National College of Ireland

Using Machine Learning Algorithms to Detect Phishing attack

MSc Research Project

MSc Cybersecurity

Shreyas Chandrakanth Kulkarni

Student ID: X23173106

School of Computing

National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland MSc Project

Submission SheetSchool of Computing

Student Name:	Shreyas Chandrakanth Kulkarni		
Student ID:	X23173106		
Programme:	MSc In Cybersecurity	**Year:**	2024
Module:	MSc Research Practicum		
Supervisor:	Jawad Salahuddin		
Submission Due Date:	12/08/2024		
Project Title:	Real-Time Detection of Social Engineering Threats in Social Media Posts.		
Word Count:		**Page Count**	26

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

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
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.	□

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

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

Using Machine Learning Algorithms to Detect Phishing attack

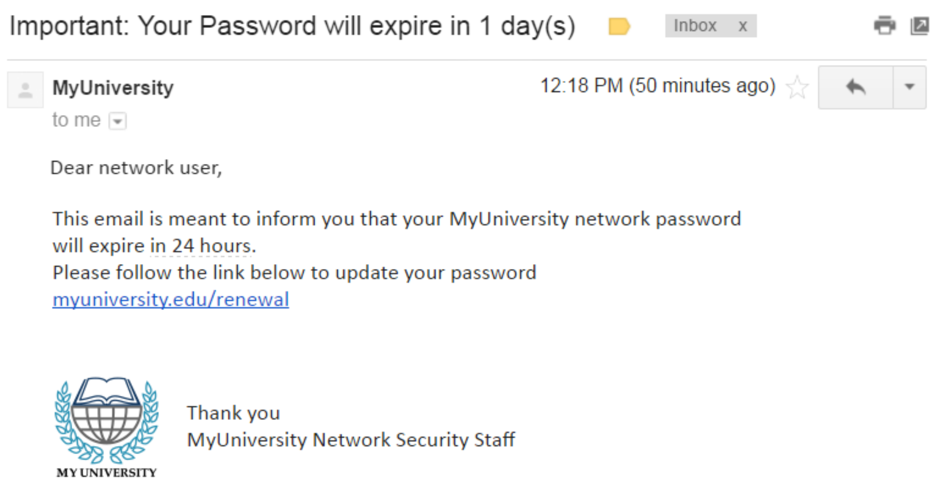
Abstract

Cyberfraud, particularly through phishing attempts, is a significant issue in corporate operations. This might impede business operations but also deprive valuable and critical information. This study's main objective is to address the issue of identifying phishing and not disturbing employees with these harmful email attacks. The proposal is to make an intelligent pipeline using machine learning to mitigate the case of phishing attacks. Various Machine Learning techniques including deep learning models, will be evaluated to determine their effectiveness in detecting inappropriate behavior in greedy search-based pipeline networks. Since machine learning has been identified as a crucial instrument in combating cybercrime, this response system can work agnostic to any organization and data to give a proper security service and maintain the server networks. These results further emphasize the significance of employing adaptable techniques to address emerging security challenges.

Keywords: DDoS attack, Phishing Email attacks, Machine Learning, Deep Learning, Response System

Chapter 1: Introduction

Phishing is one of the social engineering attacks commonly used by hackers to obtain user information, such as login passwords and credit card data. Phishing refers to the deliberate act of tricking a target into accessing an email, instant message, or text message by assuming the identity of a reliable source. Afterwards, the receiver is deceived into clicking on a harmful hyperlink, which can install harmful software.


Important: Your Password will expire in 1 day(s) Inbox x

MyUniversity
to me
12:18 PM (50 minutes ago)

Dear network user,

This email is meant to inform you that your MyUniversity network password will expire in 24 hours.
Please follow the link below to update your password
myuniversity.edu/renewal

MY UNIVERSITY

Thank you
MyUniversity Network Security Staff

Figure 1: An example depicting the suspicious email link **(Source: www.impreva.com)**

Phishing is a fraudulent technique in which false communications are transmitted across the internet, usually in the form of emails or messages, to give the false impression that they come from reliable sources. The primary objective is to deceive the receivers into sending sensitive information, such as credit card details, usernames, and passwords, along with other personal details. Phishing attacks can lead to theft, significant financial loss, and the unapproved disclosure of private data. The objective of the phishing scam depicted in Figure 1 is to contact as many users (in this case academic people) as possible using the dissemination of a phishing email that appears to have originated from *myuniversity.edu* (can be any link related to the organization or individual). By clicking on the link and give access to the entire system to the hackers.

1.1 Research Motivation

Despite two-factor authentication (2FA) being one of the most effective methods one can use to protect against phishing attempts, companies may take a variety of other measures to reduce the chance of phishing and spear phishing attacks. There is an extra layer of verification added when access is granted to essential applications. The availability of a physical device, such as a smartphone, and the user's awareness of and ability to use a password and username are the two primary aspects of two-factor authentication. Because compromised credentials are not sufficient to grant access, two-factor authentication (also known as 2FA) ensures that even compromised users are unable to utilize them. Both tight rules for password management and two-factor authentication (2FA) are equally critical practices. An example of such a policy would prohibit employees from using the same password for several apps and mandate that they change their passwords regularly. As a result of the fact that phishing attacks can take on specific forms and behaviors, it would be advantageous to make use of machine learning and artificial intelligence to recognize these attacks as well as other types of distributed denial of service (DDoS) attacks. This would allow the user or organization to take the appropriate precautions to protect themselves.

1.2 Research Aim

The integration of machine learning-based techniques with cybercrime detection is crucial for safeguarding online transactions and enhancing customer trust. Organizations can effectively combat fraudulent practices by employing machine learning algorithms (Pranali Landge, 2023). The utilization of a combination of machine learning and cyber security in this technique aids in overcoming challenges posed by harmful attacks and guarantees the ability to identify threats in real-time.

1.3 Research Questions

Based on the above discussions, we will be focusing on the following research questions,

RQ1: How can we develop machine learning algorithms so that they can recognize changing phishing strategies that take into consideration elements like content, sender behavior, and contextual cues?

RQ2: How can we differentiate between a valid legitimate link and a phishing link?

RQ3: How can we construct phishing detection systems that can adapt to new attack methods, keep detection rates high while minimizing the number of false positives, and make use of ensemble learning approaches such as integrating several classifiers or utilizing data from numerous sources?

Chapter 2: Literature Review

The field of research and development of methods for identifying fraudulent activity in online transactions has become an increasingly significant area of study. To safeguard both businesses and their customers, researchers from all over the world have been focusing their attention on developing algorithms and other approaches that can halt fraudulent activity in its tracks.

2.1 Research done towards different data imbalance techniques and anomalous data handling

The problem of class imbalance in transaction datasets has been the subject of a significant amount of study in the field of financial fraud detection. Rebalancing datasets by data stratification, minority over-sampling, and majority under-sampling has been done to enhance the ability of machine learning models to detect fraudulent patterns (Jose' A. Saez,' Bartosz Krawczyk, and Michal Wozniak, 2016). Some examples of methods include synthetic minority oversampling (SMOTE), Adaptive Synthetic Sampling

(ADASYN) and Rebalancing datasets by data stratification. The SMOTE algorithm was developed by (Chawla et al., 2002)(**Mengran Zhu, 2024**) to improve the performance of models in fraud detection scenarios. Synthetically generating data from minority groups is one method that may be utilized to address the issue of class imbalance. These approaches have as their primary objective the improvement of model performance through the correction of class imbalance.

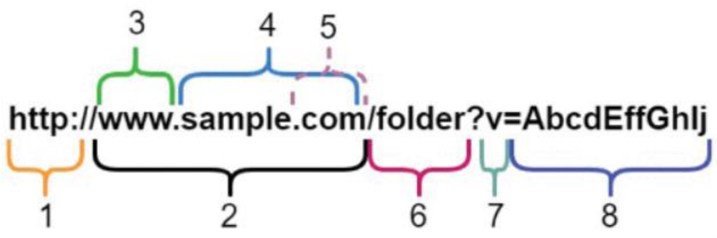


Figure 2: The structure of a phishing email (**Source: Nengran Zhu, 2024**)

As the above link shows, a phishing email contains a certain structure, this structure contains the information that can be used to mitigate the phishing. Research on the robustness of ML models against harmful attacks is becoming crucial to fraud detection. Advanced methods to modify machine learning models for evasion are at the heart of harmful attacks. It is vital to use strong machine learning models to protect fraud detection systems and successfully fight against such attacks. Some of the methods being studied to strengthen models against adversarial acts include adversarial training (**Weimin Zhao, 2022**), anonymity detection (**Ali Bou Nassif, 2021**), and model diversification. The goal is to guarantee accurate fraud detection by making the models more resistant to harmful attacks. An important area of focus in fraud detection research is making ML models resilient against adversarial attacks. To avoid detection, harmful attacks use sophisticated methods to manipulate ML models. To successfully defend against these threats and keep fraud detection systems secure, robust ML models are needed. To make models more resistant to adversarial behaviors, researchers are looking into methods such as adversarial training (Weimin Zhao, 2022), model diversification (Zhiqiang Gong, Ping Zhong, and Weidong Hu, 2019), and anomaly detection (Ali Bou Nassif, 2021). Emerging adversarial methods in fraud detection necessitate real-time detection strategies.

2.2 Machine Learning models towards the anomalous data prediction

The training method (**Alnemari, S. and Alshammari, M., 2023**) of the Random Forest ensemble learning approach makes use of several decision trees that were generated independently of one another. After totaling all the predictions given by each tree, we can derive the final forecast. Due to its exceptional performance on high-dimensional data, Random Forest is effective in facilitating the collection of intricate feature correlations. When applied to the phishing problem, Random Forest can identify trends in URLs or language that give the impression of malicious intent. Random Forest is responsible for searching for small irregularities that are related to phishing attempts. This is done to improve the accuracy of phishing detection systems. To accomplish this, it is quite beneficial to study the text and URLs from a semantic and structural point of view.

Modelling the likelihood (**Shirazi, H., et. al., 2023) of a binary result given one or more predictor elements may be accomplished through logistic regression, which is a linear classification technique.** Although it is known by a different name, logistic regression is utilized for classification applications most of the time. Logistic regression is a powerful method for detecting phishing because it calculates the chance that a certain phrase or URL is related to malicious intent. This makes it an effective strategy for phishing detection. This approach is easier to employ for locating crucial textual or structural indications of phishing attempts since it provides insights into the significance of features and results that may be interpreted after they have been obtained. By examining the correlations between the input data and their labels, logistic regression can help identify potentially malicious websites or material.

The XGBoost algorithm (**Apruzzese, G., 2023**), which is well-known for its accuracy and efficiency, is an example of a gradient-boosting strategy that performs exceptionally well when presented with large datasets. It does this by including weak learners into the model in a sequential manner, with each learner correcting the faults of its predecessor. Some examples of weak learners are decision trees. Phishing attempts may be easily spotted by XGBoost, which has a flair for recognizing subtle but revealing patterns and irregularities in URLs or linguistic characteristics. This makes it possible for XGBoost to identify phishing attempts. XGBoost searches for structural, semantic, and contextual characteristics in URLs or text that distinguish real things from fraudulent ones. This is done to make phishing detection systems more effective in general across the board. To do this, the text or URLs are analyzed.

One of the many ensembles learning (**Ajala, O.A., 2024**) approaches that is becoming increasingly popular is called AdaBoost. This method is an example of an approach that combines numerous weak classifiers into a single powerful one. It's possible that we can enhance the effectiveness of the classifier by redistributing the weights of data points that were mistakenly classified. When AdaBoost is used for phishing detection, the accuracy and robustness of the model also experience improvements. To do this, the categorization of text or URLs is modified on several occasions. The number of false positives may be reduced using AdaBoost, and the identification of subtle phishing attempts can be increased. This is accomplished by adjusting the weighting of the model appropriately and focusing on challenging situations.

Support vector machines (**Kuraku, D.S. and Kalla, D., 2023**), are supervised learning models that come in useful for tasks like classification and regression. By identifying the optimal hyperplane for the separation of data points into many categories, support vector machines (SVMs) attempt to optimize the margin between classes. In circumstances in which the class boundary is readily obvious, support vector machines (SVMs) perform remarkably well when working with high-dimensional data. When identifying phishing, support vector machines (SVMs) use the geographical distribution of textual qualities or URLs to determine if they are genuine or counterfeit. By evaluating the geometric relationships between data points, support vector machines (SVMs) can discern between real and fake items. This is accomplished by studying the data points. This dissimilarity may cause them to classify site URLs or text contents differently and cause confusion.

It is possible to utilize decision trees for both regression and classification tasks; decision trees are a sort of non-parametric supervised learning (**Shombot, E.S., Dusserre, G., Bestak, R. and Ahmed, N.B., 2024**). Through the process of recursively splitting the feature space into subsets depending on the feature values, they strive to either reduce the amount of impurity or improve the amount of information gained with each split. Decision trees can be useful for phishing detection since they enable pattern identification and feature importance comprehension. Because of this, they can be beneficial. Their legibility and simplicity make them ideal for analyzing textual or structural indicators of phishing attempts. Because decision trees can differentiate between actual and fraudulent entities, they can increase the accuracy of phishing detection systems. To accomplish this, decision trees recursively divide the feature space.

2.3 Research towards mitigating Phishing and DDoS attacks using Machine Learning

The accuracy of the model (**Ahmed, S., et. al, 2023 is evaluated based on the proportion of positive occurrences, such as URLs used for phishing, correctly identified compared to the total number of positive cases predicted.** To determine how accurate the model's positive predictions are, this statistic is applied. It is computed by dividing the total count of true positives by the sum of both true positives and false positives. When it comes to phishing detection, accuracy is an essential parameter since it demonstrates how well the model can identify dangerous URLs without simultaneously creating false positives. With a high accuracy score and a low false positive rate, the likelihood of the algorithm wrongly labelling valid URLs as phishing attempts is reduced. This is because the algorithm is less likely to make a mistake.

When compared to the total number of positive examples in the dataset (**Alashhab, A.A., et. al, 2024**), "recall," which is often referred to as "sensitivity," is the ratio of the number of occurrences that have been positively recognized (such as phishing URLs). We can assess whether the model can recognize all real positives by dividing the total number of positive findings by the sum of all positive and negative results. The capacity of the model to identify phishing attempts is evaluated by recall, and it ensures that no dangerous URLs are overlooked despite this evaluation. In the process of phishing detection, it is an essential component. As shown by its high recall value and low false negative rate, the model can effectively identify most phishing attempts.

Given that it provides an accurate evaluation of the overall classification accuracy, the F1 Score is useful in situations where reducing the number of false positives and false negatives is a top concern. As the value goes closer and closer to 1, the performance of the model gets better. By keeping a healthy equilibrium between recall and accuracy, the phishing detection model can reduce the number of false positives and negatives, resulting in a better F1 Score. This contributes to the phishing detection system being more robust (Ramprasath, J., 2024).

Table 1: Summary of the research papers read

Sl. No	Paper Name	Authors Name	Dataset Used	Models Used	Results Summary
1	Phishing Website Detection using Machine Learning Algorithms	Rishikesh Mahajan, Irfan Siddavatam	PhishTank	DecisionTree, Random Forest, SVMs	They achieved 97.14% detection accuracy using random Forest algorithm with lowest false positive rate.
2	Detecting Phishing Domains Using Machine Learning	Shouq Alnemari, Majid Alshammari	UCI Phishing Websites	Artificial neural networks, Support Vector machine, Decision Tree, Random Forest	The overall results show random forest (RF) model achieved the highest performance
3	A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN	Zainab Alshingiti, Rabeah Alaqel, Jalal Al-Muhtadi, Qazi Emad Ul Haq, Kashif Saleem, Muhammad Hamza Faheem	-	LSTM, CNN, and LSTM–CNN	CNN architecture provides excellent results compared to LSTM–CNN and LSTM
4	Phishing URL detection using machine learning methods	SK Hasane Ahammad, Sunil D. Kale, Gopal D. Upadhye, Sandeep Dwarkanath Pande, E Venkatesh Babu, Amol V. Dhumane, Mr Dilip Kumar Jang Bahadur	PhishTank	Light GBM, Random Forest, Decision Tree, Logistic Regression, and SVM	Light GBM produced good results
5	Detection of Phishing Websites using an Efficient Machine	Naresh Kumar D	-	Random Forest, Logistic Regression, KNN, Decision Tree	Random Forest classifier has better phishing detection accu-

	Learning Framework				racy
6	Detecting phishing websites using machine learning technique	Ashit Kumar Dutta	AlexaRank and PhishTank	LURL, Hung Le et al, Hong J et al	Performance of LURL is better comparing to other URL detectors
7	Phishing Detection using Machine Learning based URL Analysis: A Survey	Arathi Krishna V, Anusree A, Blessy Jose, Karthika Anilkumar, Ojus Thomas Lee	Phish Tank or Open-Phish	Logistic Regression (LR), K-Nearest Neighbour (KNN), Support Vector Machine(SVM), DecisionTree (DT), Naive Bayes (NB), XGBoost , Random Forest (RF) and Artificia l Neural Network (ANN)	Random Forest gives better results in most scenarios

Chapter 3: Methodology

3.1 Dataset

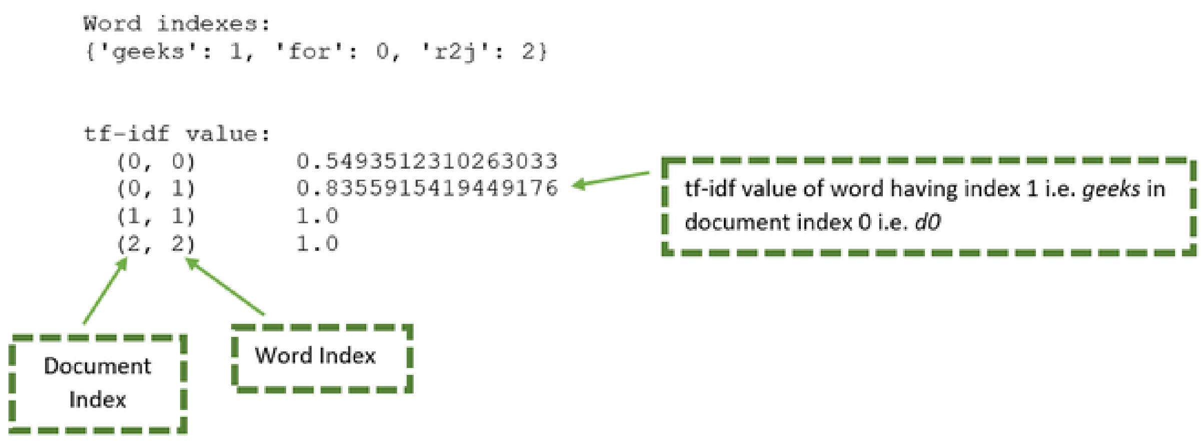
In all, the dataset has a total of 18,634 samples of emails and out of these 11,322 are safe emails and the rest 7,312 the emails that are regarded as suspicious. 'item' contains records that encompass the textual content of the email body as well as the associated email type. Below is a brief guide for research on the specific characteristics of phishing emails inclusive of the characteristics of other types of emails. This dataset is very helpful in training machine learning models in which the primary goal is to identify secure and phishing emails. Due to the availability of enriched text bodies, analysis can be quite extensive, which helps to identify desirable aspects and trends to improve the performance of detection of Phishing emails.

3.2 Data Preprocessing Techniques

The next step the preparatory process of the dataset that is needed for machine learning tasks is the splitting of the data into training and test sets. The goal of this is to enable a section of data to be used for training the model while the other part is used for testing the model which will help in making a perfect judgement of the model. Specifically, the data set is split into the training set and testing set in terms of an 80:20 ratio and ensuring data randomization to make the experiment reproducible. The next step the preparatory process of the dataset that is needed for machine learning tasks is the splitting of the data into training and test sets. The goal of this is to enable a section of data to be used for training the model while the other part is used for testing the model which will help in making a perfect judgement of the model. Specifically, the data set is split into the training set and testing set in terms of an 80:20 ratio and ensuring data randomization to make the experiment reproducible.

$$W_{x,y} = tf_{x,y} \; x \log(\frac{N}{df_x})$$

In other words, the above equation is the statistical feature extraction method's formula. This attempts to create statistical features with the corpus and the words contained in the entire document. After this preprocessing of data has been made, the training subset of data is proceeded with vectorization by using the TF-IDF vectorizer, which has been explained above. This vectorizer transforms the textual data into numerical data for analysis. This operation is entirely indispensable for translation purposes to transform the text into a format that could be used in machine learning algorithms.



```
Word indexes:
{'geeks': 1, 'for': 0, 'r2j': 2}


tf-idf value:
  (0, 0)       0.5493512310263033
  (0, 1)       0.8355915419449176
  (1, 1)       1.0
  (2, 2)       1.0
```

tf-idf value of word having index 1 i.e. *geeks* in document index 0 i.e. *d0*
Document Index
Word Index

Figure 3: An example depicting the generation of TF-IDF value

As for the vectorizer just fitted to the training data, we can subsequently calculate the term frequencies relevant to the training data as well as the inverse document frequencies corresponding to the obtained term frequencies. Following that, the testing subset is to go through the same operation of the TF-IDF vectorizer mentioned above.

3.3 Machine Learning Models

3.3.1 Logistic Regression

Logistic regression is designed for binary outcomes, unlike linear regression, which handles continuous outcomes and assumes a linear relationship between variables. In logistic regression, the logistic function models the relationship between the dependent variable and one or more independent variables. This function transforms the linear combination of predictors into a probability value between 0 and 1, making it perfect for binary classification tasks, as it ensures that predicted values can be interpreted as probabilities.

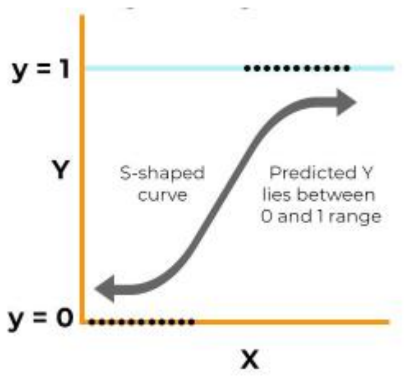


Figure 4: Sigmoid or Logistic Function

3.3.2 Random Forest Classifier

Logistic regression is primarily used for classification tasks. In ensemble learning, multiple decision trees are created during training, and their outcomes are combined to improve accuracy and robustness. These trees are trained on random subsets of the data, with each tree's splits based on a random selection of features. This approach enhances the model's generalizability and reduces the risk of overfitting, a common issue with individual decision trees.

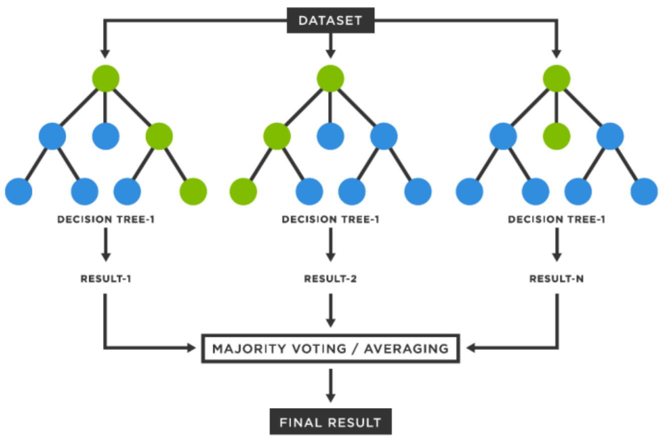


Figure 5: An example of how the results is generated as can be seen multiple trees are generated

3.2.3 Naïve Bayes Classifier

Naive Bayes classifiers calculate the posterior probability for each class by utilizing the given predictors to make predictions. To begin this process, the initial step is to merge the prior probability of each class with the likelihood of the predictors given that class. Subsequently, the probabilities are standardized across all possible classes. Due to its simplicity and efficiency, this approach is highly useful in handling complex and large datasets.

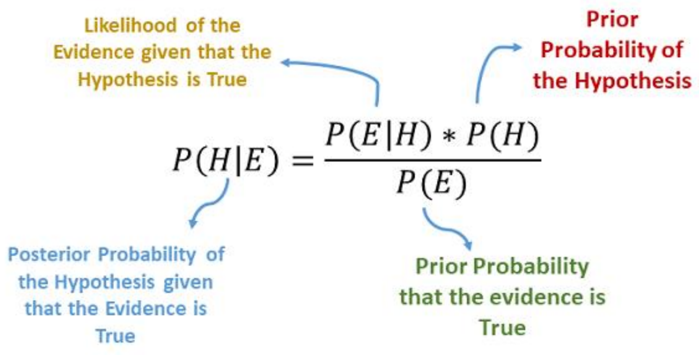

Likelihood of the Evidence given that the Hypothesis is True
Prior Probability of the Hypothesis
$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$
Posterior Probability of the Hypothesis given that the Evidence is True
Prior Probability that the evidence is True

Figure 5: Naïve Bayes Algorithm using the Bayes Theorem

3.3.4 Support Vector Machine

Support Vector Machines (SVMs) are a group of supervised learning algorithms effective in both classification and regression tasks. They excel at handling large datasets and complex decision boundaries, making them highly efficient for distinguishing between data classes. SVMs work by placing hyperplanes (or multiple hyperplanes for multi-class problems) in higher-dimensional space to separate different classes. The main goal is to find the hyperplane that maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class. A larger margin enhances the model's robustness, reducing classification errors on unseen data.

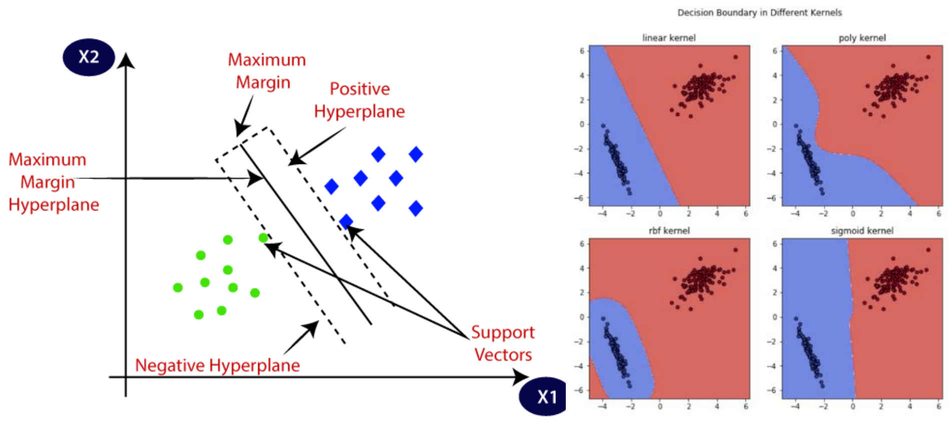


Figure 6: The example of how the gain margin is fixed based on the hyperplane and how the classes are differentiated

3.3.5 K-Nearest Neighbours

The k-nearest Neighbours (k-NN) algorithm is a widely recognized supervised learning method used for both classification and regression tasks. When it comes to organizing various types of data, it is widely recognized for its simplicity and effectiveness.

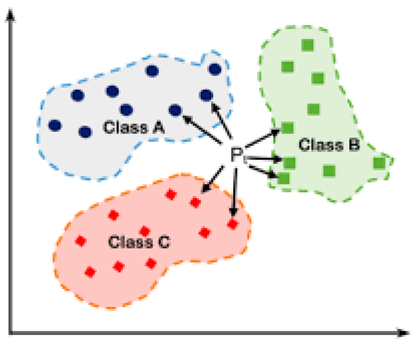


Figure 7: Example showcasing the clusters, or the classes formed based on the distance from one point i.e neighbours

Due to the need to calculate distances between the query point and all points in the training set, the k-nearest neighbours (k-NN) algorithm has several challenges, with computing complexity being one of the most critical, especially when working with large datasets. To address this problem, several optimizations and data structures, such as k-d trees and ball trees, can be employed to expedite the search for the closest neighbouring item.

3.4 Recurrent Neural Networks

3.4.1 GRU (Gated Recurrent Units)

Gated Recurrent Units, commonly referred to as GRUs, are a type of recurrent neural network (RNN) structure that is highly effective for tasks involving sequence modelling. These applications encompass the fields of time-series analysis and natural language processing.

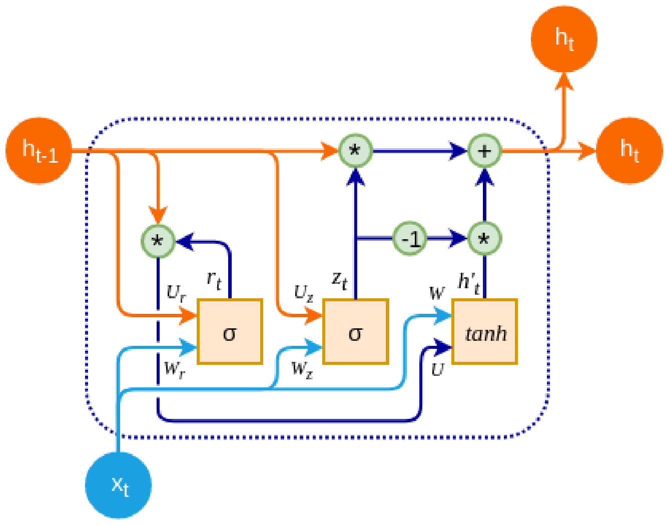


Figure 8: GRU cell

3.4.2 LSTM (Long Short-Term Memory)

Long Short-Term Memory (LSTM) networks are a specific type of recurrent neural network (RNN) designed to address the challenges of learning long-range dependencies in sequential input. LSTMs are highly advantageous for tasks that require the handling of natural language and time-series data due to their ability to retain and utilize information from long sequences.

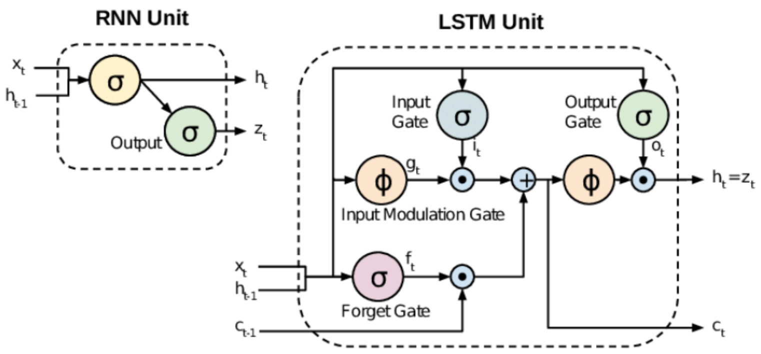


Figure 9: LSTM cell

3.5 Evaluation

Precision – It illustrates how successful the approach can be in preventing falsely identifying legitimate transactions as fraudulent ones. The accuracy of a test may be calculated by dividing the total number of positive and negative findings by the number of real positives.

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

Recall –To determine it, take the total number of positive results and divide it by the sum of all positive results and false negatives. This value is the outcome of the calculation. By analyzing all the valid cases, it is possible to determine what percentage of the forecasts were accurate.

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

F1 Score –This allows just one score to be obtained. It is helpful to have the F1 Score available in situations when the dataset is skewed toward fraudulent behaviours in comparison to non-fraudulent behaviours.

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

Threshold calibration, which involves altering the model's classification threshold, is one method that may be used to achieve the optimal balance between accuracy and recall characteristics.

Chapter 4: Implementation

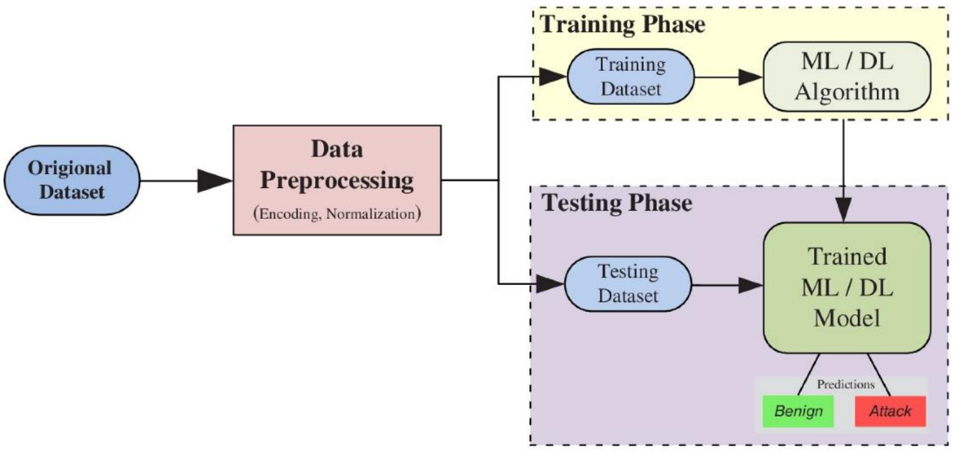


Figure 10: Architecture diagram used for this machine learning-based cybersecurity detection of phishing

Algorithm Flow

<u>Step 1 – Data collection and preprocessing</u>: We need to collect many emails, both legitimate and fake. Preprocess the emails to extract important information such as the sender's email address, the email message, any attached files, and any URLs. To get the data ready for training, the group of features need to be transformed into numerical values and the numerical features normalized. The technique of creating new features from existing ones or changing already features in a dataset to improve the performance of a machine-learning model is known as feature engineering.

The most crucial step is to determine the most relevant features for the categorization of actual and false messages by extracting the important features and evaluating the features. To detect any slight changes that would point to the sender's involvement in phishing, include other data such as the words used in the email content, URLs, senders' reputations, and many more.

<u>Step 2 – Process of selecting and training a model</u>: We need to select some of the mainstream ML techniques for classification such as, random forests and logistic regression. We will be utilizing statistical characteristics like TF-IDF because text data is available.

$$W_{x,y} = tf_{x,y}\ x\ \log(\frac{N}{df_x})$$

We then split the data into a training set and a validation set such that no data from the training set leaks into the validation set and both the classes are equally represented.

We will perform training of several models employing different hyperparameters to compare between different models and increase the learning capability.

Step 3 – Evaluation of the model: Evaluate the training models against the validation set and compare the outcomes using metrics like accuracy, precision, recall, and F1-score to assess the models.

Through cross-validation, or on the other hand, a separate test set, to evaluate the models' capacity to apply the knowledge to data that was not examined before.

Step 4 – Deployment and monitoring of models: This project has used some of the best models for deploying and results are displayed for evaultion.

To establish mechanisms to regularly check the working of the model and make the corresponding changes to adapt to changes in the types of phishing strategies and guarantee effectiveness in the future.

Integrating the model into the existing cybersecurity framework to provide an instantaneous countermeasure against phishing attacks.

Step 5 – Assessment and Enhancement: To determine the efficiency of the deployed model, we will employ an initial evaluation that compares specific variables, such as the proportion of false positive and false negative outcomes.

We intend to gather input from both security experts and users to pinpoint areas that require enhancement and make successive modifications to the model.

To improve the model's effectiveness in stopping phishing attacks, it should be updated often and expanded with new elements.

4.1 Data Source

For the periods January to May 2015 and May to June 2017 (**Tan, Choon Lin (2018),**), there were downloads of 5,000 phishing and 5,000 legal websites and 48 characteristics were extracted from the datasets. Implementing feature extraction via the browser automation framework, namely the Selenium WebDriver makes for a more accurate method of extracting features as compared to the regular expression-based parsing method. Thus, the described dataset can be effective for phishing feature analysis if the researcher needs fast proof of concept trials or comparing phishing classification models among anti-phishing professionals.

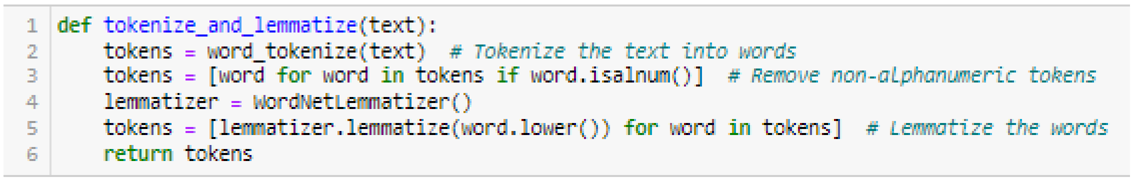
4.2 Data Pre-Processing: We then split the data into a training set and a validation set such that no data from the training set leaks into the validation set and both the classes are equally represented. We will perform training of several models employing different hyperparameters to compare between different models and increase the learning capability.

Step 3 – Evaluation of the model: To evaluate the training models by testing them against the validation set and comparing the results with the parameters such as accuracy, precision, recall, and F1-score.To check the ability of the models to apply the knowledge to unseen data by cross-validation, or on the other a different test set.

Step 4 – Deployment and monitoring of models: This paper has implemented the most effective model in an operational phishing detection system. To establish mechanisms to regularly check the working of the model and make the corresponding changes to adapt to changes in the types of phishing strategies and guarantee effectiveness in the future. Integrating the model into the existing cybersecurity framework to provide an instantaneous countermeasure against phishing attacks

Step 5 – Assessment and Enhancement: We will use a formative evaluation to determine the effectiveness of the implemented model by comparing certain variables like the percentage of false positive and false negative results. We will collect feedback from the users and security specialists to identify the areas of improvement and adjust the model consecutively. The model should be updated frequently, and new components should be included in it to increase its efficiency in preventing phishing attacks.4.1 Data SourceFor the periods January to May 2015 and May to June 2017 (**Tan, Choon Lin (2018),**), there were downloads of 5,000 phishing and 5,000 legal websites and 48 characteristics were extracted from the datasets. Implementing feature extraction via the browser automation framework, namely the Selenium WebDriver makes for a more accurate method of extracting features as compared to the regular expression-based parsing method. Thus, the described dataset can be effective for phishing feature analysis if the researcher needs fast proof of concept trials or comparing phishing classification models among anti-phishing professionals.4.2 Data Pre-ProcessingAfter that, the testing subset is subjected to the identical TF-IDF vectorizer that was described earlier. Therefore, this guarantees that the test data is converted using the same feature space as the training data, which helps to preserve consistency and enables an accurate evaluation of the trained model's performance. Through the application of the vectorizer to both the training dataset and the testing dataset, the method guarantees that the text data is appropriately prepared for the training and evaluation of robust models.

After that, the testing subset is subjected to the identical TF-IDF vectorizer that was described earlier. Therefore, this guarantees that the test data is converted using the same feature space as the training data, which helps to preserve consistency and enables an accurate evaluation of the trained model's performance. Through the application of the vectorizer to both the training dataset and the testing dataset, the method guarantees that the text data is appropriately prepared for the training and evaluation of robust models.

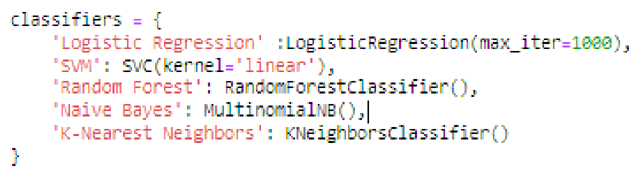


```python
def tokenize_and_lemmatize(text):
    tokens = word_tokenize(text)  # Tokenize the text into words
    tokens = [word for word in tokens if word.isalnum()]  # Remove non-alphanumeric tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word.lower()) for word in tokens]  # Lemmatize the words
    return tokens
```

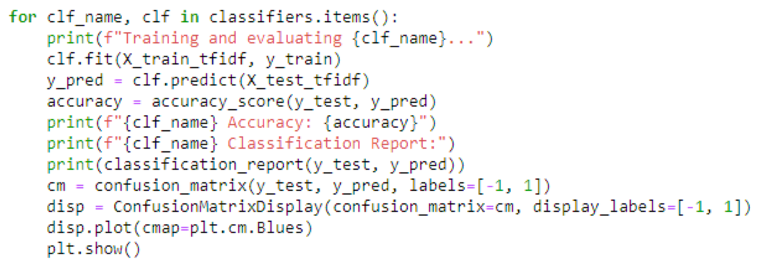
Figure 11: Code Snippet of Tokenisation

4.3 Machine Learning

We evaluated the effectiveness of several machine learning classifiers on pre-processed text data. The classifiers used include K-Nearest Neighbours, Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machine (SVM) with a linear kernel. Each algorithm was trained using TF-IDF-transformed data, allowing the models to identify patterns in the text. After training, these classifiers were tested on a separate dataset, also processed with TF-IDF, to ensure consistency in predicting labels.



```python
classifiers = {
    'Logistic Regression' :LogisticRegression(max_iter=1000),
    'SVM': SVC(kernel='linear'),
    'Random Forest': RandomForestClassifier(),
    'Naive Bayes': MultinomialNB(),
    'K-Nearest Neighbors': KNeighborsClassifier()
}
```



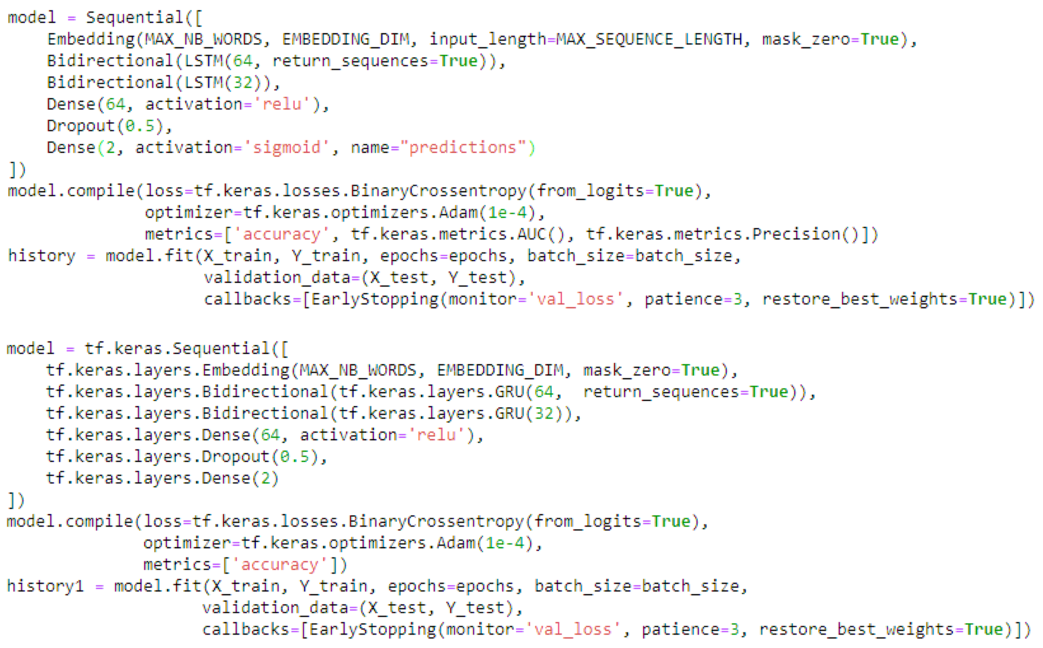
```
for clf_name, clf in classifiers.items():
    print(f"Training and evaluating {clf_name}...")
    clf.fit(X_train_tfidf, y_train)
    y_pred = clf.predict(X_test_tfidf)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{clf_name} Accuracy: {accuracy}")
    print(f"{clf_name} Classification Report:")
    print(classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred, labels=[-1, 1])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[-1, 1])
    disp.plot(cmap=plt.cm.Blues)
    plt.show()
```

Figure 12: Code Snippet for Machine Learning Modelling

4.4 Deep Learning

The first deep learning model uses a bidirectional GRU architecture, while the second uses a bidirectional LSTM architecture, both built with Keras in TensorFlow. In the first model, an embedding layer converts input text into dense vectors, followed by a 64-unit bidirectional GRU layer that processes the data in both directions to capture context. A second 32-unit GRU layer enhances feature extraction, and a 64-unit dense layer with ReLU activation adds non-linearity. A dropout layer is included to prevent overfitting, and the final output layer has two units for classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the evaluation metric, with training continuing until validation loss stabilizes to avoid overfitting.

The second model is similar but uses bidirectional LSTM layers instead of GRU layers, with 64 and 32 units, respectively, to capture long-term dependencies in the text.



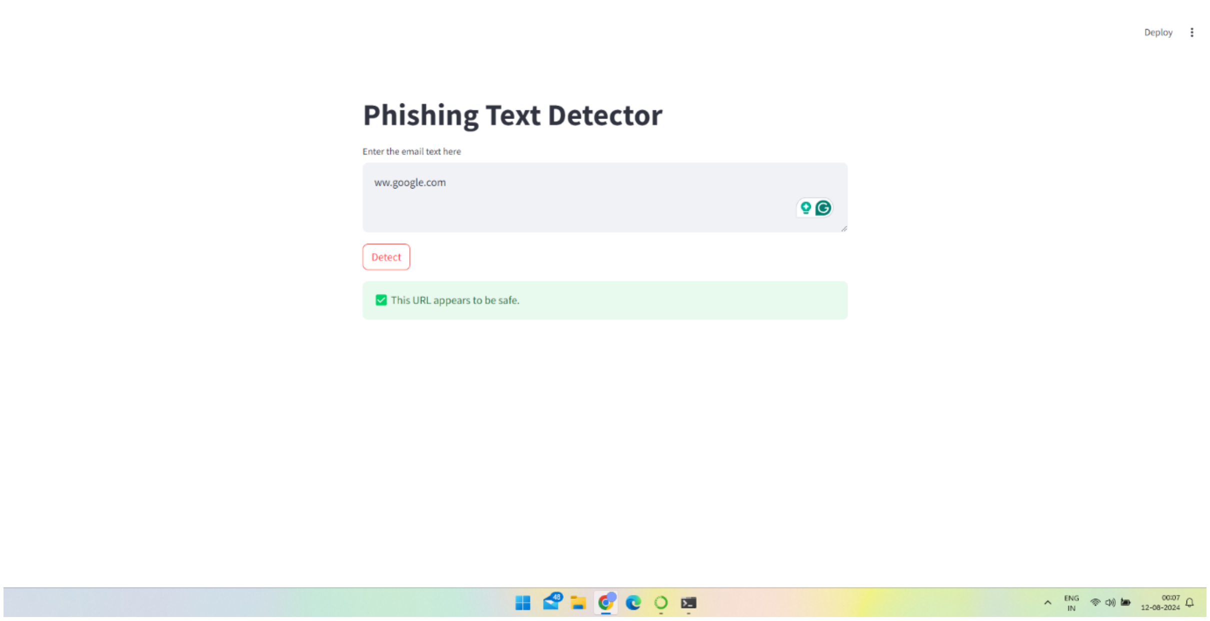
```
model = Sequential([
    Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH, mask_zero=True),
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(32)),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(2, activation='sigmoid', name="predictions")
])
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy', tf.keras.metrics.AUC(), tf.keras.metrics.Precision()])
history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,
                    validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)])

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(MAX_NB_WORDS, EMBEDDING_DIM, mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64,  return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(2)
])
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
history1 = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,
                    validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)])
```

Figure 13: Deep Learning code snippets

Chapter 5: Results and Analysis

5.1 Case study

This case study illustrates the ability to differentiate between genuine and fraudulent URLs, providing users with critical protection against phishing attacks.



Phishing Text Detector

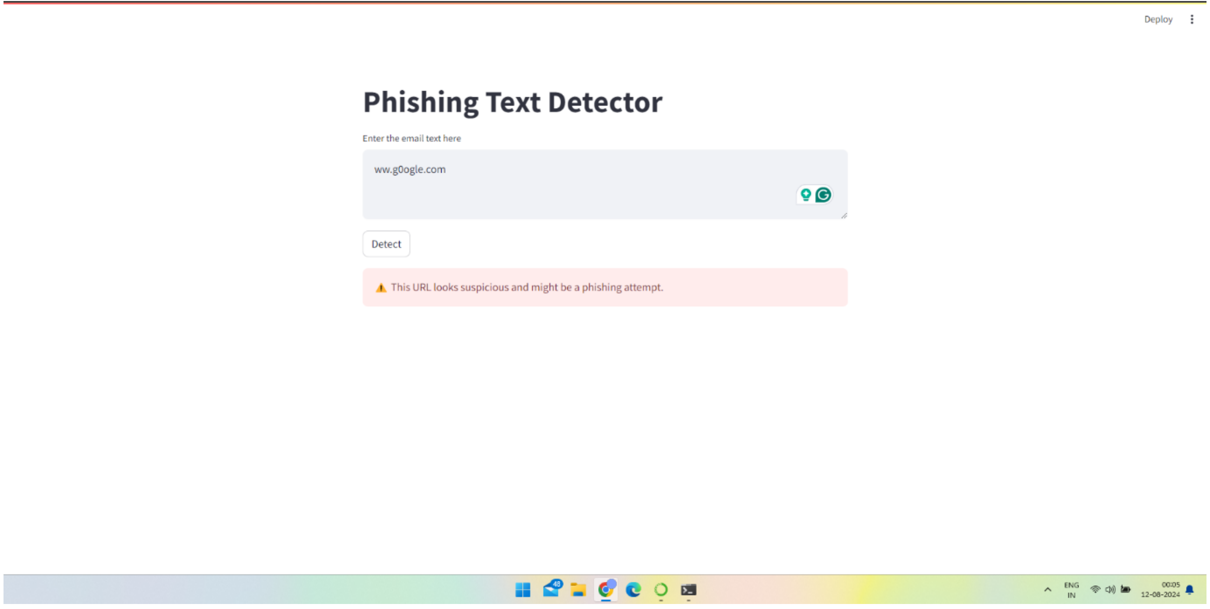
Enter the email text here

ww.google.com

This URL appears to be safe.

Figure 14: Logistic Regression performance

The first image shows the evaluation of the link www.google.com, which is recognized as legitimate. The system verifies the URL against a database of known trusted sites and confirms its authenticity.



Phishing Text Detector

Enter the email text here

ww.g0ogle.com

This URL looks suspicious and might be a phishing attempt.

Figure 15: Logistic Regression performance

The second image illustrates the detection of a phishing attempt with the link www.g0ggle.com. Here, the system identifies the subtle substitution of the letter "o" with the digit "0" as a red flag for potential phishing. This discrepancy, while minor, is enough for the tool to categorize the link as suspicious and alert the user to possible fraudulent intent.

5.2 Machine Learning

5.2.1 Logistic Regression

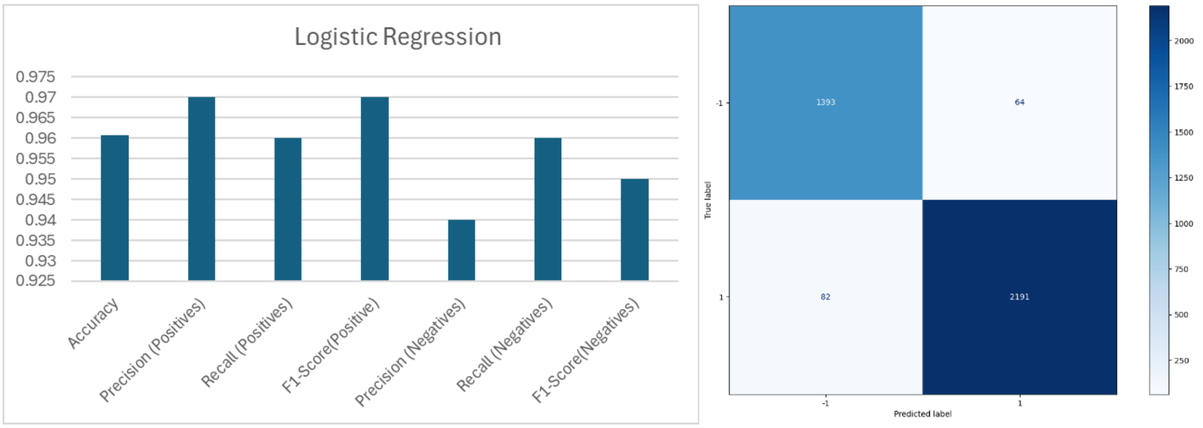


Figure 16: Logistic Regression performance

The logistic regression model exhibited strong overall performance with an accuracy of 96.08%, effectively balancing precision and recall for both positive and negative classes. For positives, the precision was 0.97, meaning the model made very few false-positive errors, while the recall of 0.96 indicates it successfully captured the truest positives. Similarly, for negatives, the precision was 0.94 and the recall was 0.96, showing that the model also handled negative cases well.

The high F1 scores of 0.97 for positives and 0.95 for negatives reflect the model's ability to maintain a good balance between precision and recall, minimizing both false positives and false negatives. This indicates that the model is not only accurate but also reliable in distinguishing between positive and negative cases. Overall, the logistic regression model's performance is robust, making it a strong candidate for use in scenarios where accurate classification is critical

5.2.2 SVM

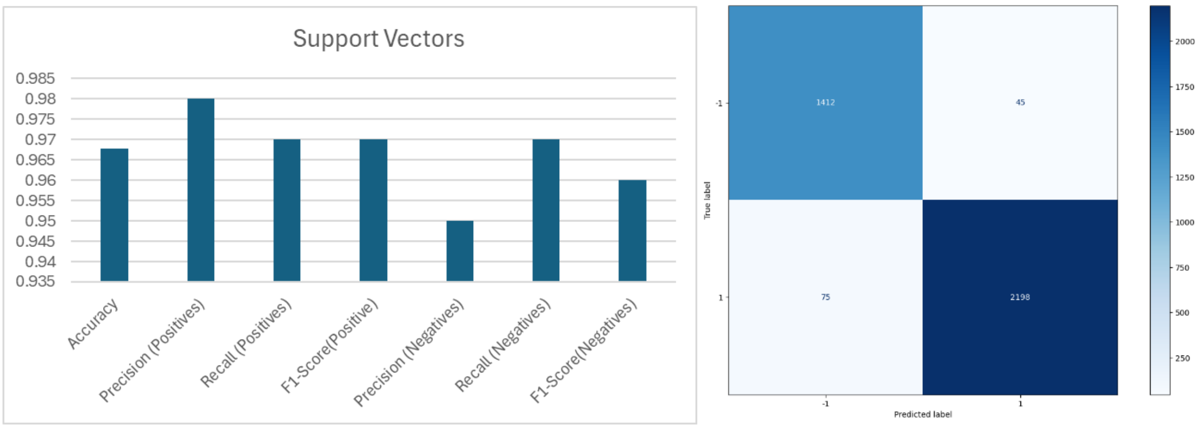


Figure 17: SVM performance

The proposed Support Vector Machine (SVM) model was very accurate and estimated an overall accuracy of 96%. The model also performed very well in the positive classifications with a precision of 0. 98, revealing that most of the positive predictions are accurate. The recall for positive cases was therefore 0. In the second group of the same patients, the model achieved a 97% accuracy, which indicates that the model identified nearly all true positive cases. The F1-Score for positive classifications was 0. 97, this shows that the precision was very good and combined with the recall provided a good balance between the two and hence the model can correctly classify the positive cases without many wrongs. On the negative side, the SVM model kept the precision at 0. It translated to 95/100, meaning that the negative predictions were 95% correct all through. The recall for negatives was slightly higher at 0. 97, this shows that the model was able to correctly classify most true negative cases. The F1-Score for negative classifications was at 0. The authors also compared the precision and recall of the model, noting that it is 96% for negative instances, which proves the model's ability to achieve a good balance between precision and recall. In conclusion, the presented SVM model has high accuracy, and has almost equal performance for both classifications as positive and negative, which proves that it is a reliable and accurate tool for accurate predictions.

5.2.3 Random Forest Classification

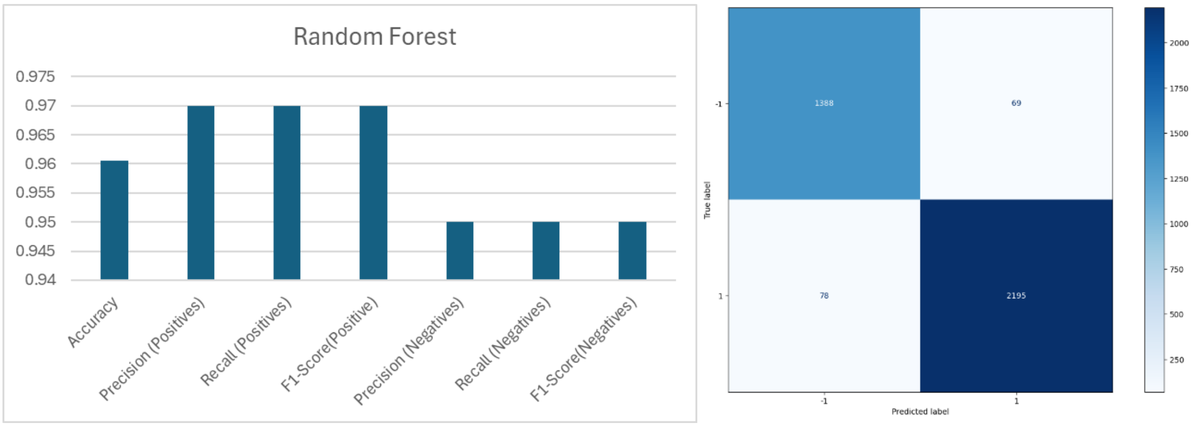


Figure 18: Random Forest performance

The accuracy of the Random Forest Classifier model was found to be 96% this means that the model was right in 96 out of 100 of the observations. 5 percent of all the occurrences contained in the given dataset. In the positive classification, which entails identifying the parts of the text that are most relevant to a story, the model had a precision of 0. 97 which implies that, out of the 100 instances which was predicted to be positive, 97 were positive. The recall for positive cases was also zero. 97, which demonstrates the ability of the model to capture most of the truly positive cases. The F1-Score regarding positive classifications was 0.97, which is good and indicates the equal ratio of precision and recall of the model. On a negative note, the model was able to attain a precision of 0. 95, which means the percentage of negative classification that was correctly identified as such of the total number of negative predictions made was 95%. The recall for negatives was 0. 95, which means that the call negative cases that were not included in the study were also correctly left out by the model. Thus, the F1-Score of the proposed model is 0.95 for negative classifications, the Random Forest Classifier keeps on performing well in the process of differentiating between positive and negative instances. In conclusion, the model's metrics for both classes present a good balance, proving its stability and effectiveness in classification problems.

5.2.4 Naïve Bayes

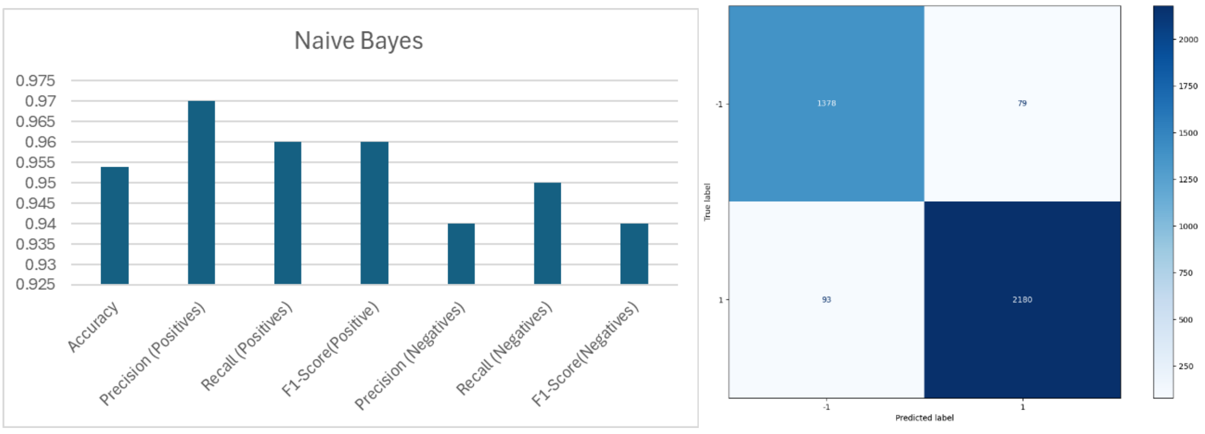


Figure 19: Naïve Bayes performance

The Naive Bayes model obtained a performance of 95% when it was evaluated. 38%, which means that the proposed model has a high level of accuracy when solving such classification problems. Regarding positive classifications, the model obtained a precision of 0. 97, which means that 97 percent of the incident accurately predicted as positive. The recall for positive cases was 0. 96' suggesting that the model was able to correctly identify all the true positive rates at a whopping 96 percent. Specifically, the F1-Score for positive classifications was 0. 96, which is a good compromise between precision and recall, thus guaranteeing the correct identification of positive cases. In the case of the negative classifications, the model achieved a precision of 0. 94, this means that in the 100 instances that were predicted to be negative, it had a 94% chance of being negative. The specificity for recall of the negative cases was 0. The true negative rate was estimated at 95 proving that the model is good at identifying the false negative instances. The F1-Score of this study was 0. 94 for negative classifications, as the above result shows that the Naive Bayes model can retain steady performance on both positive and negative classes. This equal balance across these measures highlights the accuracy of the model for classification tasks even if it uses a more basic probabilistic theory.

5.2.5 KNN

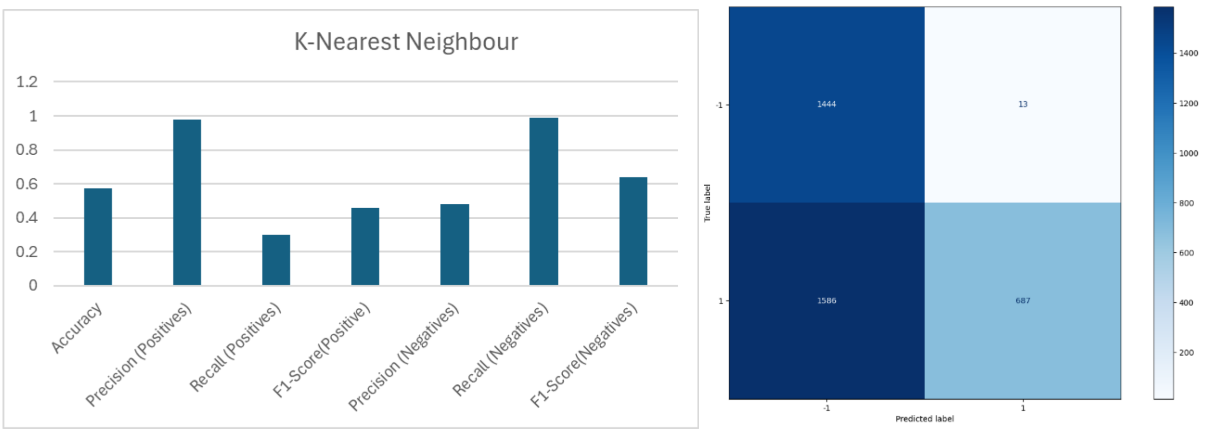


Figure 20: k-NN performance

The K-Nearest Neighbours (KNN) model was also not very impressive and demonstrated a skewed performance split. Although it obtained a precision of 0. 98 for positive classifications for negative classifications the precision was low. This clearly shows that the model labelled many true positives as negatives, that is, it failed to predict them. This situation resulted in a rather low F1-Score of 0. 46 for positives, which shows the difference between precision and recall. As for the negative classification, the desired recall was achieved at 0. 99, thus correctly classifying almost all the true negatives. The accuracy for negatives was relatively low at 0. 48, which means that more cases were diagnosed as cancerous when in actual sense they were not. The F1-Score for negatives was 0. 64 Regarding the above results, the model has outperformed positives but at the same time, it has shown that it has areas of improvement. In general, the KNN model had issues with the balanced accuracy, more specifically – it failed to provide good results with the positive cases hence it may not be the most suitable for use in datasets with imbalanced classes.

5.3 RNN- based Models

5.3.1 LSTM Approach

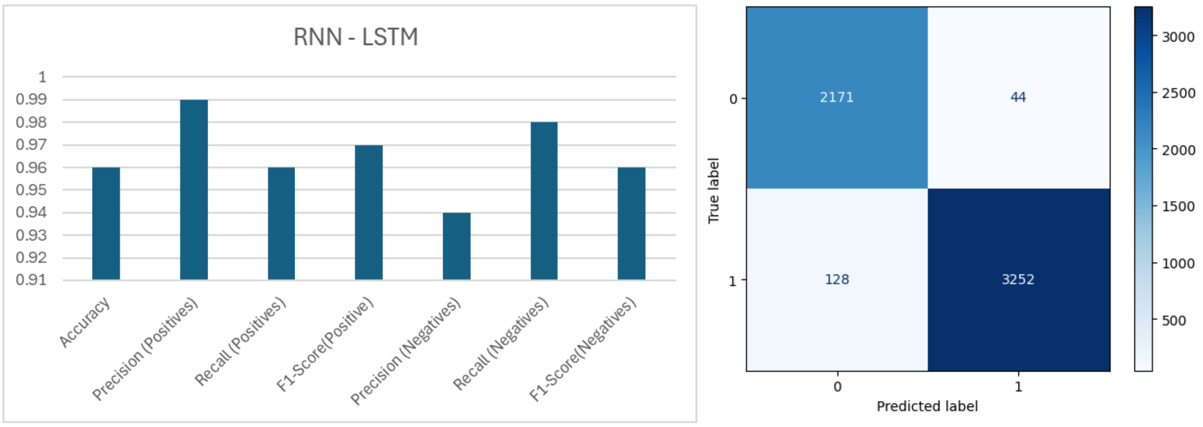


Figure 21: LSTM performance

The LSTM model demonstrates robust performance across several metrics. It achieves an accuracy of 0.96, reflecting its overall effectiveness in classifying instances correctly. For positive cases, the model exhibits a high precision of 0.99 and a recall of 0.96, resulting in a strong F1-score of 0.97. This indicates that the LSTM is highly effective at identifying positive instances while maintaining a low rate of false positives. For negative cases, the precision stands at 0.94, with a recall of 0.98, and an F1-score of 0.96. These metrics suggest that the model is also proficient at detecting negative instances, ensuring a good balance between precision and recall for both classes.

5.3.2 GRU-Based Approach

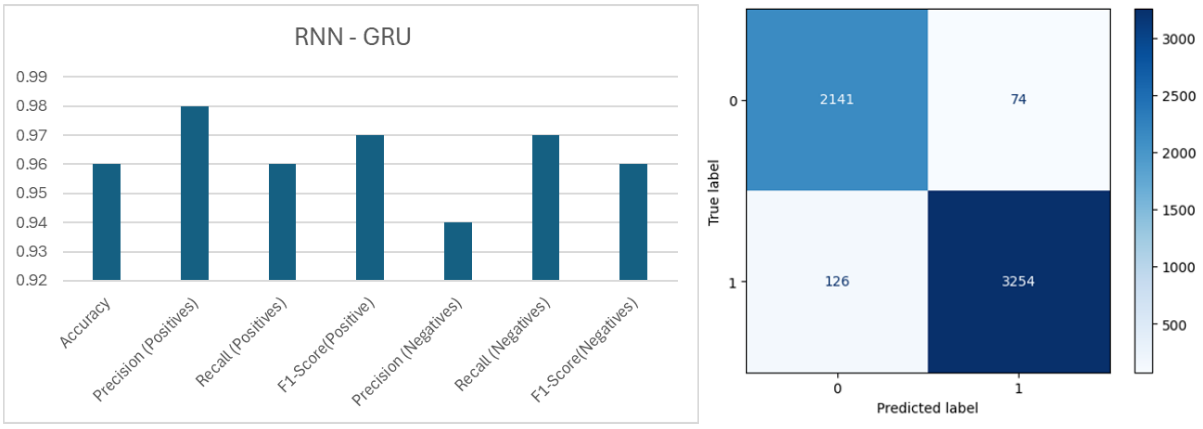


Figure 22: GRU performance

The GRU model exhibits strong performance metrics across the board. It achieves an accuracy of 0.96, indicating a high overall classification rate. For positive cases, the model has a precision of 0.98 and a recall of 0.96, resulting in an F1-score of 0.97. This demonstrates the GRU's effectiveness in accurately identifying positive instances while maintaining a low rate of false positives. For negative cases, the model reports a precision of 0.94 and a recall of 0.97, with an F1-score of 0.96. These results highlight the GRU's ability to reliably detect negative instances, maintaining a balanced performance between precision and recall for both classes.

5.4 Comparative analysis

5.4.1 Accuracy

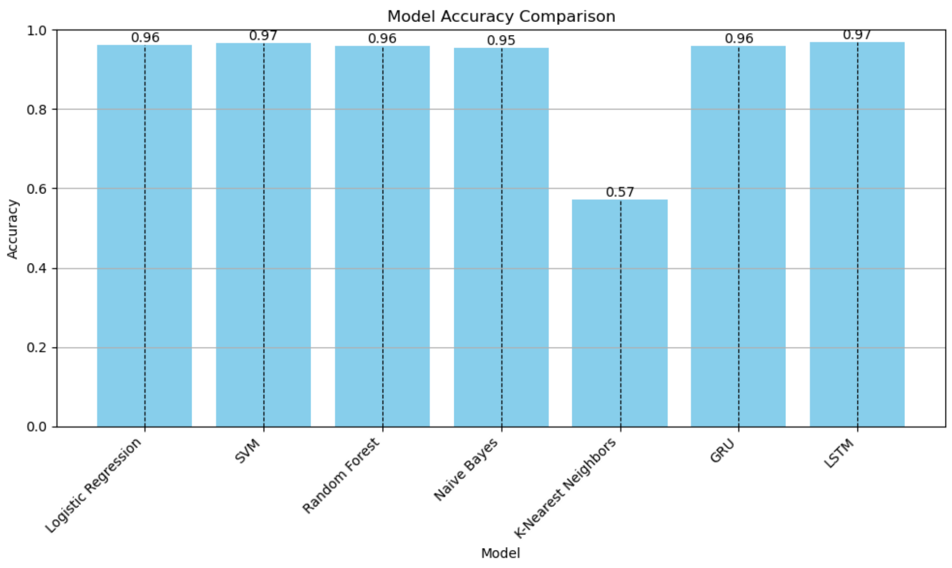


Figure 23: Accuracy comparison of all the models

The graph of the comparison of the accuracy of the models demonstrates the differences between the models. Out of all the models, SVM and LSTM have the greatest accuracy, meaning they work with complex structures in the data stream. Finally, KNN has the lowest accuracy of 0.57 indicating that perhaps the assumption that features are independent is not accurate in this case. Other model's performance lies between these two, perhaps because of the model used

5.4.2 Precision

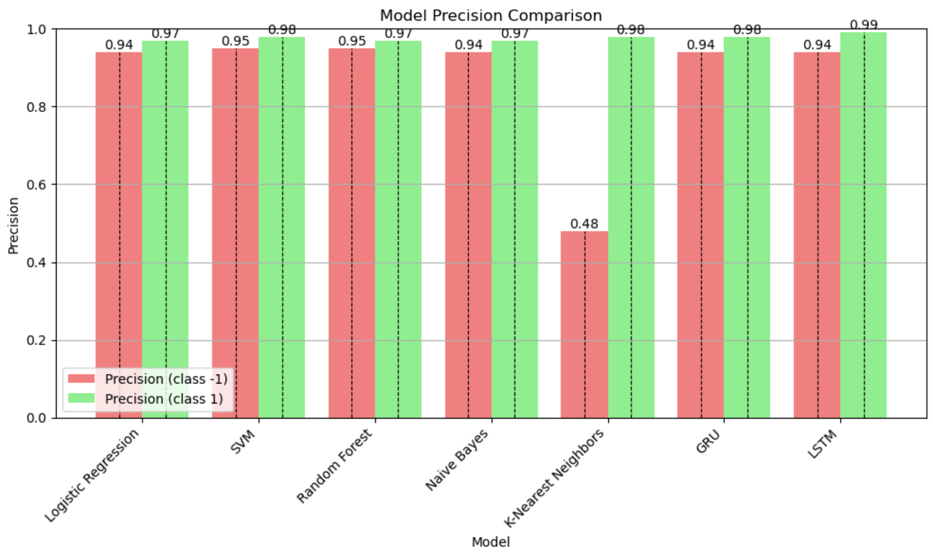


Figure 24: Precision comparison of all the models

The paragraph explains that LSTM has the highest precision for the positive class (1) at 0.99, meaning it makes very few mistakes in identifying positive cases. SVM and GRU also show high precision for the positive class, around 0.98, indicating they are very accurate in identifying true positives. Random Forest and Logistic Regression have slightly lower but still very high precision for the positive class, at 0.97, with Naive Bayes close behind.

In contrast, KNN has high precision for the positive class (0.98) but struggles with the negative class, where its precision drops to 0.48. This means KNN is less effective at correctly identifying negative cases. SVM, GRU, and LSTM perform well in both classes, with precision for the negative class ranging from 0.94 to 0.95. Overall, LSTM, SVM, and GRU show better accuracy across both classes compared to other models, while KNN has notable issues with classifying negative cases.

5.4.3 Recall

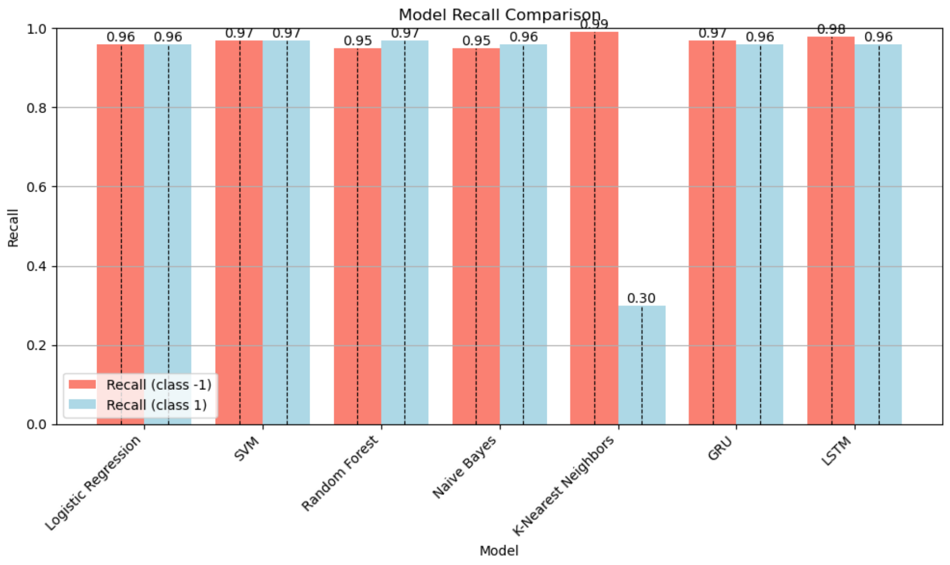


Figure 25: Recall the comparison of all the models

The graph shows that the LSTM model excels in recalling negative cases (-1) with a score of 0.98, indicating it is very effective at correctly identifying negative instances. Similarly, SVM and GRU also perform well in recalling negative cases, with scores of 0.97, which means they are good at minimizing false negatives.

Logistic Regression and Random Forest have equal recall rates of 0.96 for the negative class, showing solid performance. On the other hand, KNN has a high recall of 0.99 for the negative class but a much lower recall of 0.30 for the positive class (1). This contrast highlights that while KNN is very good at identifying negative cases, it struggles significantly with positive cases.

5.4.4 F1-Score

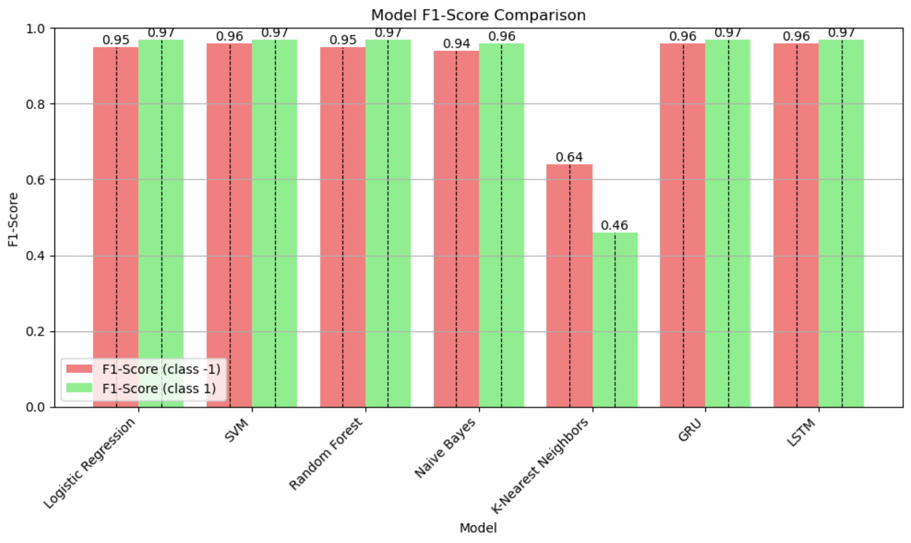


Figure 26: F1 Score comparison of all the models

The graph indicates that the LSTM model excels in terms of the F1 score for both classes, with a particularly high score of 0.97 for the positive class (1). This score reflects a strong balance between precision and recall. SVM and GRU also perform well, each achieving an F1-score of 0.97 for the positive class and slightly lower, but still consistent, scores for the negative class (-1).

Logistic Regression and Random Forest maintain solid F1-scores, with 0.97 for the positive class and 0.95 for the negative class, showing that they are reliable in balancing precision and recall. Naive Bayes has slightly lower F1 scores, with 0.96 for the positive class and 0.94 for the negative class.

KNN, however, displays a significant imbalance, with an F1-score of 0.64 for the negative class and only 0.46 for the positive class. This large disparity indicates that KNN has trouble balancing precision and recall, particularly for positive cases.

5.5 Summary

In all the evaluated measures, accuracy, precision, recall, and F1-score, LSTM and SVM models show the highest results. LSTM performs especially well, and while it has an equal F1 score, this means that it is good at avoiding both false positives and false negatives. SVM also performs well, especially in terms of precision; thus, it can be used effectively for those tasks where a false positive rate is a critical factor. Compared to it, K-Nearest Neighbors (KNN) is the least accurate model, with lower accuracy and a great gap between precision and recall. This suggests that the KNN algorithm is not very consistent and is more likely to make the wrong classification. Logistic Regression, Random Forest and Naive Bayes are all relatively stable, average performers in all metrics, and make for good default classifiers, but are not as excellent as LSTM and SVM. Overall, LSTM and SVM are the most accurate models, while KNN is the least accurate; Logistic Regression, Random Forest, and Naive Bayes are medium-accurate and stable models.

6. Conclusion

The objective of this research is to assess different machine learning models that are used to classify emails concerning phishing attacks. Among the models, K-Nearest Neighbours (KNN), Gated Recurrent Unit (GRU), Logistic Regression, Random Forest, LSTM, Naive Bayes, and Logistic Regression are tested. From the study, the SVM and LSTM models are quite effective with the model achieving accuracies of more than 96%. The algorithms have shown high performance in learning complex patterns inside the phishing data, indicating that they can be used for accurate event discovery. While the LSTM has a strong performance in identifying the long-term dependency in the sequential data, the SVM model is another set of classifiers that provide high accuracy in classifying the data. The two models help each other to refine the classifications that they come up with. However, there are limitations with the KNN model because of the feature independence assumption resulting in the model having a lowest accuracy of 57%. These limitations may not be suitable for the structure of data in phishing since the data is complex and diverse. The precision and recall of the model indicate that it might not be as efficient in detecting phishing attempts as the more complex models. Naïve Bayes and Logistic Regression are considered to have intermediate accuracy. Compared to models of a higher level, these have a rather low capability to analyse the patterns of phishing emails. As much as they seem to be, they are not quite so and may not be easily comprehendible. When it comes to the KNN model, class imbalance is a critical challenge since it significantly affects the model’s performance.

The LSTM and SVM models stand out for their strengths in reducing false positives and accurately predicting successful phishing attempts. While KNN lags in recall and precision for positive phishing detections, the GRU model performs well when recall is consistent across classes. LSTM and Random Forest are the top models for phishing detection, balancing speed and accuracy. Their ability to handle complex data patterns makes them ideal for enhancing phishing detectors. However, as phishing strategies evolve, it's essential to continually refine these models and explore their relationship with other techniques.

References

Ahmed, S., Khan, Z.A., Mohsin, S.M., Latif, S., Aslam, S., Mujlid, H., Adil, M. and Najam, Z., 2023. Effective and efficient DDoS attack detection using deep learning algorithm, multi-layer perceptron. Future Internet, 15(2), p.76.

Ajala, O.A., 2024. Leveraging AI/ML for anomaly detection, threat prediction, and automated response.

Alashhab, A.A., Zahid, M.S., Isyaku, B., Elnour, A.A., Nagmeldin, W., Abdelmaboud, A., Abdullah, T.A. and Maiwada, U., 2024. Enhancing DDoS attack detection and mitigation in SDN using an ensemble online machine learning model. IEEE Access.

Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fa-tima Mohamad Dakalbab, 2021. Machine learning for anomaly detection: A systematic review. IEEE Access, 9:78658–78700, 2021.

Ali, T.E., Chong, Y.W. and Manickam, S., 2023. Machine learning techniques to detect a DDoS attack in SDN: A systematic review. Applied Sciences, 13(5), p.3183.

Alnemari, S. and Alshammari, M., 2023. Detecting phishing domains using machine learning. Applied Sciences, 13(8), p.4649.

Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A.V. and Di Franco, F., 2023. The role of machine learning in cybersecurity. Digital Threats: Research and Practice, 4(1), pp.1-38.

Aslam, N., Srivastava, S. and Gore, M.M., 2024. A comprehensive analysis of machine learning and deep learning-based solutions for DDoS attack detection in SDN. Arabian Journal for Science and Engineering, 49(3), pp.3533-3573.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic,´ Pavel Laskov, Giorgio Giacinto, and Fabio Roli, 2013. Evasion Attacks against Machine Learning at Test Time, page 387–402. Springer Berlin Heidelberg, 2013.

Cyril Goutte and Eric Gaussier, 2005. A probabilistic interpretation of precision, recall and f-score, with implication for evalua-tion. volume 3408, pages 345–359, 04 2005.

Dakota Aaron McCarty, Hyun Woo Kim, and Hye Kyung Lee, 2020. Evaluation of light gradient boosted machine learning technique in large scale land use and land cover classification. Environments, 7(10), 2020.

Daniel Belete and Manjaiah D H, 2021. Grid search in hyperparameter optimization of machine learning models for prediction of HIV/aids test results. International Journal of Computers and Applications, 44:1–12, 09 2021.

Enzo Grossi and Massimo Buscema, 2008. Introduction to artificial neural networks. European journal of gastroenterology hepatology, 19:1046–54, 01 2008.

Felekidis, E., Malicious website detection using machine learning techniques on imbalanced datasets.

Guang-Bin Huang and Lei Chen, 2006. Enhanced random search-based incremental extreme learning machine. Neurocomput-ing, 71(16):3460–3468, 2008. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).

Haibo He, Yang Bai, Edwardo Garcia, and Shutao Li., 2008, Adasyn: Adaptive synthetic sampling approach for imbalanced learning. pages 1322 – 1328, 07 2008.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, 2015. Explaining and harnessing adversarial examples, 2015.

James Bergstra and Y. Bengio, 2012. Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13:281–305, 03 2012.

Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maq-sood, 2012. Random forests and decision trees. International Journal of Computer Science Issues (IJCSI), 9, 09 2012.

Joanne Peng, Kuk Lee, and Gary Ingersoll, 2002. An introduction to logistic regression analysis and reporting. Journal of Educational Research - J EDUC RES, 96:3–14, 09 2002.

Jose´ A. Saez,´ Bartosz Krawczyk, and Michal Wozniak, 2016. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognition, 57:164–178, 09 2016.

Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S.B. and Joga, S.R.K., 2023. Phishing detection system through hybrid machine learning based on URL. IEEE Access, 11, pp.36805-36822.

Kuraku, D.S. and Kalla, D., 2023. Phishing Website URL Detection Using NLP and Machine Learning Techniques. Journal on Artificial Intelligence-Tech Science.

Matteo Re and Giorgio Valentini, 2012. Ensemble methods: A review, pages 563–594. 01 2012.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li, 2021. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning, 2021.

Mengran Zhu, Ye Zhang, Yulu Gong, Changxin Xu, and Yafei Xiang, 2024. Enhancing credit card fraud detection: A neural network and smote integrated approach. Journal of Theory and Practice of Engineering Science, 4:23–30, 02 2024.

Mohammed Zaid, M.S., Namratha, N. and Yashaswini, B.V., 2023. Examining the emerging threat of Phishing and DDoS attacks using Machine Learning models

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer., 2008, Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321–357, June 2002.

Nicholas Carlini and David Wagner, 2017. Towards evaluating the robustness of neural networks. pages 39–57, 05 2017.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami, 2017. Practical black-box attacks against machine learning, 2017.

Pranali Landge, S Swati, Sherekar, and Editor Ijmtst, 2023. Machine learning approaches for prediction and prevention of cyber-attacks for cyber security. International Journal for Modern Trends in Science and Technology, 9:89–94, 10 2023.

Ramprasath, J., Krishnaraj, N. and Seethalakshmi, V., 2024. Mitigation services on SDN for distributed denial of service and denial of service attacks using machine learning techniques. IETE Journal of Research, 70(1), pp.70-81.

Shirazi, H., Muramudalige, S.R., Ray, I., Jayasumana, A.P. and Wang, H., 2023. Adversarial autoencoder data synthesis for enhancing machine learning-based phishing detection algorithms. IEEE Transactions on Services Computing, 16(4), pp.2411-2422.

Shombot, E.S., Dusserre, G., Bestak, R. and Ahmed, N.B., 2024. An application for predicting phishing attacks: A case of implementing a support vector machine learning model. Cyber Security and Applications, 2, p.100036.

Tan, Choon Lin (2018), "Phishing Dataset for Machine Learning: Feature Evaluation", Mendeley Data, V1, doi: 10.17632/h3cgnj8hft.1

Thakur, K., Ali, M.L., Obaidat, M.A. and Kamruzzaman, A., 2023. A systematic review on deep-learning-based phishing email detection. Electronics, 12(21), p.4545.

Tianqi Chen and Carlos Guestrin, 2016. Xgboost: A scalable tree-boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16. ACM, August 2016.

Varun Chandola, Arindam Banerjee, and Vipin Kumar, 2009. Anomaly detection: A survey. ACM Comput. Surv., 41, 07 2009.

Weimin Zhao, Sanaa Alwidian, and Qusay Mahmoud, 2022. Adversarial training methods for deep learning: A systematic review. Algorithms, 15:283, 08 2022.

Yann LeCun, Y. Bengio, and Geoffrey Hinton, 2015. Deep learning. Nature, 521:436–44, 05 2015.

Zhiqiang Gong, Ping Zhong, and Weidong Hu, 2019. Diversity in machine learning. IEEE Access, PP:1–1, 05 2019.