

Enhancing Phishing Email Detection with Sentiment Analysis: A Hybrid Approach

MSc Research Project
Cybersecurity

Saiprasad Salian
Student ID: x22183761

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Saiprasad Salian
Student ID:	x22183761
Programme:	Cybersecurity
Year:	2024
Module:	MSc Research Project
Supervisor:	Michael Prior
Submission Due Date:	12/08/2024
Project Title:	Enhancing Phishing Email Detection with Sentiment Analysis: A Hybrid Approach
Word Count:	5663
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sai Salian
Date:	11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Phishing Email Detection with Sentiment Analysis: A Hybrid Approach

Saiprasad Salian
x22183761

Abstract

In this digital era, people are surrounded by technology and the internet usage has skyrocketed. This means that there's more data available on the internet about a particular person than they can dream of. Cybercriminals use these data to launch attacks via emails and try to steal their sensitive information. Phishing is a very common type of attack used by these criminals and they try to attack large organizations in order to obtain ransom from them or to deal with their information. These attacks happening inside an organization through an organizational email has the potential to lead losses in billions. To detect such malicious attempt, this paper proposes a phishing email detection system that can analyse the sentiment behind the email and improve the level of accuracy. We use the DistilBERT model for sentiment analysis and then feed the sentiment aware embeddings to a SVM model for further classification. This proposed study provides steps on how to create a system that can not only check the regular heuristic features and key filters but also the sentiment behind the email which will help improve the overall security.

Keywords – *Phishing emails, DistilBERT, SVM, Sentiment Analysis, Phishing Detection*

1 Introduction

With the fast rise of global internet usage, people are sharing their personal information more online. But this has also made them more vulnerable to cybercrime. Phishing is one type of threat that stands out and it involves creating false emails which look real in order to deceive receivers into revealing sensitive details about themselves. Phishing email attacks have now become a big problem worldwide because they cause severe financial and data losses for many organizations and individuals.

As per a report from IT Governance EU, the cases of phishing have grown greatly in recent years. In April 2021 alone, around 200,000 phishing attacks were recorded by the Anti-Phishing Working Group (APWG). By March 2022 this number had nearly doubled to reach 384,291 cases Irwin (2022). The main focus of attacks is still on the financial sector, such as banks and insurance companies. This area experiences 23.6% of all incidents that happen Irwin (2022). The monetary damage is very high with Venari Security's estimate saying every piece of personal information that gets stolen costs around \$181 (£150) for companies Irwin (2022). Furthermore, IBM stated a rise in the normal expense for data breaches from \$4.24 million (£3.42 million) during 2021 to \$4.35 million (£3.51 million) in year 2022 Irwin (2023).

Last year, phishing attacks were at their highest point. The Anti Phishing Working Group (APWG) recorded nearly five million attempts in 2023. In the final quarter alone, more than one million assaults happened - social media platforms got targeted especially with 42.8% of all phishing attempts focusing on them specifically; this shows how much danger we face from ever-increasing and complex Business Email Compromise (BEC) attacks *APWG — Phishing Activity Trends Reports* (2023).

Advanced detection tools are crucial to fight against the rising danger of phishing emails. This project investigates the combination of sentiment analysis with Support Vector Machine (SVM) models for improved phishing email identification. Sentiment analysis, which evaluates the emotional tone expressed through text, has potential to detect patterns like manipulation and urgency that are frequently seen in phishing emails. The aim of the project is to improve the precision of detecting phishing emails by introducing DistilBERT for sentiment analysis and SVM algorithm together. The objective is to assess the effectiveness of this combination in classifying phishing emails when compared to regular techniques. We are also trying to improve the detection of ever-changing phishing tactics.

Research Question: How can integrating sentiment analysis using DistilBERT with traditional machine learning techniques such as SVM enhance the accuracy and effectiveness of phishing email detection systems?

The paper starts off with an extensive literature review which will help in understanding the previous works done in the field and the need for new techniques to challenge the ever-growing tactics of phishing emails. After the literature review the next section will be Methodology and Design Specification which will explain the thought process, steps and requirements needed to cultivate the expected results. Post that will be the Implementation and Evaluation section wherein we discuss what were major implementations blocks requires and on what basis we evaluated our results. The last part comprises of the Conclusion and Future works where we conclude what we have attained from this research and give future directions for the researchers in the same field.

2 Related Work

Detecting phishing emails is a major part of cybersecurity, which has the purpose to inform users to not share their sensitive information with anyone. Most traditional email phishing detection attempts to find patterns or perform statistical analysis using machine learning models based on a breadth of features that an email contains. While these detection systems could be greatly improved if they incorporated sentiment analysis, which could recognize the emotional manipulation so common in phishing emails. The last couple of years we have seen a lot of novelties in the field of machine learning and natural language processing (NLP), e.g. more efficient classifiers like Support Vector Machine all the way up to transformer-based models such as DistilBERT which show significant improvements over traditional measures for phishing detection.

2.1 Machine Learning Powered Phishing Detection

The paper proposed by Ma et al. (2009) brings together several machine learning (ML) methods aimed at detecting phishing emails. The authors tried out five different ML techniques to figure out which one works best for this task: Decision Tree, Random Forest, Multi-layer Perceptron, Naïve Bayes, & Support Vector Machine (SVM). They

discovered that the Random Forest algorithm gave the best results when compared to the other methods. Still, the researchers could've looked into deep learning approaches to check if those would perform better in spotting phishing emails. This study is a bit limited since it only used a small part of possible features. Adding more features could really help improve detection & classification. Future efforts are geared toward making the normalization process better by cutting down huge values & finding a threshold that can get rid of noisy data. The authors also aim to create a stable, automatic filter for detecting phishing emails with little supervision required. Plus, they plan to set up an automatic feature update mechanism that adapts the classifier as needed.

Dey et al. (2021) in their paper developed a model for detecting phishing emails and website to analyse different machine learning algorithms. The focus of this paper was about discussing and comparing various ML algorithms to verify the most efficient one for phishing detection. Random Forest, Decision Tree, Logistic Regression and Naïve Bayes were the four machine learning algorithm used by the authors. Author's goal was to create a model that can precisely detect phishing email and websites. Identifying spam and legitimate messages was done using Naïve Bayes and Multinomial Naïve Byes was used for classifying phishing emails. Accuracy of the Decision Tree algorithm drops when using for new data classification as it works in single dimension. As per the results, among all the algorithms Random Forest came out to be the least accurate in detecting phishing emails and website.

Patil et al. (2017) did some research on classifying spam & phishing emails. The authors used Support Vector Machine (SVM) and obfuscation URL detection algorithm to do the same. With the help of this combination, they were also able to check out the CSS components to match the similarity. This study helps in understanding how if we combine SVM with MapReduce framework it gives us much greater efficiency in finding out spam emails. obURL method was used to figure the URL encoding which made phishing email detection easier. They also performed various test like checking DNS records, white lists, URL encoding, and so on to improve the accuracy. One of the advantages of using SVM over other algorithms is that it can handle huge datasets better. Limitations of SVM include that it can have a difficult time when the inputs are large and to overcome this obstacle the authors used MapReduce, with the help of MapReduce the SVM can handle larger inputs easily.

The paper "Phishing Detection Using Machine Learning Technique" focuses on analysing phishing sites using various machine learning model. The authors used multiple ML algorithms such as Decision Tree, Random Forest, and SVM to figure out which one of these works the best in identifying fake and legitimate websites. Random Forest turned out to be the most efficient one with great accuracy levels as compared to others. The dataset used for the purpose of this research was not as broad as it could be which means it's doesn't cover the complete spectrum of phishing tactics used. Using more diverse datasets and combining different technologies to boost up the performance would be worth looking into Rashid et al. (2020).

A hybrid model created with the combination of Random Forest, SVM and neural network with backpropagation was used for detecting phishing by Sindhu et al. (2020). The researchers extracted lexical features from URLs and used neural network and SVM for classification between legitimate and fake websites to figure out which of these algorithms works best. As a result of the experiment, SVM outclassed the other algorithms and was integrated as a Chrome extension which was then used to alert users when they launch a suspicious website. To make the model even more robust the authors used web min-

ing techniques such as focusing on identifying IP addresses, URL length, sub-domain registration length, the presence of @symbol, requests within the URL were some of the parameters used in the feature extraction process.

Cuckoo Search SVM is a novel method for detecting phishing emails presented by Niu et al. (2017). What this classifier does is it extracts 23 features from the emails which include body-based, URL-based, and header-based features. The combination of Cuckoo Search and SVM comes together to create a model which helps in optimizing the Radial Basis Function (RBF). The Authors tested this model on a dataset that contains both old and new phishing emails, and upon testing they came out with the results such that the CS-SVM outperformed the standard SVM with default RBF parameter. For Future work the authors suggest optimizing the CS-SVM even further and to run it on a distributed platform better performance.

The authors Ripa et al. (2021) explore detection techniques using machine learning to deal with the emerging threats of phishing attacks. A spear phishing bot made by using ML was developed by the researchers. Their model focuses on detecting phishing through URLs, emails, and websites. XGBoost, KNN, Naïve Bayes, Decision Tree, Logistic Regression, Gradient Boosting, and Random Forest were the different classifiers the authors used. Random Forest proved to be the best among other classifiers with minimal time consumption. Authors give a direction to future work where in we can develop a framework that can improve the accuracy while also maintaining efficiency.

2.2 Application of Sentiment Analysis

Wankhade et al. (2022) in their paper provide an in-depth survey of sentiment analysis methods, applications and challenges faced. The researchers segregated the techniques used as lexicon-based and supervised learning. Supervised methods turned out to be more accurate while lexicon-based methods are better suited for unlabelled data. The authors conclude that a hybrid method which utilize machine learning, and lexicon-based techniques could prove to be more efficient in sentiment classification. However, there are some limitations observed by the researchers which include handling of unstructured and noisy data and context-dependent nature of sentiment.

Alaparthi and Mishra (2020) in their paper “BERT: A sentiment analysis odyssey” try to find out the effectiveness of four different sentiment analysis techniques, namely an unsupervised lexicon-based model using Sent WordNet, a traditional supervised machine learning model using logistic regression, a supervised deep learning model using Long Short-Term Memory (LSTM), and advanced supervised deep learning models using Bidirectional Encoder Representations from Transformers BERT. The researchers used a dataset containing 50,000 movie reviews from the IMDB, evaluation of these techniques is done on the basis of accuracy, recall, precision, and F1 score. Apart from BERT all other techniques were run on CPU based systems. Authors also highlight the need for proper preprocessing in sentiment analysis and included detailing steps for removing HTML tags, expanding contracted words, and lemmatization. As per the results BERT proved out to be potential in providing high quality sentiment classification, which is a valuable insight for researchers in text analysis. This paper also stands as the first comparative evaluation of BERT against other sentiment analysis models out there, proving its efficiency in handling text data.

The paper “Boosting the Phishing Detection Performance by Semantic Analysis” by Zhang et al. (2017) present a novel approach to improve phishing detection using

semantic analysis. The researchers also shed light on the limitations of present phishing detection techniques, which usually focuses on visual similarity and statistical features. They also introduced the use of the word embeddings through word2vec to capture the semantic information of web pages. With the help of multi scale statistical features and the semantic features the authors developed a more rigid phishing detection model. The combination of semantic and statistical features greatly improves the accuracy as per the obtained results. The primary concern of the researchers which is addressed in the paper is detecting phishing sites that mimic the content of real sites by focusing on textual and contextual similarities. The authors have contributed to the literature by presenting a detailed evaluation of the chosen approach on real world datasets, which showed its effectiveness in detecting phishing websites. This research presses the importance of utilizing both statistical and semantic aspects in creating an advanced tool for phishing detection.

3 Methodology

The proposed methodology in this paper provides steps for developing a model to detect phishing emails using DistilBERT for feature extraction and Support Vector Machine (SVM) for classification. This process starts with data preprocessing, moving to feature extraction then model training, and testing.

3.1 Dataset Description

The dataset used in this project is downloaded from Kaggle, specifically from the “Phishing Email Dataset” provided by Naser Abdullah Alam. A group of researchers put together this dataset to investigate tactics used in phishing emails. They gathered emails from different places, making a broad collection for study. The main dataset (Final Dataset) used is a combination of different initial datasets.

3.1.1 Initial Datasets-

- Enron and Ling Datasets: These datasets are concentrated on the main part of phishing emails, which is the content. They include subject lines, text in the email body and labels that show if an email is spam (phishing) or real.
- CEAS, Nazario, Nigerian Fraud and SpamAssassin Datasets: These datasets can give more context about the emails. They may have details such as sender information, receiver information, when it was sent and labels for spam or legitimate classification.

3.1.2 Final Dataset-

- The last dataset is a merge of the initial datasets mentioned above. It contains approximately 82,500 emails with 42,891 of them being spam emails and 39,595 of them being legitimate emails.

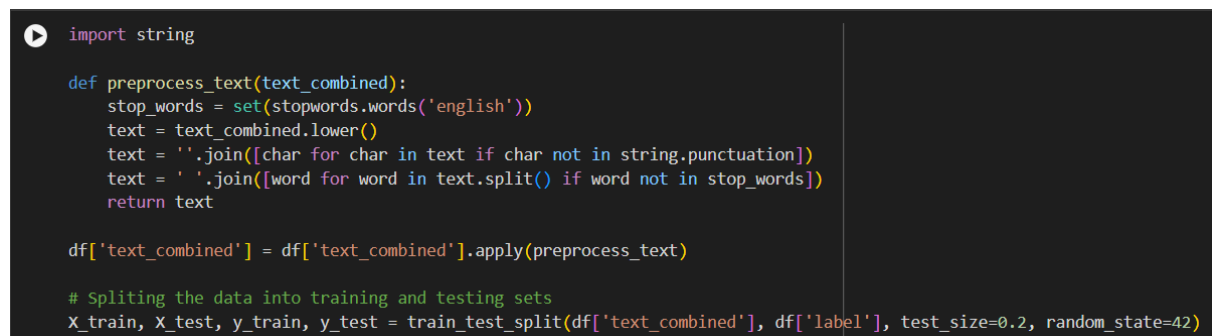
The dataset consists of important parameters such as the sender and receiver information, subject line, text in email body and any URLs which are crucial for sentiment analysis. For data splitting, the dataset is separated into 80-20 ratio.

3.2 Data Preprocessing

Data preprocessing is very important for cleaning and getting the data ready to train models. It has many stages, making sure that the information is in best format for feature extraction and later classification. At first, we load dataset into pandas DataFrame that helps with simple handling and examination of the data. Sometimes, the plain email texts may include useless components like punctuation marks, capital letters and stopwords which don't help with identifying important patterns needed for classification.

For handling this, the text data goes through a sequence of cleaning actions. At first, all text is turned into lowercase to keep consistency and remove the variation brought by case differences. Then, it takes out punctuation from the texts to decrease noise and concentrate on actual content of emails. Furthermore, we eliminate stopwords. These are usual words that don't have much meaning in the text (like 'and', 'the', 'in' etc.). We use Natural Language Toolkit (nltk) to do this. It is an important stage because it lessens the size of text data and increases model efficiency and usefulness by concentrating on more informative words.

After all the cleaning, we divide dataset into training and testing sets with a split ratio of 80-20. It means that 80% are for training the model while other 20% is saved to test how well it works. This division makes sure our model gets trained on most part of data but also gives us a good way to check its performance on new unseen data. The function used for this is 'train_test_split' from the module 'sklearn.model_selection'. This makes sure that data splits randomly and without any bias into training and testing groups. This careful preprocessing action makes certain the information provided to the DistilBERT model is neat, consistent, and without useless details.



```
import string

def preprocess_text(text_combined):
    stop_words = set(stopwords.words('english'))
    text = text_combined.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text

df['text_combined'] = df['text_combined'].apply(preprocess_text)

# Splitting the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(df['text_combined'], df['label'], test_size=0.2, random_state=42)
```

Figure 1: Snippet of data preprocessing

3.3 Feature Extraction Using DistilBERT

This step involves using DistilBERT which is a pre-trained transformer model to convert the raw text in the dataset into meaningful numerical embeddings which can then be used for classification. DistilBERT is a cheap, small, fast and light transformer model trained by distilling BERT base. It utilizes 40% less parameters, is 60% more faster and also preserves over 95% of BERT's performance. Since it is the distilled version of BERT it means that it has been trained to predict the same probabilities as the bigger model as presented by the author Sanh et al. (2020).

3.3.1 Loading the Pre-trained Model and Tokenizer

We start off this step by loading the DistilBERT model and the tokenizer required. As the model and tokenizer were pre trained and saved in the drive location we call them from the drive itself. Now, the DistilBERT model doesn't take in raw text data as inputs, so we need to turn the data into something acceptable for the model. Here is where the tokenizer comes into the picture, the process of tokenization is to simply convert the raw text into tokens which are then encoded into numbered format that we can feed to the DistilBERT for further processing. Once the tokenized data is processed by the DistilBERT it provides us with the contextual embedding which are in the form of vectors. These embeddings are then used as the input for SVM classifier.

```
[ ] # Load the saved tokenizer and DistilBERT model
tokenizer = DistilBertTokenizer.from_pretrained('/content/drive/My Drive/Thesis - Phishing Detection/Models/tokenizer')
model = DistilBertModel.from_pretrained('/content/drive/My Drive/Thesis - Phishing Detection/Models/distilbert_model')
```

Figure 2: Loading the model and tokenizer

3.3.2 Preparing the Data for the Model

As discussed earlier, DistilBERT model doesn't take text data as input and hence it needs to be tokenized and converted into tensors. The model has an input size limit of typically 512 tokens, so we must truncate the text to fit the limit and add padding. After the tokenization process is completed, we get tensors which include input IDs and attention masks. Encoded tokens are represented by Input IDs whereas the attention masks represent the real and padding tokens.

```
def get_embeddings(texts, tokenizer, model, device, batch_size=32):
    """Function to get DistilBERT embeddings for a list of texts."""
    embeddings = []
    model.eval()

    with torch.no_grad():
        for i in range(0, len(texts), batch_size):
            batch_texts = texts[i:i + batch_size]
            inputs = tokenizer(batch_texts, return_tensors='pt', truncation=True, padding=True, max_length=512).to(device)
            outputs = model(**inputs)
            batch_embeddings = outputs.last_hidden_state.mean(dim=1).cpu().numpy()
            embeddings.extend(batch_embeddings)

    return np.array(embeddings)
```

Figure 3: Preparing the data for model

3.3.3 Extracting Embeddings

At the core of feature extraction, we rely on the DistilBERT model to create embeddings from tokenized text data. These embeddings are dense vectors capturing the semantic meaning of the text at hand. In this project, we group embeddings to boost performance & save memory. For each group of text, we break them down into tokens, convert them to tensors, and then apply the DistilBERT model to access those final hidden states. We combine these hidden states by averaging the token embeddings. What we end up with is a fixed-size vector representation for each email.

```
# Get embeddings for training and testing sets
X_train_embeddings = get_embeddings(X_train.tolist(), tokenizer, model, device)
X_test_embeddings = get_embeddings(X_test.tolist(), tokenizer, model, device)
```

Figure 4: Snippet of embeddings code

The aggregation method used here involves calculating the mean of the last hidden states. This approach assists in capturing the overall context of the text. It simplifies the complexity in the model's outputs while providing a compressed format for each email, which can be beneficial for future classification tasks.

```
# Verify shapes
print(X_test_embeddings[0])
print(f'Training embeddings shape: {X_train_embeddings.shape}')
print(f'Test embeddings shape: {X_test_embeddings.shape}')
```

```
[ -2.20407471e-01  2.45572925e-01  3.05758476e-01 -2.80571193e-01
  1.10120237e-01 -1.23687595e-01  3.74695897e-01 -2.59085577e-02
  6.92025945e-02 -6.24933392e-02 -8.96160826e-02 -1.52642101e-01
 -7.14655519e-02  1.26403242e-01 -2.08766609e-02  3.02949488e-01
 -1.92975793e-02 -6.41667172e-02 -1.60487831e-01  4.62504983e-01
  4.57487226e-01 -1.96409374e-02  1.26321152e-01  4.04556155e-01
  1.61134601e-01 -2.56515667e-02  1.57411993e-01 -1.73981220e-01
  8.39695185e-02  2.64994297e-02  1.37280151e-01  4.41025309e-02
  1.65634900e-01 -3.27298582e-01  2.24182248e-01 -1.13348007e-01
  2.42888749e-01 -1.11162826e-01  4.18361127e-02  8.60983282e-02
 -2.69794822e-01 -3.73471141e-01 -6.80335909e-02  2.79589951e-01]
```

Figure 5: Snipped of embedding output from the DistilBERT

3.3.4 Handling the Embeddings

Usually, the resultant embeddings are vectors with high dimensions. These embeddings are moulded into a two-dimensional array where each row corresponds to the vector representation of an email in order to ensure compliance with the SVM classifier. In order to feed the embeddings into the SVM model for training and prediction, this step converts them into a format.

```
X_train_embeddings = X_train_embeddings.reshape(X_train_embeddings.shape[0], -1)
X_test_embeddings = X_test_embeddings.reshape(X_test_embeddings.shape[0], -1)
```

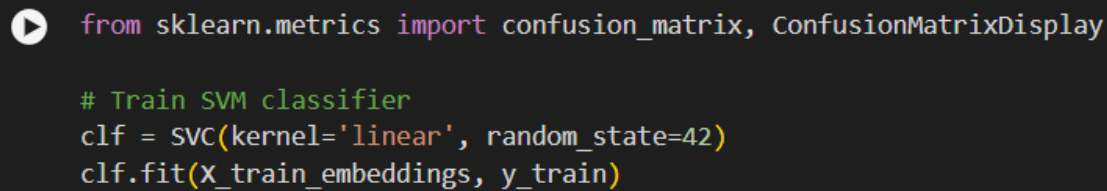
Figure 6: Converting to for the SVM model

The embeddings from the DistilBERT model have high contextually aware features which helps capture the nuances of the email content. Then these embeddings are given as input to the SVM classifier which is then used to figure out between legitimate and phishing emails.

3.4 Model Training

Model training can be divided into 2 phases, the first phase being where we extract features from the raw text data. We are using DistilBERT for the purpose of extracting features in the form of embeddings. These embeddings produced by the model are contextual representations of the text data. Since SVM need the input in a fixed length numerical vector form, the embeddings are then converted into vector form as discussed under section 3.3.4. We used SVM as our classification algorithm because it performs well with the embeddings as compared to other algorithms.

SVM model is trained using the embeddings obtained along with the labels. The algorithm can identify the hyperplane that separates the data points belonging to phishing and non-phishing emails. As it requires high dimension of feature space, we used kernel for the SVM which helps us to maintain efficiency and makes sure that the model can resist against overfitting.



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Train SVM classifier
clf = SVC(kernel='linear', random_state=42)
clf.fit(x_train_embeddings, y_train)
```

Figure 7: SVM model training

The SVM model is then used to predict labels for the test set consisting of 20% of the dataset. It assigns a label to each text based on the embeddings it receives. The process is carefully structured to evaluate how well the model performs on data which is essential for gauging its ability to generalize effectively.

3.5 Evaluation

The trained model is then tested against the test set and can evaluate its predictions using respect to actual labels. The analysis compared with various performance metrics such as precision, recall, F1-score and accuracy. These measures together suggest how well the model has done in recognizing phishing e-mail.

We calculate the performance metrics using a scikit-learn function called `classification_report`: it provides detailed information and gets computed both for each class label separately, as well as overall. Action- This evaluation is to have a clear understanding of how good SVM model explores and it meets the criteria for an efficient phishing detection as well.

The trained and tested model is then saved. The trained SVM model is then stored using 'joblib' that helps to load it as it earlier and we do not need to retrain. It is the step which takes place in real world deployment where we use our model to predict on new data.

4 Design Specification

4.1 Overview

The design specification outlines the architecture and techniques employed in the implementation of a phishing email detection system. The system leverages the DistilBERT model for feature extraction and Support Vector Machine (SVM) for classification.

4.2 Architecture

The architecture of the phishing detection system is composed of 4 key phases:

4.2.1 Data Collection and Preprocessing:

- **Source:** We used the dataset from Kaggle named ‘Phishing Email Dataset’ for the training and testing of the model. The dataset had approximately 82500 emails and it comprises of 2 columns one text_combined and the other label which indicate if the email is real or phishing.
- **Preprocessing:** Before starting with anything we first need to preprocess the text data to remove punctuation, convert the text into lowercase and eliminate stopwords. After this the data is considered to be clean, but it still can’t be used as input for the DistilBERT model. We then tokenize the data in order to feed it to the DistilBERT model.

4.2.2 Feature Extraction:

- **Model:** For the purpose of feature extraction in our project we have used the DistilBERT model which is a smaller and faster version of BERT. We obtain meaningful embeddings from the model as output. As discussed earlier in section 3.3. this model can retain 95% of BERT’s prediction accuracy all while being more efficient due to being a distilled version.
- **Process:** The tokenized raw text data from the preprocessing step is then fed to the model in order for it to produce embeddings which are in the form of numerical representations. These embeddings are able to capture a high level of contextual features and semantic nuances of the text which will help in improving classification accuracy. These embeddings obtained are then converted into vectors for the SVM to work on.

4.2.3 Classification:

- **Model:** We are using Support Vector Machine (SVM) with a linear kernel for the task of classification. Using the linear kernel helps maintain the efficiency. The purpose of using this model is because it has the higher probability of working well with the embeddings that we get from the DistilBERT model.
- **Training:** We used the feature vectors produced by the DistilBERT with the respective labels to train the SVM model. SVM model then tries to find the hyperplane that differentiates between real and phishing emails. Here we are using 80% of the total dataset to train the model.

4.2.4 Evaluation:

- **Metrics:** Metrics such as precision, recall, accuracy and f1-scores are used to evaluate the model's performance. Along with this the confusion matrix is also studied to understand the performance in different classes.
- **Validation:** We also validated the results by training and testing the SVM model individually without the DistilBERT on the same dataset. The individual SVM model produced an accuracy of 94% which is less than what we are getting when we use SVM along with the DistilBERT model.

4.3 Technical Requirements

- **Software Requirements:**

- Python 3
- Google Colab
- Libraries: Transformers, scikit-learn, pandas, NumPy, torch, nltk, joblib, sklearn
- Data Storage: Google Drive (for storing datasets and models)

- **Hardware Requirements:**

- CPU and GPU for training (GPU for faster training the DistilBERT)
- 14 GB of System RAM and 16 GB of GPU RAM to handle large datasets and model computations

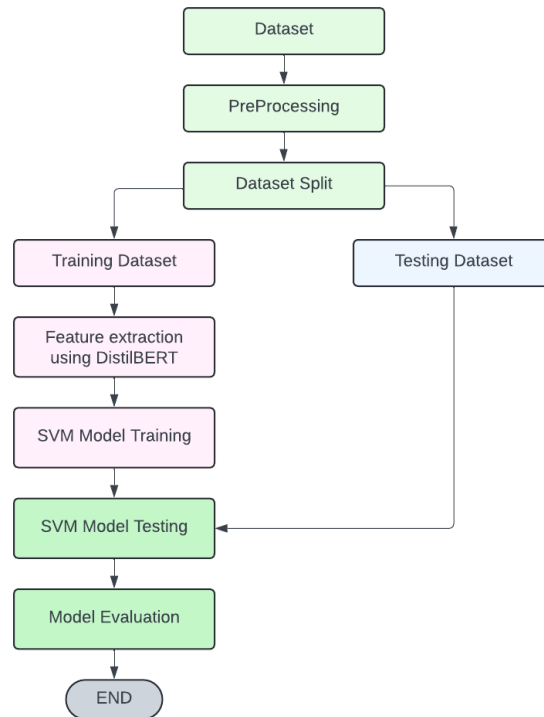


Figure 8: Model Design

5 Implementation

The project makes use of a transformer-based model, mainly DistilBERT, for grabbing features from text data. Then a machine learning classifier steps in to phishing emails. The whole process has a few key stages: preparing the data, extracting features with DistilBERT, & training and evaluating the model. Each step is organized carefully to make sure the final model is spot on & runs efficiently when detecting phishing attempts

5.1 Preparation of Phishing Email Dataset

First up, we get the phishing email dataset from Kaggle and preprocess it to make sure it's ready for extracting features & classifying. This dataset has emails tagged as either phishing or legit. To start, we clean it up by getting rid of any stuff that doesn't matter, like duplicates, null values, or anything that's not text. After that, we tokenize the text and keep it consistent like turning all the texts into lowercase and removing special characters. Next, we split the dataset into training & testing sets using the 'train_test_split' function from 'sklearn.model.selection'. We go with an 80:20 ratio so that we train on a good chunk of data but still have enough left for testing how well it performs.

5.2 Feature Extraction Using DistilBERT

Now we start with extracting features using DistilBERT. This model helps change text data into numerical embeddings that are good for classification tasks. DistilBERT is like a lighter, faster version of BERT that gives us powerful text embeddings. These capture what words mean in context within emails. Each email gets pushed through the DistilBERT model, and we pull out the hidden state's of the [CLS] token, which sums up the whole sentence as feature vectors. We store these embeddings to use them later in our machine learning classifier.

5.3 Training the Classifier and Evaluation

When we're done with feature extraction, we can jump into training our machine learning classifier. For this project, we're using a Support Vector Machine classifier because it's strong & does a great job with complex embeddings. So during training, we use that training set where our DistilBERT embeddings are seen as input features while email labels (phishing or legitimate) are our target variable. The training involves adjusting hyperparameters to boost performance & keep from overfitting. After the model is trained and tested we checked the accuracy, precision, recall and F1-score of the model to see how effectively its detecting phishing emails.

Once we verified the efficiency of our model we saved the model in the drive for future use. By doing so we can implement the already trained model for real life scenarios where it can identify phishing emails. Libraries like 'transformers' for DistilBERT, 'scikit-learn' for machine learning, and 'pandas' for data handling and easier manipulation were used.

6 Evaluation

Evaluating the performance of the phishing email detection model is very important to make sure it can accurately tell apart between real and fake emails. In this evaluation, we look at a number of main measurements: Accuracy, Precision, Recall and F1-score. We calculate these using the ‘sklearn.metrics’ library which gives us full understanding about how well our model is doing.

1. Accuracy:

Accuracy is the size of correctly classified emails (including both phishing and legitimate) from all emails. It is a basic measure that shows the model’s performance on whole dataset. The formula for calculating accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

2. Precision:

Precision is the standard of correctly anticipated phishing emails divided by the total number of emails forecasted as phishing. If precision is high, it suggests that there are not many false positives - in other words, the model does well at reducing instances where legitimate emails are wrongly marked as being fake.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3. Recall:

Recall, also called sensitivity or true positive rate, is about how well the model can identify phishing emails among all real phishing emails. Having a high recall means that most of the phishing emails are correctly detected.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4. F1-Score:

The F1-score is the harmonic average of Precision and Recall, which gives us a single number that considers both aspects. It comes in handy when there is an uneven distribution among classes or if dealing with either false positives or false negatives holds high cost implications.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the evaluation, we also created the confusion matrix to show how well the model performed in different classes. The confusion matrix comprises of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) which helps us if there is any kind of error our model is making.

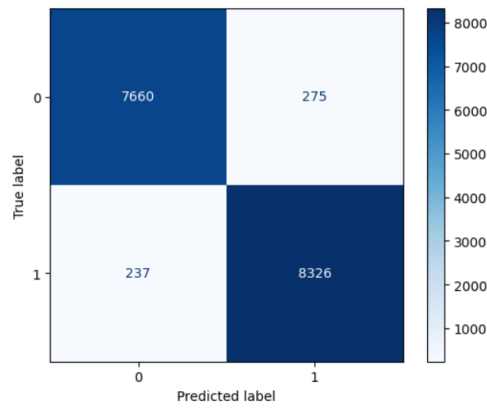


Figure 9: Confusion Matrix

	precision	recall	f1-score	support
0	0.97	0.97	0.97	7935
1	0.97	0.97	0.97	8563
accuracy			0.97	16498
macro avg	0.97	0.97	0.97	16498
weighted avg	0.97	0.97	0.97	16498

Figure 10: Accuracy from the Hybrid Model

The results of this project show that the phishing email detection model has 97% accuracy, with a precision and recall also at 97%, which results in the F1-score being 97%. As per the results we've obtained we can say that the model is highly efficient in detecting phishing emails.

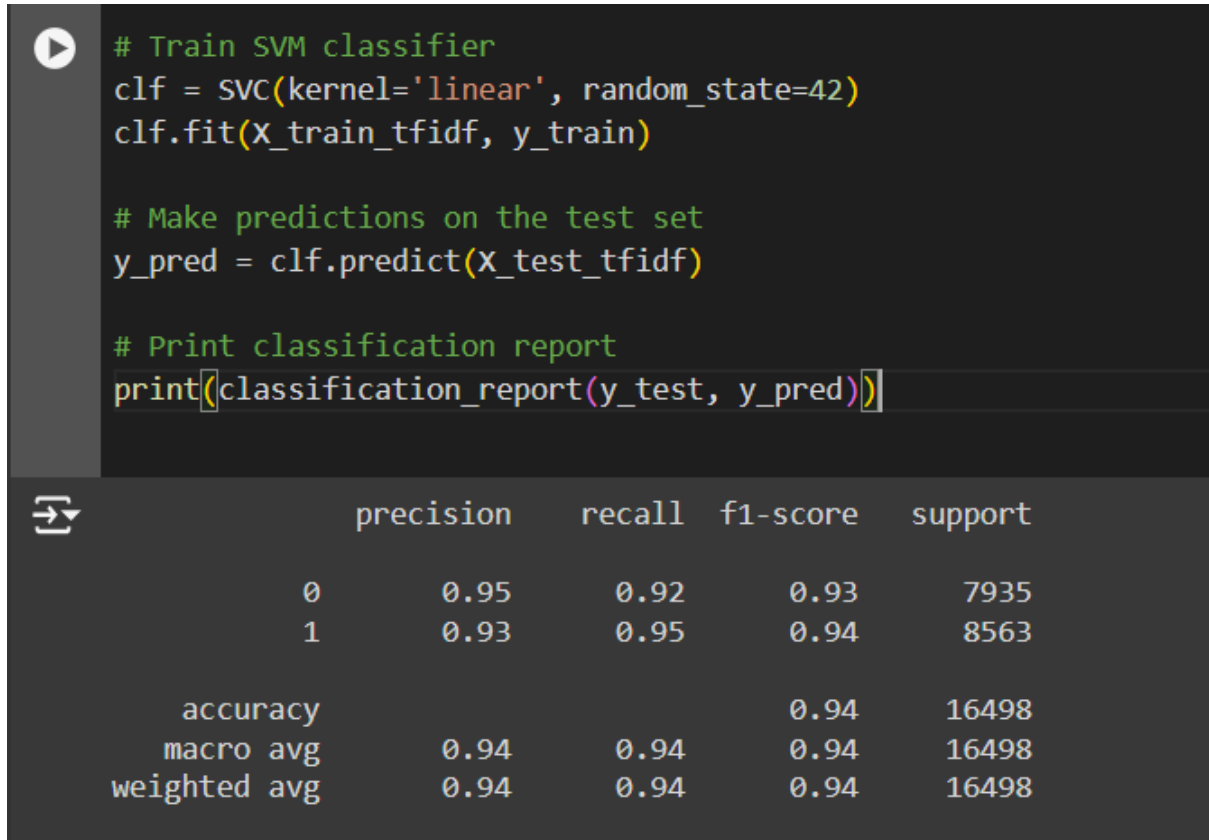


Figure 11: Accuracy from the Hybrid Model

Apart from looking at the metrics and studying the confusion matrix we also validated our model against a individual SVM model. We wrote a program using the same dataset to detect phishing emails but this time instead of using both DistilBERT and SVM together we just used SVM. By doing so we were able to understand if there is any actual improvement from using sentiment analysis with regular algorithms. When using just the SVM model for the dataset we received a accuracy of 94%, precision of 93%, recall of 92% and f1-score of 93%. This shows that our approach of using sentiment analysis with SVM has helped us improve the overall performance of the system.

6.1 Discussion

The primary goal of this paper is to improve the detection of phishing emails by combining sentiment analysis using DistilBERT and Support Vector Machines (SVM). We planned to determine how well this hybrid technique works in detecting phishing emails.

Our test showed that combining the sentiment analysis-based features with SVM significantly increased the accuracy of phishing email detection when compared to using just SVM individually. When compared to the presently available models that use heuristic rules or keyword-based filters, our sentiment aware SVM model performed better. Sentiment analysis assisted in identifying the deceptive language and emotional tone frequently found in phishing emails, which are not usually handled by traditional techniques.

The SVM classifier successfully used sentiment variables (Embeddings) to distinguish between phishing and real emails. This finding supports the theory that sentiment analysis can identify minute language details that point to phishing. Sentiment features, for

example, improved the accuracy of classifying emails with threatening or urgent language as phishing.

Although we also figured out that using just Sentiment Analysis for phishing would not be sufficient enough to capture phishing emails. Capturing phishing emails should also entail the present techniques used such as the email ID check, URL's and images in the mail, and so on.

7 Conclusion and Future Work

This study proposed a sentiment analysis based phishing detection model which also uses machine learning algorithm. Working towards this topic in the era of every increasing phishing attacks is important and noteworthy. With the help of the phishing email dataset we produced the embeddings from the DistilBERT model and then used that embeddings to train and test our SVM model. The advantage of such a model is that it can increase the efficiency of the system to detect phishing emails by introducing semantic details from the raw email text which other models don't really capture. Using DistilBERT will also be helpful for the organizations working in the area as it does not require as many resources as the BERT model which will eventually reduce the resource cost and increase the accuracy of the overall system. Due to the limited time, this model cannot be implemented for a real time scenario. Although we were prepared to do it by using our own personal Gmail API to read the recent emails and detect if they are real or phishing email. We also created an OAuth Client to pass the API through and give the permission to access the emails but due to the time constraints we had to stop the process midway. Nevertheless, this can be implemented in the future as a real time phishing detection system which uses sentiment analysis for improved accuracy and can help as guiding light for the researchers who want to work in the similar field.

References

- Alaparthi, S. and Mishra, M. (2020). Bidirectional encoder representations from transformers (bert): A sentiment analysis odyssey, *arXiv*. Available at: <https://doi.org/10.48550/arXiv.2007.01127>.
- APWG — *Phishing Activity Trends Reports* (2023). Available at: <https://apwg.org/trendsreports/> (Accessed: 23 April 2024).
- Dey, N. et al. (2021). Analysis of machine learning algorithms by developing a phishing email and website detection model, *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pp. 1–7. Available at: <https://doi.org/10.1109/CSITSS54238.2021.9683131>.
- Irwin, L. (2022). Reported phishing attacks reach an all-time high — it governance eu, IT Governance Blog En. Available at: <https://www.itgovernance.eu/blog/en/reported-phishing-attacks-reach-an-all-time-high> (Accessed: 23 April 2024).
- Irwin, L. (2023). 51 must-know phishing statistics for 2023 — it governance, IT Governance UK Blog. Available at: <https://www.itgovernance.co.uk/blog/51-must-know-phishing-statistics-for-2023> (Accessed: 23 April 2024).

- Ma, L. et al. (2009). Detecting phishing emails using hybrid features, *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, pp. 493–497. Available at: <https://doi.org/10.1109/UIC-ATC.2009.103>.
- Niu, W. et al. (2017). Phishing emails detection using cs-svm, *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pp. 1054–1059. Available at: <https://doi.org/10.1109/ISPA/IUCC.2017.00160>.
- Patil, P., Rane, R. and Bhalekar, M. (2017). Detecting spam and phishing mails using svm and obfuscation url detection algorithm, *2017 International Conference on Inventive Systems and Control (ICISC)*, pp. 1–4. Available at: <https://doi.org/10.1109/ICISC.2017.8068633>.
- Rashid, J. et al. (2020). Phishing detection using machine learning technique, *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pp. 43–46. Available at: <https://doi.org/10.1109/SMART-TECH49988.2020.00026>.
- Ripa, S., Islam, F. and Arifuzzaman, M. (2021). The emergence threat of phishing attack and the detection techniques using machine learning models, *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, pp. 1–6. Available at: <https://doi.org/10.1109/ACMI53878.2021.9528204>.
- Sanh, V. et al. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, *arXiv*. Available at: <http://arxiv.org/abs/1910.01108> (Accessed: 11 August 2024).
- Sindhu, S. et al. (2020). Phishing detection using random forest, svm and neural network with backpropagation, *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pp. 391–394. Available at: <https://doi.org/10.1109/ICSTCEE49637.2020.9277256>.
- Wankhade, M., Rao, A. and Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges, *Artificial Intelligence Review* **55**(7): 5731–5780. Available at: <https://doi.org/10.1007/s10462-022-10144-1>.
- Zhang, X. et al. (2017). Boosting the phishing detection performance by semantic analysis, *2017 IEEE International Conference on Big Data (Big Data)*, pp. 1063–1070. Available at: <https://doi.org/10.1109/BigData.2017.8258030>.