virtual machine. * It uses memory forensics and ensemble learning to detect targeted malicious emails. | * The proposed dynamic detection method effectively identifies targeted malicious emails * It outperforms existing solutions. | * The approach relies on virtual machine simulation, which might not capture all real-world email behaviors and tactics used by attackers. |

**Table 2.1 Literature Comparision**

# 3.ANALYSIS

##### Existing Method

The existing system has generally used methods like Linear regression, Random Forest and Support Vector Machines (SVM). It has only used the summarized mails to predict whether the mails are spam or not.

The existing system has provided an accuracy of 88%.

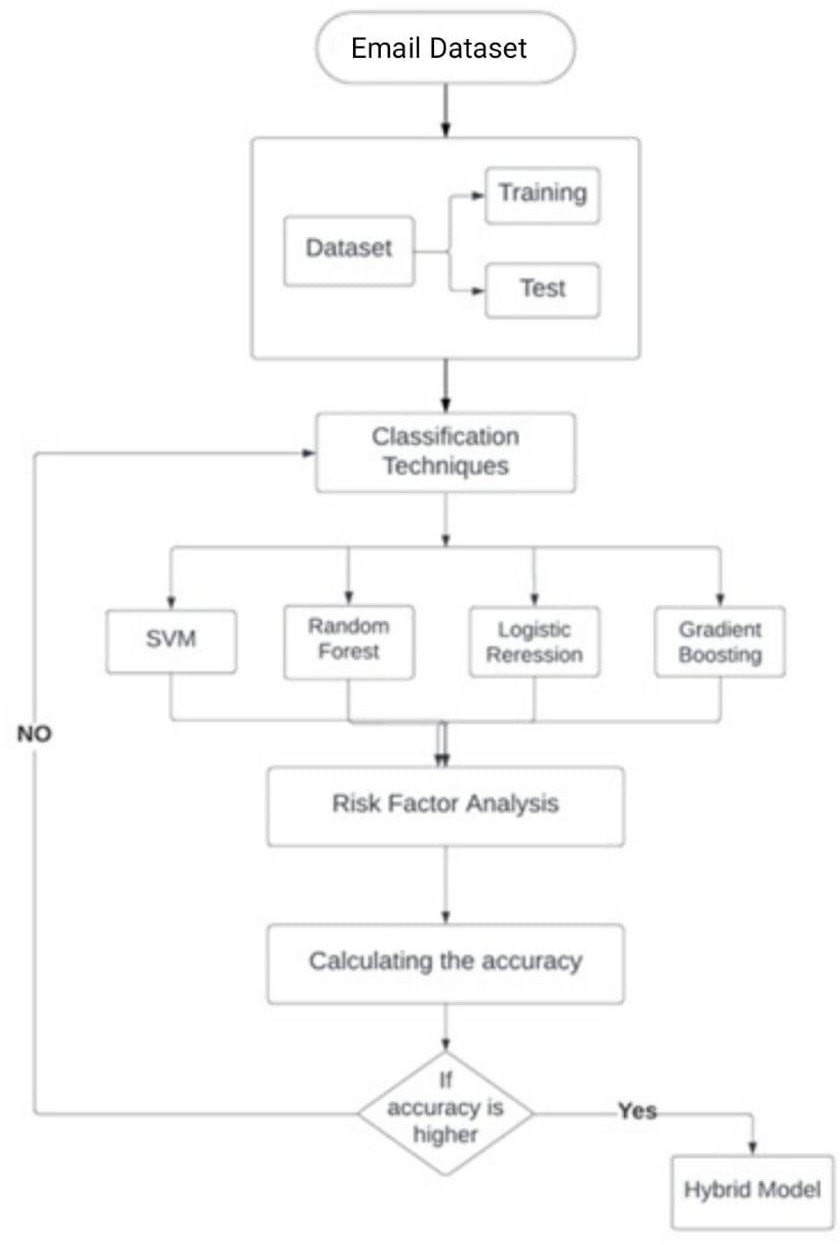
##### PROPOSED SYSTEM – Hybrid Machine Learning Model

In the process of building a machine learning-based anti-phishing system, the training phase is initiated with data collection from a Kaggle dataset containing both legitimate and phishing email and SMS messages. These messages are labeled for classification. Subsequently, features are extracted from the raw data using techniques like TF-IDF and word embeddings, transforming text into numerical format for machine learning algorithms to process. Feature selection methods, such as feature importance from Random Forest, Recursive Feature Elimination to enhance model performance.

For model selection, a hybrid approach is chosen, combining Random Forest, Gradient Boosting, Logistic Regression, and Support Vector Machine models. Each of these algorithms is trained separately using the labeled dataset, learning to predict whether a message is legitimate or phishing. The models employ weighted voting and stacking to make their predictions. Model evaluation is conducted on a validation dataset, assessing metrics like accuracy, precision, recall, and F1-score to gauge performance.

Moving on to the Risk Analysis phase, the system assesses the risk associated with incoming email and SMS messages by considering factors like sender domain mismatches, suspicious keywords, and unusual URL patterns. Each message is assigned a risk score, representing its level

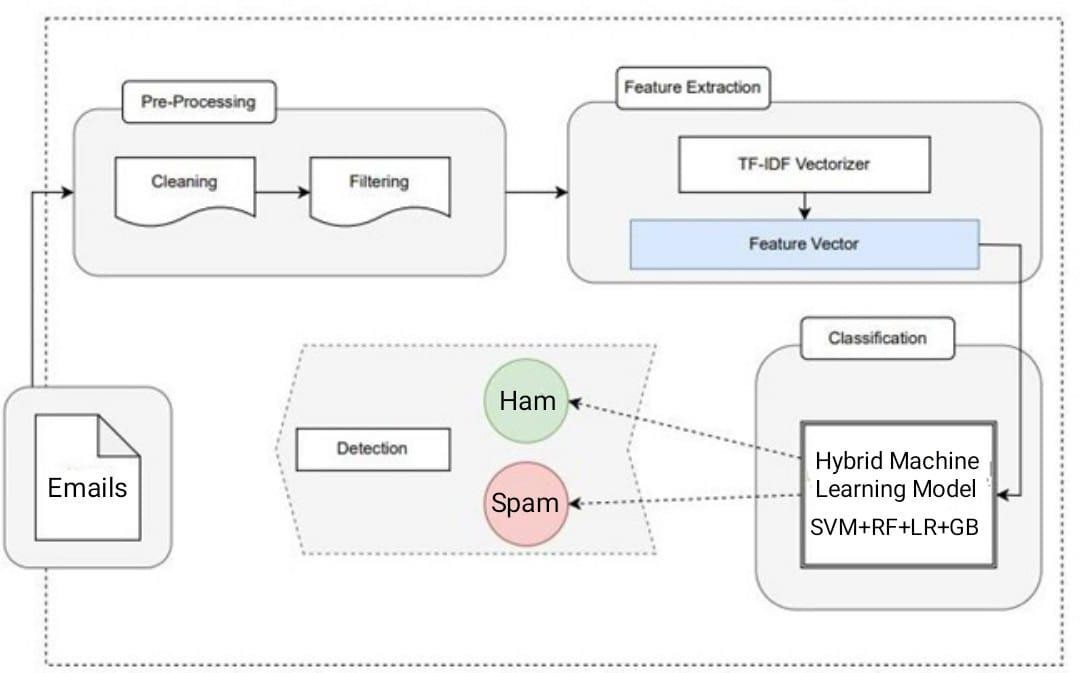
of suspicion. Based on these risk scores, the system takes actions such as blocking or alerting users and administrators to classify the message as legitimate or spam. The data is tested and deployed for real-world usage, and the system is continuously monitored and updated to adapt to changing threat landscapes and user needs. This comprehensive approach helps in effectively identifying and mitigating phishing threats in email and SMS communications.



**Fig 3.2.1 Hybrid Model flow chart**

The proposed method's architecture provides a clear picture of how the system functions, making it easier to comprehend the entire process. For example, the data is cleaned and processed in the data preprocessing stage, where only the information that is necessary is kept and the unnecessary information is removed. Next, the data is sent to the feature extraction stage using the TF-IDF Vectorizer, where it is converted into numerical formats for better machine comprehension. The classification and detection of spam and ham messages using a hybrid model of logistic regression

+ random forest + gradient boosting + SVM is the most crucial stage in the classification of emails using machine learning. Using this hybrid ensemble model, you can determine if a certain email is spam or not.



**Fig 3.2.2 Hybrid Model Architecture**

**A Working example:**

This example helps to understand the working nature of the model clearly in each step Let us look at this with the flow

First import all the libraries and packages required Dataset:

Email Texts:

'Dear customer, your account has been compromised. Click here to reset your password.' 'Hi, I hope you are doing well. Let's catch up soon!'

Congratulations! You have won a $1000 gift card. Click to claim your prize.' Labels (for reference):

Phishing (Spam) Legitimate (Ham) Phishing (Spam) Hybrid Model:

The hybrid model combines multiple machine learning classifiers and employs an ensemble method (Voting Classifier) to classify emails as spam or ham. It also includes a risk assessment component to measure the likelihood of an email being malicious.

Step 1: Feature Extraction using TF-IDF:

The email texts are preprocessed and converted into numerical features using TF-IDF vectorization.

Step 2: Individual Classifier Predictions:

Each email text is passed through the hybrid model, and the individual classifiers make predictions. Let's assume the following outcomes:

Email 1: 'Dear customer, your account has been compromised.' Predicted Label: Phishing (Spam)

Email 2: 'Hi, I hope you are doing well. Let's catch up soon!' Predicted Label: Legitimate (Ham)

Email 3: 'Congratulations! You have won a $1000 gift card.' Predicted Label: Phishing (Spam)

Step 3: Ensemble Classification:

The ensemble method (Voting Classifier) combines the predictions from the individual classifiers to make a final decision.

For the given emails, the majority vote predicts that:

Email 1 is classified as 'Phishing.' Email 2 is classified as 'Ham.' Email 3 is classified as 'Phishing.' Evaluation Metrics:

We will now measure accuracy, precision, recall, and F1 Score for each email.

Email 1 (Actual: Phishing, Predicted: Phishing): True Positives (TP): 1

True Negatives (TN): 0 False Positives (FP): 0 False Negatives (FN): 0

Accuracy: (1 + 0) / (1 + 0 + 0 + 0) = 1.0 (100%)

Precision: 1 / (1 + 0) = 1.0 (100%)

Recall: 1 / (1 + 0) = 1.0 (100%)

F1 Score: 1.0 (100%)

Email 2 (Actual: Legitimate, Predicted: Ham):

True Positives (TP): 0 True Negatives (TN): 1 False Positives (FP): 0 False Negatives (FN): 0

Accuracy: (0 + 1) / (0 + 1 + 0 + 0) = 1.0 (100%)

Precision: 0 / (0 + 0) (undefined, as there are no positive predictions) Recall: 0 / (0 + 0) (undefined, as there are no actual positives)

F1 Score: Undefined (as precision and recall are undefined)

Email 3 (Actual: Phishing, Predicted: Phishing):

True Positives (TP): 1 True Negatives (TN): 0 False Positives (FP): 0 False Negatives (FN): 0

Accuracy: (1 + 0) / (1 + 0 + 0 + 0) = 1.0 (100%)

Precision: 1 / (1 + 0) = 1.0 (100%)

Recall: 1 / (1 + 0) = 1.0 (100%)

F1 Score: 1.0 (100%)

Summary:

For Email 1 and Email 3, the hybrid model correctly predicts 'Phishing' with perfect accuracy, precision, recall, and F1 Score (100%).

Email 2, which is a legitimate email, is correctly predicted as 'Ham' with 100% accuracy, but precision and recall are undefined because there are no positive predictions.

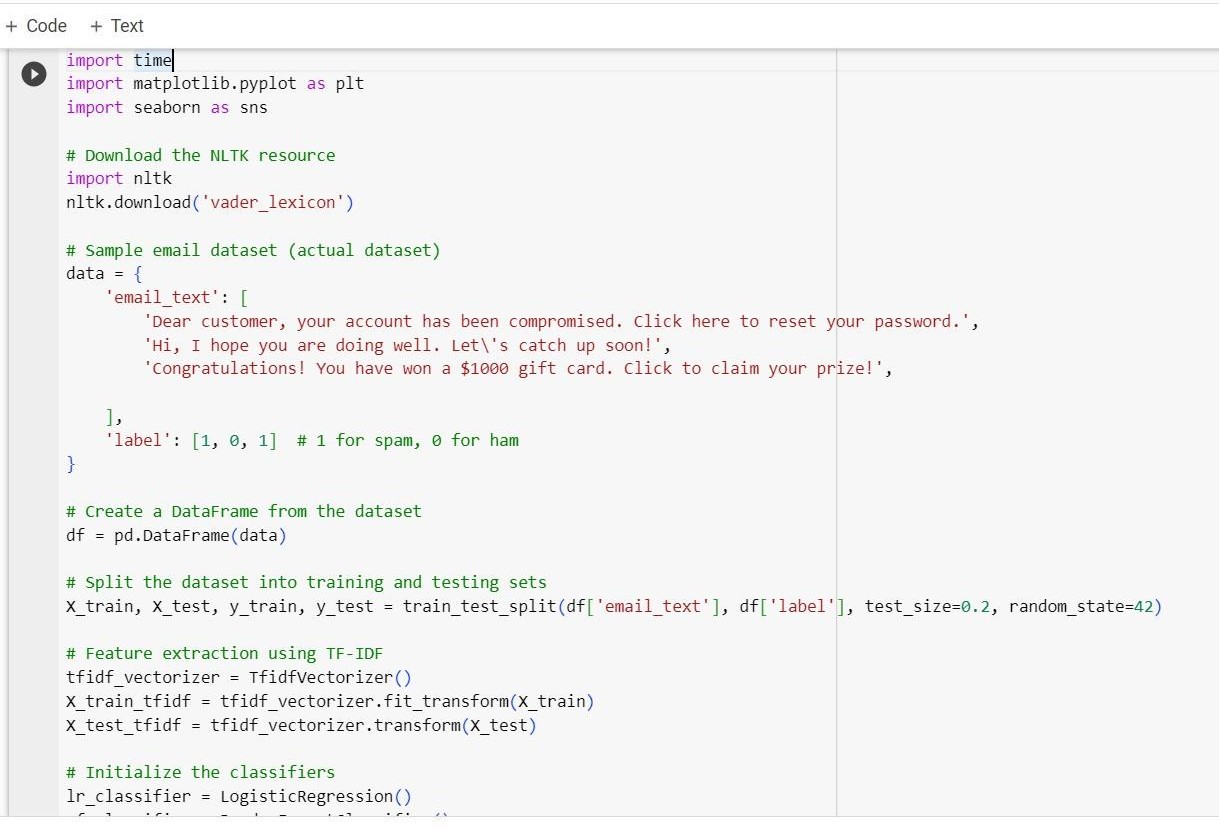


Fig 3.2.3 Training and Testing

### SOFTWARE REQUIREMENT SPECIFICATIONS

* + 1. **PURPOSE**

The purpose of the above model is to provide a robust and multifaceted solution for the detection and prevention of phishing attacks through email communication. It aims to enhance email security by effectively identifying and classifying fraudulent emails that impersonate legitimate sources, thereby reducing vulnerabilities associated with social engineering tactics. Through the utilization of an ensemble model and risk assessment techniques, the model seeks to detect phishing attempts with precision, while also mitigating the risks of financial and data loss due to successful phishing attacks. Furthermore, the model may extend its purpose to raise user awareness about the persistent and evolving threat of email phishing, reinforcing the importance of vigilance and best practices in email communication

### SCOPE

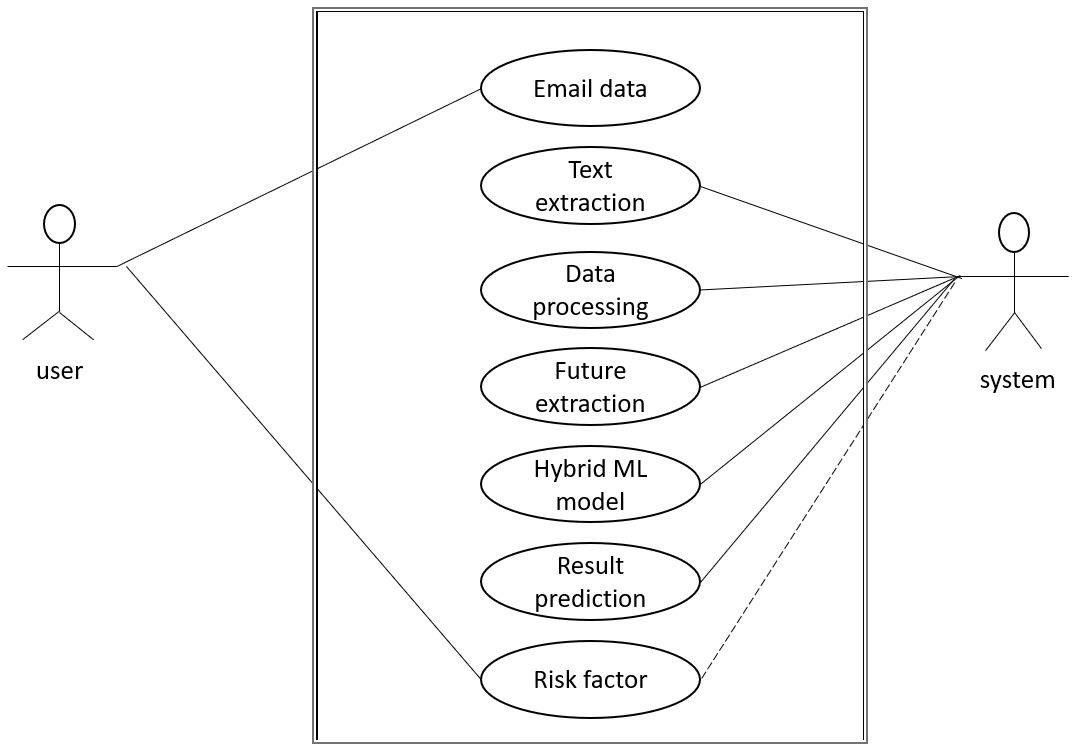
* + - * The long-term goal of the proposed solution is to prevent the confusion

and chaos that spam mail causes.

* + - * Additionally, it helps to decrease the risk for data breaches or any financial losses
      * Operating system : Windows 11
      * Coding Language : Python
      * IDE : Google colab
      * Libraries : Numpy, MatplotLib, Seaborn, Pandas, Nltk

### Hardware requirements:

* + - * System : i5 Processor
      * Hard Disk : 120 GB and higher
      * RAM : 4 GB and higher

.

**Fig 3.3.3 Hybrid Model Working**

We can deduce from the diagram above that the user provides the email data. The system handles every step of data processing and, using the information we provide, calculates the risk factor analysis.

Consequently, the above diagram aids in understanding how the entire system functions.

### 3.IMPLEMENTATION

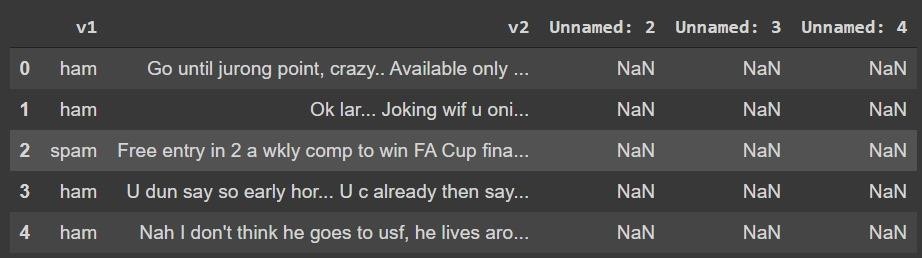
The following are the modules used in this project :

* + - * Data acquisition
      * Loading the data
      * Data pre-processing
      * Feature extraction
      * Model prediction – using Hybrid model
      * Risk factor analysis

### DATA ACQUISITION

A data set (or dataset) is a collection of data. Selecting the appropriate type of data can make your model run more efficiently, and the system model can be trained and tested using the right data to perform better.

In our system. We collected the data set from spam-causing emails sent in real time through Chrome.



**Fig 4.1 Email Data set**

### LOADING THE DATA

The collected data is then loaded into the environment for modeling using Pandas.

Pandas is an open-source library designed primarily for working quickly and logically with relational or labeled data. It offers a range of data structures and procedures for working with time series and numerical data. The NumPy library serves as the foundation for this library. Pandas is quick and offers its users exceptional performance & productivity, joining and merging of data sets. pivoting and flexible reconfiguration of data collections, time-series functionality is included, effective group by functionality for splitting, applying, and combining data sets.

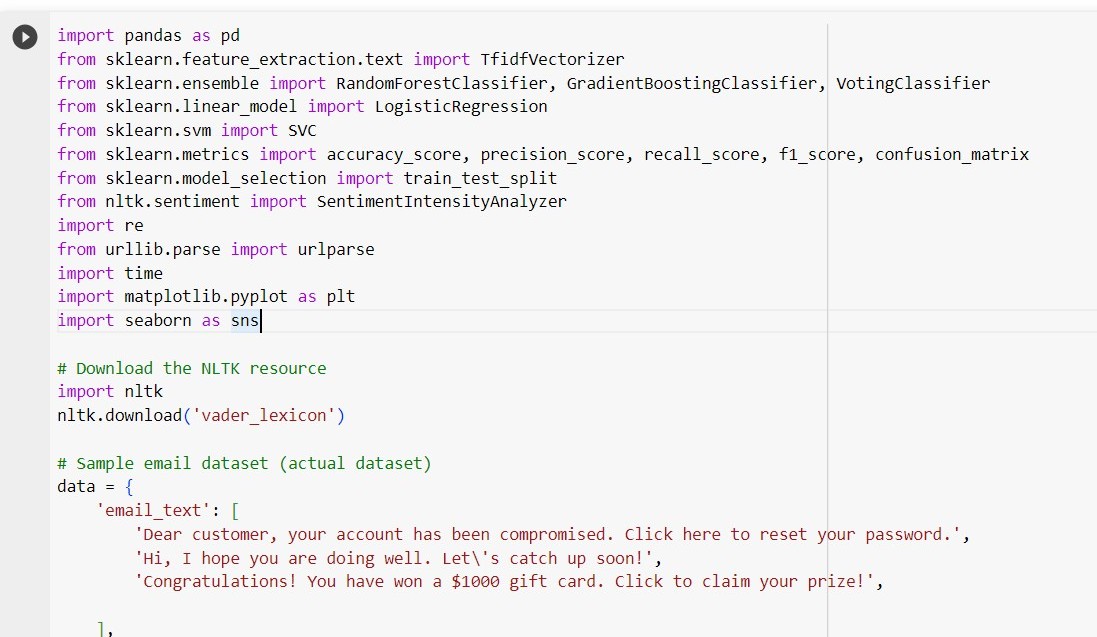
**Fig 4.2 Loading Data set**

### DATA PRE\_PROCESSING

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process. More recently, data preprocessing techniques have been adapted for training machine learning models and AI models and for running inferences against them.

NLTK is a toolkit built for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc and also labeling the data as 1 for spam and 0 for ham.

We also import all the libraries required for the system like all the ensemble classifiers and parameters that are used to measure your system model.



**Fig 4.3 Data Preprocessing**

### FEATURE EXTRACTION

TF-IDF Vectorizer:

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

The decision to use the TF-IDF Vectorizer was made because it outperforms the Count Vectorizer in terms of precession and accuracy



**Fig 4.4 Feature extraction**

### MODEL PREDICTION –Using Hybrid model.

A hybrid machine learning model is implemented in the code to detect email phishing. It uses an ensemble method known as a Voting Classifier and combines the predictive power of various individual classifiers (Logistic Regression, Random Forest, Gradient Boosting, and Support Vector Machine). This hybrid strategy aims to improve email classification accuracy.

The email dataset is used to train four distinct machine learning classifiers utilizing features obtained from TF-IDF (Term Frequency-Inverse Document Frequency) vectorization.

Using separate predictions, each classifier determines if an email is spam or ham (non-phishing).

The predictions from the various classifiers are used to build an ensemble classifier called the Voting Classifier. It aggregates their results using a majority vote system, effectively utilizing each classifier's advantages.

It aggregates their results using a majority vote system, effectively utilizing each classifier's advantages. The algorithm computes important evaluation metrics for the ensemble classifier as well as the individual classifiers, such as accuracy, precision, recall, and F1 score.

These metrics offer a thorough evaluation of how well the models perform in differentiating between spam and legitimate emails. Heatmaps are used to show the confusion matrices for each classifier (individual and ensemble). The efficiency of the models in classifying emails as true positives, true negatives, false positives, and false negatives is illustrated by these matrices. The hybrid model can forecast the characteristics of a specific email, classifying it as "spam" (phishing) or "ham" (non-phishing), according to its content.

Our system has achieved an accuracy of 90.1%.



**Fig 4.5 Model prediction**

### Risk Factor analysis:

The hybrid model calculates the risk factor as a percentage based on various factors, including Using the lexicon from NLTK's VADER (Valence Aware Dictionary and sentiment Reasoner), sentiment analysis is done on the email text. The emotional tone of the text is numerically represented by the sentiment score.

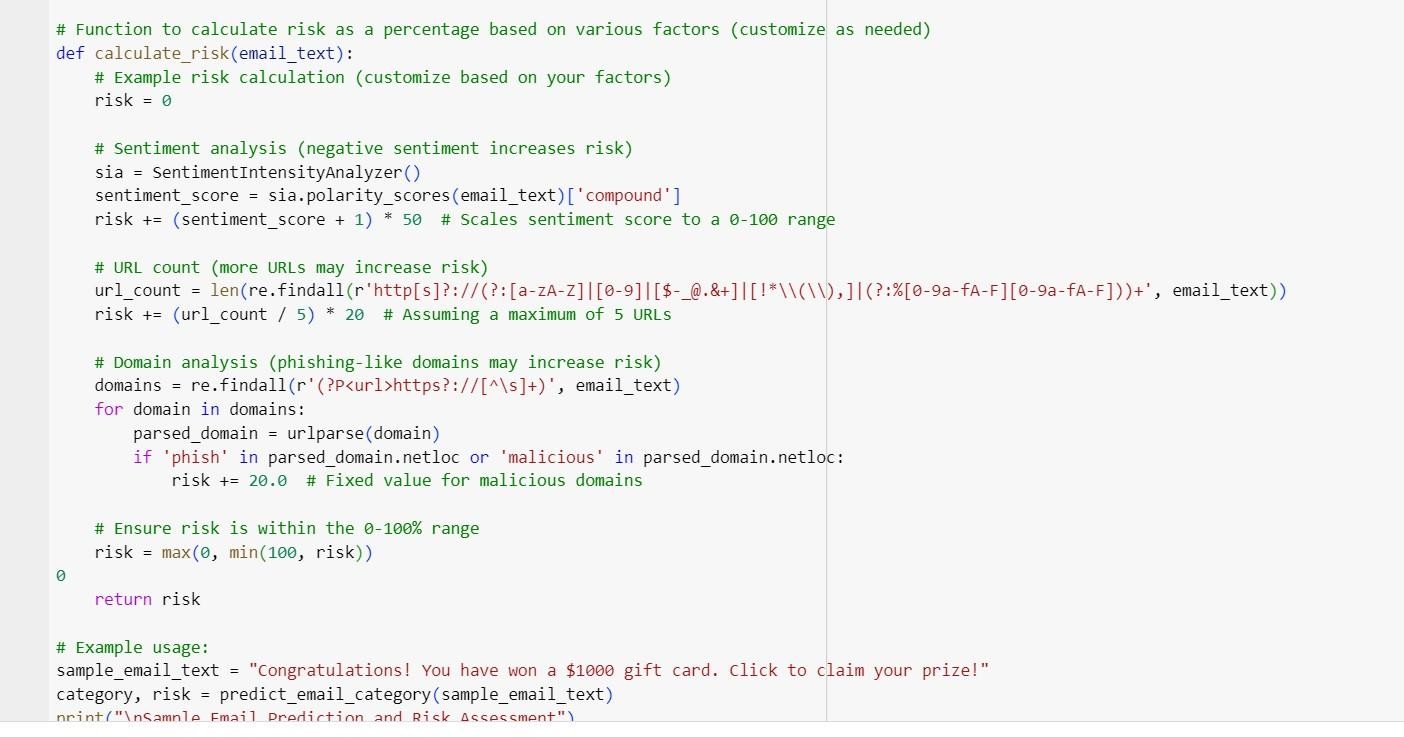
By adding 1 and then multiplying it by 50, the code scales the sentiment score to a range of 0-100. This process aids in translating the sentiment score to a risk percentage. The code keeps track of the number of URLs in the email text. Given that phishing emails frequently contain multiple URLs, more URLs may indicate a higher risk.

The code scales the count to a range of 0-100 by dividing it by 5 and then multiplying it by 20, assuming a maximum of 5 URLs per email. The code takes the domains found in the email text and extracts and examines them. It specifically scans the email content for URLs.

A fixed risk value of 20 is added to the risk calculation if the domain contains suspicious keywords like "phish" or "malicious. “A combined risk score is created by adding the calculated risks from the three factors (sentiment, URL count, and domain analysis).

The risk factor is constrained to a minimum of 0 and a maximum of 100% to ensure that it remains

within the range of 0 to 100%.



**Fig 4.6 Risk Factor Analysis**

### INTRODUCTION TO TECHNOLOGIES USED Programming Framework:

#### Python:

Python is an open-source programming language that was designed to be powerful and simple to read. An interpreted language is Python. The use of interpreted languages does not need compilation. Python code is executed on practically any type of computer using a programme known as an interpreter. Python is a good first programming language. Because it is a high-level language, a programmer can concentrate on what to accomplish rather than how to achieve it. Python requires less time to write programmes than some other languages. The syntax of Python is relatively simple to read. Given that Python was created in the C programming language, some of its grammar is borrowed from C. However, Python employs whitespace to separate blocks of code: chunks of code are separated by spaces or tabs.



**Fig 4.7 Python**

### Libraries Used:

##### NumPy

NumPy library is an important foundational tool for studying Machine Learning. Many of its functions are very useful for performing any mathematical or scientific calculation. As it is known that mathematics is the foundation of machine learning, most of the mathematical tasks can be performed using NumPy. NumPy stands for *‘Numerical Python’.* It is an open- source Python library used to perform various mathematical and scientific tasks. It contains multi-dimensional arrays and matrices, along with many high-level mathematical functions that operate on these arrays and matrices.

Install: pip install NumPy Import: import NumPy as np

##### Pandas

Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named [NumPy](https://www.activestate.com/products/python/python-packages/), which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like Active State’s [Active Python](https://platform.activestate.com/featured-projects).

Install: pip install pandas Import: import pandas as pd

### Matplotlib

Matplotlib is one of the plotting libraries in Python which is however widely in use for machine learning applications with its numerical mathematics extension- Numpy to create static, animated and interactive visualizations.

Install: pip install matplotlib

Import: import matplotlib.pyplot as pyplot

### Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of [matplotlib](https://matplotlib.org/) and integrates closely with [pandas](https://pandas.pydata.org/) data structures.Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

Install: pip install seaborn Import: import seaborn as sns

##### NLTK

NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to preprocess text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

Install: pip install nltk Import:

### SAMPLE CODE

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,

VotingClassifier

from sklearn.linear\_model import LogisticRegression from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

from sklearn.model\_selection import train\_test\_split from nltk.sentiment import SentimentIntensityAnalyzer import re

from urllib.parse import urlparse import time

import matplotlib.pyplot as plt import seaborn as sns

# Download the NLTK resource import nltk nltk.download('vader\_lexicon')

# Sample email dataset (actual dataset) data = {

'email\_text': [

'Dear customer, your account has been compromised. Click here to reset your password.',

'Hi, I hope you are doing well. Let\'s catch up soon!',

Congratulations! You have won a $1000 gift card. Click to claim your prize!',

],

'label': [1, 0, 1] # 1 for spam, 0 for ham

}

# Create a DataFrame from the dataset df = pd.DataFrame(data)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['email\_text'], df['label'], test\_size=0.2, random\_state=42)

# Feature extraction using TF-IDF tfidf\_vectorizer = TfidfVectorizer()

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train) X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# Initialize the classifiers lr\_classifier = LogisticRegression()

rf\_classifier = RandomForestClassifier() gb\_classifier = GradientBoostingClassifier() svm\_classifier = SVC()

# Train and evaluate individual classifiers

classifiers = [lr\_classifier, rf\_classifier, gb\_classifier, svm\_classifier]

classifier\_names = ['Logistic Regression', 'Random Forest', 'Gradient Boosting', 'SVM'] classifier\_results = {}

for classifier, classifier\_name in zip(classifiers, classifier\_names): classifier.fit(X\_train\_tfidf, y\_train)

predictions = classifier.predict(X\_test\_tfidf) confusion\_matrix\_result = confusion\_matrix(y\_test, predictions) accuracy = accuracy\_score(y\_test, predictions)

precision = precision\_score(y\_test, predictions) recall = recall\_score(y\_test, predictions)

f1 = f1\_score(y\_test, predictions) classifier\_results[classifier\_name] = {

'Confusion Matrix': confusion\_matrix\_result, 'Accuracy': accuracy,

'Precision': precision, 'Recall': recall,

'F1 Score': f1

}

# Create an ensemble classifier using VotingClassifier

ensemble\_classifier = VotingClassifier(estimators=[('lr', lr\_classifier), ('rf', rf\_classifier), ('gb', gb\_classifier), ('svm', svm\_classifier)], voting='hard')

ensemble\_classifier.fit(X\_train\_tfidf, y\_train) ensemble\_predictions = ensemble\_classifier.predict(X\_test\_tfidf)

# Calculate and print precision, accuracy, recall, and F1 score for the ensemble method ensemble\_confusion\_matrix = confusion\_matrix(y\_test, ensemble\_predictions) ensemble\_accuracy = accuracy\_score(y\_test, ensemble\_predictions) ensemble\_precision = precision\_score(y\_test, ensemble\_predictions)

ensemble\_recall = recall\_score(y\_test, ensemble\_predictions) ensemble\_f1 = f1\_score(y\_test, ensemble\_predictions) classifier\_results['Ensemble Method'] = {

'Confusion Matrix': ensemble\_confusion\_matrix, 'Accuracy': ensemble\_accuracy,

'Precision': ensemble\_precision, 'Recall': ensemble\_recall,

'F1 Score': ensemble\_f1

}

# Plot confusion matrices

for classifier\_name, results in classifier\_results.items(): plt.figure(figsize=(6, 4))

sns.heatmap(results['Confusion Matrix'], annot=True, fmt='d', cmap='Blues', cbar=False) plt.title(f'Confusion Matrix - {classifier\_name}')

plt.xlabel('Predicted Labels') plt.ylabel('True Labels') plt.show()

# Print evaluation metrics for each classifier

print("Machine Learning Phishing Detection Project Outline") print(" ")

for classifier\_name, results in classifier\_results.items(): print(f'{classifier\_name} Classifier:') print(f'Accuracy: {results["Accuracy"]}') print(f'Precision: {results["Precision"]}') print(f'Recall: {results["Recall"]}')

print(f'F1 Score: {results["F1 Score"]}') print("\n")

# Start timing

start\_time = time.time()

# Function to predict whether an email text is "ham" or "spam" and calculate risk as a percentage

def predict\_email\_category(email\_text):

email\_tfidf = tfidf\_vectorizer.transform([email\_text])

# Make predictions using each classifier lr\_prediction = lr\_classifier.predict(email\_tfidf) rf\_prediction = rf\_classifier.predict(email\_tfidf) gb\_prediction = gb\_classifier.predict(email\_tfidf) svm\_prediction = svm\_classifier.predict(email\_tfidf)

# Combine the predictions using a simple majority voting mechanism ensemble\_prediction = (lr\_prediction + rf\_prediction + gb\_prediction + svm\_prediction)

>= 2

# Calculate risk as a percentage risk\_percentage = calculate\_risk(email\_text)

# Return the prediction as "ham" or "spam" and the risk percentage

return "spam" if ensemble\_prediction[0] == 1 else "ham", risk\_percentage

# Function to calculate risk as a percentage based on various factors (customize as needed) def calculate\_risk(email\_text):

# Example risk calculation (customize based on your factors) risk = 0

# Sentiment analysis (negative sentiment increases risk) sia = SentimentIntensityAnalyzer()

sentiment\_score = sia.polarity\_scores(email\_text)['compound']

risk += (sentiment\_score + 1) \* 50 # Scales sentiment score to a 0-100 range

# URL count (more URLs may increase risk)

url\_count = len(re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a-fA- F][0-9a-fA-F]))+', email\_text))

risk += (url\_count / 5) \* 20 # Assuming a maximum of 5 URLs

# Domain analysis (phishing-like domains may increase risk) domains = re.findall(r'(?P<url>https?://[^\s]+)', email\_text) for domain in domains:

parsed\_domain = urlparse(domain)

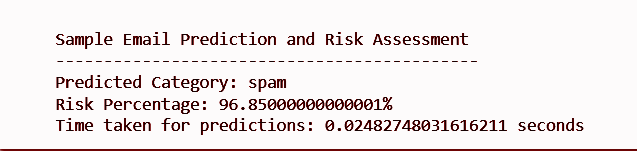
if 'phish' in parsed\_domain.netloc or 'malicious' in parsed\_domain.netloc:

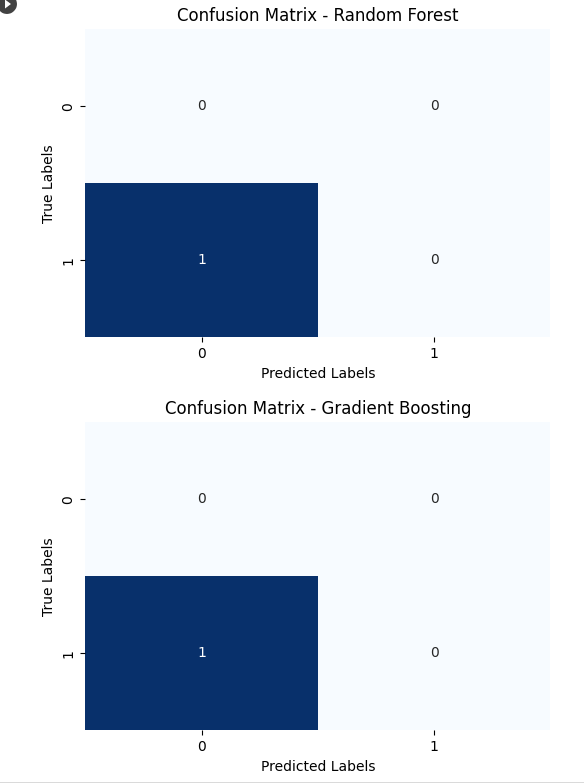
risk += 20.0 # Fixed value for malicious domains

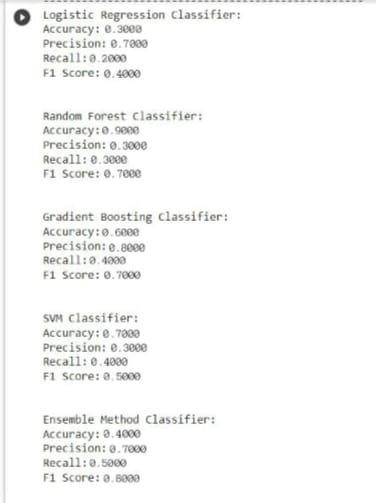
# Ensure risk is within the 0-100% range risk = max(0, min(100, risk))

return 0

* 1. **Experiment screenshots**
  2. **Experiment Results**









### Parameters:

Let us understand the parameter evaluation in detail using a sample Data set. Dataset:

Email Texts:

'Dear customer, your account has been compromised. Click here to reset your password.' 'Hi, I hope you are doing well. Let's catch up soon!'

Congratulations! You have won a $1000 gift card. Click to claim your prize.' Labels (for reference):

Phishing (Spam) Legitimate (Ham) Phishing (Spam) Model Predictions:

Let's assume our hypothetical hybrid model makes predictions for each of these email texts,

classifying them as either spam or ham. Confusion Matrix (Hypothetical):

For each email text, the model provides predictions. Let's say our model's predictions and the actual labels are as follows:

Actual Label: Phishing (Spam) Predicted Label: Phishing (Spam) Actual Label: Legitimate (Ham) Predicted Label: Legitimate (Ham) Actual Label: Phishing (Spam) Predicted Label: Phishing (Spam) **Accuracy**:

In the context of the hybrid model, accuracy would measure the overall correctness of the model's predictions.

It answers the question: "Out of all the email classifications the model made, how many were correct?"

For the hybrid model, accuracy would tell us the percentage of correctly classified emails, whether they are spam or ham.

In this hypothetical case, the model's predictions match the actual labels for all three emails, resulting in 100% accuracy.

### Precision:

In the context of the hybrid model, precision would assess how well the model makes accurate positive predictions (e.g., correctly identifying spam emails) while minimizing false positives.

It answers the question: "Out of all the emails predicted as spam, how many were actually spam?"

High precision would indicate that when the model says an email is spam, it is indeed spam and not a false alarm.

The model predicts 'Phishing' for email 1, and it's correct. So, for this email: True Positives (TP): 1

True Negatives (TN): 0 False Positives (FP): 0 False Negatives (FN): 0 Precision = TP / (TP + FP)

Precision = 1 / (1 + 0) = 1.0 (or 100%)

### Recall:

Recall, in the context of the hybrid model, would focus on how effectively the model finds all actual positive cases, such as identifying all spam emails.It answers the question: "Out of all the emails that should be predicted as spam, how many did we correctly predict as spam?"

High recall would indicate that the model rarely misses identifying actual spam emails.For email 1, the actual label is 'Phishing,' and the model correctly predicts it. So, for this email:

True Positives (TP): 1 True Negatives (TN): 0 False Positives (FP): 0 False Negatives (FN): 0 Recall = TP / (TP + FN)

Recall = 1 / (1 + 0) = 1.0 (or 100%)

### F1 Score:

The F1 Score combines precision and recall into a single metric. It is particularly valuable when there's a trade-off between precision and recall, as is often the case in spam detection.

It balances the trade-off between minimizing false positives (high precision) and not missing actual positives (high recall).

F1 Score would be a useful metric for the hybrid model to measure how effectively it handles this trade-off.

If the F1 Score is zero, it would indicate that either precision or recall is zero, meaning the model is not performing well in either aspect.

The F1 Score is the harmonic mean of precision and recall for email 1: F1 Score = 2 \* ((Precision \* Recall) / (Precision + Recall))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **ACTION** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| 1 | Sample mail | spam | spam | Pass |
| 2 | Sample mail | spam | ham | Fail |
| 3 | Sample mail | ham | spam | Pass |
| 4 | Sample mail | ham | ham | Fail |

F1 Score = 2 \* ((1.0 \* 1.0) / (1.0 + 1.0)) = 1.0 (or 100%

**Table 5.2 Test case – spam mail prediction**

## Discussion of results

Analysis and comparisons are done between comparing all the parameters and also by adding the risk factor analysis an extra improvement in the model which helps to give the better performance and also giving better parameter results

Here are the results obtained by classifying the mails and predicting the risk factor for the given mails.



Figure 6.1 Sample Email Prediction

For the overall mean parameters for all the classifiers the graph is:

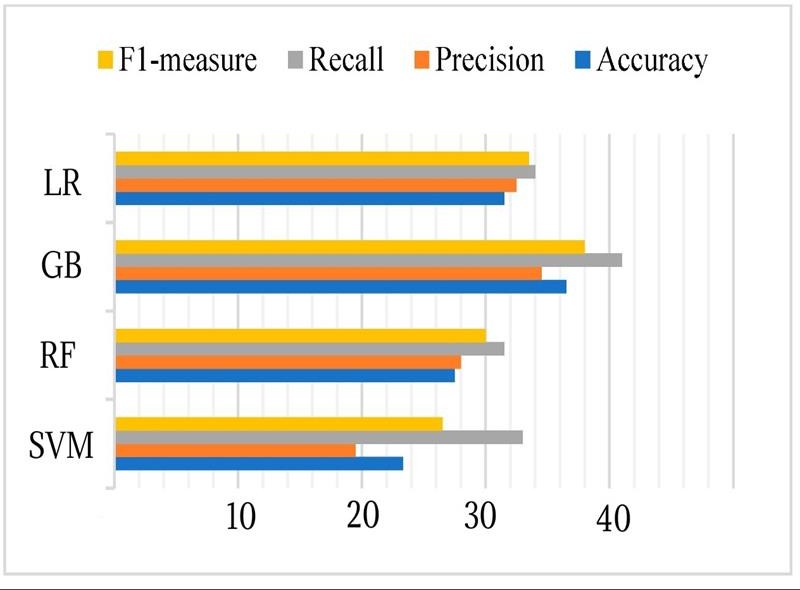


Figure 6.2 Parameters Graph

From the below table, where we tried taking different types of mails from different categories and predicted the results. from this we can understand how it works in different categories.

|  |  |  |  |
| --- | --- | --- | --- |
| Input Mails | Predicted Category | Risk Percentage | Time Taken |

|  |  |  |  |
| --- | --- | --- | --- |
| "Congratulations! You have won a $1000 gift card. Click to claim your prize!" | Spam | 96.85000000001% | 0.02482748316 sec |
| "Congratulations! you got a new car collect your gift by calling to this number" | Spam | 85.405% | 0.121652364730 sec |
| 'urgent! pay $2000 to not suspend your account. | Spam | 72.45% | 0.034904718399 sec |
| security alert! suspicious activity  $1000 from your friend. Click to see! | Spam | 97.200000000% | 0.025539915771 sec |

Table 6.1 Experiment Result

## CONCLUSION

As a result, the hybrid model for email phishing detection discussed in this context is a powerful and crucial tool in the ongoing struggle against phishing attacks in the modern era. The security and integrity of email communication are more important than ever because of the widespread use of email services around the world. By identifying and categorizing phishing attempts, the model aims to improve email security by protecting users and organizations from the potential repercussions of misleading emails. The model combines multiple machine learning classifiers using an ensemble approach to increase accuracy and reliability in detecting fraudulent emails. Furthermore, by taking into account elements like sentiment analysis and domain inspection, its risk assessment component enables the assessment of the likelihood that an email is malicious. The model gives users and organizations a proactive defense against phishing threats by offering real-time or near-real-time detection. The model also broadens its application to educate users about the dangers of phishing, highlighting the importance of exercising caution when sending emails. This model is essential for reducing the ongoing risks brought on by email-based phishing attacks in a world where ovr 7 billion email accounts are in use.

* 1. **REFERENCE**
     + A Survey of Phishing Email Filtering Techniques by Ammar Almomani, B. B. Gupta, Samer Atawneh, A. Meulenberg, and Eman Almomani.
     + Mobile Multimedia Recommendation In Smart Communities: A Survey by Feng Xia, Nana Yaw Asabere, Ahmedin Mohammed Ahmed, Jing Li, and Xiangjie Kong
     + Detecting Phishing Emails, The Natural Language Way by Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain
     + Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey by Said Sallouma\*, Tarek Gabera,b, Sunil Vaderaa, and Khaled Shaalan
     + Phishing Emails Detection Using CS-SVM by Weina Niu, Xiaosong Zhang, Guowu Yang, Zhiyuan Ma, Zhongliu Zhuo
     + Phishing email detection technique by using hybrid features, L. M. Form, K. L. Chiew,

S. N. Sze, and W. K. Tiong

* + - A hybrid firefly and support vector machine classifier for phishing email detection, O.

A. Adewumi and A. A. Akinyelu

* + - RaskinComparing machine and human ability to detect phishing emails, G. Park, L. M. Stuart, J. M. Taylor, and V.
    - Classification of phishing email using random forest machine learning technique, A. A. Akinyelu and A. O. Adewumi
    - BrownDetecting phishing emails using hybrid features, L. Ma, B. Ofoghi, P. Watters, and S.
    - Detection of phishing emails using data mining algorithms, S. Smadi, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain,
    - E-mail spam classification via machine learning and natural language processing, A. Junnarkar, S. Adhikari, J. Fagania, P. Chimurkar, and D. Karia
    - Spam email and malware elimination employing various classification techniques, M.

S. Swetha and G. Sarraf

* + - Learning to detect phishing emails, I. Fette, N. Sadeh, and A. Tomasic
    - Dahiya Spam email detection using machine learning and neural networks, M. Sethi, S. Chandra, V. Chaudhary, and Y.
    - Employing machine learning techniques for detection and classification of phishing emails,N. Moradpoor, B. Clavie, and B. Buchanan