A Phishing Detection System for Enhanced Cybersecurity Using Machine Learning

Adwaith Atholi Thiruvoth and Pushkar Ogale[a]
Stephen F. Austin State University, 1936 North Street, Nacogdoches, U.S.A.

Keywords: Email Phishing, Machine Learning, Email Classification, Cybersecurity, Phishing Detection, Supervised Learning, Random Forest, Support Vector Machine, GUI, Feature Extraction, Data Preprocessing, Model evaluation, SecureInbox.

Abstract: Email phishing is a pressing cybersecurity challenge that requires efficient detection methods. Emails that look legitimate lead users to malicious sites. Our work aims to develop a machine learning-driven email classification system, named SecureInbox. A comparative study of classical machine learning techniques like Random Forest, Naive Bayes, Decision Tree, SVM, and gradient boosting regression trees was conducted, and it was found to be successful in achieving high accuracy and effectiveness in distinguishing between legitimate and phishing emails. This study makes use of various statistical methods, classification algorithms to develop a user-friendly graphical interface (GUI) for seamless email classification. SecureInbox automatically fetches the mailbox file associated with the current user in a Linux environment and classifies their emails as phishing or not phishing while displaying the results interactively. Our work helps to strengthen email security by providing a convenient tool for phishing email identification, thereby enhancing defence against cyber threats.

1 INTRODUCTION

In an era dominated by digital communication, email is a widely used tool for personal and professional communication. However, it also serves as a medium for cyber-attacks, especially phishing attacks. Phishing emails trick individuals into revealing sensitive information, such as passwords and personally identifiable information, posing risk to individuals and organizations. Traditional email filtering methods struggle to keep pace with the evolving attack tactics employed by cybercriminals (Tessian, 2022). Consequently, there is a pressing need for innovative approaches to enhance email classification and phishing detection capabilities. Our work focusses on using machine learning techniques to classify email in real time so that any user of a computing system can scan his email to determine a phishing effort. Our work introduces SecureInbox, a machine learning-based phishing detection system specifically optimized for Linux environments running RedHat 8.

The widespread problem of phishing attacks highlights how easily email systems can be tricked. Conventional email filtering mechanisms which rely on rule-based heuristics, signature detection, and blocklisting, often fall short in accurately identifying sophisticated phishing attempts. Cybercriminals refine their tactics to evade detection and this reduces the effectiveness of these conventional email filtering mechanisms. Moreover, it is hard for humans to spot every phishing email because cybercriminals know how to make these emails look real. Therefore, there is a critical need to enhance existing email security measures with advanced technological solutions, particularly those leveraging machine learning algorithms, to bolster email classification and phishing detection capabilities.

The proposed solution to address the challenge of phishing involves the development of an advanced email classification system utilizing machine learning techniques. This system is empowered by machine learning algorithms to effectively distinguish between safe and phishing emails. Leveraging labeled datasets, various algorithms—Naive Bayes (Zamora,

[a] https://orcid.org/0000-0002-0417-1996

Thiruvoth, A. A. and Ogale, P.
A Phishing Detection System for Enhanced Cybersecurity Using Machine Learning.
DOI: 10.5220/0013570800003964
In *Proceedings of the 20th International Conference on Software Technologies (ICSOFT 2025)*, pages 355-360
ISBN: 978-989-758-757-3; ISSN: 2184-2833

2024), Decision Tree, Random Forest, Gradient Boosting Regression Trees, and Support Vector Machine (Godfried, 2022)—are used for training the models and evaluated for their effectiveness. Feature extraction methods like Term Frequency-Inverse Document Frequency (TF-IDF) are used to transform raw email text to a suitable numerical format (Zamora, 2024). By selecting the optimal algorithm based on performance evaluation, this system ensures enhanced cybersecurity measures.

Our work provide an algorithmic benchmarking of the classical machine learning models that are mentioned earlier, conducting a comprehensive comparative analysis. Secondly we introduce a Linux-first design through SecureInbox, representing a dedicated phishing classification tool that is optimized for RedHat 8 environments. Finally we emphasize reproducibility and extensibility by making publicly available with a modular architecture that enables easy adaptation to other Linyx distributions or integration with additional datasets, facilitating future research and development in this domain.

2 RELATED WORKS

The evolving sophistication of phishing attacks has driven significant research into more effective detection methods. Traditional approaches relying on rule-based filters and signature detection (Tessian, 2022) fall short against evolving social engineering strategies, as bad actors continuously modify their strategies to bypass static defences. This led to the search for a solution which can adapt to the new threat. Recent work (Zamora, 2024) demonstrated the effectiveness of TF-IDF feature extraction combined with Naive Bayes classification, while (Godfried, 2022) showed impressive performance from ensemble methods like Random Forest and Gradient Boosting for email classification tasks.

Current state-of-the-art systems primarily employ supervised learning techniques, with particular success from Support Vector Machines in handling high-dimensional text data (Zamora, 2024).

Our work investigates the performance characteristics of multiple machine learning models (including Naive Bayes, Decision Trees, and SVM) in a Linux environment. Furthermore, we extend beyond pure algorithm evaluation by implementing SecureInbox as a complete, RedHat-8-optimized solution that maintains detection accuracy while meeting the performance constraints of local deployment. The system's modular design and open availability represent an additional contribution, enabling future research to build upon our work for other Linux distributions or integrate additional detection features.

This combination of rigorous algorithm benchmarking, platform-specific optimization, and commitment to reproducible research distinguishes our approach from previous work in the field, while maintaining compatibility with established best practices in feature extraction and model evaluation.

3 METHODOLOGY

Our phishing detection framework follows a structured machine learning pipeline, and it involve the following steps shown in Figure 1.

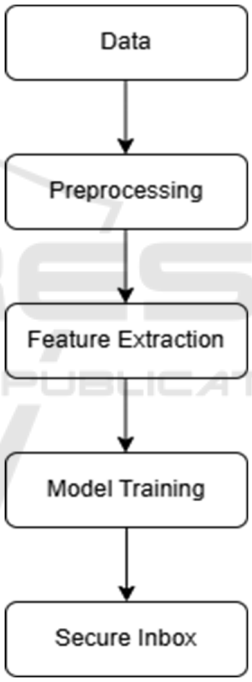

Figure 1: End-to-end Workflow.

3.1 Data Collection

The dataset used in this project was ‘Phishing_Emails.csv’, taken from Kaggle (Zamora, 2024), a website that provides a diverse collection of datasets for research purposes. Our dataset comprises labelled email samples, which are categorized as Safe or Phishing Emails, and consists of two main features or columns: “Email Text” containing the content of emails, and “Email Type” indicating the label of each email, labelled as “Safe Email” and “Phishing Email”.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18650 entries, 0 to 18649
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  18650 non-null  int64
 1   Email Text  18634 non-null  object
 2   Email Type  18650 non-null  object
dtypes: int64(1), object(2)
memory usage: 437.2+ KB
```

Figure 2: The Data Frame Structure.

Figure 2 shows the Data Frame structure, including the total number of rows and columns, column names, non-null counts per column, and data types (Dtype) of each column.

```
 1   Email Text  18634 non-null  object
 2   Email Type  18650 non-null  object
dtypes: int64(1), object(2)
memory usage: 437.2+ KB
None
   Unnamed: 0                                         Email Text     Email Type
0           0  re : 6 . 1100 , disc : uniformitarianism , re ...     Safe Email
1           1  the other side of * galicismos * * galicismo *...     Safe Email
2           2  re : equistar deal tickets are you still avail...     Safe Email
3           3  \nHello I am your hot lil horny toy.\n    I am...  Phishing Email
4           4  software at incredibly low prices ( 86 % lower...  Phishing Email
```

Figure 3: Data Frame details

Figure 3 displays the first five rows of the Data Frame df, along with the values in each column, providing a preview of the data to understand its structure and contents.

3.2 Data Preprocessing

The acquired dataset undergoes preprocessing to ensure data cleanliness and compatibility with machine learning algorithms. Tasks include standardizing text formats and handling missing values.

3.3 Feature Extraction

Techniques such as TF-IDF are employed to transform the content of emails into numerical feature vectors. We prioritize TF-IDF over transformer-based embeddings (e.g., BERT) due to its computational advantages and proven effectiveness in resource-constrained environments. Comparative studies demonstrate TF-IDF's 12–18× faster processing speeds and 10× lower memory usage than BERT for text classification, with minimal accuracy trade-offs (Gomes et al., 2023).

TF-IDF vectorisation process assigns weight to each word in the email text based on its frequency and rarity across the whole document. This allows capturing of the important characteristics of the email content, which serve as input for the machine learning algorithms.

3.4 Model Training

Supervised learning algorithms, including Naive Bayes, Decision Tree, Random Forest, Gradient Boosting Regression Trees, and Support Vector Machine (SVM), are trained on the pre-processed dataset [3].

During training, the algorithms learn to classify emails as legitimate, or phishing based on the extracted features.

3.5 Model Evaluation

The trained models are evaluated using performance metrics such as accuracy, precision, recall, and F1 score to assess their effectiveness in email classification (Kanstrén 2020).

Accuracy (1) describes the number of correct predictions.

$$\text{Accuracy} = \frac{\#\text{ of correct predictions}}{\#\text{ of all predictions}} \quad (1)$$

Precision (2) is the measure of how many positive instances predicted by the model were correct.

$$\text{Precision} = \frac{\text{True positives}}{\text{Predictive positives}} \quad (2)$$

Recall (3) is the measure of how many positive cases the model correctly predicted, over all the positive cases in the dataset.

$$\text{Recall} = \frac{\text{True positives}}{\text{Actual positives}} \quad (3)$$

The F1 score (4) is the harmonic mean of precision and recall. F1 score provides a single metric that weighs the two ratios (precision and recall) in a balanced way.

$$\text{F1 Score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

This evaluation ensures that the selected model is capable of accurately distinguishing between legitimate and phishing emails.

3.6 Integration and Deployment

After the evaluation, the best-performing machine learning model is selected for the development of the SecureInbox email classification system. SecureInbox incorporates a user-friendly graphical interface (GUI), allowing the users to interact with the email classification system seamlessly.

4 IMPLEMENTATION

The implementation of the SecureInbox email classification system was completed on a Dell Server, running a RedHat 8 Linux environment. Interactions with the Linux server were through a client program MobaXterm Personal Edition v24.0.

The client program could be run from a personal computer, in this case a Windows environment. This environment provided the necessary resources for SSH sessions, SSH compression, SSH-browser functionality, and X11-forwarding for remote display.

4.1 Implementation Details

SecureInbox is designed specifically for Linux systems. The application runs on a Linux system running the RedHat 8 distribution with kernel version 4.18.0 and is not compatible with Windows systems. The implementation of the email classification system involves several key components and processes, as outlined below in the following sub sections.

4.1.1 Programming Languages and Libraries

Python is the primary programming language used for developing SecureInbox as well as the underlying classifier system.

Various libraries and frameworks are utilized, including ‘scikit-learn’, ‘pandas’, ‘joblib’, and ‘tkinter’. These libraries provide essential functionalities for machine learning, data manipulation, and graphical user interface development.

4.1.2 Data Collection Module

This module is responsible for collecting labelled email datasets from various sources and ensuring data integrity and quality.

4.1.3 Preprocessing Module

This module manages preprocessing tasks such as noise removal, formatting standardization, and missing value handling to prepare the dataset for training. The following code snippet depicts the pre-processing module.

```
def load_dataset(self, filename):
  try:
      df = pd.read_csv(filename,
encoding='utf-8')
      except UnicodeDecodeError:
           df = pd.read_csv(filename,
      encoding='ISO-8859-1')

  # Check for missing values and handle
them accordingly
  if df.isnull().values.any():
  # Fill missing values with empty
strings
      df.fillna('', inplace=True)
```

The code snippet also illustrates the code pertaining to missing value handling and standardization.

4.1.4 Feature Extraction Module

This module utilizes TF-IDF and other feature extraction techniques to transform raw email text into numerical feature vectors. The ‘TfidfVectorizer’ class from ‘scikit-learn’ is used for feature extraction (Zamora, 2024). The following code snippet depicts the feature extraction module.

```
# Feature extraction using TF-IDF
vectorizer = TfidfVectorizer(stop_words
='english')
X = vectorizer.fit_transform(df['Email
Text'])

# Label encoding
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df
['Email Type'] )
```

The TF-IDF represents the importance of each term in distinguishing between legitimate and phishing emails.

4.1.5 Model Training Module

This module trains multiple supervised learning algorithms on the pre-processed dataset to develop robust email classification models.

4.1.6 Model Evaluation Module

This module evaluates the performance of trained models using appropriate metrics to select the best-performing model for deployment. It displays the accuracy, precision, recall and F1 score from the evaluation.

4.2 Test Methodology

After training, the model undergoes testing to assess their performance in real-world scenarios. This involves evaluating the models' ability to accurately classify unseen email data, including both legitimate and phishing emails. Figure 5 depicts the 'test_emails' function which facilitates the testing process by loading emails from a mailbox file and using the trained classifier to predict the label (legitimate or phishing) for each email. Preprocessing takes place similarly to the training phase when extracting the emails from the mailbox.

To provide a graphical user interface (GUI) for the application, a user-friendly GUI was developed using the 'tkinter' library for SecureInbox to facilitate seamless interaction with the email classification system.

5 RESULTS

The comparison of the algorithms used in this project was based on their performance in accurately classifying emails as either legitimate or phishing. By evaluating metrics such as accuracy, precision, recall, and F1 score, we assess the effectiveness of each algorithm in distinguishing between the two classes of emails. This comparative analysis aims to identify the most suitable algorithm for developing a robust email classification system.

Results indicated that each algorithm achieved high performance in classifying emails as phishing and legitimate. SVM and Random Forest produced the best results with an F1 score of 97.75% and 96.73%. The comparative analysis is depicted in Figure 4. The four bars of the histogram refer to accuracy, precision, recall and F1 scores, respectively.

A histogram was plotted based on the results obtained after evaluating the model. Note that SVM exhibited the highest overall efficiency followed by the Random Forest model. Even though both these models are suitable candidates to build SecureInbox, we considered factors such as computational efficiency and scalability. We determined that Random Forest was the optimal choice for the task, exhibiting acceptably high levels of accuracy, precision, recall, and F1 score.

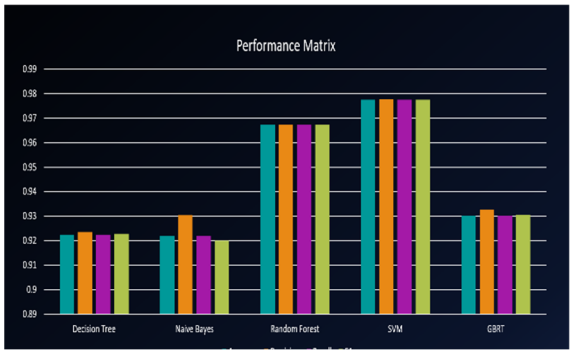

Figure 4: Histogram of accuracy, precision, recall and F1 scores for various Machine learning algorithms for the specific data set.

The selection of the algorithm was followed by developing the SecureInbox application with an integrated Graphical User Interface.

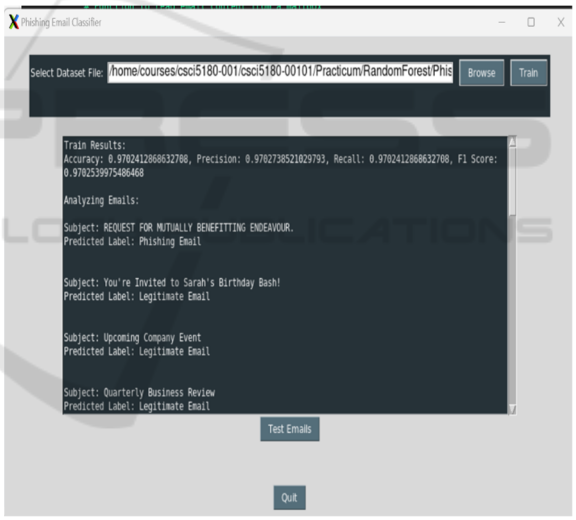

Figure 5: SecureInbox Results Output.

Figure 5 shows the screenshot of the results produced by the SecureInbox after training. This interface allows users to train the Random Forest model with their dataset. Once they are trained, the performance of the model will be displayed, and your model is ready to evaluate and classify emails.

6 CONCLUSION

Traditional email filtering mechanisms have become increasingly outdated in the face of evolving cyber threats, particularly phishing attacks. Our work heavily focused on machine learning algorithms to tackle the persistent challenge of phishing emails.

We employed various supervised learning techniques such as Naive Bayes, Decision Tree, Random Forest, Gradient Boosting Regression Trees, and Support Vector Machine (SVM), and compared the results to choose the best algorithm for developing SecureInbox.

Through rigorous testing and evaluation, we assessed the performance of these algorithms using metrics like accuracy, precision, recall, and F1 score. Our results indicated high efficacy across multiple algorithms, with SVM and Random Forest standing out as top performers, achieving F1 scores of 97.75% and 96.73%, respectively. Based on computational efficiency and scalability, we determined Random Forest to be the optimal choice for our email classification system.

The model was successfully integrated into the SecureInbox application with a user-friendly Graphical User Interface, allowing users to train with their dataset and analyse the emails to accurately classify them as legitimate or phishing. Our work demonstrates that machine learning is an effective tool that can be used to detect phishing attempts through email.

7 FUTURE WORK

While this study provided insights into various supervised learning algorithms for email classification and demonstrated the use of machine learning as an effective classification tool, there is room for future research and development. First, we will advance feature engineering by investigating distilled BERT embeddings alongside novel linguistic pattern extraction. For real-world deployment, we are developing Postfix/MTA plugins for real-time scanning and implementing incremental learning to adapt to emerging attack patterns.

Currently the tool enables email analysis on a Linux system. We can adapt this tool in the future to work within a Windows environment, providing broader accessibility and integration with common email clients and server configuration. To broaden accessibility, cross-platform expansion will include Windows support via Docker containers.

REFERENCES

Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. I. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, *5*(6).

Godfried, I. (2022, January 4). Decision Trees, Random Forests, and Gradient Boosting: What's the Difference? Towards Data Science. Retrieved from https://towards datascience.com/decision-trees-random-forests-and-gradient-boosting-whats-the-difference-ae435cbb67ad

Gomes, L., da Silva Torres, R., & Côrtes, M. L. (2023). BERT-and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: a comparative study. *Information and Software Technology*, *160*, 107217.

Harikrishnan, N. B., Vinayakumar, R., & Soman, K. P. (2018, March). A machine learning approach towards phishing email detection. In *Proceedings of the anti-phishing pilot at ACM International workshop on security and privacy analytics (IWSPA AP)* (Vol. 2013, pp. 455-468).

Kanstrén, T. (2020, September 11). A Look at Precision, Recall, and F1 Score: Exploring the relations between machine learning metrics. *Towards Data Science*. Retrieved from https://towardsdatascience.com/a look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Tessian. (2022, January 12). Phishing Statistics 2020. *Tessian* Blog. Retrieved from https://www.tessian. com/blog/phishing-statistics-2020/#how-delivered

Zamora, N. (2024). Phishing detection: Bayes model. *Kaggle*. https://www.kaggle.com/code/nordszamora/ phishing-detection-bayesmode