Hyperparameter Optimization for Malicious URL Detection: Leveraging Optuna and Random Search in Machine Learning and Deep Learning Models

Miloud Khaldi, Zohra Alilat, Hana Bendoubba, and Nadir Mahammed
LabRI-SBA Lab., Ecole Supérieure en Informatique, Sidi Bel Abbès, Algeria
E-mail: m.khaldi@esi-sba.dz, z.alilat@esi-sba.dz, h.bendoubba@esi-sba.dz, n.mahammed@esi-sba.dz

Keywords: Malicious URLs, hyperparameter optimization, Optuna, random search, BERT, LightGBM

Received: May 2, 2025

Uniform Resource Locators (URLs) are critical indicators for identifying malicious online activities such as malware distribution, phishing attacks, and website defacement. This research presents a robust approach for detecting these threats using both deep learning (DL) and machine learning (ML) techniques. We emphasize hyperparameter optimization, employing Optuna—a Bayesian optimization framework—and Random Search to systematically enhance model performance. Unlike many prior studies, which often overlook thorough hyperparameter tuning, our approach demonstrates improvements over state of the art methods. Our Bidirectional Encoder Representations from Transformers (BERT) model achieved an accuracy of 98.84%, with an F1 score of 99.02%, while the Light Gradient Boosting Machine (LightGBM) attained an accuracy of 98.46% and an F1 score of 98.45%

Povzetek: Prispevek predstavi optimizacijo hiperparametrov za zaznavanje zlonamernih URL-jev z uporabo Optuna in Random Search v modelih BERT in LightGBM, kar izboljša zaznavanje nevarnosti.

1 Introduction

The Internet has experienced rapid expansion in recent years, becoming an essential part of daily life in the digital age. It serves as a global communication platform and a vast repository of information across sectors such as government, business, academia, and personal use. With the widespread adoption of smartphones and increasing connectivity, users can now access online content from anywhere. However, this accessibility also exposes them to both reliable and malicious sources, increasing their vulnerability to cyberattacks and online fraud.

Malicious websites are frequently employed in cyberattacks to steal sensitive user data. Attackers often create counterfeit websites that closely resemble legitimate ones, deceiving users into unknowingly submitting personal information [1]. According to the Anti-Phishing Working Group (APWG), phishing attacks are escalating. In Q3 2024 alone, 932,923 phishing attempts were recorded—an increase from the previous quarter—with social media platforms being the most targeted sector, accounting for 30.5% of all attacks [2]. Additionally, reports indicate that over 40,000 new malicious URLs are created daily, resulting in estimated financial losses of $17,700 per minute [3].

One notable example is the February 2025 Bybit exchange hack, in which attackers exploited standard wallet transfer procedures to steal approximately $1.5 billion in Ethereum [4]. This incident highlights the evolving sophistication of cyber threats and the urgent need for robust detection mechanisms.

URLs serve as the unique identifiers of websites, determining their location and access methods. When users access a URL, it typically connects to a server's database and returns the website's content. URLs may be benign or malicious. In phishing attacks, malicious URLs closely imitate legitimate ones, often distributed via email or deceptive ads. Once clicked, these links may download malware or redirect users to credential-stealing sites. Given the large number of new websites created daily and the growing sophistication of cyberattacks, accurately distinguishing between benign and malicious URLs has become increasingly challenging.

Due to the structural similarities between benign and malicious URLs, effective classification requires extracting and analyzing specific features. The ease with which attackers can now create harmful URLs underscores the need for automated, scalable, and accurate detection methods. Recent research has turned to Machine Learning (ML) approaches to enhance malicious website detection. However, most existing solutions focus on individual web pages and rely on manually extracted features (such as text or images), limiting their scalability and generalization across diverse domains [5, 6].

In this paper, we propose an advanced approach for malicious URL detection by integrating deep learning and traditional ML with automated hyperparameter optimization. Specifically, we develop two models: (1) a BERT-based classifier enhanced with Optuna for automated hyperparameter tuning, leveraging BERT's powerful contextual understanding to capture subtle patterns in URL structures, and (2) a lightweight yet effective LightGBM classifier optimized with Random Search to balance performance and computational efficiency.

The primary contributions of this work are as follows:

– We introduce a BERT-Optuna model that combines contextual language modeling with automated hyperparameter tuning to improve malicious URL detection accuracy.

– We develop a LightGBM-Random Search model to efficiently explore hyperparameter space for robust and interpretable detection results.

– We conduct a comprehensive evaluation of the proposed models on a real-world dataset of over 650,000 URLs.

– We perform a comparative analysis against state-of-the-art classifiers to validate the effectiveness of our approach.

The remainder of this paper is structured as follows: Section 2 reviews related work. Section 3 presents our preprocessing techniques. Section 4 outlines the classification methods. Section 5 describes the proposed models. Section 6 details the experimental setup and evaluation metrics, and Section 7 presents and discusses the results. Finally, Section 8 concludes the paper and outlines directions for future research.

2 Related research

In this section we are going to explain the methods used to detect malicious URLs. Prior studies have explored harmful URL detection and classification via diverse ML and DL methods, yielding varying accuracy percentages. These studies have enhanced the comprehension of efficient methodologies and techniques for detecting and categorizing malicious URLs. In order to evaluate the contribution of these methodologies in comparison to earlier research, we have collected different contents of articles and sources to uncover characteristics or trends that separate benign from malicious url. These methodes use various algorithms to classify and detect malicious URLs.

Su and Su[7] suggested BERT-based harmful URL detection. By tokenizing URL strings with BERT, the self-attention mechanism may understand their semantic linkages. The model uses raw URL strings or Random Forest feature strings, depending on the dataset. BERT is fine-tuned against public datasets (Kaggle, GitHub, ISCX 2016) and adaptable to multiple data formats, delivering excellent classification accuracy 98.78%, 96.71%, and 99.98% on the three datasets.

Nanda, Saraswat, and Sharma[8] proposed a phishing URL detection model using Bidirectional Long Short-Term Memory (BiLSTM), Convolutional Neural Networks (CNN), and a Gated Highway Attention (GHA). The model has several key steps: pre-processing URLs to a uniform length of 255 characters; BiLSTM and CNN work together to extract both global and local features; highway networks improve information flow to deal with problems like vanishing gradients; attention mechanisms are used to improve feature learning and reduce the effect of irrelevant data; the model achieved better performance metrics, with an accuracy of 99.78%, precision of 98.97%, recall of 99.35%; beating other traditional ML and DL techniques.

Wang et al.[9] provided a DCNN malicious URL detection model. A dynamic convolution technique replaces the pooling layer with a k-max pooling layer and adds a folding layer. The pooling settings are dynamically modified based on the input URL length and convolutional depth to extract deeper features. The model uses character-level and word-level embedding to represent unusual words and special characters while minimizing memory usage. This hybrid embedding achieved an accuracy of 98.7% and 98.7% as F1-score.

The work of Sheikhi and Kostakos[6] configures a hybrid approach combining the Firefly Algorithm for feature selection and Particle Swarm Optimization (PSO) for optimizing XGBoost hyperparameter improving malicious website detection. The hybrid model achieved an astonishing 98.42% accuracy and an F1-score of 98.4% in binary classification, while it yielded a multiclass classification accuracy of over 98%.

The research of Gupta et al.[10] shows how ML-based URL lexical characteristics can detect phishing. The technique constructs nine lexical features—token count, URL length, and domain delimiters—and uses Random Forest, K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machine (SVM) classifiers. The approach requires no third-party information, enabling real-time performance with low computational resources. Random Forest had the highest accuracy of 99.57%, a low false positive rate of 0.53%, and a rapid run time of 51.5ms.

Ullah et al.[11] proposed a multi-model visual representation and transformers-based transfer learning explainable malware detection system is presented. The malware categorization method uses network traffic text and images. BERT extracts first-learned textual properties from HTTP and TCP flows. After feature extraction using the FAST extractor and BRIEF descriptor, a malware-to-picture transformation transforms network byte streams into images. After class imbalance is resolved via Synthetic Minority Over-Sampling (SMote), CNN deep feature extraction is performed. A voting-based ensemble learning approach leads the final categorization. Malware detection and classification rates of 98.44% and 99.16%.

Wu et al.[12] proposed an NLP method based on a bidirectional gated recurrent unit (BiGRU).It used The dropout mechanism in the input layer to avoid overfitting, the attention mechanism in the middle layer to learn the correlation between sequences, Word2Vec for the preprocessing to train the word vector of URLs and the BiGRU to extract all the sequence information of URLs. This proposed method (DA-BiGRU) achieved better experimental results in detecting malicious URLs than the MLP, Att-BiLSTM, Att-BiGRU, DA-BiLSTM with an accuracy of 0.9792, precision of 0.9834, reacall of 0.9553 and F1 score of 0.9691.

Alshingiti et al.[13] proposed a DL technique to detect phishing URLs. This methodology employs Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), and a hybrid LSTM−CNN model. The three systems adhere to a four-step process: extracting features, preprocessing data using SelectKBest for feature selection and MinMaxScaler for scaling, training the models, and ultimately categorizing URLs as legitimate or phishing. They refined hyperparameters such as number of layers, dropout rate, learning rate, number of epochs and batch size to improve performance. Among the models, CNN achieved the highest accuracy at 99.2%, surpassing LSTM−CNN at 97.6% and LSTM at 96.8%.

R, Patil, and Mohana[14] proposed a malicious URL detection and classification method using machine learning. The method utilizes a dataset of 651,191 URLs, categorized into Phishing, Benign, Defacement and Malware types. Three machine learning algorithms, Random Forest, LightGBM and XGBoost, were implemented. The models were trained and validated on the dataset, and their performance was evaluated using metrics like accuracy, precision, recall and F1 score. The Random Forest classifier achieved the best results with an accuracy of 96.6%.

Alsaedi et al.[15] introduced the CTI-MURLD model for the detection of malicious URLs, which incorporates various data sources such as URL content features, Google-based Cyber Threat Intelligence (CTI), and Whois-based CTI. Utilizing external threat intelligence. The model comprises several phases: Data preprocessing involves the application of NLP techniques to normalize textual data. Feature extraction through N-gram and TF-IDF. Feature Selection: Mutual Information is employed to select the 5000 most informative features. Ensemble Learning for Classification employs a two-stage ensemble approach: Three Random Forest classifiers are trained independently on separate feature subsets: URL features, Google-CTI, and Whois-CTI. The probabilistic outputs from the RF classifiers are aggregated and input into a Multi-Layer Perceptron (MLP) neural network, which is optimized using the BFGS quasi-Newton algorithm to enhance weight tuning and accelerate convergence. Grid Search was employed to optimize performance, as Random Forest algorithms necessitate meticulous hyperparameter selection, increased accuracy and reduced false positive and false negative rates relative to baseline models. The best result was that of RF + GS with an accurcy of 97.25%,recall of 97,26% and a precision of 97.36%.

Yu et al.[16] The study proposed an improved M-BERT model for malicious URL detection, leveraging Transformer encoders for feature extraction and classification. It outperformed traditional methods, achieving 94.42% precision. Future work focuses on scalability, multimodal data integration, and privacy protection using federated learning.

Maneriker et al.[17] introduce URLTran, a phishing URL detecting transformer-based model. URLTran achieves a true positive rate (TPR) of 86.80% at a 0.01% false positive rate (FPR), a 21.9% improvement over past DL models such as URLNet by adjusting BERT and RoBERTa and including domain-specific pre-training. After adversarial fine-tuning, the model is tested against adversarial attacks like homoglyph replacements and compound word splitting, hence proving increased resilience.

Mankar et al.[18]conducted a comparative evaluation of machine learning models for malicious URL detection. The study utilized a dataset of over 500,000 URLs and examined models including decision trees, random forests, KNN, and Naive Bayes. Feature engineering was employed to transform URLs into numeric features for model training. The Random Forest and Extra Trees ensemble models achieved the best performance, demonstrating over 91% accuracy in distinguishing between benign and malicious URLs. The findings indicate the potential of ensemble machine learning techniques for automated malicious URL detection.

The following table 1 summarizes those findings of several research studies.

3 Methodology

The proposed methods are designed to develop innovative models for accurately detecting malicious websites. The implementation details, dataset, and summary of the proposed models are all detailed in this section.

3.1 Data collection

In this research, we evaluated the proposed models using a malicious URL dataset sourced from [19]. The dataset consists of a total of 651,191 URLs, categorized into four distinct classes: benign, defacement, phishing, and malware. The primary goal is to use this dataset to develop ML and DL models capable of identifying malicious URLs to prevent some of cybersecurity threats.

Distribution of URLs

Here is the distribution of URLs in the dataset as shown in the table 2:

The dataset is curated from five different sources to ensure a comprehensive collection of URL examples. The sources include ISCX-URL-2016, Malware Domain Blacklist, Faizan Git Repository, Phishtank, and PhishStorm datasets [20]. The dataset is structured in a tabular format with two main columns: URL and Label. The URL column contains the actual web addresses, and the Type column indicates the category of each URL (benign, defacement, phishing, or malware) as in Figure 1.

Table 1: Literature review

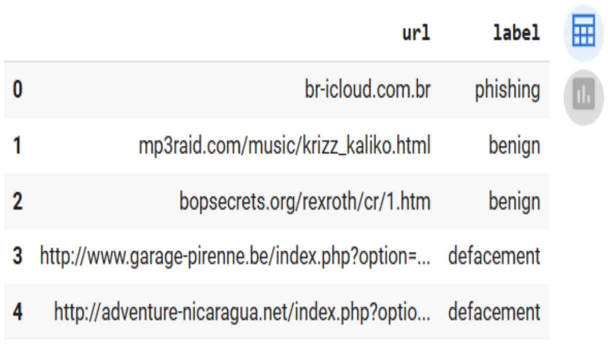
Work	Dataset	Methodology	Feature Engineering	Hyperparams optimization	Results (%)	Limitations
[7], 2023	Kaggle, GitHub, ISCX-URL-2016	BERT+self-attention mechanisms+Classifier	RF + SMOTE	–	Acc1: 98.78, Acc2: 96.71 , Acc3: 99.98 respectively	Limited zero-day attack handling
[9], 2021	GitHub, Kaggle, Alexa	DCNN	Word Embedding Based on Character Embedding	–	Acc: 98.7, F1: 98.7, P: 99.3, R: 98.1	High computational cost
[6], 2024	ISCX-URL-2016	XGBoost	Firefly Algorithm + CFSSubsetEval	Partical Swarm Optimization(PSO)	Binary Acc: 98.42, Multi Acc: 98	Needs real-world validation
[10], 2021	ISCX-URL-2016	ML (RF, KNN, SVM, LR)	Lexical Features + Feature Selection	–	Acc: 99.57, P: 99.7, R: 99.46, F1: 99.58	Ignores content-based features
[11], 2022	CICMalDroid 2020+CIC - InvesAndMal2019	BERT	FAST extractor,BRIEF descriptor	–	Acc: 99.16, F1: 98.44	Possible flaws include the method's computational complexity
[12], 2022	Kaggle	BiGRU	Word2Vec + Dropout + Attention	–	Acc: 97.92, P: 98.34, R: 95.53, F1: 96.91	Needs optimization
[8], 2024	Web Phishing Dataset	BiLSTM + HABCNN	Word2Vec + Feature Extraction	–	Acc: 99.78, P: 98.97, R: 99.35, F1: 98.99	Struggles with shortened URLs
[13], 2023	ISCX-URL2016	CNN, LSTM, CNN-LSTM	SelectKBest + MinMaxScaler	–	Acc: 99.2, P: 99, R: 99.2, F1: 99.2	Ignores URL activity status
[17], 2021	Microsoft Edge+Internet Explorer telemetry	BERT, RoBERTa	WordPiece Tokenization	–	Acc: 99.67, F1: 99.71	Advanced reinforcement learning-based attacks could weaken the model
[14], 2023	651,191 URLs (Kaggle)	ML (Random Forest, LightGBM, XGBoost)	Feature extraction (counts, lengths, lexical tokens), PCA	Manual tuning	ACC: RF: 96.6%, LGBM: 95.6%, XGBoost: 93.2%	basic hyperparameter optimization
[15], 2022	Various sources (URL, Google CTI, Whois CTI)	CTI-MURLD: Two-stage ensemble (RF + MLP) with BFGS optimization	URL content features, Google-based CTI, Whois-based CTI, N-gram, TF-IDF, Mutual Information for feature selection	Grid Search	RF + GS: 97.25% accuracy, 97.26% recall, 97.36% precision	Complex architecture, requires extensive feature engineering
[16], 2024	Dataset of URLs	Improved M-BERT model using Transformer encoders	Implicit feature extraction via DL (Transformer-based)	–	Precision: 94.42%	Scalability and multimodal data integration not yet addressed, future work needed

3.2 Bidirectional encoder representations from transformers (BERT)

It is a pre-trained language model trained on 3.3 billion English tokens using two objectives: masked language modeling and next sentence prediction. It uses special tokens like [CLS] and [SEP] to structure inputs and enable context-aware understanding. The base version of BERT consists of 12 transformer layers with 12 attention heads each, and can be fine-tuned to achieve high performance across various

Table 2: Distribution of URLs in the dataset

Type	Total	Percentage
Benign URLs	428,103	65.72%
Defacement URLs	96,457	14.81%
Phishing URLs	94,111	14.45%
Malware URLs	32,520	5.00%



	url	label
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement

Figure 1: URL dataset

NLP tasks such as sentiment analysis and question answering [21].

3.3 Light gradient boosting machine (LightGBM)

It is a decision tree-based boosting algorithm designed to be fast and efficient on large datasets. It uses two main techniques: GOSS (Gradient-based One-Side Sampling) to reduce computational costs, and EFB (Exclusive Feature Bundling) to reduce dimensionality. These approaches allow LightGBM to be faster than conventional methods while maintaining high accuracy [22].

This two algorithms have been used in the proposed classification model to identify malicious websites with the highest accuracy.

3.4 Data preprocessing

Preprocessing is a critical step in any classification task, especially in the context of unbalanced text data such as malicious URLs. Our preprocessing pipeline was carefully designed to meet the specific requirements of each of the two models (BERT and LightGBM), while ensuring overall consistency in data preparation.
Cleaning: URLs were first converted to lowercase to standardize case. Duplicates were then removed, resulting in a 1.56% reduction in the total data volume.
Target Encoding: The label corresponding to the URL type (benign, phishing, malware, defacement) was processed using LabelEncoder strategy to transform classes into integers
URL Tokenization: The vector representation of URLs was adapted to the two methods we used:

– **For BERT model**: Tokenization was performed using the tokenizer `bert-base-uncased`, with a maximum sequence length set to 256 tokens. This step includes truncation, masking of special tokens `[CLS]`, `[SEP]`, and padding.

– **For LightGBM model**: TF-IDF vectorization [23] (Term Frequency Inverse Document Frequency: which is a traditional text analysis method that determines the importance of a word by calculating its frequency in the document and its inverse document frequency across the document set [24]) was applied, using **character n-grams** of size 1 to 3 filters out overly frequent terms (max_df = 0.85) and rare terms (min_df = 10). The maximum number of features (`max_features`) was set to 120,000, allowing the extraction of relevant syntactic patterns while maintaining a manageable vector space.

3.4.1 Dataset partitioning

A stratified partitioning of the dataset was performed to preserve class distributions across all subsets as shown in figure 2. The data was initially divided into two main sets: 85% for model development and 15% as a held-out test set used exclusively for final evaluation. To reduce computational cost during hyperparameter optimization (HPO), a stratified 20% subset of the 85% development set was sampled. HPO was performed on this subset. Once optimal hyperparameters were identified, the final models were retrained on the full 85% development set and evaluated only on the untouched 15% test set, ensuring that performance metrics reflect true generalization ability.

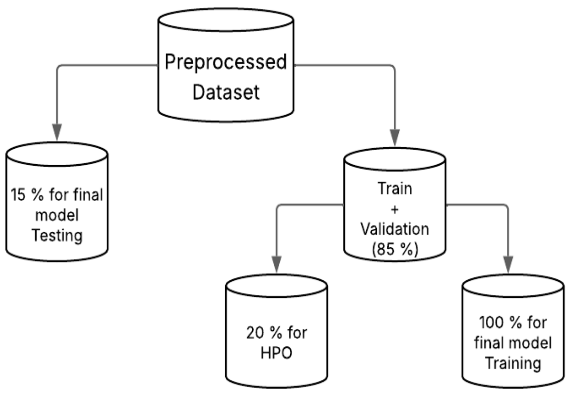

Preprocessed Dataset
15 % for final model Testing
Train + Validation (85 %)
20 % for HPO
100 % for final model Training

Figure 2: Dataset spliting

3.5 Optuna

A modern optimization framework that efficiently searches for optimal hyperparameters using pruning and dynamic trial allocation, its architecture is shown in Figure 3

Optuna is an emerging tool with three advantages for model selection or hyperparameters determination. The first advantage Optuna provides is the define-by-run style API. The second advantage is an efficient pruning and sampling mechanism. The third advantage is that it is easy to set up.[25]

This approach uses the Tree-structured Parzen Estimator (TPE) algorithm, which is an efficient Bayesian approach to optimize continuous and discrete search spaces. This algorithm constructs a probability distribution of hyperparameters by distinguishing promising configurations from poor performers[26].

3.6 Random search

The random search method involves testing a set number of random combinations of hyperparameters, followed by evaluating their performance and selecting the best results. This approach is efficient and works well with high-dimensional data. The process consists of several steps: first, the number of iterations (n_iters) is set; then, all parameter values are initialized to zero. In each iteration, the parameter values are randomly adjusted, and the model is trained on the data. The classifications are tested using real data to evaluate the performance, and the best parameter values and classification results are saved for future use [28].

4 Classification methods

Our work shows how hyperparameter optimization affects DL (BERT) and ML (LightGBM) models, which are frequently used but rarely fine-tuned. Our approach to optimization is rigorous: for BERT, we use Optuna to tune key hyperparameters like learning rate, batch size, number of epochs and dropout; for LightGBM, we use random search to adjust tree depth, learning rate, number of leaves and others. To reduce computational cost, we performed hyperparameter optimization (HPO) on a stratified 20% subset drawn from the 85% training/validation portion of the data. This ensures both efficiency and class representativity. The optimal hyperparameters were then applied to retrain models on the full 85% development set. Final evaluation was conducted on the held-out 15% test set, never seen during training or optimization.

5 Hyperparameter optimization

Hyperparameter optimization is a crucial step in the training process for ML and DL models. Careful hyperparameter configuration can significantly improve a model's performance, particularly in unbalanced classification contexts such as malicious URLs. In this study, we explored two distinct optimization strategies tailored to the nature of the models used.

5.1 BERT-Optuna model

Our first model is based on the BERT architecture and is specifically designed for malicious URL classification. We use the bert-base-uncased variant, which contains 110 million parameters and has proven effective for sequence classification tasks. To enhance performance, we applied a Bayesian hyperparameter optimization approach using Optuna's Tree-structured Parzen Estimator (TPE) algorithm, which efficiently explores the search space. The objective of the optimization was to maximize the weighted F1-score, a suitable metric given the class imbalance in our dataset. The model architecture is illustrated in Figure 4.

The hyperparameter search space included the following: learning rate, batch size, dropout rate, and number of epochs. Their respective ranges and results are presented in Table 3. In response to the structural characteristics of URLs and the imbalance in class distribution, we also introduced several adaptations. Notably, we used automatically computed class weights, inversely proportional to class frequencies, within the CrossEntropyLoss function to improve minority class recognition. Additionally, we limited input sequences to 256 tokens, striking a balance between information preservation and computational efficiency.

5.2 LightGBM-RandomSearch model

Our second approach presents a comparative study of two LightGBM-based architectures for malicious URL classification, highlighting the importance of preprocessing and hyperparameter optimization. Initially, we evaluated a baseline LightGBM model using default parameters, coupled with carefully tuned TF-IDF vectorization settings. To enhance performance and mitigate overfitting, we then applied systematic hyperparameter tuning via RandomizedSearchCV, enabling efficient exploration of a large search space with lower computational overhead compared to exhaustive methods. The model architecture is illustrated in Figure5.

The optimization process focused on key parameters: number of leaves and maximum tree depth to manage model complexity, learning rate and number of estimators to influence training dynamics, feature_fraction and bagging_fraction for subsampling and robustness. We also tuned the number of boosting iterations to control overall complexity. The full range of explored hyperparameters and their optimal values is detailed in Table 4. Overall, the final model achieved a strong balance between expressiveness and generalization.

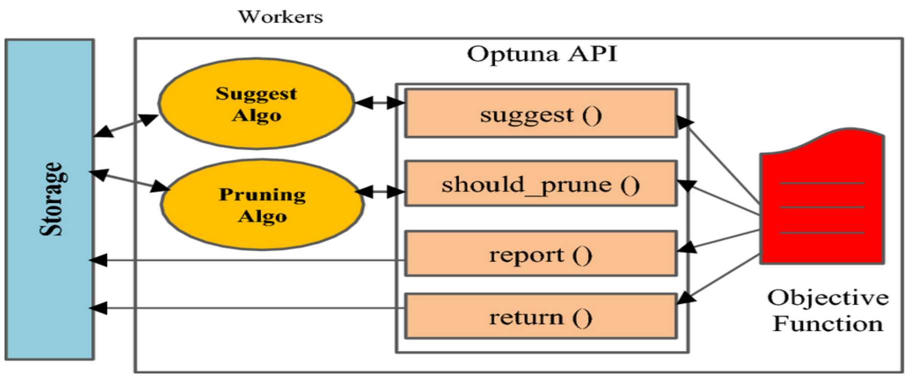


Figure 3: Overview of Optuna's system design [27]

Table 3: Hyperparameters optimization for BERT model

Hyperparameter	Value range	Optimal value
Learning rate	log-uniform between 1×10^{-5} and 5×10^{-5}	$3.49312729495079 \times 10^{-5}$
Batch size	{16, 32}	32
Dropout	[0.1, 0.5]	0.1479659107702988
Number of epochs	4 to 10 with early stopping	6

6 Experimental setup

In this section, we look at the experimental setup.In subsection 6.1, the performance evaluation metrics that will be used to judge how well the suggested method works are explained. Subsection 6.2 explains how the experiment was set up and how it was done.

6.1 Evaluation metrics

In the experiments, to improve the evaluation effectiveness of the proposed models, we utilized several recognized metrics, namely the F-measure, precision, recall, classification accuracy (ACC) and ROC-AUC (Area under the Receiver Operator Characteristic Curve). In confusion matrix, TP represents entries that are appropriately categorized as malicious sites. FN is the number of entries mistakenly classified as harmful websites.The FP represents the mistakenly identified harmful sites. TN is the number of successfully detected authentic websites.

The classification accuracy **(ACC)** measures the proximity among assessments of projected records and the overall count of valid and harmful websites. The value of ACC is measured by Eq.1

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision: It is the proportion of webpages the suggested model successfully identified as harmful. It shows the proposed model's correctness. The value of precision is measured using Eq.2

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall: It indicates the proportion of harmful websites that were accurately identified as harmful. It shows the wholeness. It calculated using Eq.3

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F-measure: It is defined as the harmonic mean of precision and recall.It calculated using Eq.4

$$F - measure = \frac{2 \times Precision \times Recall}{Precioision + Recall} \tag{4}$$

FAR: FAR is for False Acceptance Rate. It calculated using Eq.5

$$FAR = \frac{FP}{FP + TN} \tag{5}$$

AUC-ROC (Area under the Receiver Operator Characteristic Curve): lets you check how well the predicted classes match up with the real classes. If the AUC-PR score is high, it means that the model can tell the difference between negative and positive case points better. Most of the time, AUC values are shown on a range from 0 to 1, where:

- AUC = 0.5 shows that the model performs similarly to random guessing.
- AUC = 1 indicates that the classifier differentiates between negative and positive instances.

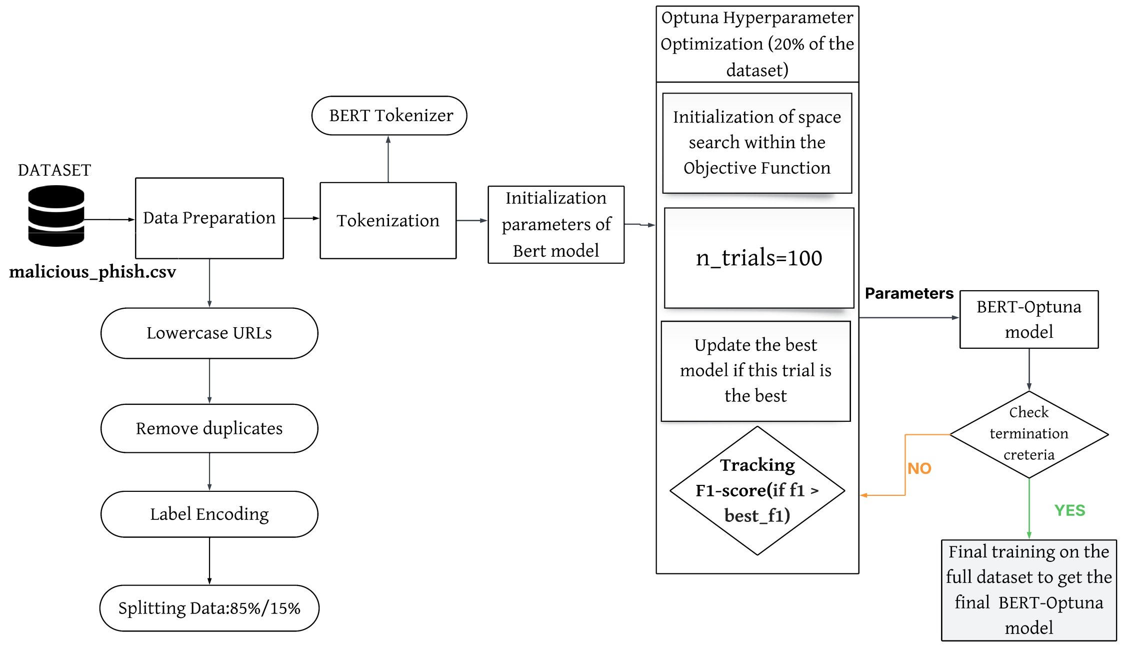

DATASET
malicious_phish.csv
Data Preparation
Tokenization
BERT Tokenizer
Initialization parameters of Bert model
Lowercase URLs
Remove duplicates
Label Encoding
Splitting Data:85%/15%
Optuna Hyperparameter Optimization (20% of the dataset)
Initialization of space search within the Objective Function
n_trials=100
Update the best model if this trial is the best
Tracking F1-score(if f1 > best_f1)
Parameters
BERT-Optuna model
Check termination creteria
NO
YES
Final training on the full dataset to get the final BERT-Optuna model

Figure 4: The proposed BERT approach

Table 4: Hyperparameters optimization for LightGBM model

Hyperparameter	Value range	Optimal value
num_leaves	[30, 80]	68
max_depth	[5, 15]	10
learning rate	[0.1, 0.25]	0.10560222831238217
n_estimators	[1800, 3000]	2443
feature fraction	[0.5, 0.9]	0.6091500749817148
bagging fraction	[0.5, 1]	0.959083326056901
max_bin	[63, 255]	205
min_data_in_leaf	[5, 30]	23
num_iterations	[10, 1000]	912

6.2 System setup

The optimization and training of the BERT model were carried out on a server equipped with dual NVIDIA RTX 3060 GPUs, each offering 12 GB of dedicated memory, utilizing PyTorch 1.12 and Transformers 4.24 to fully exploit CUDA acceleration. The LightGBM model was trained and optimized in a Google Colab environment with access to 334 GB of RAM. Although the runtime type supported TPU acceleration, LightGBM computations were executed on the CPU, as the library does not support TPU-based processing. A range of Python libraries, including Scikit-learn, NumPy, SciPy, Pandas, and Matplotlib, were employed to develop and evaluate the proposed models.

7 Experimental results and discussion

The experimental results and performance metrics of Bert with Optuna optimization model, LightGBM with tunned TF-IDF model and LightGBM with Random Search optimization and tunned TF-IDF are shown in Table 5 and Figure 6.
All the results reported in this section were obtained on the 15% hold-out test set, which was never used during hyperparameter optimization or training. This ensures that the metrics accurately reflect the model's generalization performance.

7.1 Performance analysis

Bert-Optuna achieved the highest results with 99.02% F1-score, 98.84% accuracy, an AUC-ROC of 0.9982 and

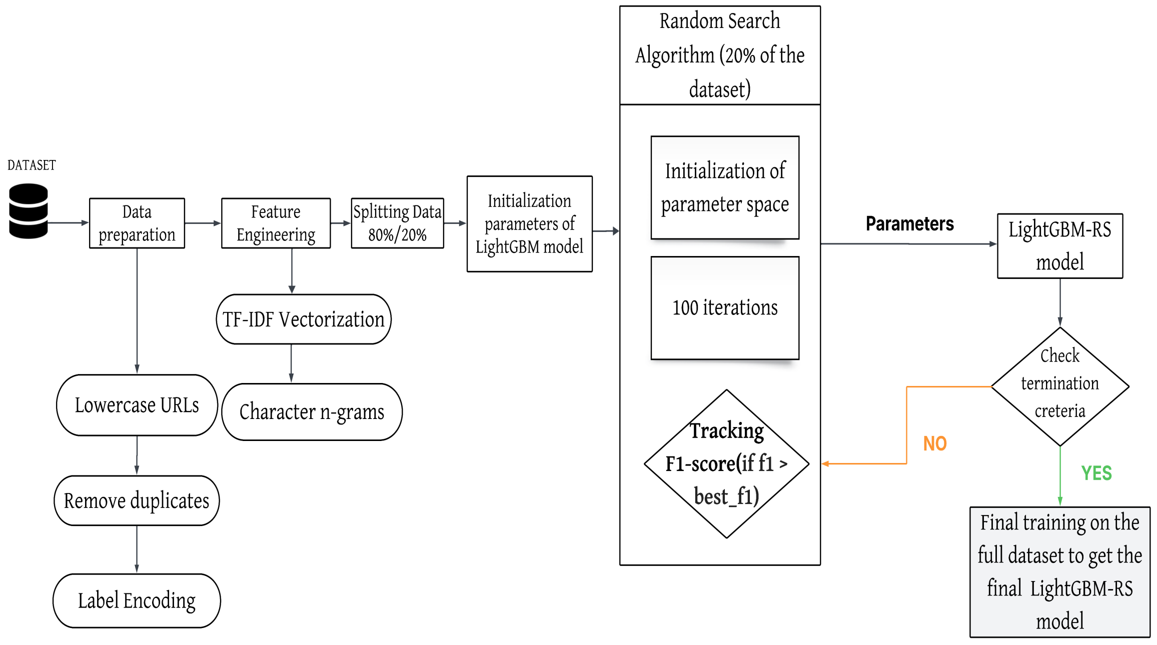

DATASET
Data preparation
Feature Engineering
Splitting Data 80%/20%
Initialization parameters of LightGBM model
Random Search Algorithm (20% of the dataset)
Initialization of parameter space
100 iterations
Tracking F1-score(if f1 > best_f1)
Parameters
LightGBM-RS model
Check termination creteria
NO
YES
Final training on the full dataset to get the final LightGBM-RS model
TF-IDF Vectorization
Character n-grams
Lowercase URLs
Remove duplicates
Label Encoding

Figure 5: The proposed LightGBM framework

False Rate Alarm of 0.005, outperforming all other models. LightGBM-Tfidf-RandomSeach followed with 98.46% accuracy, 98.45% F1-score, and an AUC-ROC of 0.998, showing clear gains from Random Search optimization. Tfidf-LightGBM also performed well, reaching 97.63% accuracy and 97.61% F1-score using only TF-IDF tuning. These results highlight the impact of optimization and preprocessing on model performance.

7.2 Performance comparison with existing studies

Compared to the first part of Su and Su[7] on the Kaggle dataset, , our Optuna-BERT model improves the F1-score from 98.57% to 99.02% so reduces the false alarm rate. Similarly, in comparison to the work by R, Patil, and Mohana [14], our Tfidf-LightGBM and Tfidf-RandomSearch-LightGBM models show clear improvements, with accuracy rising from 95.6% to 98.46% and a higher AUC-ROC score. These comparisons confirm that our optimized models offer significant advancements over existing approaches in malicious URL detection.

7.3 Why did BERT outperform LightGBM and state-of-the-art models?

BERT's performance superiority stems from its deep contextual understanding of URL text sequences. Unlike LightGBM, which relies on static vectorized features (TF-IDF), BERT captures both token-level and sentence-level semantics. This makes it particularly effective in recognizing subtle variations that are common in malicious URLs.

Further, using a longer token length (256 vs. 128 in earlier studies) allowed BERT to retain more contextual information from long or obfuscated URLs. The optimized configuration — a lower dropout (0.15) and larger batch size (32) — helped enhance generalization and stability.

7.4 Role of hyperparameter optimization

Hyperparameter tuning played a pivotal role. For BERT, Optuna selected a learning rate of 3.462×10^{-5} and a dropout of 0.147, with early stopping after 6 epochs to prevent overfitting. These adjustments boosted model stability and performance, particularly on minority classes.

For LightGBM, random search optimization increased its capacity by selecting a larger number of leaves (68) and a tree depth of 10. At the same time, reducing the feature fraction to 0.61 improved generalization by encouraging diversity across trees. These improvements closed a significant part of the performance gap with BERT.

7.5 Handling class imbalance: per-class performance

The dataset exhibits class imbalance, with benign URLs overrepresented and phishing/malware underrepresented. This imbalance affected model sensitivity to minority classes. Table 6 provides a breakdown of per-class F1-scores:

BERT-Optuna achieved consistently high F1-scores across all classes, outperforming other models especially on the more challenging classes like phishing (97.11%) and mal-

Table 5: Performance comparison of malicious URL detection approaches

Research	**Dataset**	**Model**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Su and Su [7]	Malicious urls dataset [19]	Fine-tuning BERT + Classifier	98.78%	99.12%	98.02%	98.57%
R, Patil, and Mohana [14]	Malicious urls dataset [19]	Light GBM Classifier	95.6%	95%	96%	95%
This study	Malicious urls dataset [19]	TF-IDF LightGBM	97.63%	97.61%	97.63%	97.61%
	Malicious urls dataset [19]	TF-IDF Random-Search Light-GBM	98.46%	98.45%	98.46%	98.45%
	Malicious urls dataset [19]	OptunaBERT	98.84%	99.20%	98.84 %	99.02%

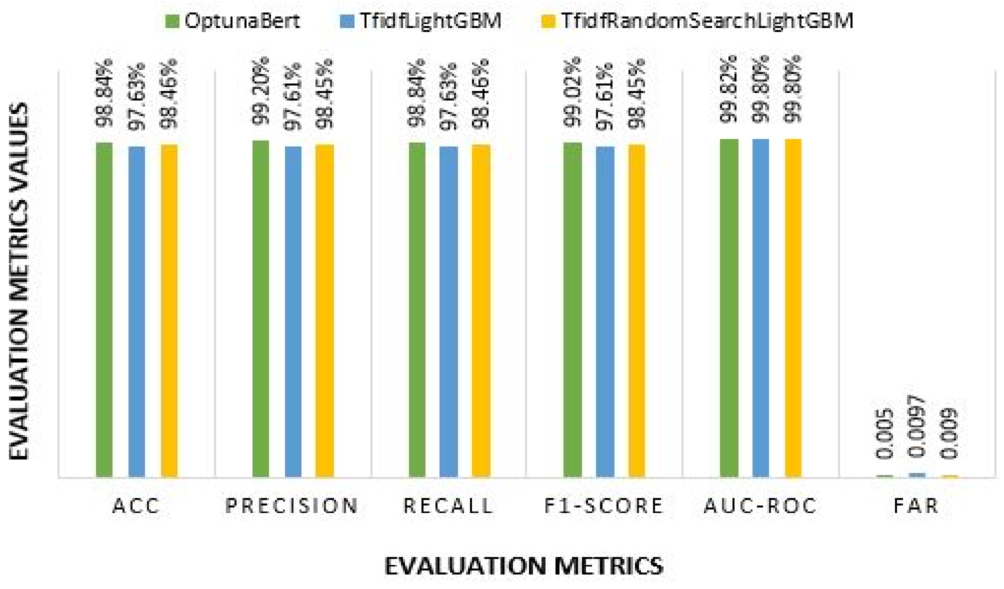


Figure 6: Comparison of the proposed models' evaluation metrics

ware (96.14%). This reflects the advantage of transformer-based architectures in capturing contextual features. In contrast, LightGBM-based approaches showed slightly lower F1 performance on phishing and malware, indicating some limitations in handling complex or possibly less-represented patterns.

8 Conclusion and future work

Malicious websites pose significant cybersecurity threats by targeting users to steal sensitive data and disrupt online activity. This study proposed a novel approach combining machine learning (ML) and deep learning (DL) techniques to effectively detect and classify such threats. We developed and evaluated two optimized models: (1) a BERT-based classifier enhanced with the Optuna hyperparameter optimization framework, and (2) a LightGBM model fine-tuned via RandomizedSearchCV to efficiently explore a wide hyperparameter space.

Experiments conducted on a real-world dataset containing over 651,000 URLs demonstrated the effectiveness of our approach. The Optuna-BERT model achieved an F1-

Table 6: Per-class performance (F1-score) comparison

Model	Benign	Defacement	Phishing	Malware
BERT-Optuna	99.32%	99.76%	97.11%	96.14%
LightGBM-TFIDF-RandomSearch	99%	99.98%	97%	95%
TFIDF-LightGBM	99%	99%	96%	92%

score of 99.02%, while the optimized LightGBM model reached 98.45% accuracy. These results outperformed several existing benchmark methods in terms of F1-score, false alarm rate, and AUC-ROC.

The study highlights the importance of combining advanced feature representations, contextual language modeling, and systematic hyperparameter optimization to enhance detection performance. It also shows that properly optimized classical models like LightGBM can still provide competitive results.

Future work will focus on extending the framework to detect evolving cyber threats in more diverse environments, such as IoT ecosystems. We also plan to investigate ensemble techniques and alternative feature extraction strategies to further improve robustness, scalability, and generalization across threat types and domains.

References

[1] Abdul Basit et al. "A comprehensive survey of AI-enabled phishing attacks detection techniques". In: *Springer Science* 2020. https://doi.org/10.1007/s11235-020-00733-2.

[2] Minal Chawla and Siddarth Singh Chouhan. "A Survey of Phishing Attack Techniques". In: 2014. http://doi.org/10.5120/16197-5460.

[3] A. Saleem Raja, R. Vinodini, and A. Kavitha. "Lexical Features-Based Malicious URL Detection Using Machine Learning Techniques". In: *Materials Today: Proceedings* 2021. https://doi.org/10.1016/j.matpr.2021.04.041.

[4] David L. Krause. "The $1.4 Billion Bybit Hack: Cybersecurity Failures and the Risks of Cryptocurrency Deregulation". In: 2025. https://doi.org/10.2139/ssrn.5150171.

[5] Saeid Sheikhi. "An effective fake news detection method using WOA-xgbTree algorithm and content-based features". In: *Elsevier* 2021. https://doi.org/10.1016/j.asoc.2021.107559.

[6] Saeid Sheikhi and Panos Kostakos. "Safeguarding cyberspace: Enhancing malicious website detection with PSO-optimized XGBoost and firefly-based feature selection". In: *Computers & Security* 2024. https://doi.org/10.1016/j.cose.2024.103885..

[7] Ming-Yang Su and Kuan-Lin Su. "BERT-Based Approaches to Identifying Malicious URLs". In: *Sensors* 23 2023, p. 8499. https://doi.org/10.3390/s23208499.

[8] Manika Nanda, Mala Saraswat, and Pankaj Kumar Sharma. "Enhancing cybersecurity: A review and comparative analysis of convolutional neural network approaches for detecting URL-based phishing attacks". In: *Elsevier e-prime* 2024. https://doi.org/10.1016/j.prime.2024.100533.

[9] Zhiqiang Wang et al. "A Malicious URL Detection Model Based on Convolutional Neural Network". In: *Hindawi Secur. Commun. Networks* 2021. https://doi.org/10.1155/2021/5518528.

[10] Brij B. Gupta et al. "A Novel Approach for Phishing URLs Detection Using Lexical-Based Machine Learning in a Real-Time Environment". In: *Computer Communications* 2021. https://doi.org/10.1016/j.comcom.2021.04.023.

[11] Farhan Ullah et al. "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation". In: *Italian National Conference on Sensors* 2022. https://doi.org/10.3390/s22186766.

[12] Tiefeng Wu et al. "Malicious URL Detection Model Based on Bidirectional Gated Recurrent Unit and Attention Mechanism". In: *applied Sciences* 2022. https://doi.org/10.3390/app122312367.

[13] Zainab Alshingiti et al. "A Deep Learning-Based Phishing Detection System Using CNN,LSTM, and LSTM-CNN". In: *electronics MDPI* 2023. https://doi.org/10.3390/electronics12010232.

[14] U. S. D. R, Anusha Patil, and Mohana Mohana. "Malicious URL Detection and Classification Analysis using Machine Learning Models". In: *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)* 2023. https://doi.org/10.1109/IDCIoT56793.2023.10053422.

[15] Mohammed Alsaedi et al. "Cyber Threat Intelligence-Based Malicious URL Detection Model Using Ensemble Learning". In: *Sensors* 2022. https://doi.org/10.3390/s22093373.

[16] Boyang Yu et al. “Efficient Classification of Malicious URLs: M-BERT—A Modified BERT Variant for Enhanced Semantic Understanding”. In: *IEEE Access* 2024. https://doi.org/10.1109/ACCESS.2024.3357095.

[17] Pranav Maneriker et al. “URLTran: Improving Phishing URL Detection Using Transformers”. In: *arXiv* 2021. https://arxiv.org/abs/2106.05256.

[18] Nikhilesh P Mankar et al. “Comparative Evaluation of Machine Learning Models for Malicious URL Detection”. In: *2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon)* 2024. https://doi.org/10.1109/MITADTSoCiCon60330.2024.10575452.

[19] *Malicious URLs Dataset*. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset. Accessed: 26 August 2023.

[20] Hafiz Muhammad Junaid Khan. “A MACHINE LEARNING BASED WEB SERVICE FOR MALICIOUS URL DETECTION IN A BROWSER”. MA thesis. Electrical and Computer Engineering Department Hammond, Indiana, 2019.

[21] Kevin Clark et al. “What Does BERT Look At? An Analysis of BERT’s Attention”. In: *arXiv preprint arXiv:1906.04341* 2019. https://doi.org/10.18653/v1/W19-4828.

[22] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Advances in Neural Information Processing Systems* 30 2017. https://dl.acm.org/doi/10.5555/3294996.3295074.

[23] Shahzad Qaiser and R. Ali. “Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents”. In: *International Journal of Computer Applications* 2018. https://doi.org/10.5120/ijca2018917395.

[24] Bing Wen et al. “File Compliance Detection Using a Word2Vec-Based Semantic Similarity Framework”. In: *Informatica* 49.18 2025, pp. 51–66. https://doi.org/10.31449/inf.v49i18.7421.

[25] Jung-Pin Lai et al. “Tree-Based Machine Learning Models with Optuna in Predicting Impedance Values for Circuit Analysis”. In: *Micromachines* 14 2023. https://doi.org/10.3390/mi14020265.

[26] Saeid Hanifi et al. “Advanced hyperparameter optimization of deep learning models for wind power prediction”. In: *Renewable Energy* 221 2024, p. 119700. https://doi.org/10.1016/j.renene.2023.119700.

[27] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *CoRR* abs/1907.10902 2019. https://doi.org/10.48550/arXiv.1907.10902.

[28] Dimas Aryo Anggoro and Salsa Sasmita Mukti. “Performance Comparison of Grid Search and Random Search Methods for Hyperparameter Tuning in Extreme Gradient Boosting Algorithm to Predict Chronic Kidney Failure.” In: *International Journal of Intelligent Engineering & Systems* 14.6 2021. https://doi.org/10.22266/ijies2021.1231.19.