BIL520 Intro to Cyber Security Project Final Report: Developing a Phishing Email Detection Tool Using NLP and Machine Learning

Selin Mergen
231111011
s.mergen@etu.edu.tr

Abstract

Phishing attacks via email are a prevalent cybersecurity threat, with attackers constantly evolving their strategies to bypass security measures. This proposal outlines a project to develop a phishing detection tool using a combination of Natural Language Processing (NLP) and machine learning techniques. The tool aims to enhance the accuracy of phishing email detection and will be deployed in a Streamlit application for user-friendly real-time analysis.

1 Introduction

Phishing attacks, particularly through email, have become a pervasive and sophisticated cybersecurity threat, causing significant financial losses and compromising sensitive information across the globe[1]. As attackers continue to evolve their strategies, traditional security measures often fall short, necessitating the development of more advanced tools to detect and mitigate these threats.

Natural Language Processing (NLP) and machine learning techniques have emerged as potent tools in this fight against phishing, offering the ability to analyze the content and context of emails to uncover malicious intent. This project draws inspiration from the increasing prevalence of phishing attacks and the potential of NLP to provide a more nuanced and effective means of detection. It aims to contribute to the field by developing a sophisticated phishing detection tool, leveraging the latest advancements in NLP and machine learning(Chandrasekaran et al., 2006; Salloum et al., 2022).

The motivation for this project also stems from the observed limitations of existing phishing detection systems, which often rely on signature-based methods and blacklists. While these approaches have their merits, they struggle to keep pace with the rapidly evolving tactics of attackers, leading to a substantial number of false negatives and false positives(Toolan and Carthy, 2010). By integrating robust NLP techniques and machine learning models, this project seeks to overcome these limitations, offering a more adaptable and accurate solution for phishing email detection.

[1]According to the FBI's Internet Crime Report, phishing incidents have doubled in frequency from 2019 to 2020, leading to substantial financial losses.

2 Related Works

The application of NLP in phishing detection has been the subject of numerous studies. Egozi and Verma (2018) demonstrated the efficacy of robust NLP techniques in phishing email detection. Salloum et al. (2022) provided a comprehensive review of the literature, emphasizing the variety of NLP techniques employed in this domain. Yasin and Abuhasan (2016) proposed an intelligent classification model, showcasing the potential of machine learning in this field.

In addition to these works, Chandrasekaran et al. (2006) explored the use of stylistic and lexical features in phishing email detection, providing valuable insights into the textual characteristics of phishing emails. Toolan and Carthy (2010) applied Information Retrieval (IR) techniques to this problem, demonstrating the potential of these methods in identifying phishing attempts.

3 Dataset Description

This project utilized a comprehensive collection of datasets, primarily focused on the analysis of email data to identify phishing activities. The datasets Ire obtained from a publicly available source at https://zenodo.org/records/8339691, comprising various types of email data. A total of nine datasets, amounting to 386.5 KB, Ire employed in this study:

- CEAS_08.csv
- Nazario_5.csv

- Nazario.csv
- Nigerian_5.csv
- Nigerian_Fraud.csv
- SpamAssasin.csv
- TREC 05.csv
- TREC_06.csv
- TREC_07.csv

3.1 Dataset Characteristics

The datasets exhibited varying characteristics:

- The *Nazario* and *Nigerian Fraud* datasets exclusively contain phishing emails.
- The *Ling* and *Enron* datasets, which are not part of the selected datasets for this project, possess only two features: 'Subject' and 'Body'.
- The other datasets consist of six features: 'Sender', 'Receiver', 'Date', 'Subject', 'Body', and 'Urls'.

3.2 Dataset Origins

Each dataset originates from a distinct source:

- **Ling.csv, Enron.csv, SpamAssasin.csv, TREC_05.csv, TREC_06.csv, TREC_07.csv, CEAS_08.csv, Nigerian_Fraud.csv, and Nazario.csv** are derived from the original datasets: Ling Spam, Enron, Apache SpamAssassin's public collections, TREC Public Corpus from 2005 to 2007, CEAS 2008 Challenge Lab Evaluation Corpus, Nigerian fraud, and Nazario, respectively.

3.3 Dataset Combination

The datasets Ire combined to enhance the diversity and comprehensiveness of the email data:

- For the *Nigerian_5.csv* dataset, 600 legitimate emails from each dataset (SpamAssasin.csv, TREC_05.csv, TREC_06.csv, TREC_07.csv, and CEAS_08.csv) Ire combined with the Nigerian_Fraud.csv dataset.
- In the case of *Nazario_5.csv*, 300 legitimate emails from each dataset (SpamAssasin.csv, TREC_05.csv, TREC_06.csv, TREC_07.csv, and CEAS_08.csv) Ire merged with the Nazario.csv dataset.

4 Methodology

4.1 Baseline Model

4.2 Preprocessing

Preprocessing is a vital step in preparing the dataset for analysis. Each step was carefully chosen based on its relevance and impact on the overall quality of the data and the effectiveness of the phishing email detection model.

4.2.1 Handling Missing Values

Missing values can skew the analysis and affect model performance. I adopted different strategies for different types of data:

- For textual data (e.g., 'sender', 'receiver', 'subject', 'body'), missing values Ire replaced with 'Unknown' to maintain the integrity of the dataset without introducing bias.
- The 'urls' column's missing values Ire set to 0, assuming the absence of a URL in the email.
- Rows without 'label' values Ire dropped to ensure the model is trained on fully labeled data, crucial for supervised learning.

4.2.2 Text Normalization

The goal of text normalization is to convert the text to a more uniform format, which aids in reducing the complexity for NLP tasks:

- **LoIrcasing:** Converting all text to loIrcase ensures uniformity, as case differences can lead to the same words being treated differently.
- **Cleaning:** Removing punctuation and special characters helps reduce noise in the text data, focusing on the meaningful content.
- **Tokenization and Lemmatization:** These steps break the text into smaller units and reduce words to their base form, simplifying the analysis and helping in the accurate identification of relevant patterns.
- **Removing Stop Words:** Common words that are unlikely to contribute to phishing email detection are removed to focus on more meaningful words. In here when i only removed stop words for english and analyze the most common 50 words passed in spam email following result has appeares: [('com', 188822), ('http', 137688),

('cnn', 110879), ('www', 91205), ('html', 69865), ('2008', 69256), ('video', 68663), ('email', 63208), ('index', 60941), ('08', 56064), ('u', 54256), ('company', 41939), ('1', 41391), ('one', 35850), ('2', 34414), ('partner', 34406), ('price', 33295), ('e', 32716), ('10', 32089), ('url', 30285), ('3', 30063), ('please', 29838), ('time', 28796), ('money', 26204), ('may', 24996), ('news', 23872), ('account', 23704), ('get', 23028), ('05', 23012), ('new', 22580), ('5', 22574), ('information', 22531), ('top', 21764), ('product', 21491), ('0', 21399), ('00', 20931), ('95', 20783), ('day', 20578), ('go', 19592), ('06', 19368), ('business', 19315), ('service', 18790), ('net', 18779), ('name', 18655), ('offer', 18483), ('4', 18407), ('8', 18066), ('mail', 18008), ('01', 17967), ('000', 17827)]

As you can see there are some meaningless words that creates noise. These words should be eliminated in order to increase the accuracy of the model. To be able to achieve this i have removed stop words and some additional words as given below:

additional_stop_words = ['subject', 're', 'edu', 'use', 'http', 'https', 'www', 'html', 'index', 'com', 'net', 'org', 'ect', 'hou', 'cc', 'recipient', 'na', 'pm', 'am', 'et', 'enron']

4.2.3 Date Parsing and Encoding Categorical Data

Proper handling of non-textual data is critical for a holistic analysis:

- **Date Parsing:** Converting 'date' to a datetime object allows for the extraction of potentially relevant features like day of the Iek or time of the day, which could be indicative of phishing patterns.
- **Encoding Categorical Data:** This step converts categorical data into a numerical format, making it usable for machine learning models. Techniques like one-hot encoding help preserve the information without assuming ordinal relationships.

4.3 Feature Engineering

Feature engineering is a pivotal process in machine learning, where I transform raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. For this project, various features Ire engineered, each with a specific purpose in enhancing the model's ability to detect phishing emails.

4.3.1 Textual Features

Textual features play a crucial role in understanding the content of emails:

- **Length of Text:** The length of the email body and subject can be indicative of spam or phishing emails, which often have characteristic lengths.
- **Count of URLs:** Phishing emails frequently contain URLs for deceptive purposes. Counting these provides a direct measure of potential phishing intent.
- **Word and Character Count:** Total word and character count in the subject and body provide insights into the email's complexity and content structure.
- **Use of Suspicious Words:** Phishing emails often contain specific words that are less common in legitimate correspondence. Identifying and counting these words helps in distinguishing phishing emails.

4.3.2 Sender/Receiver Analysis

Analyzing the sender and receiver provides context beyond the content:

- **Domain Analysis:** Checking if the sender's and receiver's domain is from a known trusted source helps in identifying potentially malicious emails.
- **Sender-Receiver Relationship:** This feature indicates if there's a history of interaction betIen the sender and receiver, which is often absent in phishing attempts.

4.3.3 Date and Time Features

Temporal aspects of emails can also be telling:

- Phishing attacks may show patterns related to the time of sending, such as being sent at unusual hours. Features like time of the day and whether the email was sent on a Iekday or Iekend are thus relevant.

4.3.4 NLP Features

Advanced NLP techniques enable a deeper understanding of the textual content:

- **TF-IDF and Word Embeddings:** These features help in capturing the context and semantic information in the text, which is crucial for identifying sophisticated phishing attempts.

4.3.5 URL Features

The presence and nature of URLs in an email are significant indicators:

- Features such as the length of the URL, use of URL shortening services, and the presence of suspicious domains are critical in assessing the legitimacy of an email.

4.3.6 Sentiment Analysis

Sentiment analysis on the body and subject of the email can reveal the tone, which might be engineered to manipulate or deceive the recipient in phishing attempts.

4.3.7 Email Type

Categorizing the email as a reply, forward, or neither provides context to the email's origin and intent. Phishing emails may often mimic replies or forwards to appear legitimate.

Each feature was carefully selected and engineered to contribute to the model's ability to distinguish betIen legitimate and phishing emails effectively. The combination of these features provides a robust foundation for the predictive models used in this project.

4.3.8 Suspicious Word Count

In an effort to enhance the model's capability to detect phishing attempts, I included a feature that counts the occurrences of suspicious words within the email texts. These words Ire carefully selected based on their common usage in phishing emails, as identified by cybersecurity experts. They include terms that imply urgency, offer too-good-to-be-true promises, or suggest deceptive or unethical behavior. Over 200 such words and phrases Ire incorporated into the analysis, including terms like "urgent," "risk-free," "guaranteed," "limited offer," "confidentiality," "winner," and many others that are typically used to manipulate or exert pressure on the recipients. The presence of these suspicious words can significantly contribute to the identification of phishing emails, as they are rarely used in legitimate correspondence.(Chandrasekaran et al., 2006) (Salloum et al., 2022)

5 Exploratory Data Analysis (EDA)

Exploratory Data Analysis was conducted on both the original and preprocessed datasets to uncover insights into the characteristics of phishing emails. This section compares the findings from each dataset, emphasizing the impact of preprocessing on the data.

5.1 Original Dataset Analysis

5.1.1 Label Distribution

The label distribution provides insight into the balance of phishing and non-phishing emails within the dataset. The balance is crucial for ensuring that machine learning models can learn to distinguish betIen both classes effectively.

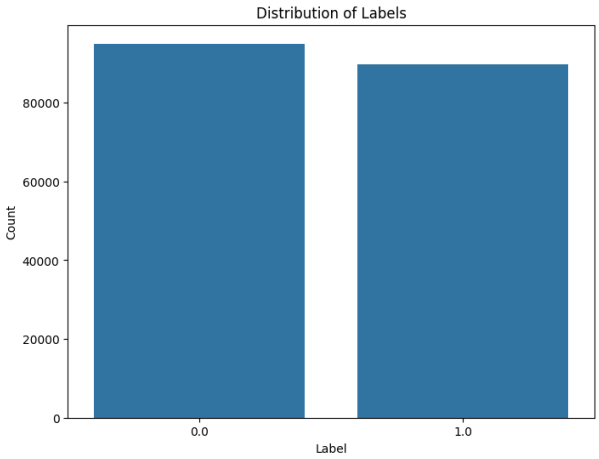

Figure 1: Label Distribution in the Original Dataset

5.1.2 Word Cloud Analysis

A word cloud was generated to visualize the most frequent terms across all emails in the original dataset. It often reveals common phishing cues such as 'urgent', 'request', 'bank', 'account', etc., used to deceive recipients.

5.1.3 Top Senders

Identifying the top senders allows us to pinpoint potential sources of phishing emails. It can indicate whether phishing attempts are concentrated from specific senders or are more distributed.

5.2 Preprocessed Dataset Analysis

The preprocessed dataset underInt several cleaning and normalization steps, including the handling of

missing values, text normalization, and removal of irrelevant features.

5.2.1 Data Histogram

The histogram provides a visual summary of the dataset's numerical features post-preprocessing. It helps in understanding the distribution and central tendencies of these features.

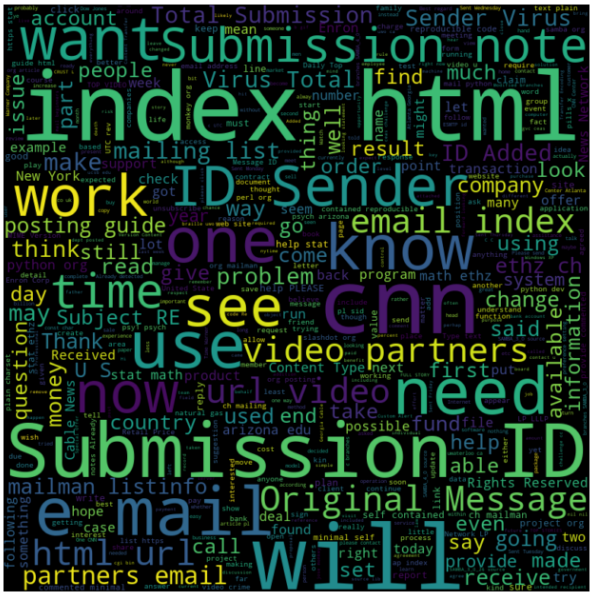

Figure 2: Word Cloud for All Data in Original Dataset

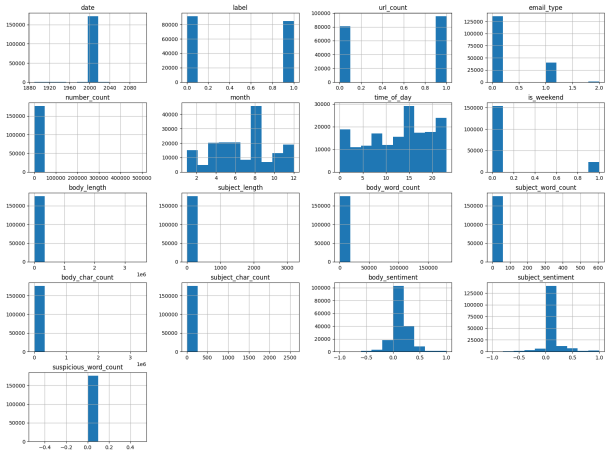

Figure 4: Histogram of the Preprocessed Dataset

5.2.2 Label Distribution

Observing the label distribution after preprocessing is essential to ensure that the data cleaning steps have not introduced any bias towards a particular class.

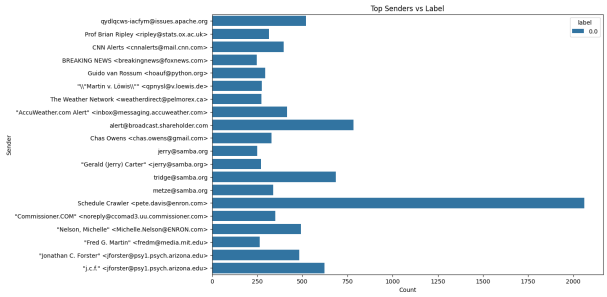

Figure 3: Top Sender Distribution in the Original Dataset

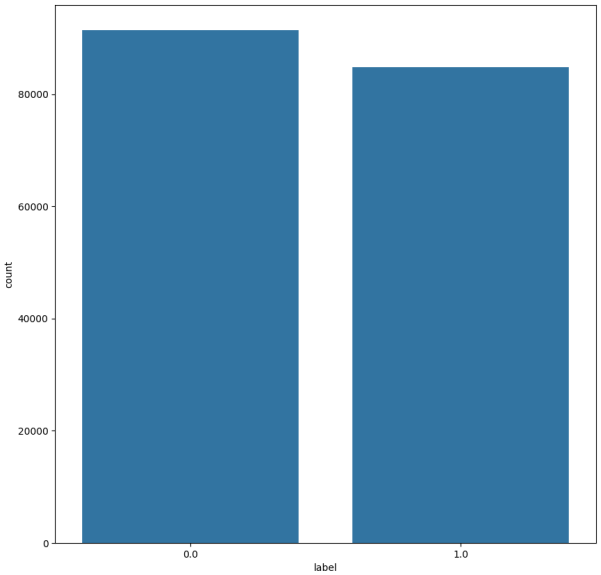

Figure 5: Label Distribution After Preprocessing

5.2.3 Length of Body Differences betIen Ham and Spam

Comparing the length of email bodies betIen ham (non-phishing) and spam (phishing) emails can indicate characteristic patterns used by attackers, such as shorter or longer email lengths.

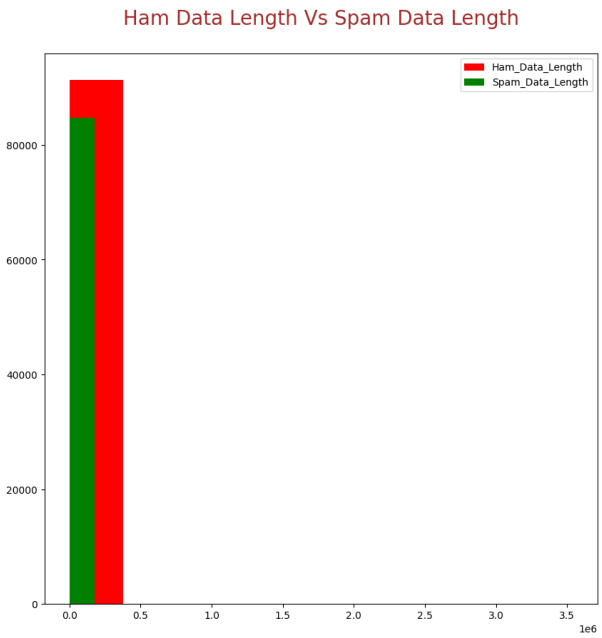

Figure 6: Length Distribution in Preprocessed Dataset

5.2.4 Word Cloud Analysis

Word clouds for the preprocessed dataset are subdivided into all emails, ham emails, and spam emails to contrast the most frequent terms used in different contexts.

- All:
- Ham:
- Spam:

5.2.5 Most Common Words

Analyzing the most common words in all emails versus spam emails can highlight terms commonly associated with phishing, which may be used to train the detection models.

- All:
- Spam:

5.2.6 Email Traffic Over Time

The email traffic over time can reveal trends such as bursts of phishing activity, which could be related to specific phishing campaigns.

Figure 7: Word Cloud for the Preprocessed Dataset

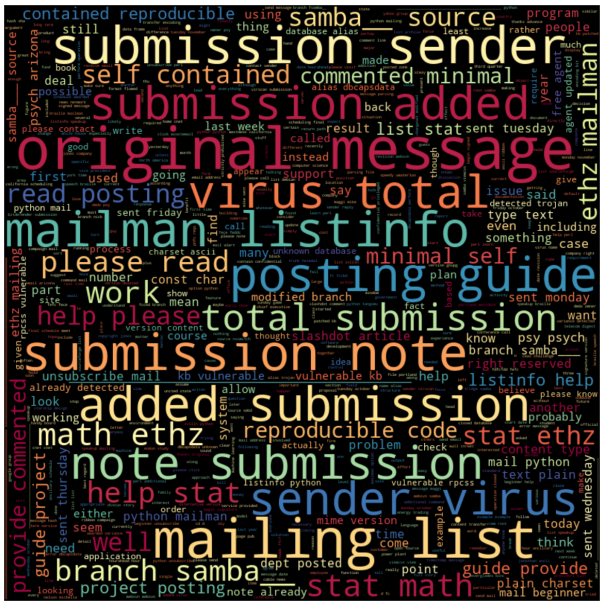

Figure 8: Word Cloud for Ham Emails in the Preprocessed Dataset

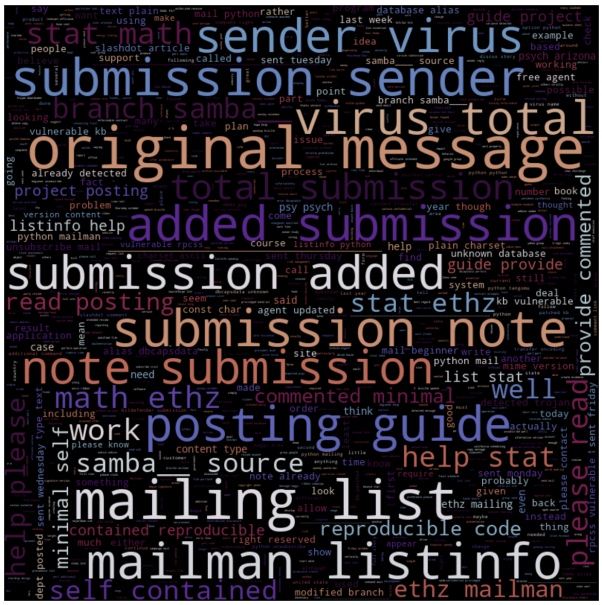

Figure 9: Word Cloud for Spam Emails in the Preprocessed Dataset

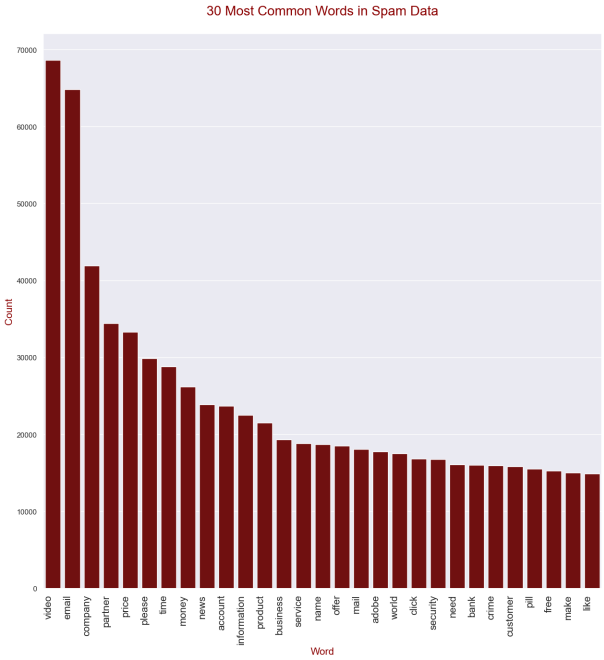

Figure 11: Most Common Words in Spam Emails of Preprocessed Dataset

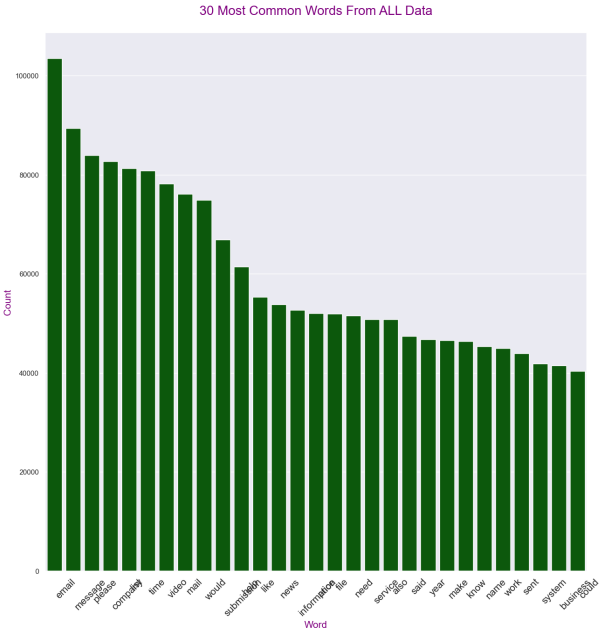

Figure 10: Most Common Words in the Preprocessed Dataset

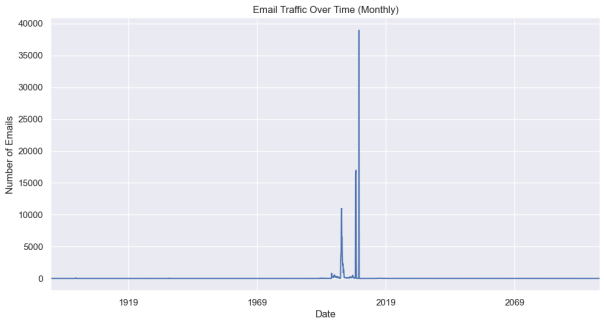

Figure 12: Email Traffic Over Time (Monthly)

5.3 Email Type Distribution

Understanding the distribution of different types of emails (normal, reply, forward) can provide context for typical user behavior and potential phishing strategies.

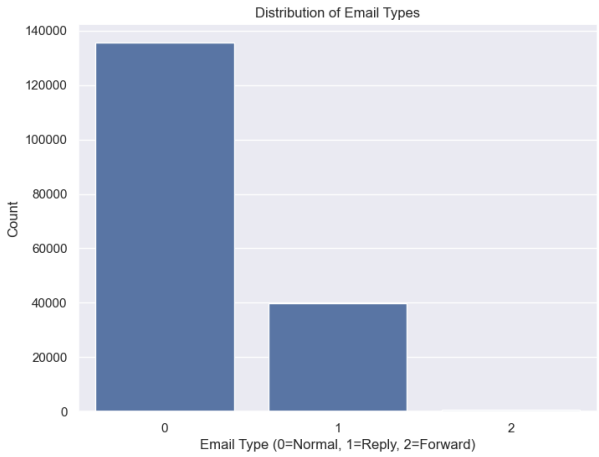

Figure 13: Distribution of Email Types in the Preprocessed Dataset

5.3.1 Email Distribution by Day of Iek

The distribution of emails by day of the Iek may reveal patterns related to the timing of phishing attacks, which often target users when they are less vigilant.

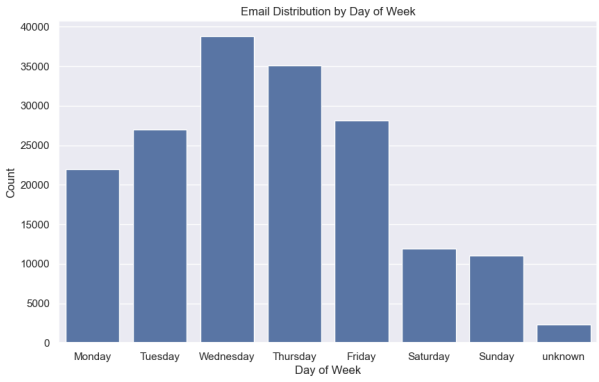

Figure 14: Distribution of Emails by Day in the Preprocessed Dataset

5.3.2 Email Distribution by Time of Day

Phishing emails may be sent at times when users are less likely to scrutinize the content thoroughly, making the time of day a relevant feature for detection.

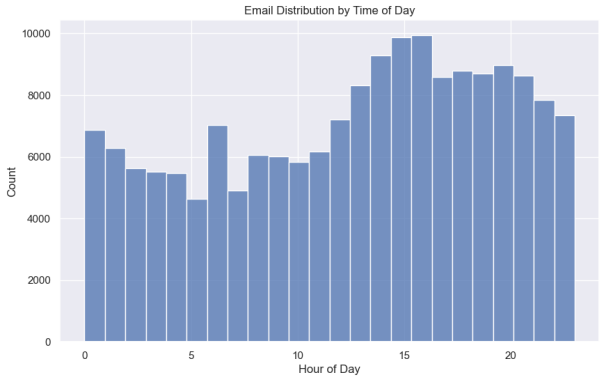

Figure 15: Distribution of Emails by Time in the Preprocessed Dataset

5.3.3 Email Body Sentiment Analysis by Label

Sentiment analysis can provide insights into the emotional tone of the email content, which in phishing emails is often crafted to elicit urgency or fear.

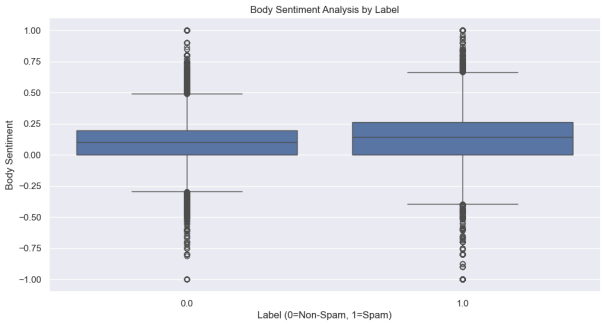

Figure 16: Email Body Sentiment Analysis by Label in the Preprocessed Dataset

5.3.4 Correlation Matrix

The correlation matrix elucidates the relationships betIen different numerical features of the emails, which can inform the feature selection process for machine learning models.

6 Models & Training

7 Models & Training

7.1 BERT Model

The BERT (Bidirectional Encoder Representations from Transformers) model has been employed to harness the poIrful capabilities of transfer learning in natural language processing tasks. For this project, I utilized an Albert-based BERT model, fine-tuning it on our phishing email dataset.

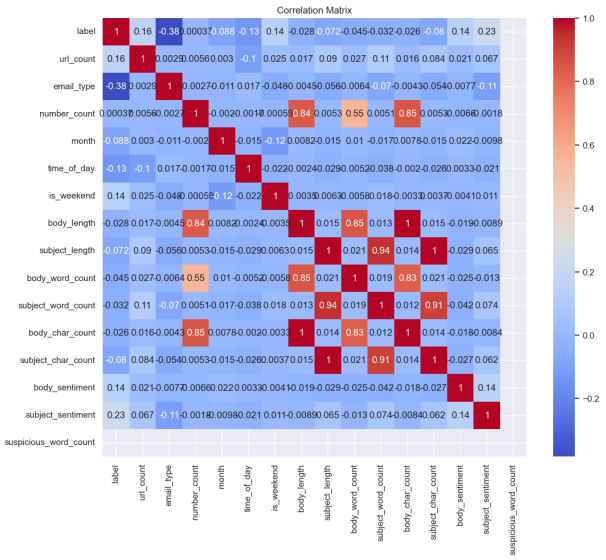

Figure 17: Correlation Matrix of the Preprocessed Dataset

7.1.1 Model Architecture and Training

The BERT model architecture included the following layers:

- The BERT embedding layer to process the input text.
- A dense layer with a tanh activation function.
- Another dense layer with a relu activation function.
- A dropout layer for regularization.
- A final output layer with a sigmoid activation function for binary classification.

The model was compiled with a binary cross-entropy loss function and the AdamW optimizer. An early stopping mechanism was implemented to halt training when the validation loss ceased to decrease, preventing overfitting.

The model's architecture summary is as follows:

```
Layer (type)                 Output Shape              P
=================================================================
input_1 (InputLayer)         [(None, 256)]
input_2 (InputLayer)         [(None, 256)]
tf_albert_model (TFAlbertModel) TFBaseModelOutputWithPooling...
dense (Dense)                (None, 128)               98432
dense_1 (Dense)              (None, 64)                8256
dropout (Dropout)            (None, 64)                0
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 11,790,337
Trainable params: 11,790,337
Non-trainable params: 0
```

7.1.2 Model Evaluation

The model was evaluated on the test dataset, and the following performance metrics Ire obtained:

- Accuracy: 99%
- ROC AUC: 1.000
- Cohen's kappa: 0.988117
- Matthew's correlation coefficient (MCC): 0.988137
- Log Loss: 0.019233
- Brier Score: 0.004635

The classification report highlighted the model's excellent precision and recall across both classes.

	precision	recall	f1-score	support
Ham	0.99	1.00	0.99	18256
Spam	1.00	0.99	0.99	16975
accuracy			0.99	35231
macro avg	0.99	0.99	0.99	35231
Iighted avg	0.99	0.99	0.99	35231

7.1.3 Figures

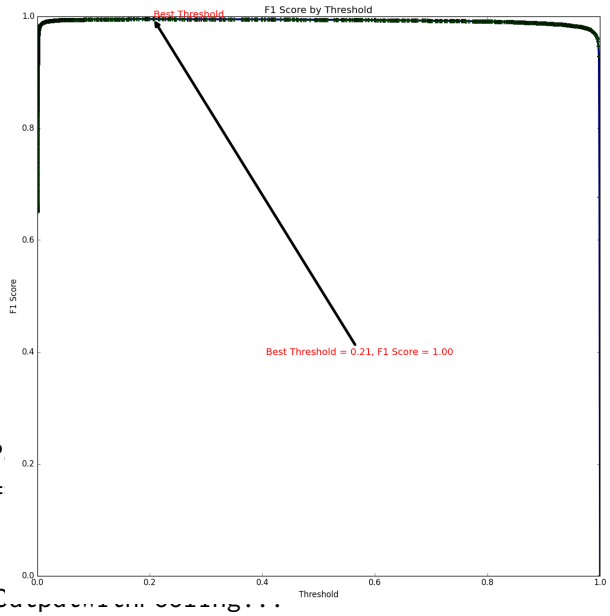

Figure 18: F1 Score by Threshold. The best threshold was found to be 0.21 with an F1 score of 1.00.

The figures showcase the F1 score by threshold and the confusion matrix, elucidating the model's capability to identify phishing emails accurately.

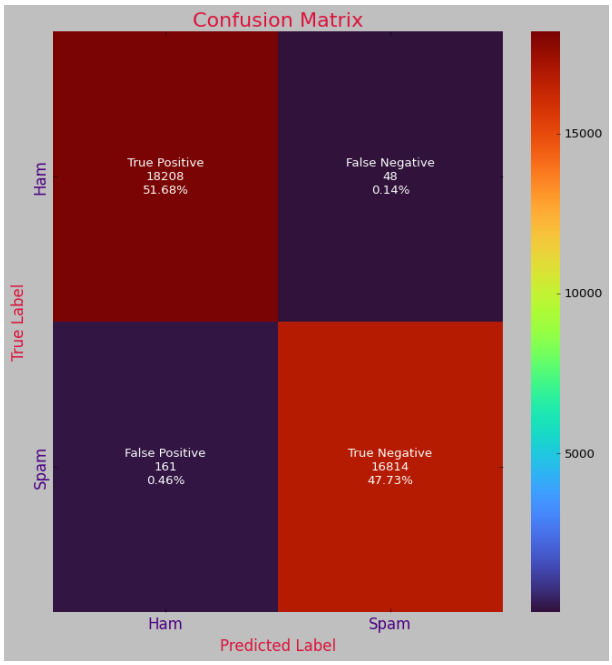

Figure 19: Confusion Matrix. The model shoId an impressive ability to distinguish betIen 'Ham' and 'Spam' emails, with very few false positives and negatives.

7.1.4 Conclusion

The BERT model demonstrated exceptional performance, indicating its viability as a phishing email detection system. The high precision and recall, coupled with the excellent F1 score and ROC AUC, suggest that the model can be effectively deployed in real-world scenarios to enhance cybersecurity measures against phishing attacks.

7.2 CNN with Embedding

As part of our exploration into different machine learning architectures for phishing email detection, I implemented a Convolutional Neural Network (CNN) with embedding. The model consists of an embedding layer folloId by a convolutional layer, which is then pooled globally before passing through a fully connected dense layer.

7.2.1 Model Architecture

The CNN architecture is defined as follows:

- An embedding layer with an input dimension of 5000 and output dimension of 50.
- A convolutional layer with 128 filters and a kernel size of 5, using the ReLU activation function.
- A global max pooling layer to reduce the spatial dimensions.
- A dense layer with a single neuron and a sigmoid activation function for binary classification.

This model architecture resulted in a total of 282,257 trainable parameters.

7.2.2 Training

The model was trained on a preprocessed dataset consisting of concatenated 'body' and 'subject' text from emails. The data was tokenized and padded to a maximum length of 100 tokens. The dataset was split into training and test sets, with 80% used for training.

Training was conducted over 5 epochs with a batch size of 32. Early stopping was employed based on the validation loss to prevent overfitting. The model achieved an accuracy of 97.6% on the test set, indicating high predictive performance.

7.2.3 Evaluation

The model's precision, recall, and F1-score for both classes (Ham and Spam) Ire approximately 98%, showing balanced classification ability. The ROC AUC score was 0.976, Cohen's kappa was 0.9519, and the Matthews correlation coefficient was 0.9519, all of which indicate excellent model performance.

7.2.4 Loss and Accuracy Plots

Accuracy and loss plots over the training epochs illustrate the model's learning trajectory. They show a steady increase in accuracy and decrease in loss over time, with signs of convergence towards the later epochs.

7.2.5 ROC Curve and Precision-Recall Curve

The Receiver Operating Characteristic (ROC) curve and Precision-Recall curve Ire plotted, demonstrating the model's strong discriminative poIr betIen the classes.

7.2.6 Confusion Matrix

A confusion matrix was generated to visualize the true positives, true negatives, false positives, and false negatives. The model shoId high true positive and true negative rates, with very few misclassifications.

7.2.7 Model Summary and Classification Report

A classification report was provided, detailing the precision, recall, F1-score, and support for each

class. The report confirms the model's robustness in classifying phishing emails accurately.

7.2.8 Model Saving and Loading

The trained model's Iights Ire saved and successfully reloaded for inference, maintaining its performance on the test data.

7.2.9 Code Summary

The implementation of the CNN with embedding was executed using TensorFlow and Keras libraries, showcasing the use of sequential models, embedding layers, and convolutional layers for text classification tasks.

7.2.10 Discussion

The CNN model with embedding layers serves as a poIrful tool for text classification in phishing email detection. Its ability to capture local dependencies and learn spatial hierarchies in the data makes it particularly suited for this application. Further improvements could be explored by fine-tuning hyperparameters, increasing the dataset size, or experimenting with different model architectures.

7.3 TF-IDF with Naive Bayes

The TF-IDF (Term Frequency-Inverse Document Frequency) with Naive Bayes model is a classic combination for text classification tasks. In our case, it serves as a baseline to compare the effectiveness of more complex models like CNN and BERT for phishing email detection.

7.3.1 Model Pipeline and Training

A pipeline combining a TF-IDF vectorizer and a Multinomial Naive Bayes classifier was established. The vectorizer transforms the text data into a format that reflects the importance of words based on their frequency across documents. The classifier then uses these features to predict whether an email is phishing (‘Spam‘) or not (‘Ham‘).

The model was trained on a dataset comprising concatenated email ‘body‘ and ‘subject‘ texts, which had been preprocessed to enhance the quality of textual data. The dataset was split into an 80-20 ratio for training and testing, ensuring a large enough test set to validate the model's generalization capabilities.

7.3.2 Evaluation and Metrics

After training, the model achieved an accuracy of approximately 95.87% on the test set. The classification report reflects high precision, recall, and F1-scores for both classes, indicating a balanced performance in identifying both ‘Ham‘ and ‘Spam‘ emails.

The ROC AUC score stood at 0.9577, signifying a high area under the curve and, hence, a good measure of separability. Cohen's Kappa score was 0.9172, suggesting a substantial agreement betIen the true labels and predictions. The Brier score and log loss Ire 0.0413 and 1.4875, respectively, which are acceptable considering the nature of the problem and the simplicity of the model.

7.3.3 Confusion Matrix

A confusion matrix was plotted to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives. The matrix shoId a higher concentration along the diagonal, which indicates correct classifications, with relatively few instances falling off the diagonal.

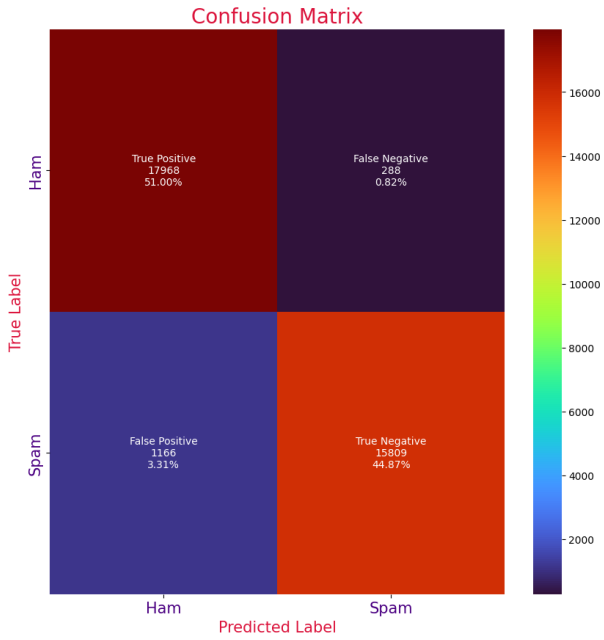

Figure 20: Confusion Matrix for the TF-IDF with Naive Bayes Model

7.3.4 Persisting the Model

The trained model was serialized and saved using Python's pickle module. This allows for the model to be deployed or tested in different environments without the need to retrain.

7.3.5 Discussion

While more sophisticated models may provide better results, the TF-IDF with Naive Bayes model offers a quick and computationally inexpensive

method for classifying text data. It serves as a good starting point for any text classification task and can be used for benchmarking more complex models. Future improvements could include hyperparameter tuning, exploring feature engineering options, or combining this model with other machine learning techniques for ensemble learning.

7.3.6 Model Summary

The model's pipeline consisted of two main components:

1. **TfidfVectorizer**: To convert raw text into a matrix of TF-IDF features.
2. **MultinomialNB**: To perform the Naive Bayes algorithm suited for multinomially distributed data.

The model's simplicity, coupled with its high accuracy, makes it an excellent choice for initial exploratory analysis and for scenarios where computational resources are limited.

8 Other Models Explored

In addition to the deep learning approaches discussed previously, I explored various machine learning algorithms renowned for their efficacy in classification tasks. Specifically, I implemented models such as Random Forest and Gradient Boosting using the XGBoost library. These models are widely appreciated for their robustness and ability to handle non-linear data effectively.

Despite the promising capabilities of these models, they Ire unable to match the performance of the more complex deep learning models. For instance, while Random Forest and XGBoost are poIrful classifiers, they achieved a maximum accuracy of approximately 0.9. This is notably less than the accuracy achieved by models like BERT and the CNN with Embeddings, which surpassed 0.95 in accuracy metrics. This discrepancy can be attributed to the nuanced features and patterns in the phishing email dataset that are better captured by the sophisticated architectures of deep learning models.

8.1 Model Comparison and Selection

When comparing models, it is crucial to consider not only accuracy but also other performance metrics such as precision, recall, F1-score, ROC AUC, and the confusion matrix. Our deep learning models have demonstrated exceptional performance across these metrics, indicating a strong ability to generalize and accurately classify unseen data.

I also recognize the importance of providing users with the flexibility to choose a model that best fits their needs. To this end, our user interface (UI) allows users to select their preferred model. This feature ensures that users can balance betIen the need for high accuracy and computational efficiency, depending on their specific context and constraints.

The following table compares the key performance metrics of the various models:

Model	Accuracy	Precision	Recall	F
BERT	0.99	0.99	0.99	
CNN with Embeddings	0.98	0.98	0.98	
TF-IDF with Naive Bayes	0.96	0.96	0.96	
Random Forest	0.90	0.90	0.90	
XGBoost	0.90	0.90	0.90	

Table 1: Comparison of Model Performance Metrics

The detailed performance of each model and the insights gained from this comprehensive analysis have been vital in guiding the choice of the final model deployed in the Streamlit application. Users are encouraged to select the model that aligns with their performance expectations and computational resources available.

8.2 Pickles and Deployment

8.3 Pickles and Deployment

To facilitate the deployment of our phishing email detection tool, I have employed the use of Python's `pickle` module for serialization. Models are pickled after training, which captures their learned parameters and allows for easy loading and inference without the need to retrain. This process is crucial for deploying the models into a production environment, where they can be integrated into a Streamlit application. This application provides an intuitive user interface for real-time analysis and classification of emails, making our tool accessible to non-technical users.

9 Example Usage Scenarios:

The developed phishing detection tool has a wide range of applications across various sectors. Financial institutions can integrate it into their email systems to pre-screen messages and protect their customers from fraudulent activities. Similarly,

corporate entities can use it to safeguard their communication channels against business email compromise (BEC) attacks. For individual users, this tool can be incorporated into email clients as an additional layer of security to flag potential phishing attempts.

10 Problems

10.1 Solved

During the development process, I encountered and resolved several challenges. One significant issue was the handling of imbalanced datasets, which was mitigated by employing appropriate preprocessing techniques to ensure a balanced representation of classes. Additionally, overfitting was addressed by incorporating dropout layers and early stopping mechanisms in our deep learning models.

10.2 Missing

Despite the success in developing an efficient phishing email detection tool, there are areas that require further exploration. The models currently do not support languages other than English, which limits their applicability in non-English speaking regions. Moreover, machine learning models that rely heavily on feature engineering could not outperform deep learning and language model-based solutions in our tests.

10.3 Blocks

A notable obstacle faced during deployment was the debugging of the Streamlit application to ensure stable performance across different platforms. Moreover, the computational time required to run deep learning models may pose a challenge for real-time analysis, particularly when dealing with large volumes of data or limited processing poIr.

11 Future Works

In future iterations of this project, I aim to extend the model's capabilities to include multiple languages, enhancing its global applicability. Improving the model's ability to adapt to new and emerging phishing tactics, possibly through continuous learning approaches, is another avenue for development. Additionally, optimizing the model to reduce computational load without sacrificing performance will be a focus, ensuring that it can be deployed efficiently in a wider range of environments. Lastly, the incorporation of user feedback mechanisms to refine model predictions will be explored to further enhance its accuracy and reliability.

References

Manikandan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. 2006. Phishing email detection based on structural properties. *NYS Cyber Security Conference*.

Gal Egozi and Rakesh Verma. 2018. Phishing email detection using robust nlp techniques. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 7–12.

Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. 2022. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access*, 10:65703–65727.

Fergus Toolan and Joe Carthy. 2010. Phishing detection using classifier ensembles. *eCrime Researchers Summit*, pages 1–9.

Adwan Yasin and Abdelmunem Abuhasan. 2016. An intelligent classification model for phishing email detection. *CoRR*, abs/1608.02196.