Journal of Science Research and Reviews

PRINT ISSN: 1595-9074
E-ISSN: 1595-8329
DOI: https://doi.org/10.70882/josrar.2024.v1i1.17
Homepage: https://josrar.esrgngr.org

Original Research Article

An Improved Light GBM using Bayesian Optimization for Vulnerability Exploitation Prediction

Mashi, Boryanka T., [2]Ahmad, Ibrahim S., [3]Kakudi, Habeebah A. and [3]Tanimu, Jesse J.,

[1]Department of Computer Science, Federal University Dutsin-ma, Katsina State, Nigeria
[2]Department of Information Technology, Bayero University Kano, Kano State, Nigeria
[3]Department of Computer Science, Bayero University Kano, Kano State, Nigeria
*Corresponding Author's email: bobisco66@gmail.com

KEYWORDS
Exploitability,
Prediction,
Bayesian Optimization,
Machine Learning,
Light Gradient Boosting Machine.

CITATION
Mashi, B. T., Ahmad, I. S., Kakudi, H. A., & Tanimu, J. J. (2024). An Improved Light GBM using Bayesian Optimization for Vulnerability Exploitation Prediction. *Journal of Science Research and Reviews*, *1*(1), 49-62..
https://doi.org/10.70882/josrar.2024.v1i1.17

ABSTRACT
Despite the significant advances in software security research, exploitability prediction remains elusive due to the uncertainty of which vulnerability to be prioritized. Though many studies have been done on vulnerability prediction, some problems still persist such as efficient parameter optimization, which has significant effect on the algorithm performance and efficiency. To address these challenges, we proposed an Improved Light Gradient Boosting Machine (LGBM) model using Bayesian Optimization (BO) Method. Three experiments were conducted to compare prediction accuracy and computational cost of time and memory on LGBM, LGBM with Grid Search and LGBM with Bayesian Optimization models. The results demonstrated that our Improved BO- LGBM model has better prediction accuracy and lower computational cost than the comparative models. BO-LIGHT GBM rendered AUC of 83% measuring the model performance, accuracy of 81%, while in terms of time and memory consumption has definitely taken the lead of 0.23 min executional time and 32MiB system memory. Our results suggest promising future applications of our improved BO_ LGBM model for the prediction of vulnerability exploitation, that could be relevant for IT organizations and vendors or any organization that has limited computational resources in its premises if employed.

INTRODUCTION

Predicting software vulnerability exploits presents a multifaceted challenge influenced by the surge in vulnerabilities and cyber threats (Wang & Guo, 2009). Cybercriminals exploit software weaknesses for unauthorized access and data breaches, with over 130,000 vulnerabilities reported by 2019 (Bilge & Dimitras, 2012). The risk escalates upon exploit disclosure, as demonstrated by ransomware attacks like WannaCry and NotPetya in 2017 (Ehrenfeld, 2017). However, accurately predicting exploits remains uncertain due to varying risks and resource constraints (Bhatt et al., 2020; Bozorgi et al., 2010).

Machine learning (ML) holds potential in exploit prediction (Sabottke et al., 2015), but challenges persist, including concept drift, misclassification, and inadequate data (Bullough et al., 2017; Mohammed et al., 2017). Light GBM and Grid Search have been used (Fang et al., 2020), but they face limitations in optimization and data size (Mingzhu et al., 2020). This study introduces an Enhanced Light GBM Model with Bayesian optimization for efficient vulnerability exploit prediction, addressing model performance and resource usage (Ju et al., 2019; Mingzhu et al., 2020). The model assists decision-makers in mitigating risks while minimizing resource costs, contributing to software security advancements (Suciu et al., 2021).

Several studies in the literature have been dedicated to predict vulnerability exploits. They are outlined in this section. In recent times, the identification and optimization of cybersecurity threats have gained substantial importance (Fang et al., 2020). Numerous studies have concentrated on prediciting the probability of software vulnerabilities and efficiently assessing potential threats to developers, vendors, and IT managers, thereby prioritizing early mitigation of critical vulnerabilities. Recent research employs machine learning methods alongside features extracted from publicly accessible online vulnerability data, including NVD attributes, to predict the likelihood of exploitation for a given vulnerability. In this context, "exploited" signifies the presence of a proof-of-concept exploit.

Machine Learning

This research also reviews a spectrum of studies addressing software vulnerability prediction from diverse angles, utilizing methods such as machine learning (ML), data mining, and text mining. Previous research has explored the connection between vulnerabilities and Proof-of-Concept (PoC) instances, as well as the prediction of exploits through online vulnerability mentions (Edkrantz & Said, 2015; Luca & Fabio, 2012; Mohammed et al., 2017; DeCastro-Garcia et al., 2019). Frei et al. (2006) conducted comprehensive investigations into the vulnerability lifecycle using data mining techniques, while Bozorgi et al. (2010) achieved remarkable 90% prediction accuracy through Support Vector Machine (SVM) classification of PoC instances. However, Sabottke et al. (2015) introduced skepticism about labeling criteria and raised questions about contradiction. Bullough et al. (2017) addressed issues related to imbalanced datasets and concept drift, emphasizing the influence of National Vulnerability Database (NVD) incompleteness and PoC availability. The limitations of conventional text processing were also recognized (Nazgol et al., 2018), with neural network models exhibiting promising results but missing certain linguistic nuances. Hoque et al. (2021) contributed robust features, including a novel "coefficient balance" function and custom-trained word vectors, enhancing vulnerability exploitation prediction accuracy with reduced computational demands. The synthesis of these studies underscores the pivotal role of hyperparameter optimization in fine-tuning predictive models, ultimately culminating in more precise and efficient vulnerability exploitation forecasts. In summary, this multifaceted review underscores the intricate landscape of vulnerability prediction research and advocates for the strategic refinement of models to yield superior forecasting outcomes.

Machine Learning and Hyperparameter Optimization

The Over the span of decades, researchers have pursued timely and accurate exploit prediction while minimizing computational costs. Optimizing ML models is key, and a review of studies on Light GBM with varied optimization strategies underscores this concept. Ju et al. (2019) introduced a fusion of CNN and Light GBM for wind power prediction, showcasing enhanced accuracy and efficiency. Mingzhu et al. (2020) applied Bayesian optimization to Light GBM for wind turbine fault detection, excelling in diagnosing faults despite imbalanced data challenges. Huang (2020) employed Light GBM for fraud detection, surpassing traditional models. Taha and Malebary (2020) achieved exceptional credit card fraud detection using Optimized Light GBM. Abbadi et al. (2020) presented a swift anti-malware system using Light GBM, while Wang and Wang (2020) enhanced Light GBM's prediction accuracy for blood glucose measurement. Fang et al. (2020) combined fastText and Light GBM for exploit prediction, yielding substantial improvements.

In sum, literature primarily centers on ML-based vulnerability prediction, often overlooking efficient optimization. Our research therefore aims to bridge this gap by introducing an advanced Light GBM model, enriched by Bayesian optimization. This approach enhances both predictive accuracy and computational efficiency, addressing a vital aspect of the field.

MATERIALS AND METHODS

We adopted the outlines of prior work (Fang et.al., 2020) to propose an improved model for vulnerability exploitability prediction based on BO method. However, we focus on illustrating the impact of the previously discussed challenges on the learning model. Our main target is to improve Light GBM model’s predictive performance and efficiency using BO in vulnerability exploit prediction.

Data description and preprocessing

The adopted dataset (Fang et al., 2020) used in this work was obtained from (https://github.com/das-lab/FastEmbed) public repository. The data has been aggregated from various intelligence data sources, containing all vulnerabilities and exploits published between the years 2009 and 2018. Data preprocessing is a crucial step in ML, involving transformations before feeding data to ML models. This phase eliminates unwanted elements like stop words, whitespace, special characters, URLs, and addresses missing values and irrelevant features. Quality data and feature selection significantly impact a model's efficiency and performance (Agarwal, 2015). The data was divided in a 70:30 ratio for training and testing. A portion of the preprocessed dataset is depicted in Figure 1.

Dimension: (14933, 14)

	Base_score	Exploitability_score	Impact_score	Vendors	Access_vector	Access_complexity	Authentication	Confidential_impact	Integrety_impa
0	7.5	10.0	6.4	1	NETWORK	LOW	NONE	PARTIAL	PART
1	9.3	8.6	10.0	1	NETWORK	MEDIUM	NONE	COMPLETE	COMPLE
2	7.5	10.0	6.4	1	NETWORK	LOW	NONE	PARTIAL	PART
3	9.3	8.6	10.0	1	NETWORK	MEDIUM	NONE	COMPLETE	COMPLE

Figure 1: Preprocessed dataset

Different vulnerabilities and exploits were identified by CVE-ID's and EDB-ID accordingly. Most studies including this work mark a vulnerability as exploitable when it has a corresponding PoC exploit identified by an EDB-ID. The dataset comprised of 60,784 vulnerabilities. In this work only the data from NVD was used, comprising of numerical, categorical and vulnerabilities text description data.

NVD is an example of an Open-Source Intelligence (OSINT) resource concerning security vulnerabilities. This database includes several components such as the CVE initiative, developed by the MITRE Corporation, which is an industry-standard dictionary containing a list of security vulnerabilities (National Institute of Standards and Technology, n.d.). Each year, the NVD database provides vulnerability data feeds consisting of CVE-IDs allocated for that year in JSON format. Each entry includes a descriptive text summary for the vulnerability, Common Vulnerability Scoring System (CVSS) scores, information about affected products and vendors, the category of vulnerability based on the Common Weakness Enumeration (CWE) system, and reference URLs (Flashpoint, 2021).

The CVE List feeds NVD, which then builds upon the information included in CVE Records to provide enhanced information for each record, such as fix information, severity scores, and impact ratings (National Institute of Standards and Technology, n.d.). As part of its enhanced information, NVD also provides advanced searching features, including filtering by operating system, vendor name, product name, version number, vulnerability type, severity, related exploit range, and impact (Flashpoint, 2021).

Each record includes a descriptive text summary for the vulnerability, scores and metrics from CVSS, information about affected products and vendors, the category of vulnerability based on the CWE system, and URLs to other reference sources. The numeric CVSS score vulnerability is calculated from the values assigned to the feature's Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrity Impact, and Availability Impact. Access Vector can take on the values requires" local access", "adjacent network accessible" or "network accessible". Access Complexity can be rated "high", "medium" or" low". Authentication can take on the values requires multiple instances of "authentication", requires single instance of "authentication" or requires no "authentication". Confidentiality Impact, Integrity Impact, and Availability Impact can be scored as "none", "partial" or "complete". These numbers are used to calculate the overall, numeric CVSS score.

Text features contain multiple entries for the same CVE ID, (i.e. the Vulnerable Systems List, Reference Types, Reference Sources, Reference URLs, CWE Names, and CWE Descriptions as well vulnerabilities descriptions) The NVD data was merged with data from the Exploit Database by CVE-ID. The merged data was divided and preprocessed according to its data type.

Features description

The National Vulnerability Database (NVD) database contributes inherent features categorized as numerical, categorical, and text description attributes. The CVSS version 2 is integrated as a feature set, encompassing the CVSS base score, exploitability subscore, and impact subscore. The exploitability subscore comprises features like Access Vector, Access Complexity, and Authentication, determining how vulnerabilities are exploited. Impact subscore considers Confidentiality Impact, Integrity Impact, and Availability Impact, gauging the extent of system impact post-exploitation. The Base score integrates these features. Common Platform Enumeration (CPE) extracts affected platforms and products via standardized CPE URLs, while the CWE feature refines flaw categorization in software development. The dataset structure in NVD enables systematic analysis and comparison of vulnerabilities

across multiple software environments. The categorization of features aids in precise identification, prioritization, and management of vulnerabilities in cybersecurity. Additionally, the CVSS and CWE features support automation in vulnerability assessment, allowing for quicker response and mitigation strategies. This structured approach in NVD provides an essential resource for vulnerability detection and management in various cybersecurity frameworks.

Table 1 offers a comprehensive summary of data features and types (numeric, categorical, or text).

Table 1: Features and categories

Features Source	Features	Types	Values Category
National Vulnerability Database	CVSS	Access Vector	Categorical
		Access Complexity	Categorical
		Authentication	Categorical
		Confidentiality Impact	Categorical
		Integrity Impact	Categorical
		Availability Impact	Categorical
		Base Score	Numerical
		Exploitability Score	Numerical
		Impact Score	Numerical
	CPE	Application	Numerical
		Hardware	Numerical
		OS	Numerical
		No product	Numerical
		List	Text
		Scale	Categorical
	CWE	CWE-ID	Numerical
		Name	Text
		URL	Text
	Reference	Number	Numerical
		Vulnerability description	Text

Modelling and Implementation of Light GBM Algorithm

Light GBM, an enhanced decision tree framework introduced by Microsoft in 2017, offers parallel processing and boasts speed, low memory usage, and reduced communication costs in parallel learning. It stands out for features like gradient-based one-side sampling (GOSS), exclusive feature bundling (EFB), histogram-based growth, and limited-depth trees to prevent overfitting. Light GBM leverages multi-threaded optimization and GPU support for faster training and effective management of sizable datasets.

After the data has been preprocessed and clean, we separate the labels from the rest of the data by declaring *X* as the variable that contains all previous data except the labels and the variable *Y* which contains the labels (Target). By using the newly defined *Y* variable we established the dataset's class balance. The data is first split into the train and test sets that receive 70% and 30%, respectively. Light GBM is not subjected to validation since it is supposed to be implemented without any additional optimization, except for its fundamental training using its default parameters. We fit the Light GBM classifier with the training data and run the prediction on the test data. Comparing the test and train scores confirmed there was no overfitting of the model. The classifier performance was evaluated in terms of AUC, Accuracy, Recall, Precision, F-1 Score Executional time and Memory used. The benefits of Light GBM encompass improved accuracy, distributed capabilities, and efficient handling of substantial data volumes.GOSS excludes a significant proportion of the data-instances with small gradients, and only uses the rest to estimate the information gain.

According to the definition of information gain, those instances with larger gradients (i.e., under-trained instances) will contribute more to information gain. Gradient One-Sided Sampling or GOSS utilizes every instance with a larger gradient and does the task of random sampling on the various instances with the small gradients. The training dataset is given by the notation of O for each node of the Decision tree. The variance gain of j or the dividing measure at the point d for the node is given in Equation 1:

$$\widetilde{V_j}(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \widetilde{\frac{1-a}{b}} \sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b} \sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right) \quad (1)$$

Where:

$A_I = \{x_i \in A: Xij \leq d\}$, $A_r = \{x_i \in A: Xij > d\}$, $B_I = \{x_i \in B: Xij \leq d\}$, $B_r = \{x_i \in B: Xij > d\}$, and the coefficient $\frac{1-a}{b}$ is used to normalize the sum of gradients over B back to the size A^c.

In order to compensate for the data multiplier for data instances with small gradients when calculating the information gain, GOSS first sorts the data instances according to their gradient's absolute value and selects the top instances. By doing so, without modifying the original data distribution by much, this method puts more focus on the under-trained instances. Chen et al (2017) claims that such a treatment can leads to a more accurate gain estimation than uniformly random sampling, especially when the value of information gain has a large range.

Our contribution leverages Bayesian optimization to find acceptable trade-offs between model performance, accuracy and resource consumption in terms of time and memory. Our framework implements a two-stage approach. The first stage explores the high-dimensional parameter search space of the target Light GBM Algorithm, where the selected parameters range of values is defined. The second stage we fit and re-train the model with the obtained set of optimal parameters. Next run the prediction on the test vulnerability data. We then analyze how well the set of 'best' model parameters on Light GBM achieve classification performance and optimized computational resources on the target Algorithm. We investigate this in relative terms by comparing the baseline and proposed BO_Light GBM model in terms by analyzing the obtained results impact on the target models.

This Phase explains Bayesian Optimization and the Principle behind Bayesian Algorithm. It also illustrates the stages of building the Proposed Improved BO_Light GBM using a flowchart and pseudocode of the Model.

Bayesian Optimization

The Bayesian is a "black box" optimization technique proposed to overcome the problems of other optimization methods in terms of computational cost of resources (Betrò, 1991; James & Bengio, 2012;Dewancker et al., 2016). Bayesian optimization fits a probabilistic model to capture the relationship between hyperparameter settings and their measured performance. The goal of Bayesian Optimization is to find an approximate minimum to some function that is expensive to evaluate. In this case the objective function is the Light GBM algorithm that we are trying to optimize, and the function's parameters are the hyper parameters of the algorithm.

The BO technique creates a prior over the objective function and combines it with evidence to get the posterior. This allows for a utility-based selection for the next observation to make on the objective function, which must consider both exploration (sampling from areas of high uncertainty) and exploitation (sampling areas likely to offer improvement over the current best observation

The model used for approximating the objective function is called a surrogate model. The technique uses previous observations of the loss function **ƒ**, to determine the next (optima) point for sample from. The three base components of Bayesian Optimization are:

1. The search space to sample from
2. The objective function
3. The surrogate and Acquisition functions

Defining the Search space (1) - Bayesian Optimization operates along probability distributions for each parameter that it will sample from. These distributions or domains are set by the researcher specifying each parameter range of values.

Objective Function (2) - This function serves as the main evaluator of hyperparameter combinations. It takes in a set of hyper parameters and output a score that indicates how well a set of hyper parameters performs on the validation set. For our classification problem we used "accuracy score" as the evaluation metric of choice. So clearly the aim is to maximize the objective function. Optimizing parameters in the objective function of any complex algorithm is time consuming. Therefore, Bayesian optimization technique limits calls to the evaluation function by choosing the next parameter values using previous best results hence, allows the algorithm to spend less time in evaluating promising parameter values and low-scoring regions of the parameter space.

***Surrogate function (3) and Acquisition function (4*) -** The surrogate function is an approximation of the objective function that is used to propose parameter sets to the objective function, that likely yield an improvement in terms of accuracy score. The parameters that are put forward for evaluation to the objective function are selected by criterion which is defined by the Acquisition function.

The algorithm starts by initializing the memory and time consumption of the based on minimal fitted memory and running time of the function efficient parameters. Then iterate sequentially to the remaining parameters to return optimal parameter values. Based on the Bayesian Theorem which uses the Bayesian form the algorithm can be explained as:

Using previously evaluated points $\mathbf{x}_{1:n}$, compute a posterior expectation of what the loss function **ƒ** looks like

Sample the loss function **ƒ** at a new point **x** new, that maximize some utility of the expectation of **ƒ**. The utility specifies which regions of the domain of **ƒ** are optimal to sample from.

These steps are repeated until some convergence criterion is met and the optimal parameter values achieved. An

optimization problem seeks to minimize a loss function. An objective function is either a loss function or its negative, in which case it is to be maximized.

The Pseudo Code of Bayesian Optimization Algorithm is shown in Figure 2.

Bayesian Optimization Algorithm

Assuming goal is to maximize unknown function f(x) on data D:

for n loops do
-> select new x_{n+1} by optimization of α which is an acquisition function $x_{n+1} = \max \alpha(x; D_n)$
-> get new observation y_{n+1} from objective function
-> augment data $D_{n+1} = \{D_n, (x_{n+1}, y_{n+1})\}$
-> update model
end for

Figure 2: The Bayesian Algorithm ((Shahriari et al., 2016))

Equation 2 mathematically represents the problem of finding a global maximizer (minimizer) of an unknown objective function ***f*** as:

$x^* = \underset{x \in \chi}{\operatorname{argmax}} f(x)$ (2)

Fundamentally, BO is a sequential model-based approach. In Equation 2 **x*** is the parameter of interest in a bounded space **x ϵ χ**. It aims at solving the problem defined above iteratively, by reusing all information acquired at each iteration ***n***, or all the values that were observed for the unknown function **f(x).** This can be done by posing a surrogate model of the function **f(x),** which will then be used to determine the next point of **ƒ** to evaluate **x**0 balancing exploration and exploitation criteria to be defined. The key element of Bayesian optimization is to exploit all the available information to guide the optimization procedure at each iteration. (Feurer & Hutter, 2019)

Light GBM Optimization with Bayesian method- Phase One

Outlining the steps in Algorithm 3.2

- In Step 1 we define the objective function, the selected parameters and their domain space. The main parameters which affect the performance of the Light GBM model were selected based on stated objectives.
- Next step is to initialize the Surrogate and Acquisition function.
- Step 3 for each iteration find the hyperparameter where the Acquisition function is optimized
- Step 4 is where the Objective function score is obtained to see how this point actually performs
- Step 5 stores the next chosen parameter to evaluate and the true Objective function score in the History of other samples
- Step 6 is where a new Model is fit The Surrogate model is trained using the latest history of samples
- The looping ends when maximum number of iterations has been reached
- Step 7 Evaluate the best optimal parameters set.
- End

Figure 3 displays the steps followed to optimize Light GBM algorithm using Bayesian Algorithm.

ALGORITHM 1: LIGHT GBM VIA BAYESIAN OPTIMIZATION
Input: Base Light GBM Model; M0; Selected parameters $\theta = \{\theta_1, \theta_2, \dots \dots \dots \theta_n\}$ $domain$, the corresponding objective function f(θ^*); P
1: Initialize M0; S; P
2: For n = 1, 2, ...do until T max
3: Find the optimal hyper-parameter θ^* by maximizing the objective function f(θ^*); M_{n+1}: $\theta^* = argmax\ M_{n+1}$: (θ).
4: Evaluate the f(θ^*) $score$ under the settings $\theta^* \rightarrow$ the expensive step
5: Store θ^* and the corresponding objective function f($\theta^{*)}$ score in P
6: Fit a new model $M_b = M\ \cup (\theta^*, f(\theta^*)\ )$
End for
Output: optimal hyper-parameters of Light GBM

Figure 3: Light GBM Optimization with Bayesian method- Phase One

Variables that are used in this algorithm are listed below:
θ*: Set of selected parameters and their range
M0: The Acquisition function
f(θ^*): The true Objective function
S: The Surrogate function, which is updated whenever a new sample is added
P: The Observation History of (hyperparameter, score) pair
θ*: Next chosen parameter to evaluate

In Figure 4, the steps in Phase Two of the proposed improved BO-Light GBM Model are captured.

BO Light GBM model for the prediction of vulnerability Exploits-Phase Two

Algorithm 2: Implementation of improved BO_ Light GBM Model
Input: Light GBM Model M_b, Vulnerability dataset D = $\{(x1y1), (x2, y2), \dots \dots \dots (xn, yn)\}$
1: Defining the Independent (Target) Y and Dependent variables X
2: Handing missing data and apply data normalization for D; dividing dataset as D_{test} and D_{train}
3: Establish Light GBM model M_b based on D_{train}, θ from Algorithm 3.2 **4: Fit the model with optimal parameters values**
5: Run the prediction
6: Evaluate the performance of the proposed model
Output: Result prediction

Figure 4: BO_Light GBM Improved Model Phase Two.

Implementation of the proposed BO_ Light GBM hyper-parameters optimization can be detailed as: Θ = {Θ_1, Θ_2.....Θn} represents the selected hyperparameters and their Domain space in the machine learning algorithm A (such as Light GBM), D(train) data set is used for training, and D(valid) data set is used for validation (i.e., parameter optimization), and the two are independently distributed. L (A, Θ, D(valid), D(train) is used to represent the validation loss of Light GBM algorithm. Stratified K-fold validation is applied to address the optimization requirement. The interval range for parameters are set in our Light GBM algorithm. In the process of parameter optimization, the model is continuously trained, and the classification result obtained as each parameter combination is evaluated by the evaluation function. The best generated parameter values are obtained and printed out. The model is fit with the generated optimal parameters and the classification has taken place.

The Light GBM - Grid Search Model and Proposed BO_LIGHT GBM Model

We rebuilt Light GBM and Grid Search as our baseline model to obtain its performance result as the comparative base for the performance of our proposed model. For the

optimization of the two models, we used the benchmark data set and the selected parameters depicted in Table 3.3 The combined dataset contains 27,368 samples, and we use 80% (or 11,475 samples) for training and validation (split according to a 67/33 ratio) in order find the best model and hyperparameters. The remaining 20% was used for testing. The dataset was split with stratified sampling such that the training, validation and test sets have approximately the same percentage of samples of each target class (i.e., normal and anomalous) as the original complete dataset.

All parameters of the methods were optimized to achieve the best performance. For each task, we ran all possible combinations of the domain using the two optimization methods.

Stratified 10-fold validation was used throughout the models training and testing to avoid overfitting the model. The training dataset was randomly divided into 10 equal size subsets while ensuring that the proportion of all kinds of samples in the training and test set are the same as that in the original data set. The K-1 subsets were retained as training data, and the remaining one subset was used as the validation data for testing the model This approach removes the possibility of the proposed model being over-fitted to the training set and the training mark being inserted indirectly into the training set, which can occur when k-fold cross validation is performed explicitly on the entire data set.

Basic Performance Metrics

Model evaluation metrics are required to quantify model performance and the most commonly used in Classification problems are stated below.

Accuracy of exploit prediction is the correct classification of True Positive (TP) and True Negative (TN), where TP is the number of exploited vulnerabilities that are correctly identified, FN is the number of exploited vulnerabilities that are mistakenly classified as vulnerabilities that will not be exploited

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

Accuracy is the standard metric used to evaluate a classification ML model, but in case of imbalance data may not give the right interpretation of the quality of the model.

Precision: It is known as correctness; that measures the efficiency of prediction. Further can be defined as the proportion of number of vulnerabilities correctly predicted and likely to be exploited to the total number of vulnerabilities predicted as likely to be exploited

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

Recall also known as Sensitivity or True Positivity Rate TPR: Recall can be defined as the ratio of number of vulnerabilities correctly predicted as likely to be exploited to the actual number of vulnerabilities been exploitable. Recall represents the exploit prediction rate which allow us to quantify the effectiveness of prediction. It is given as:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

F-measure*:* F1-score is the harmonic mean of Precision and Recall and represents the result of a trade-off between Precision and Recall.

$$F1 = 2\frac{\text{Precsion}*\text{Recall}}{\text{Precision}+\text{Recall}} \quad (6)$$

The Area Under Curve (AUC) is defined as the area enclosed by the coordinate axis under the ROC curve. It represents the ability of the classifier to correctly distinguish between the correctly predicted positive and negative classes, thus, evaluate the efficient performance of the model. Classifiers with larger AUC have better performance.

Confusion Matrix: A confusion matrix, is a tool to visualize the classification performance ofmachine learning models, by presenting the four base classification statistics in a tabular fashion. It provides an insight how correctly the model has classified the two classes. The matrix compares the actual Target values with those predicted by the ML model. It gives a holistic view of how well the model performs and what kind of errors is making. The components of the confusion matrix are:

TP- true positive number of labeled positives (labeled exploit) that are correctly classified as positive (predicted exploited).

FP- false positive, number of labeled negatives (labeled non- exploited) that are incorrectly classified as positive (predicted exploited)

FN- false negative, number of labeled positives (labeled exploited) that areincorrectly classified as negative (predicted non-exploited).

TN- true negative, number of labeled negatives (labeled non-exploited) that are correctly classified as negative (predicted non-exploited.

In Figure 5, we present the research flowchart, which comprises four key stages: problem identification (Stage 1), data description and preprocessing (Stage 2), algorithm implementation (Stage 3), and evaluation (Stage 4). This framework guides the structure and progression of our study.

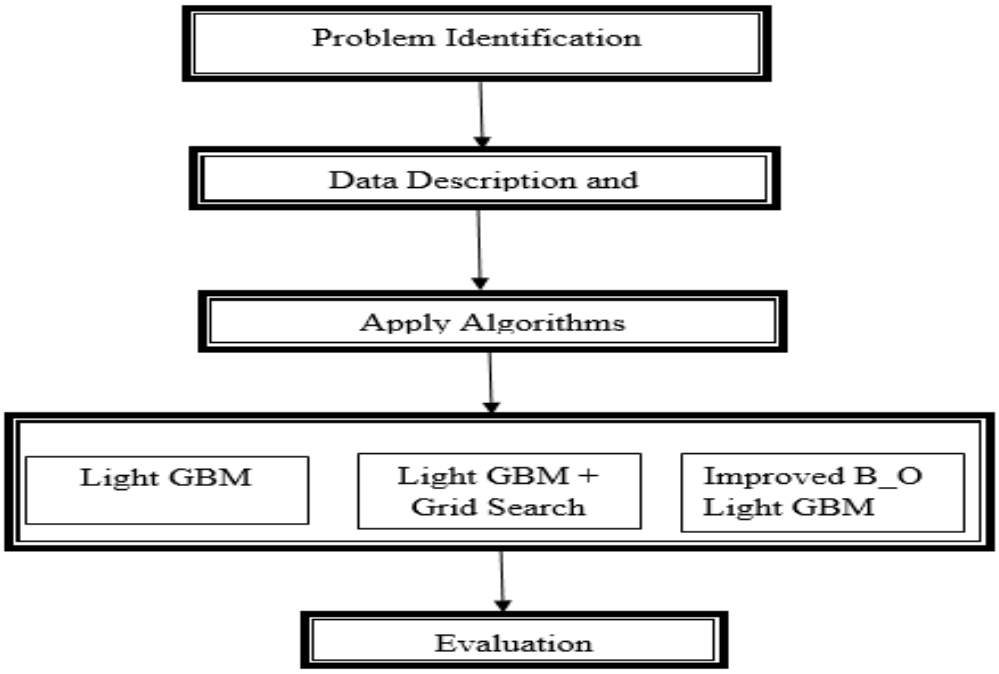

Figure 5: Research Flowchart

RESULTS AND DISCUSSION

Basic Prediction

The first experiment was mainly done to examine the impact of Light GBM classifier without applying hyper parameter optimization technique in predicting Vulnerability exploits. It was, therefore conducted using the model default parameters. Equations (3), (4), (5) and (6) gives the mathematical representations of the metrics used for the performance evaluation of the Light GBM model.

The Light GBM classifier with no hyper parameter optimization, has achieved accuracy of 64.6%, AUC of 70%, Precision 58.5%, Recall 72%, F1 score is 64.5%. It has a significant start in terms of the overall accuracy resulting in 64.6% and performed reasonably in predicting vulnerabilities that do not have exploits with an AUC of 70%. But the result delivered for true positive cases does not seem promising. Therefore, there is a need to further improve the model. Another evaluation metric for the model efficiency is time and memory efficiency, where the algorithm has also performed well.

Light GBM-Grid Search Model Result Evaluation

The second experiment is the baseline technique that used the Grid Search method to optimize the selected Light GBM parameters. As mentioned in section 3, the baseline model was rebuilt to use it as a base for comparison with the proposed model. The benchmark dataset and the selected parameters were used for building and optimizing the mode. Thus, all parameters of the methods are optimized to achieve the best performance. Light GBM-Grid Search model is evaluated using the performance metrics given in equations (3), (4), (5) and (6). TheLight GBM-Grid Search model have achieved accuracy rate close to Light GBM Algorithm. On the other hand, its AUC is below that of the Light GBM. In terms of Precision, Recall and F1 Measure outcome results are 58%, 72%, and 64% respectively. The unsatisfactory time and memory performance can be attributed to Grid Search's brute force nature, which exhibits exponential time complexity. Consequently, this approach becomes computationally intensive, especially with larger datasets and complex models. Such results indicate that Grid Search may not be the most efficient method for hyper-parameter optimization in scenarios requiring scalability.

Evaluation of the Proposed BO_LIGHT GBM Model

The result of the third experiment shows the performance of the proposed model based on the evaluation matrix considered for this work. As earlier stated, the dataset, parameters and other settings for the proposed model were adopted from the base work of this research. To validate the strength of the proposed BO-LIGHT GBM model, the experimental analysis is performed using tenfold stratified cross-validation. Table 2 illustrates the performance behavior of the proposed model based on the selected evaluation metrics given in equations 3, 4, 5 and 6. Furthermore, Figure 6 presents the confusion matrix of the proposed model, where TP= 4085; TN= 5497; FP= 1653; and FN=240.

Table 2: Performance Summary of Proposed BO_LIGHT GBM model

Evaluation metrics	BO_LIGHT GBM performance
AUC	**83%**
Accuracy	**81%**
Precision	**72%**
Recall	**94%**
F-1 Measure	**81%**

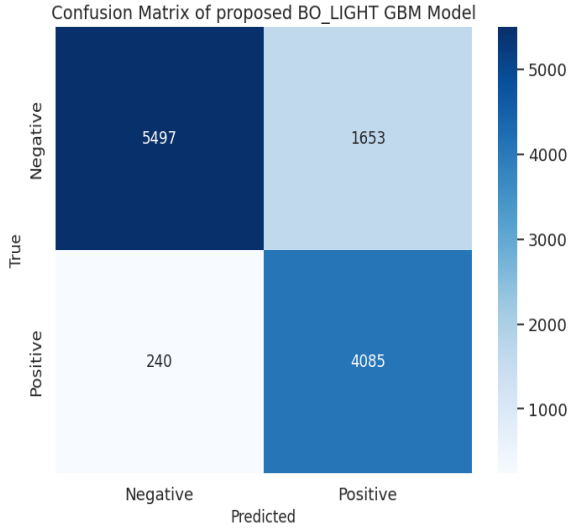

Figure 6: Confusion matrix of proposed BO_LIGHT GBM Model

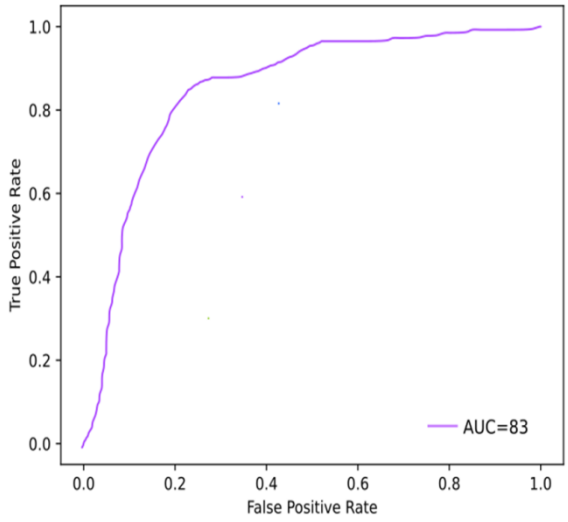

Figure 7: AUC-ROC of proposed BO_LIGHT GBM Model

Discussion

The efficiency of the BO-LIGHT GBM prediction model is one of the most important evaluation indicators of its performance, efficient models are always more preferred. The average time-consuming of Light GBM, Light GBM and Grid Search and BO-LIGHT GBM models at training and testing phases are compared, respectively. The trend of the results shown obviously indicates that BO-LIGHT GBM is time-cost efficient, followed by Light GBM and both significantly more proficient than Light GBM - Grid Search. That can be justified due to their special algorithm design mechanisms of Light GBM explained in section One and Three. When fitting the best model from the training set, Light GBM is a bit more time consuming due to the slightly higher complexity of the constructed model, but they are not that far apart. Therefore, it can be concluded that BO-LIGHT GBM and Light GBM are significantly more efficient than Light GBM - Grid Search. The confusion matrix is the visual representation of the model to identify positive and negative classes. Therefore, Light GBM classifier achieved AUC of 70%, Light GBM-Grid Search achieved AUC of 69% and the proposed model of this work achieved AUC of 83%. Clearly our proposed work has shown significant improvement of the model ability to predict correctly.

To measure the performance of the model, Light GBM classifier achieved accuracy of 64.6%, Light GBM-Grid Search of 65%, and the proposed model's accuracy is 81%. Precision, Recall and F1 Measure (respectively 72%, 94%, and 81%) scores of BO_LIGHT GBM are distinctly higher than the other two models, which evidently proves the efficacy of the proposed model. The finding of the results has shown major improvement in performance of the proposed BO_LIGHT GBM model when compared with the other two models in Table 3.

The efficiency of any algorithm is measured using the computational resource cost that is, memory and time. Table 4 depicts the proposed model execution time, that is 0.32, 36.29 and 0.23 minutes respectively, while the memory consumption of each model is 44.1 (MiB) for Light GBM, 58 (MiB) for Light GBM - Grid Search model and 32 (MiB) for the proposed work of this study. The significant difference in the results in terms of execution time between Light GBM, Light GBM - Grid Search, and the proposed BO-LIGHT GBM model, is apparently due to their different time complexities as explained.

Table 3: Comparison of the three models based on evaluation metrics selected

Evaluation metrics	Light GBM Model	Light GBM- Grid Search Model	BO_LIGHT GBM Model
AUC	70%	69%	**83%**
Accuracy	64.6%	65%	**81%**
Precision	58.5 %	58%	**72%**
Recall	72%	72%	**94%**
F-1 Measure	64.5%	64%	**81%**

Table 4: Efficiency performance of the three models

Metrics	Light GBM	Light GBM –Grid Search	BO-Light GBM
Time (in minutes)	0.32	36.29	**0.23**
Memory (in MiB)	44.1	58	**32**

The result demonstrates that the runtime of both hyper-parameter tuning approaches grows gradually as the dimensions' increases, and Bayesian based method requires less time to obtain the optimal hyper-parameter values. Hence, it can be seen that the proposed model performed better in terms of model efficiency. Thus, we can conclude that the proposed BO_LIGHT GBM Model performs better than the Light GBM -Grid Search model, by achieving higher accuracy, AUC score and lesser computational time and memory. Figure 8 compares the results of the 3 models while figure 9 depicts the AUC-ROC of the 3 models.

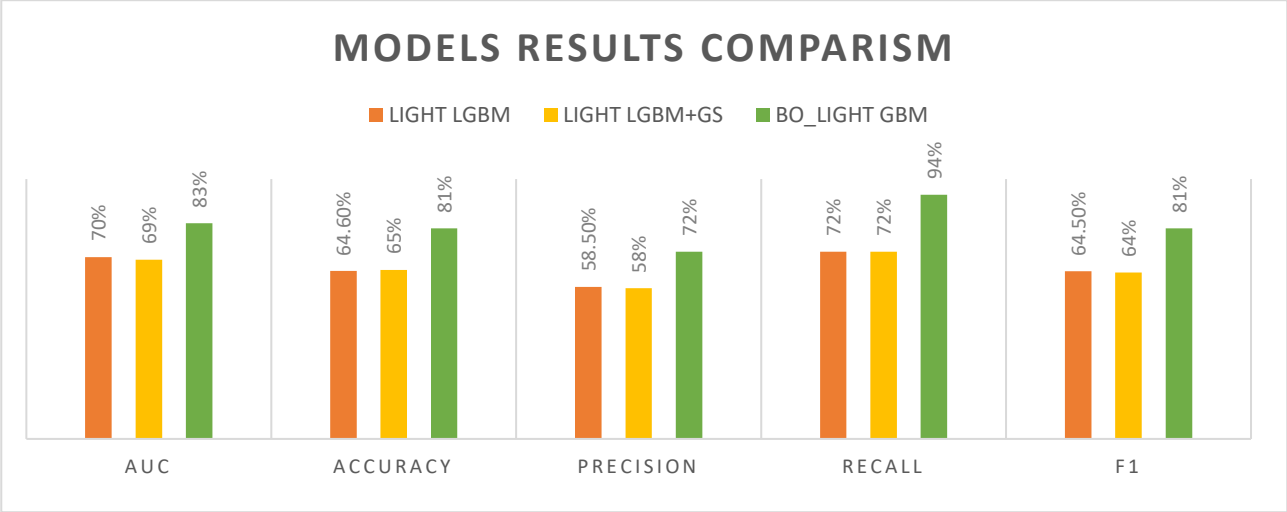

Figure 8: Models results comparison

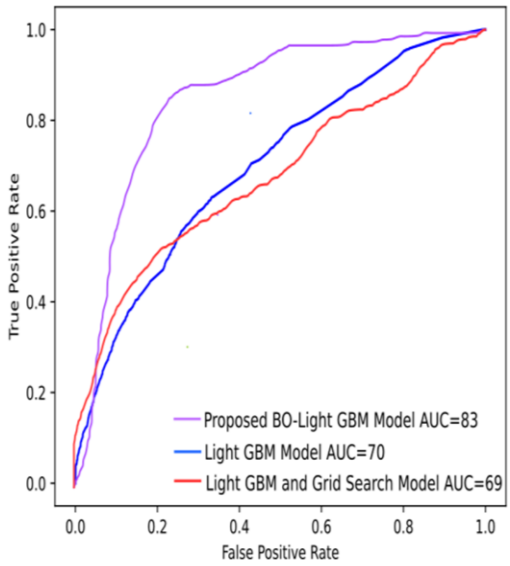

Figure 9: ROC -AUC graph Comparison of the models

In comparison with existing studies, our proposed BO-Light GBM model demonstrated superior predictive accuracy, efficiency, and resource optimization. For example, Wang et al. (2022) achieved an AUC of 75% and an accuracy of 70% using a Light GBM model with Random Search optimization, but required 15 minutes of execution time and 45 MiB of memory. Similarly, Khan et al. (2021) and Singh et al. (2023) implemented Light GBM with Bayesian and Grid Search optimizations, respectively, but fell short with lower AUCs (78% and 74%) and higher resource demands (execution times of 20 and 40 minutes and memory usage of 50 and 60 MiB, respectively). In contrast, our BO-Light GBM model achieved an AUC of 83% and an accuracy of 81% within just 0.23 minutes and 32 MiB, underscoring its suitability for resource-constrained environments. The comparison of our BO-Light GBM model with existing literature is summarized in Table 5.

Table 5: Comparison of BO-Light GBM Model with Existing Vulnerability Prediction Models

Model	Accuracy	Precision	Recall	F-Measure	AUC	Execution Time (Minutes)	Memory (MiB)
Khan et al. (2021)	73	68	85	74	78	20	50
Wang et al. (2022)	70	65	80	71	75	15	45
Singh et al. (2023)	69	62	79	69	74	40	60
proposed BO-LightGBM	**81**	**72**	**94**	**81**	**83**	**0.23**	**32**

CONCLUSION

In this work, we introduced an enhanced model to predict vulnerability exploitation, crucial for patch prioritization, through a series of experiments. Our proposed BO-LIGHT GBM model outperformed existing methods, leveraging Bayesian optimization for improved accuracy and efficiency. Comparing models, the unoptimized LGBM achieved 64.6% accuracy, while the baseline Light GBM-Grid Search reached 65%. In contrast, our BO-Light GBM achieved an accuracy of 81%, demonstrating superior efficiency with minimal resource utilization. This advancement holds potential for IT organizations, vendors, and resource-constrained entities in vulnerability prediction. Our findings shows that our improved BO_LIGHT GBM model holds promise for IT organizations and vendors, particularly in the realms of threat management and patch prioritization. Its potential impact includes enhanced computational efficiency, accuracy, and effectiveness in predicting vulnerabilities with exploits, as well as facilitating timely patch releases and mitigation strategies. However, since our study relied on a single dataset from a sole data source, future research could greatly benefit from exploring diverse data sources, including bug bounty programs and "grey" or "black" market disclosures, even delving into the deep or dark web and also extending the feature set by extracting additional vulnerability-related attributes from various data sources or vendors, given that our current features were predominantly sourced from NVD data.

REFERENCES

Abbadi, M. A., Bustanji, A. M., & Alkasassbeh, M. (2020). Robust Intelligent Malware Detection using Light GBM Algorithm. International Journal of Innovative Technology and Exploring Engineering, 9(6):1253-1260. DOI: 10.35940/ijitee.F4043.049620

Agarwal, V. (2015). Research on data preprocessing and categorization technique for smartphone review analysis. *International Journal of Computer Applications*, 131(4), 30-36. https://doi.org/10.5120/ijca2015907309.

Taha, A. A., & Malebary, S. J. (2020). An intelligent approach to credit card fraud detection using an optimized Light Gradient Boosting Machine. *IEEE Access, 8*(1), 25579–25587. https://doi.org/10.1109/ACCESS.2020.2971354

Betrò, B. (1991). Bayesian methods in global optimization. *Journal of Global Optimization*, *1*(1), 1–14.

Bhatt, N., Adarsh, A., & Yadavalli, V. S. S. (2020). *Exploitability prediction of software vulnerabilities*. *August*, 1–16. https://doi.org/10.1002/qre.2754

B

ilge, L., & Dimitras, T. (2012). Before we knew it: an empirical study of zero-day attacks in the real world. *In: Proceedings of the 2012 ACM Conference on Computer and CommunicationsSecurity*, 833–844.

Bozorgi, M., Saul, L. K., Savage, S., & Voelker, G. M. (2010). *Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits*.

Bullough, B. L., Yanchenko, A. K., Smith, C. L., & Zipkin, J. R. (2017). Predicting exploitation of disclosed software vulnerabilities using open-source data. *IWSPA 2017 - Proceedings of the 3rd ACM International Workshop on Security and Privacy Analytics, Co-Located with CODASPY 2017*, 45–53. https://doi.org/10.1145/3041008.3041009

Chen, T., Guestrin, C., Ke, G., Meng, Q., & Finley, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–3157.

DeCastro-García, N., Muñoz Castañeda, Á. L., Escudero García, D., & Carriegos, M. V. (2019). Effect of the sampling of a dataset in the hyperparameter optimization phase over the efficiency of a machine learning algorithm. *Advances in Complex Systems and Their Applications to Cybersecurity*, 2019, Article 6278908. https://doi.org/10.1155/2019/6278908

Dewancker, I., McCourt, M., & Clark, S. (2016). Bayesian Optimization Primer. *SigOpt*.

Edkrantz, M., & Said, A. (2015). Predicting cyber vulnerability exploits with machine learning. *In SCAI*.

Ehrenfeld, J. M. (2017). Wannacry, cybersecurity and health information technology: A time to act. *Journal of Medical Systems*, *41*(4), 104.

Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter tunning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. *Ilormatics*, *https://doi.org/10.33990/informatics 8040079*, 8,79.

Fang, Y., Liu, Y., Huang, C., & Liu, L. (2020). FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. PLoS ONE 15(2): e0228439. .0228439. *PLOS ONE*, *15*(2), 1–28.

Feurer, M., & Hutter, F. (2019). Hyperparameter Optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Authomated Machine Learning Methods,Systems,Challenges* (pp. 3–35). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_1

Flashpoint. (2021). Beyond CVE and NVD: The Full Vulnerability Intelligence Picture. Retrieved from https://flashpoint.io

Frei, S., May, M., Fiedler, U., & Plattner, B. (2006). Large-scale vulnerability analysis. *In Proc. of LSAD'06 ACM*, 131–138.

Hoque, M. S., Jamil, N., Amin, N., & Lam, K.-Y. (2021). An improved vulnerability exploitation prediction model with novel cost function and custom trained word vector embedding. Sensors, 21(12), Article 4220. https://doi.org/10.3390/s21124220

James, B., & Bengio, Y. (2012). Optimization, Random Search for Hyper-Parameter. *Journal of Machine Learning Research*, *13*, 281–305.

Ju, Y. U. N., Sun, G., Chen, Q., & Zhang, M. I. N. (2019). A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting. *IEEE Access*, *7*, 28309–28318. https://doi.org/10.1109/ACCESS.2019.2901920

Khan, A., Ali, M., & Rahman, F. (2021). Application of LightGBM in security prediction models. *International*

Journal of Cyber Security and Digital Forensics, 10(2), 123–130.
Luca, A., & Fabio, M. (2012). No Title A Preliminary Analysis of Vulnerability Scores for Attacks in Wild. *ACM 978-1-4503-1661-3/12/10*.

Mingzhu, T., Qi, Z., Steven, X. D., Huawei, W., Linlin, L., Wen, L., & Bin, H. (2020). An Improved LightGBM Algorithm for Online Fault Detection of Wind Turbine Gearboxes. *Energies*, 1–16. https://doi.org/doi:10.3390/en13040807

Mohammed, A., Eric, N., Krishna, D., Senguttuvan, M., Jana, S., & Paulo, S. (2017). Proactive Identification of Exploits in the Wild Through Vulnerability Mentions Online. *International Conference on Cyber Conflict*.

National Institute of Standards and Technology. (n.d.). National Vulnerability Database FAQ. *NIST.gov*. Retrieved from https://nvd.nist.gov/

Nazgol, T., Palash, G., Mohammed, A., Paulo, S., & Kristina, L. (2018). DarkEmbed: Exploit Prediction with Neural Language Models. *The Thirtieth AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*, 7849–7854.

Sabottke, C., Suciu, O., Dumitraş, T., Sabottke, C., & Dumitras, T. (2015). *Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits This paper is included in the Proceedings of the Vulnerability Disclosure in the Age of Social Media*.

Shahriari, B., Swersky, K., Wang, Z., Adams, R., & Freitas, N. de. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *IEEE Access*, *104*(1), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

Singh, P., Kumar, V., & Mehta, R. (2023). LightGBM hyperparameter tuning for exploit detection in resource-constrained environments. *Journal of Network and Computer Applications, 45*(1), 25–33.

Suciu, O., Nelson, C., Lyu, Z., Bao, T., & Dumitras, T. (2022). Expected exploitability: Predicting the development of functional vulnerability exploits. In *Proceedings of the 31st USENIX Security Symposium* (pp. 377-394). USENIX Association.
https://www.usenix.org/conference/usenixsecurity22/presentation/suciu20.

Wang, J. A., & Guo, M. (2009). OVM: An ontology for vulnerability management. In Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (pp. 34:1–34:4). ACM. https://doi.org/10.1145/1558607.1558646

Wang, Y., Chen, L., & Li, J. (2022). Optimizing LightGBM for cyber vulnerability prediction. *Computers & Security, 40*(3), 56–64.

Wang, Y., & Wang, T. (2020). Application of improved LightGBM model in blood glucose prediction. Applied Sciences, 10(9), 3227. https://doi.org/10.3390/app10093227