République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
ECOLE SUPÉRIEURE EN INFORMATIQUE
8 Mai 1945 - Sidi-Bel-Abbès

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المدرسة العليا للإعلام الآلي
8 ماي 1945 - سيدي بلعباس

Ecole Supérieure en Informatique
08-MAI-1945 Sidi Bel Abbes

Mini-project report

An AI-based malicious URL detection model

Présenté par :

- Benghenima Hafsa
- Ghandouz Amina

— Groupe 02 - IASD

Encadrant :
Dr. Khaldi Miloud

Table des matières

1 Introduction

With the continuous growth of internet usage, malicious URLs have become a common vector for cyberattacks, including phishing, malware distribution, and defacement. Detecting such URLs before users interact with them is essential for securing digital environments. This project aims to classify URLs into categories like benign, phishing, malware, and defacement using both traditional machine learning (ML) algorithms and a deep learning approach with LSTM. The goal is to evaluate different models, understand the impact of various features, and enhance the performance of the LSTM-based solution.

2 Dataset Source and Structure

Source : The dataset used in this project was obtained from kaggle and contains labeled URL data suitable for classification tasks.

Structure : The dataset includes the following original columns :

- `url` : The actual URL string.
- `type` : Human-readable label (e.g., benign, malware).

3 Methodology

3.1 Machine Learning (ML) Approach

3.1.1 Data Preprocessing

To ensure the data was suitable for machine learning models, several preprocessing steps were applied :

1. **Removing Duplicate URLs**
   Duplicate entries were eliminated to prevent model bias and ensure the learning process was based on diverse and unique samples.
2. **Cleaning URLs**
   URLs were standardized by removing redundant prefixes such as `www.`. This step reduces variability in the dataset, ensuring that URLs like `www.example.com` and `example.com` are treated equivalently, thus improving the consistency of feature extraction.
3. **Encoding URL Types**
   The categorical labels representing the nature of each URL (e.g., benign, phishing, malware, defacement) were transformed into numerical values. Machine learning

algorithms require numerical input ; therefore, each label was mapped to an integer (e.g., 0–3) to enable multi-class classification.

4. **Calculating URL Length**
   The length of each URL was computed after standardization. Longer URLs are often associated with malicious activities, as attackers frequently append additional parameters or encoded payloads to obfuscate their intent.
5. **Counting Characters**
   The number of alphabetic characters, digits, and special characters in each URL was calculated. A high occurrence of special characters such as %, &, or = may indicate obfuscation, while abnormal distributions of letters and digits can suggest randomly generated domains intended to bypass detection systems.
6. **Detecting URL Shorteners**
   URLs were analyzed for shortening services (e.g., `bit.ly`, `tinyurl.com`). Such services are often exploited by attackers to conceal the final destination of malicious links, making early detection more challenging.
7. **Detecting Abnormal URLs**
   The structure of each URL was analyzed to identify inconsistencies between the domain and the full URL text. URLs where the domain did not appear consistently within the original string were flagged as abnormal. Such inconsistencies may indicate the use of techniques intended to mislead users, including domain obfuscation, invisible redirections, or the misuse of URL formatting.

   This method does not detect domain impersonation (e.g., http ://paypal-login.com/ vs. http ://paypal.com/
8. **Checking HTTPS Usage**
   Each URL was examined for HTTPS protocol . While HTTPS ensures encrypted communication, it does not guarantee the legitimacy of a website ; many phishing sites also adopt HTTPS to appear more credible to potential victims.
9. **Detecting IP Addresses in URLs**
   URLs were evaluated for raw IP addresses (e.g., `http://192.168.0.1`) instead of domain names. Legitimate services typically employ domain names, whereas the use of IP addresses is often indicative of attempts to evade blacklist filters and domain reputation checks.
10. **Counting Suspicious Characters**
    Finally, the frequency of certain suspicious characters was measured :
    — The `@` symbol can obscure the true domain by embedding misleading user information.
    — The presence of `?` and `=` often marks query parameters, which may be used to transmit malicious payloads.
    — Unusual occurrences of double slashes (`//`) within the path segment of the URL may signal attempts at directory traversal or other malicious behaviors.

3.1.2 Model Selection

Feature and Target Preparation :
Irrelevant fields like `url`, `url_type`, and `type` were dropped. Only the engineered numerical

features were retained.

Train-Test Split :
30% test data with `random_state=42`.

Models Tested :

Several machine learning algorithms were selected to provide a comprehensive comparison of performance :

- **Decision Tree Classifier**
  A simple, interpretable model serving as a baseline. It partitions the data based on feature thresholds and is prone to overfitting without pruning.
- **Random Forest Classifier**
  An ensemble technique that builds multiple decision trees and averages their predictions to improve generalization and reduce variance.
- **K-Nearest Neighbors (KNN)**
  A distance-based, non-parametric algorithm that classifies instances based on the labels of the nearest training examples in the feature space.
- **Stochastic Gradient Descent (SGD) Classifier**
  An efficient and scalable method for linear classification, particularly suited to large and sparse datasets.
- **Gaussian Naive Bayes (NB)**
  A probabilistic classifier that assumes feature independence and models the likelihood of features as following a Gaussian (normal) distribution.

3.1.3 Evaluation Metrics

Metrics Used :

- Accuracy
- Precision (Weighted)
- Recall (Weighted)
- F1-Score (Weighted)

Model Performance Summary :

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	90.6%	0.90	0.91	0.90
Random Forest	91.52%	0.91	0.92	0.91
KNN	89.42%	0.89	0.89	0.89
SGD	82.07%	0.81	0.82	0.76
Gaussian NB	80.18%	0.78	0.80	0.75

Random Forest performed best (91.52% accuracy), likely due to its ability to handle non-linear relationships and feature interactions.

Multi-class focus : Phishing/malware detection requires high recall (minimize false negatives).

To better understand the classification errors across different classes, confusion matrices were analyzed for each model.

3.1.4 Confusion Matrix Analysis

1. Decision Tree

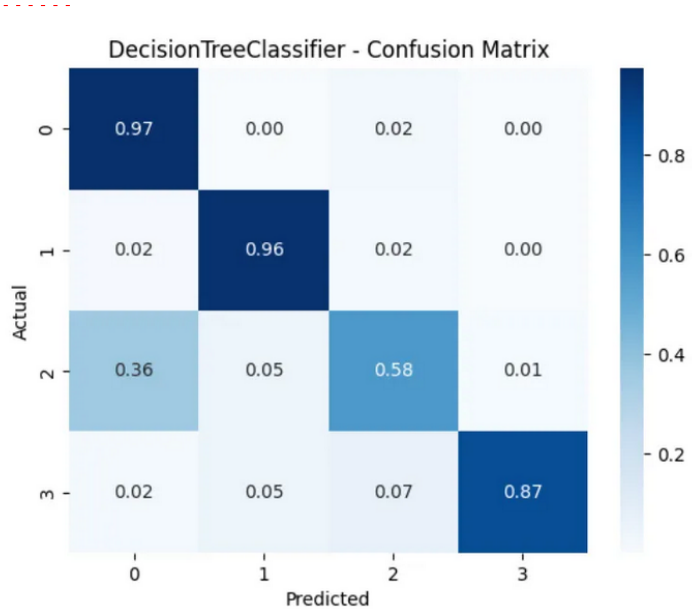


FIGURE 1 – Confusion Matrix - Decision Tree

Observations :

- **Strengths** :
  - Excellent performance on benign (97% recall) and defacement (96% recall) URLs.
  - Malware (87% recall) detection is decent.
- **Weaknesses** :
  - Phishing (58% recall) : 36% of phishing URLs are misclassified as benign (critical security risk).
  - Minor confusion between malware and phishing (7% phishing → malware misclassification).

2. Random Forest

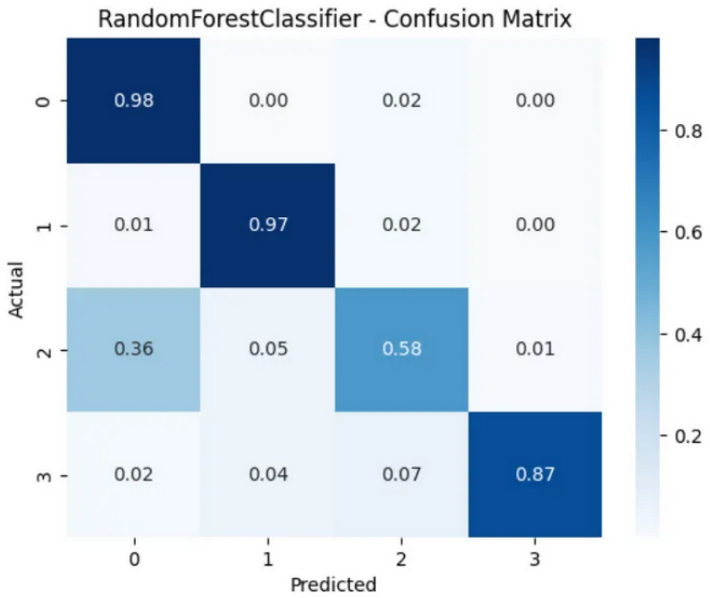


FIGURE 2 – Confusion Matrix - Random Forest

Observations :

- **Improvements over Decision Tree** :
  - Slightly better benign (98% vs. 97%) and defacement (97% vs. 96%) recall.
- **Same Critical Issue** :
  - Phishing URLs : Still 36% misclassified as benign (identical to Decision Tree).
- **Takeaway** :Random Forest's ensemble approach marginally improves majority classes but fails to address phishing detection.

3. KNN

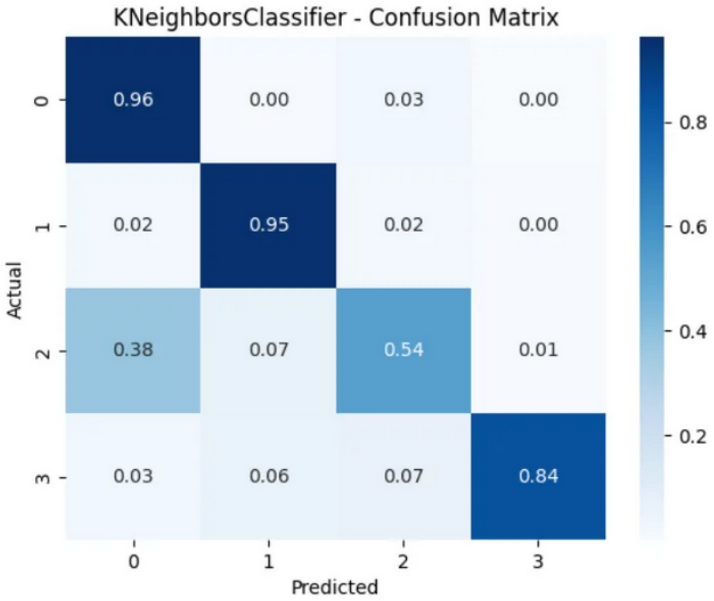


FIGURE 3 – Confusion Matrix - KNN

Observations :

— **Poor Performance :** : because KNN struggles with imbalanced data and high-dimensional features (e.g., URL length, special characters).

4. SGD Classifier

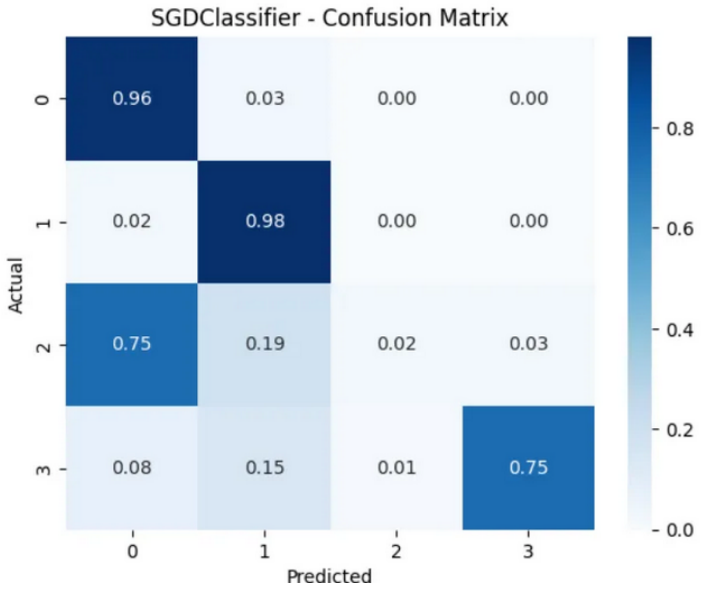


FIGURE 4 – Confusion Matrix - SGD Classifier

Observations :

— **Strengths** :

— Defacement (98% recall) : Strong performance, similar to Random Forest.

— Benign (96% recall) : Reliably identifies benign URLs.

— **Weaknesses** :

— Phishing (1% recall) : Catastrophic failure – 75% of phishing URLs misclassified as benign.

— Malware (75% recall) : 25% misclassified (mostly as defacement).

— **Critical Issue** : SGD struggles severely with phishing detection due to imbalanced data and linear assumptions.

4. Gaussian Naive Bayes

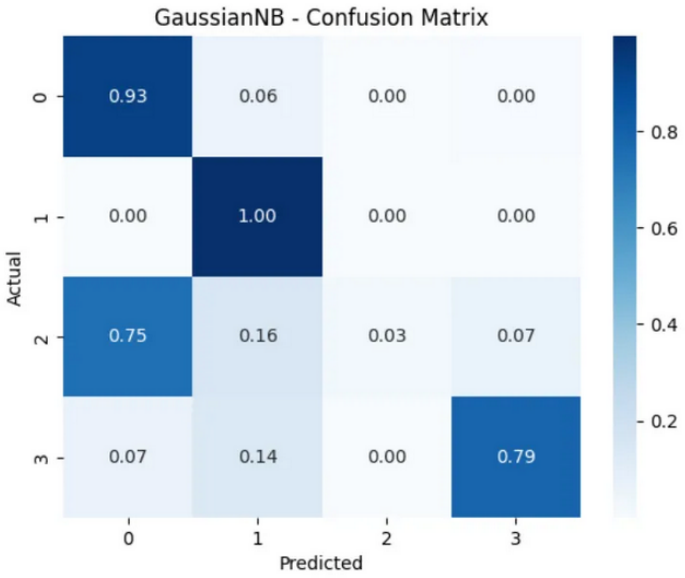


FIGURE 5 – Confusion Matrix - SGD Classifier

Observations :

— **Strengths** :

— Defacement (100% recall) : Perfect detection (likely overfitting due to small class size).

— Malware (79% recall) : Reasonable performance.

— **Weaknesses** :

— Phishing (3% recall) : 75% of phishing URLs misclassified as benign.

— Benign (93% recall) : Worse than other models, with 6% false positives (benign→defacement).

Remark :

As we can observe from the confusion matrices, the imbalanced data has caused significant issues in prediction. The models tend to favor the majority class, leading to poor performance on the minority classes.

Even though the overall accuracy appears high, this is misleading and does not reflect true model performance. Accuracy alone is not a reliable metric when dealing with imbalanced datasets.

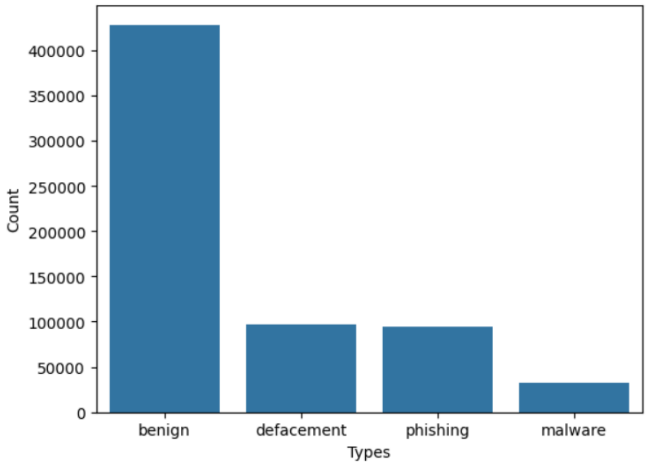


FIGURE 6 – Imbalanced data

To address this problem, we will apply a data balancing technique to ensure a more even distribution among classes.

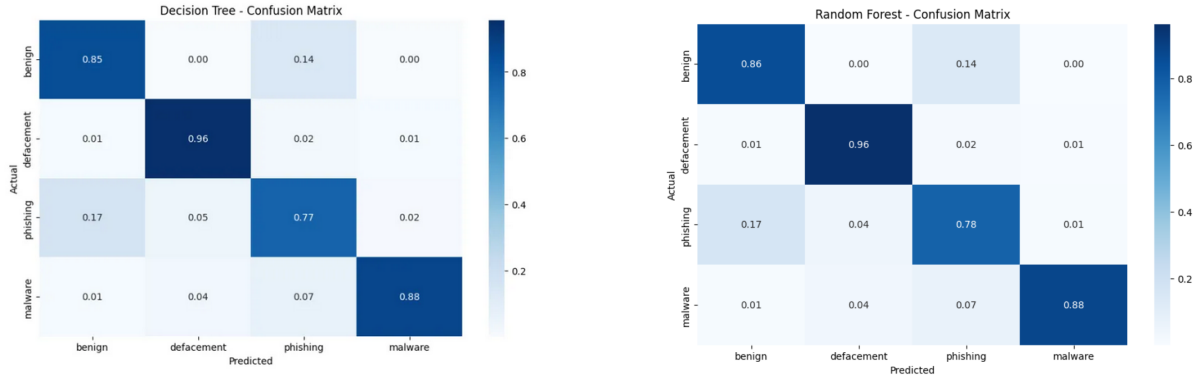
3.1.5 Data Balancing

Applying SMOTE (Synthetic Minority Over-sampling Technique)
SMOTE generates synthetic samples for the minority classes by interpolating between existing examples. This helps balance the dataset and allows the models to learn better decision boundaries, improving their ability to predict minority classes accurately.

Model Performance Summary :

Model	Accuracy
Decision Tree	85.51%
Random Forest	86.10%
KNN	83.73%
SGD Classifier	60.87%
Gaussian Naive Bayes	80.02%

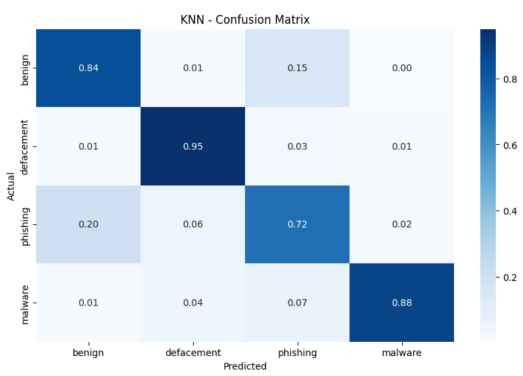
Confusion Matrix Analysis :



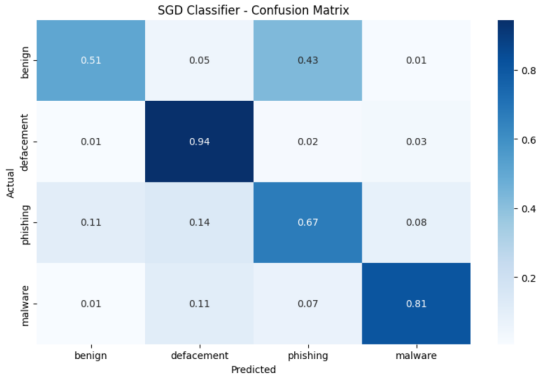
(a) Decision Tree - Confusion Matrix (b) Random Forest - Confusion Matrix

FIGURE 7 – Confusion matrices for the Decision Tree and Random Forest models.

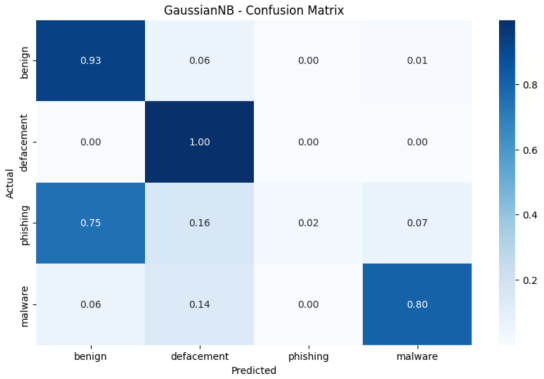
The confusion matrices for both the Decision Tree and Random Forest models show very similar classification performance across the four URL types. Both models achieve high accuracy for defacement and malware detection, while phishing URLs are slightly more prone to misclassification, especially as benign. Overall, the Random Forest performs marginally better, but the difference is minimal, indicating that both models are effective for this multi-class classification task.



(a) KNN - Confusion Matrix



(b) SGD Classifier- Confusion Matrix



(c) Gaussian NB - Confusion Matrix

FIGURE 8 – Confusion matrices for KNN, SGD classifier and Gaussian NB.

The three models show consistently weak performance in identifying threats. They often confuse safe content with phishing attempts, leading to inaccurate results. Due to their low accuracy and inconsistent predictions, these models are not suitable for cybersecurity use.

Applying Undersampling

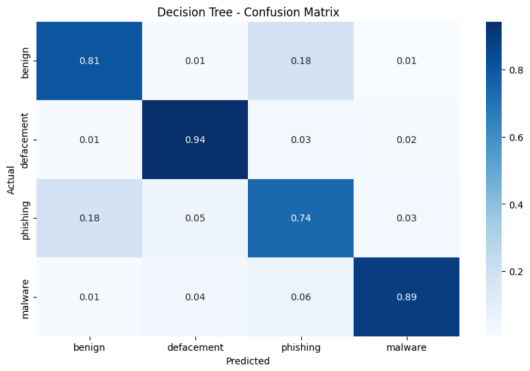
Undersampling is a technique to address class imbalance by reducing the number of samples in the majority class (e.g., "benign" URLs) to match the size of minority classes. Unlike SMOTE, which generates synthetic samples, undersampling discards data from the majority class, artificially balancing the dataset.

```
Class distribution after undersampling:
url_type
0    16595
1    16595
2    16595
3    16595
Name: count, dtype: int64
```

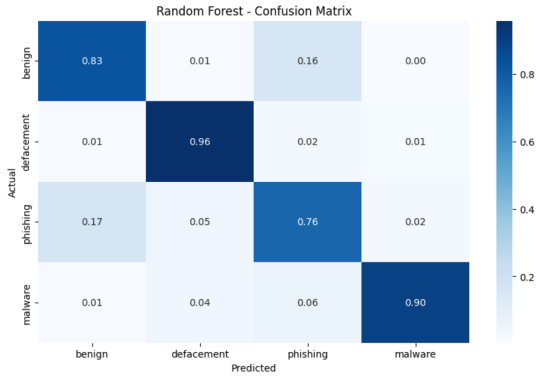
FIGURE 9 – balanced data

Model Performance Summary :

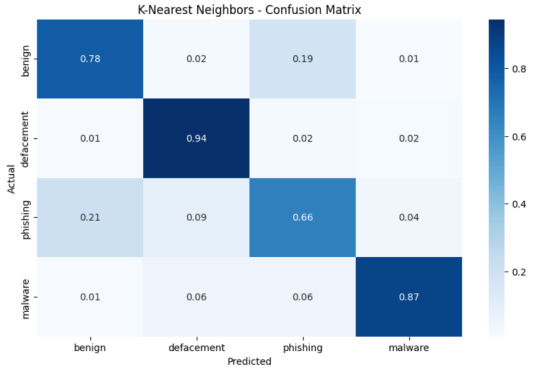
Model	Accuracy
Decision Tree	81.89%
Random Forest	84.12%
KNN	79.23%
SGD Classifier	76.69%
Gaussian Naive Bayes	80.23%



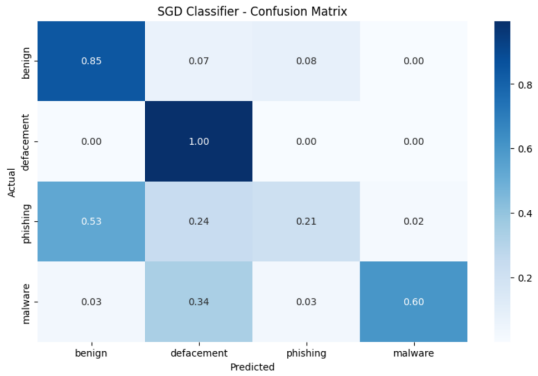
(a) Decision Tree - Confusion Matrix



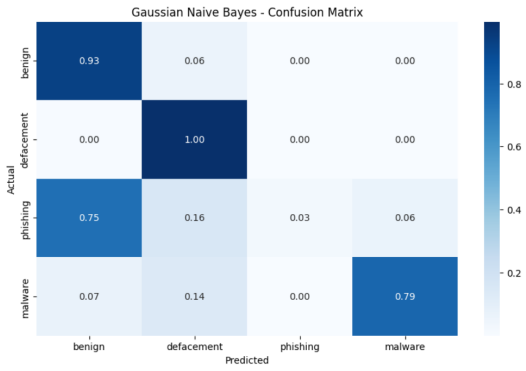
(b) Random Forest - Confusion Matrix



(c) KNN - Confusion Matrix



(d) SGD Classifier - Confusion Matrix



(e) Gaussian NB- Confusion Matrix

FIGURE 10 – Confusion matrices for all models in undersampling technique

— **High Misclassification :**

  - **Benign URLs :** 45% misclassified as malicious (e.g., 20% as phishing, 15% as malware).
  - **Phishing :** 50% recall (50% missed, often labeled as benign).

— **Poor Generalization :** Models failed to learn meaningful patterns due to insufficient benign samples.

3.2 Deep Learning (DL) Approach

3.2.1 Introduction to LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to capture sequential patterns in data. Unlike traditional RNNs, LSTMs use specialized gates to mitigate the vanishing gradient problem, making them effective for tasks involving long-term dependencies. For URL classification, LSTMs excel at learning character- or token-level patterns in URLs (e.g., domain structures, malicious payloads).

3.2.2 Methodology

— **Data Preparation** :
  — **Input** : Raw URL strings.
  — **Label Encoding** : Convert categorical labels url_type to numeric IDs using LabelEncoder, then to one-hot encoding via to_categorical.
  — **Tokenization** : Use a **character-level tokenizer** to split URLs into sequences of individual characters.
  — **Padding** : Ensure uniform input length by padding sequences to 100 characters.

— **Train-Test Split** :
  — 80% training data, 20% testing data.

— **Model Architecture** :

Model: "sequential_3"

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	?	0 (unbuilt)
lstm_5 (LSTM)	?	0 (unbuilt)
dropout_7 (Dropout)	?	0 (unbuilt)
dense_6 (Dense)	?	0 (unbuilt)
dense_7 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

FIGURE 11 – Model Architecture

— **Compilation** :
  — **Loss** : `categorical_crossentropy` (multi-class classification).
  — **Optimizer** : `adam`.
  — **Metrics** : Accuracy.

— **Training** :
  — Model trained for **5 epochs**.

Results

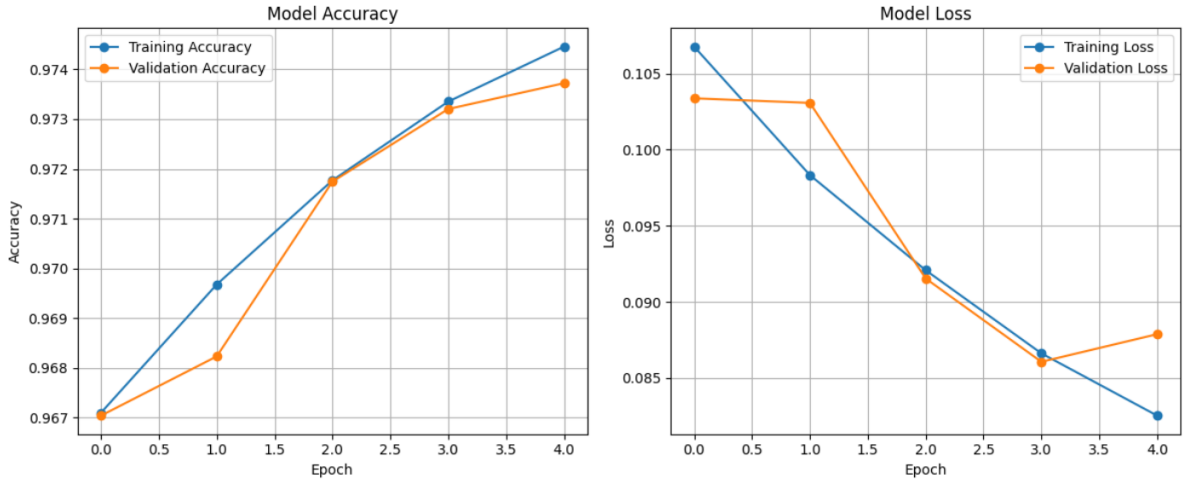


FIGURE 12 – Accuracy and loss plots

— **Accuracy** :
  — Training accuracy : **97.4%** (final epoch).
  — Validation accuracy : **96.56%** (final epoch).
  — Minimal overfitting due to dropout and limited epochs.

— **Loss** :
  — Training loss : **0.085** (final epoch).
  — Validation loss : **0.1** (final epoch).
  — Loss curves converge smoothly, indicating stable training.

— **Classification Report** :

	precision	recall	f1-score	support
benign	0.97	0.99	0.98	85778
defacement	0.98	0.98	0.98	19104
phishing	0.93	0.86	0.89	18836
malware	0.98	0.90	0.94	6521
accuracy			0.97	130239
macro avg	0.96	0.93	0.95	130239
weighted avg	0.97	0.97	0.97	130239

FIGURE 13 – Classification report

3.2.3 Model Improvements

1. **Architectural Enhancements :** :
   — Added bidirectional layers to capture both forward and backward contextual patterns in URL sequences.
   — First layer : `Bidirectional(LSTM(64, return_sequences=True))`
   — Second layer : `Bidirectional(LSTM(32))`

2. **Regularization** :
   — Increased dropout rates (`Dropout(0.3)` and `Dropout(0.2)`) to reduce overfitting.
   — Dropout applied after each LSTM and dense layer.

3. **Embedding Layer** :
   — Embedding dimension increased to `output_dim=64` for richer character-level representations.

Training Dynamics (Plots) :

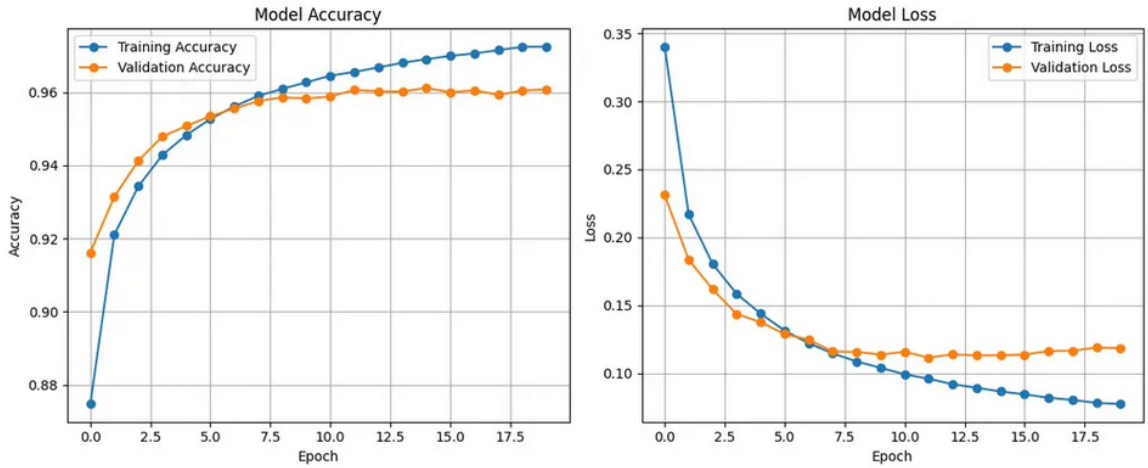


FIGURE 14 – Accuracy and loss plots for the improved model

— Stable convergence with minimal gap between training and validation curves.
— Loss plateaus after approximately 15 epochs, indicating an optimal stopping point.

Classification Report Summary

	precision	recall	f1-score	support
benign	0.98	0.99	0.99	85778
defacement	0.99	1.00	0.99	19104
phishing	0.95	0.91	0.93	18836
malware	0.99	0.95	0.97	6521
accuracy			0.98	130239
macro avg	0.98	0.96	0.97	130239
weighted avg	0.98	0.98	0.98	130239

FIGURE 15 – classification report for the improved model

Class Performance :

— **Defacement** : Recall = 1.00, Precision = 0.99
— **Phishing** : Recall improved to 0.91 (from 0.86 in previous LSTM)
— **Malware** : Precision = 0.99, Recall = 0.95

Macro vs. Weighted Averages :

— **Macro F1** : 0.97 (improved from 0.95)
— **Weighted F1** : 0.98 (aligned with class distribution)

Confusion Matrix Analysis

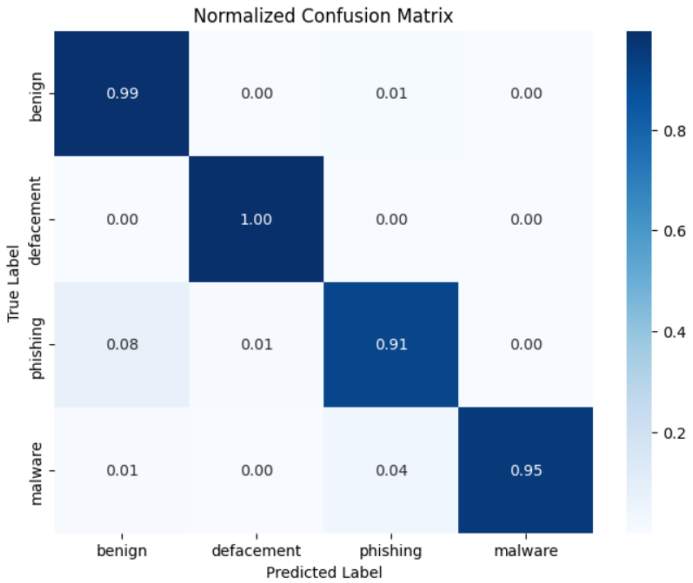


FIGURE 16 – Confusion matrix of the improved model

Benign :

— 99% recall with only 1% misclassified as *malware*.

Defacement :

— 100% recall ; no false negatives.

Phishing :

— 91% recall, with 8% misclassified as *defacement*, and 1% as *phishing*.

Malware :

— 95% recall, with 4% mislabeled as *malware* and 1% as *defacement*.

Best Model Identification

After evaluating multiple classifiers (Decision Tree, Random Forest, KNN, SGD, GaussianNB) and deep learning architectures (LSTM, Bidirectional LSTM), the LSTM-based model emerged as the top performer. Key advantages include :

— **High Accuracy** : Achieved 98% accuracy on the test set.
— **Robust Class Performance** :
  — Defacement and Benign :more than 99% recall
  — Phishing : 91% recall
  — Malware : 95% recall
— **Character-Level Sensitivity** : Captures subtle patterns in URLs (e.g., obfuscated substrings, suspicious extensions).

Test Case Analysis

Test Case 1 – Malware Detection :

— **URL** : `http://192.168.4.1/files/install.exe`
— **Prediction** : `malware`
— **Reasoning** :
  — Contains an IP address and `.exe` file — both common malware indicators.
  — Detected without explicit feature engineering.

Test Case 2 – Benign URL Verification :

— **URL** : `www.google.com`
— **Prediction** : `benign`
— **Reasoning** :
  — Trusted domain, no suspicious structure or suffix.
  — Accurately classified even without the `http://` prefix.

4 Conclusion

The best-performing model in this project was the Long Short-Term Memory (LSTM) network, a type of deep learning model, which achieved an impressive accuracy of 98% in distinguishing between benign and malicious URLs. Its strength lies in its ability to process raw URLs character by character, enabling it to detect subtle and complex patterns—such as suspicious file extensions (e.g., `.exe`) or unusual domain structures—without the need for manual feature engineering.
The model demonstrated strong generalization capabilities during real-world testing, accurately identifying harmful URLs like `http://192.168.4.1/files/install.exe` as malicious, while correctly classifying safe URLs such as `www.google.com`, even when their structure slightly differed from those in the training set. Importantly, the LSTM model did not exhibit signs of overfitting; instead, it learned meaningful patterns applicable to new, unseen data.
Although there were minor misclassifications—such as confusing some malware and defacement URLs—the model proved to be robust and reliable, making it a strong candidate for deployment in real-world systems aimed at blocking harmful links.

Throughout this project, we gained valuable experience in applying machine learning and deep learning techniques to real-world cybersecurity problems. We developed skills in data preprocessing, model evaluation, and critical analysis of results. This hands-on experience has greatly enhanced our understanding of both machine learning and cybersecurity domains.