DOI: 10.15849/ IJASCA.230720.03

Int. J. Advance Soft Compu. Appl, Vol. 15, No. 2, July 2023
Print ISSN: 2710-1274, Online ISSN: 2074-8523
Copyright © Al-Zaytoonah University of Jordan (ZUJ)

Building Machine Learning Model with Hybrid Feature Selection Technique for Keylogger Detection

Mutaz Saad Alsubaie, Samer H. Atawneh, and Mohammed Said Abual-Rub

College of Computing and Informatics, Saudi Electronic University, Riyadh 11673, Saudi Arabia
e-mail: G200319885@seu.edu.sa; motaz.sa1@gmail.com
College of Computing and Informatics, Saudi Electronic University, Riyadh 11673, Saudi Arabia
e-mail: satawneh@seu.edu.sa
Department of Management Information Systems, School of Business, King Faisal University, Hofuf, Saudi Arabia
e-mail: mabualrub@kfu.edu.sa

Abstract

The option to steal a significant quantity of important information without the owner of the message's consent is provided by keyloggers, which are tools designed to record each keystroke made on the computer. Online criminals often employ malware-infected software to attack mobile devices like smartphones and tablets. In addition, hackers are becoming smarter over time. It will be easier for them to add a keylogger to a website than a software program because users must download and install it on their devices before accessing it. All these processes, however, are not required for a website. Websites can be run on any platform. They might pick any user as their target. Thus, social networking sites, internet banking, and emails are accessible to hackers. This paper aims to develop a machine learning-based model for an in-website keylogger detection for platform-independent devices to enhance internet users' privacy and security. The study employs Random Forest, LightGBM, and CatBoost as classifiers, and uses a hybrid feature selection method, known as Hybrid Ensemble Feature Selection (HEFS), which makes the identification process robust and less runtime complex. When comparing the selected and full features on the adopted classifiers, Random Forest was found to be the best in performance; it experienced a minimal accuracy deterioration of 1.59% while achieving a massive 84.5% reduction in feature space.

Keywords: *Keylogger; Machine Learning; Hybrid Feature Selection; LightGBM; CatBoost; Random Forest.*

1 Introduction

Any software that is expressly meant to harm a computer is referred to as malware. Malware comes in a variety of forms, such as spyware and ransomware. Malware is a tool used by cybercriminals to lock down computers, corrupt files, and obtain illegal access to devices. Malware frequently functions covertly, and the user won't become aware of infection until damage has already been done to the machine [1]. A type of

Received 22 March 2023; Accepted 1 May 2023

malware called ransomware is created to extort money from the victim. Using ransomware, cybercriminals can encrypt a computer and demand payment to unlock it. They can erase the files or damage the hardware if the user doesn't agree with their requirements [2]. Another excellent example of malware that threatens IT environments is spyware. Spyware can lead to events like identity theft and data breaches. For instance, many people now carry out their banking online. Users who unwittingly install spyware on their devices risk providing attackers with information about their banking partners, account numbers, routing numbers, and credit card details [3]. Malwarebytes outlines the various types of spyware, including keyloggers, banking trojans, info-stealers, and password stealers [4].

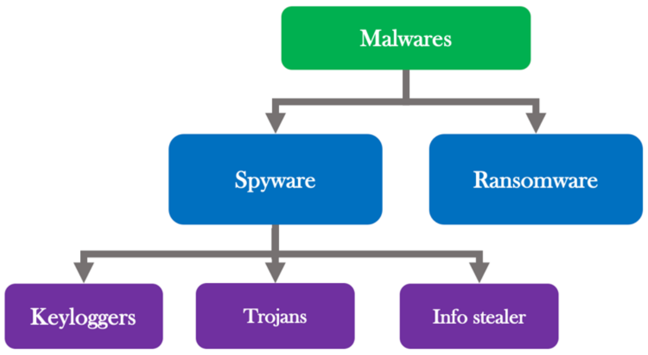


Figure 1. Types of malwares.

Keylogger software is an example of spyware that is reasonably simple to install but has a very negative impact. Keyloggers are programs that are installed on computers that record every keystroke made there. Because of this, hackers can obtain credentials, such as usernames and passwords, from the many websites a user visits. Because there is a real potential that keyloggers could be installed on public devices, it is best to avoid using them for sensitive transactions or to enter usernames and passwords [5]. Fig. 1 summarizes the relationship between malware, spyware, and keyloggers.

Over the years, various theories and strategies have been implemented to address the broad issue of harmful software. However, none of the current methods are adequate when used to address the unique issue of identifying keyloggers. Since signature-based solutions are readily circumvented and require isolating and extracting valid signatures before identifying a potential threat, their usefulness is limited. The implementation of a keylogger is seldom difficult. Even novice programmers may quickly create new iterations of current keyloggers, rendering a previously effective signature useless. Even when analyzing just keyloggers used for illegal activities in Fig. 2, the development of new variations soon renders any signature-based solution useless.

Feature selection is a method of filtering out the important features, as all the features present in the dataset are not equally important. This study shows the usage of a hybrid feature selection technique to avoid overfitting and run-time complexity issues while maintaining the classifiers' performance. The feature selection technique that will be used is called Hybrid Ensemble Feature Selection (HEFS).

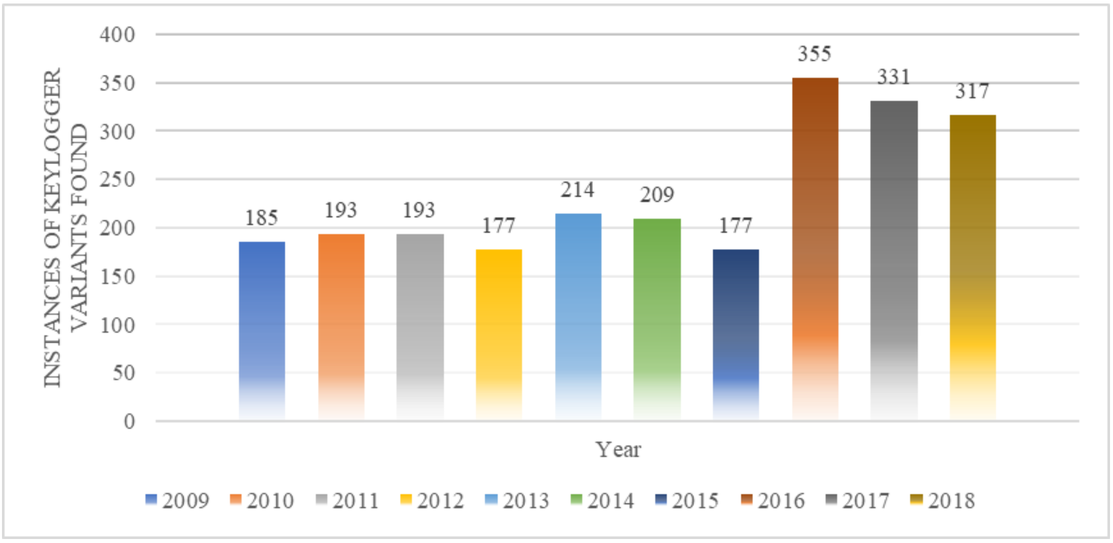


Figure 2. Increasing varieties of keyloggers are being used in crimes.

The main objective of this research is to enhance users' privacy and security by developing a machine-learning detection model for platform-independent devices. The proposed model will use a supervised classification algorithm to analyze the network activity of a computer and try to find the existence of a keylogger. A HEFS method with Random Forest, LightGBM, and CatBoost classifiers will be utilized to make the identification process robust, scalable, and efficient. This research will achieve the following objectives:

- To identify keyloggers by examining the network activities, whether they are installed as distinct processes or operate remotely through APIs.
- To propose a machine learning-based model to detect keyloggers that will operate without human intervention.
- To utilize a robust and scalable hybrid feature selection technique that will help to reduce the runtime analysis while detecting the keylogger.

To identify keyloggers, we will develop and use supervised machine learning-based detection techniques. The method keeps track of all network's transmitted and received activities for all active browser processes. In the presence of a strong correlation, detection is claimed. Both deployment and execution are possible without any special permission. The HEFS method will speed up the classification process while maintaining the classifiers' accuracy.

2 Background

This section encompasses a detailed background study of keyloggers as well as their categories.

2.1 Keyloggers

Keylogger is software that keeps track of all computer activity. Some activities include taking screenshots of every action taken on the computer screen, logging browser activity, and creating distinct keystrokes for every action taken on the computer. Due to their diverse traits, keyloggers are challenging for any detecting program to locate.

Keyloggers were first created during the 1970s [6]. The Union of Soviet Socialist Republics (USSR) later developed and utilized it. Researchers used a hardware keylogger designed specifically for typewriters. It was known as an "electric glitch" because it exploited small changes in the local magnetic flux brought on by the spin and motions of the filament to track the movements of IBM Electric typewriters [6]. Perry Kivolowitz created the first keylogger on November 17, 1983. Keyloggers get more powerful as technology advances and many distinct types of keyloggers are available in the industry that was created utilizing technological advancements. Hardware and software keyloggers are the two basic categories into which keyloggers fall.

2.2 Hardware Keyloggers

Hardware keyloggers may be discreetly installed between the Central Processing Unit and the keyboard cord or built into the computer's internal hardware, making them easy to operate. However, installing the hardware keylogger requires the cybercriminal to have unnoticed physical access to the computer system [7].

2.3 Software Keyloggers

Software keyloggers, as opposed to hardware keyloggers, are simple to install on the victim's computer. Software keystroke loggers are computer programs designed to record keystrokes on the keyboard. It gathers keyboard occurrences, records them remotely, and then sends them to the hacker, who puts the keylogger in place [8]. By utilizing a search engine to look for "keyloggers," we discovered a wide variety of software keyloggers. However, we may classify them into several groups based on their internal design.

2.4 Different Types of Software Keyloggers

Keyloggers based on the kernel may intercept keystrokes as they flow through the kernel by gaining root access and blending in with the operating system. These keyloggers operate at the kernel level, making them formidable and difficult to find. Keyloggers created using this technique may also drive the keyboard. It requires a lot of technical coding to design a keylogger in this manner [9]. The keyboard data is captured through hooking mechanisms used by API-based keyloggers. Keyloggers that rely on APIs connect to active applications' keyboard APIs. This keylogger records keystrokes similarly to other apps [10]. This approach is the one most often used to create keyloggers. Keyloggers with a memory injection basis operate by swapping memory tables connected to system operations and web browsers. By directly injecting malware into memory, malware writers can bypass Windows User Account Control [11]. Fig. 3 depicts the summary of the above discussion.

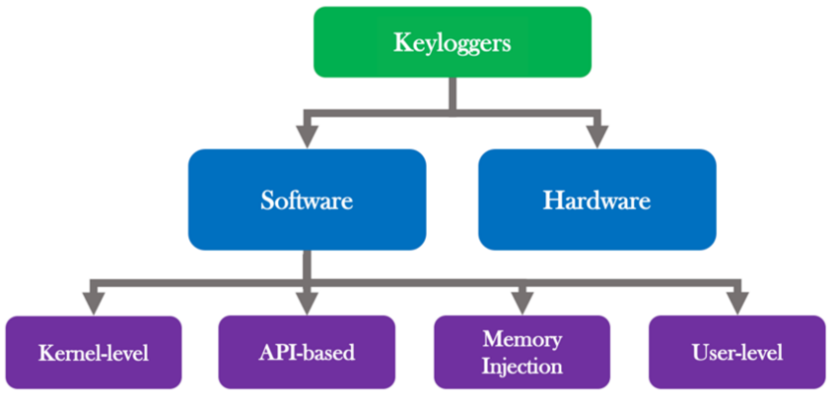


Figure 3. Different types of keyloggers.

3 Related Work

Keyloggers are extremely detrimental tools that keep track of every single our computer's activity. A keylogger is a code fragment that can be put inside any software and will record every keystroke user makes. Our computers are infected with keyloggers due to the passive invasion. When someone inputs their login and password on their computer, the attacker seeks to get the password. The intruder collects all data provided by the victim via a footprinting method, and the information is then used for other unrelated reasons [12]. Malware assaults are particularly unpleasant and difficult to identify and defend against in the cyber world. The script and malware are both included in a keylogger's single software. The keylogger's capability allows it to record every key the user presses, save it in a log file, and subsequently email the file to the specified internet protocol address. It poses a serious risk to the financial system, which is utilized for daily commercial purposes. Several keyloggers' kinds, activities, and features are well outlined [13].

Keylogger detection methods based on software and machine learning haven't been proposed in huge numbers. Here are several studies on the evaluation of keylogger detection. Pillai demonstrated a modified SVM-based framework to identify keyloggers installed on a computer. In their system, 8 open source keyloggers were installed [14]. The suggested approach failed to detect 4 of the 8 keyloggers.

Software keylogger detection employing Anti-Hook, HoneyID, bot identification, and dendritic cell algorithm (DCA) was provided by [15] in 2016. These methods are used to identify keyloggers on computers.

Brown presented popular algorithms, such as XOR, GEFeS, and SDM, to classify keyloggers in Android [16]. For the identification of "self" and "non-self," Two separate detector sets were generated by mAIS. Instances of "self" app detection are distinguished from "non-self" app detection by the "non-self" detector sets. The outputs from both detector sets are utilized to identify and categorize Android keyloggers. XOR achieved 88.33% accuracy, whereas GEFeS and SDM combinedly achieved 93.33%.

Wen L. et al. proposed the supervised learning classifier SVM and the unsupervised dimensionality reduction algorithm PCA-RELIEF [17]. The client and server are the two essential components of the suggested system. If the application's MD5 value matches one of the malicious programs' saved MD5 values, the user interface on the client side will notify them. If not, the installed application's estimated MD5 will be sent to the server. The proposed machine learning framework's accuracy and false positive rates were 95% and 13.3%, respectively.

Using 4 separate classifiers, Hatcher proposed a brand-new machine learning-based architecture [18]. It was designed to function in a cloud setting where several devices may be supported simultaneously. The platform under consideration included an Android app, an Analysis Module, a Google Cloud Messaging (GCM), and Security Web Server. ZeroR, OneR, and J48 have respective accuracy values of 49.7%, 100%, and 100%. The false positive rate for Nave Bayes was noted as 20%, but the accuracy was not given. But for J48, Naive Bayes, OneR, and ZeroR, corresponding detection rates are 45%, 83%, 94.59%, and 90%, respectively.

Anyone trying to compromise the privacy of smartphone users should focus on Android, the world's most widely used operating system. The innovative dataset presented in [19] was gathered in a realistic setting and acquired using a brand-new data-collecting approach based on a unified activity list. The information is broken down into three primary groups: the first group includes typical smartphone traffic, the second group provides traffic data for the deployment of spyware, and the third group represents traffic data for the functioning of malware. It was decided to use the random forest classification approach to verify the suggested model with this dataset. Binary-class (where the number of target variables is two) and multi-class (where the number of target variables is more than two) classification are the two approaches used for data classification. Accuracy-wise, good performance was obtained. For the binary class (where the number of target variables is two) and the multi-class (where the number of target variables is more than two), the overall average accuracy was 79% and 77%. All dataset elements included in the scope of the surveillance included access to social media, phone calls, microphone access, OS activity, and keyloggers.

Innovative cybersecurity methods that recognize harmful Internet Protocol (IP) addresses before communication are needed to stop cybercrimes. IP reputation management is among the greatest methods. A cyber-physical system's security risks may be profiled using IP reputation systems. In their innovative hybrid technique, [20] suggested combining Cyber Threat Intelligence, Dynamic Malware Analysis, Data Forensics, and Machine Learning (ML). A zero-day attack's related IP notoriety is anticipated in the pre-acceptance stage using the idea of big data forensics, and its associated zero-day assaults are classified using behavioral analysis and the Decision Tree approach.

Anti-virus software's traditional protection techniques, such as signature-based malware detection, cannot keep up with the problems posed by modern malware. Malware analysis and detection were represented as machine learning and deep learning problems [21]. When creating these models, the writers followed industry standards. They overcame the dimensionality curse by utilizing various feature reduction techniques, including AutoEncoder. The models created using Random Forest and several layered Deep Neural networks were then compared. According to the findings, random forest performs better in malware detection than deep neural network models. Random Forest attained the maximum accuracy at 99.78%.

To identify Android malware, the authors of [22] proposed a hybrid strategy utilizing machine learning. They built the application's flowchart to learn more about the API. The authors developed time-series and Boolean frequency data sets in an original way using the API information. Depending on three data sets, three detection models for Android malware detection were built, considering API calls, API sequence factors, and API frequency. Finally, an ensemble model is built for conformance. Through many trials, they evaluated and contrasted the reliability and accuracy of the detection models. On 10010 good applications and 10683 bad applications, the trials were run. The findings indicate that the detection approach accomplishes a detection accuracy of 98.98% and has a high degree of precision and consistency.

Using four classifiers—ID3, K-Nearest Neighbors, Decision Tree, C4.5 Decision Tree, and linear SVM, the authors of [23] demonstrated a supervised classification method for identifying Android malware (SVM). The classifiers use the metadata included in the apps' bytecode, including information about important API calls, package-level data, and potentially harmful arguments called, to identify if an application or software is malicious or not. A summarized comparison of methodologies from previous studies is enlisted in Table 1.

Table 1. Comparison of different methodologies from previous studies.

SN	Methodologies	Methods	Research Topic
1	Solairaj et al., 2016	Anit-Hook, HoneyID, Bot Detection, and Dendritic Cell Algorithm	PC Keylogger Detection
2	Hatcher et al., 2016	ZeroR, OneR, Naïve Bayes, and J48	Android Malware Detection
3	Brown et al., 2017	XOR, GEFs, and SDM	Android Keylogger Detection
4	Wen et al., 2017	Support Vector Machine and PCA-RELIEF	Android Keylogger Detection
5	Rathore et al., 2018	Random Forest and Deep Neural Network with Autoencoder	Malware Analysis and Detection
6	Ma et al., 2019	Control Flow Graph with Deep Neural Network, C4.5, and Long Short-Term Memory	Android Malware Detection
7	Pillai et al., 2019	SVM	PC Keylogger Detection
8	Usman et al., 2021	Machine Learning, Dynamic Malware Analysis, Data Forensics, and Cyber Threat Intelligence	Malware Analysis and Detection
9	Qabalin et al., 2022	Random Forest	Spyware Detection

This study provides a machine learning-based model to identify keyloggers operating remotely through websites. A thorough analysis of the literature revealed that few studies were machine learning-based, and no one had created a detection technique for keyloggers operating remotely through websites. Section 4 presents the proposed model.

4 The Proposed Method

This section describes methods, resources, and datasets used in this study to detect keyloggers. An overview of the proposed machine-learning model is shown in Fig. 4. To facilitate comprehension, a top-down presenting method was used, in which the key elements and functions are initially presented without going into detail about the associated algorithm.

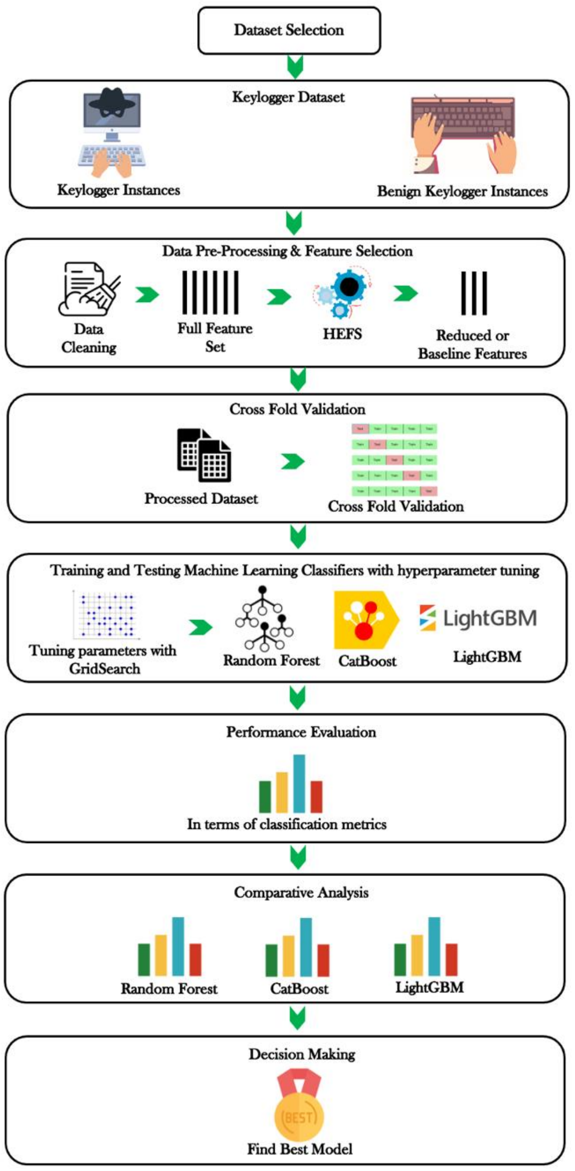


Figure 4. Overview of the proposed machine learning model.

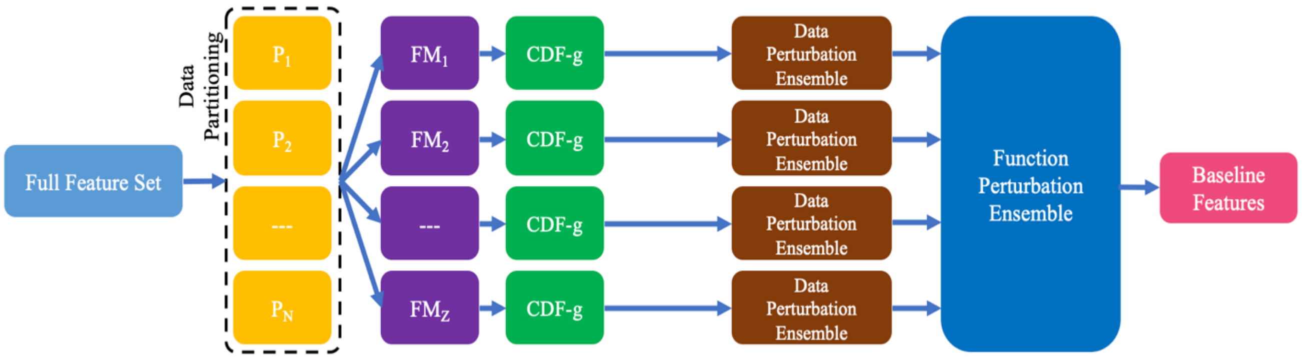
Data collection, data pre-processing, feature selection, implementation of the grid-search algorithm to determine the best parameter for each algorithm, implementation of cross-fold validation to analyze the classification performance of the ML classifiers, analysis of various ML algorithms, and selection of the best algorithms more thoroughly for keylogger detection will all be part of the research methodology.

The initial step in data pre-processing is to look for duplicate occurrences. We will easily eliminate duplicate entries and leave only the unique ones if any are identified. Then, if there are too many NULL entries, we can interpolate those values; otherwise, we can drop them.

We will encode the labels if the dataset contains any string values. We will also examine class distribution to determine whether the dataset is balanced. We can use some oversampling or under-sampling approaches if the dataset is unbalanced. Then, to reduce the number of features and prevent the machine learning classifiers from overfitting, we will feed full feature set data into the HEFS feature selection approach. For later usage, the transformed dataset will be stored in CSV format. GridSearch will choose the classifier parameter that will yield the best results for each technique, and cross-validation will make sure that our model iterates over the whole dataset to assess its classification-related resilience. The ML classifiers are evaluated using classification metrics following training and testing. This procedure will keep going until GridSearch determines the classifiers' ideal parameter. Then, we compared the three classifiers and selected the most effective model for the keylogger detection task.

4.1 Hybrid Ensemble Feature Selection Technique (HEFS)

The following is a concise overview of the suggested HEFS model. As indicated in Fig. 5, the SSN and FMZ abbreviations should be used to represent the N-th partition and Z-th filter measure, respectively. Before being separated into N divisions by stratified subsampling, the whole dataset is randomly selected. To make the samples in each partition more evenly distributed, randomization corresponds to the relocation of the sample rows.


Data Partitioning
Full Feature Set
P_1
P_2

P_N
FM_1
FM_2

FM_Z
CDF-g
CDF-g
CDF-g
CDF-g
Data Perturbation Ensemble
Data Perturbation Ensemble
Data Perturbation Ensemble
Data Perturbation Ensemble
Function Perturbation Ensemble
Baseline Features

Figure 5. Overview of HEFS.

The filter measure FMZ calculates the filter measure values for a dataset partition SSN using the partition's raw attribute values. As soon as the filter measure values are ordered, they are sent on to the next step, where the CDF-g method generates the cut-off rank for the features. A list of feature cut-off rankings is generated for each dataset division using filter measure FMZ after repeating this process for N = 1, 2..., N, a list of feature cut-off ranks {τ1, Z, τ2, Z..., τJ, Z} is created for each dataset partition by using filter measure FMZ. The CDF-g approach is broadened into a hybrid ensemble structure comprising data perturbation and function perturbation techniques to increase the robustness of baseline features and lessen the generalization problem. While function perturbation includes using several filter measures within the same dataset, data perturbation implies subsampling the dataset into distinct segments. The whole feature selection architecture was referred to as the Hybrid Ensemble Feature Selection by [24].

4.2 Classification Algorithms

In this section, the machine learning classification algorithms employed in this paper for detecting keyloggers are described in detail.

4.2.1 CatBoost

Dorogush proposed CatBoost, a novel gradient-boosting technique that reliably employs categorical variables with the minimum percentage of information loss [25]. Different from previous gradient-boosting methods is CatBoost. Target leaking is initially addressed using sorted boosting, a useful variation of gradient boosting techniques. This technique also performs well with small datasets. CatBoost can also control category variables. This handling, which comprises converting the initial categorical variables to one or more numerical values, is often completed during the pre-processing stage. Additionally, CatBoost can be utilized successfully with various data forms and types. CatBoost has recently been used in the financial industry with several different data formats, including time series data [26]. A new binary feature for each category replaces the original variable. Using random permutations to estimate leaf values when choosing the tree structure, the algorithm, according to [26], avoids overfitting by conventional gradient boosting algorithms. CatBoost's main predictor is a binary decision tree.

4.2.2 Random Forest

A very well-known ML classification algorithm that has been extensively used to solve several classification problems is named Random Forest [27]. Random Forest uses decision trees and ensemble architecture to improve classification accuracy. Voting takes place between each tree, and each variable is then assigned to the output class that is most likely to be created. The output function is derived by calculating the mean as follows:

$$R = argmax\ \frac{1}{Z} \sum_{s=1}^{T} d_t(\frac{y}{k}) \tag{1}$$

where $d_t\left(\frac{y}{k}\right)$ Jack represents each tree's probability distribution, and k represents an instance of the test set. For building a framework of decision trees to handle complicated outliers, Random Forest is a reasonable solution. Random forest was further encouraged because it was easy to construct and understand, which enhanced predicting.

4.2.3 LightGBM

A data model called LightGBM is based on Microsoft's GRADIENT BOOSTING DECISION TREE (GBDT), proposed in 2017. Like other boosting algorithms, GBDT integrates weak classifiers to create an effective learner. The Gradient Boosting Decision Tree approach can only be utilized with regression trees since each tree learns the inferences and variances of all previous trees in the process. The variation of each projected outcome and the target value is used as the goal of further learning, creating a current residual regression tree. The decision trees' results are combined to create the final expected output. Despite the good learning impacts that Gradient Boosting Decision Tree has had on various machine learning applications. Currently, the LightGBM algorithm has been proposed. It significantly increases forecasting speed while maintaining prediction accuracy and uses less memory. Creating a decision tree frequently requires using the traditional GBDT algorithm.

5 Performance Evaluation

In this section, we discussed the dataset, performance metrics, and methodology evaluation, as well as the results and discussions, are presented.

5.1 Dataset Descriptions

This study utilizes a publicly available keylogger dataset licensed under GNU Free Documentation to carry out the experiment. This section describes the properties and a few statistics related to the used datasets. The dataset has been viewed over 3500 times and downloaded 245 times. The dataset originated from Canadian Institute for Cybersecurity (CIC) website (Keylogger Detection, 2021) [28]. It contains 523617 samples, 309415 benign samples and 214202 keylogger samples, and 85 features. Table 2 contains the full information regarding the dataset.

Table 2. Detail description of the keylogger dataset.

Associated Task	**Number of Instances**	**Number of Features**	**Attribute Characteristics**	**Keylogger Samples**	**Benign Samples**	**Published Date**
Classification	523617	85	Discrete & Continuous	214202	309415	September 2021

5.2 Technology Used

In this study research, a hybrid feature selection method is proposed, and Python, which has many source libraries, is used to converse machine learning algorithms. Additionally, it gives the capability of carrying out the classification operation with high Accuracy using fundamental characteristics. The following procedures are used to test the general methodology:

Software packages: This experiment used the Jupyter Notebook. It is a web-based interactive computing environment. Many popular data science languages, including Python, Julia, Scala, R, and others, may be used with the Jupyter Notebook. Python was selected as the programming language because of its comprehensive machine-learning packages, which include Sci-Kit Learn, Pandas, Numpy, Matplotlib, Seaborn, etc.

Workstation: A workstation with the following configuration was utilized to experiment.

- Operating System: macOS Monterey (Version 12.6)
- Processor: Intel Core i9 @2.3 GHz
- Memory: 16 Gigabytes of Memory
- Secondary Storage: 1TB of SSD
- Graphics Card: AMD Radeon Pro 5500M 8GB

5.3 Performance Metrics

The proposed systems' effectiveness will be evaluated using a variety of performance criteria, including Accuracy, Precision, Recall, and F1-score. This section has gone into detail about those performance metrics.
Accuracy is an established classification evaluation statistic. It is recognized as the percentage of properly categorized instances to all instances, compared to the error value, which employs wrongly classified samples rather than correctly identified ones. A mathematical formula for Accuracy is shown in Equation 2.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \qquad (2)$$

The ratio between positively predicted instances to all instances in the true positives class is known as Recall. The recall question is how many keylogger instances have the machine learning system properly labeled. Another name for Recall is sensitivity. Equation 3 illustrates the mathematical formula for Recall or sensitivity.

$$Recall\ or\ Sensitivity = \frac{TP}{TP+FN} \qquad (3)$$

In terms of positive instances, precision is the proportion of accurately projected instances to all expected positive instances. This measure answers the question of how many instances listed as keyloggers are true. Low false positive rates are associated with high precisions value. Equation 4 illustrates the precision formula in mathematics.

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

The harmonic mean between Precision and Recall is considered an F1-Score. Hence, this metric accounts for false negatives (FN) and false positives (FP). Although it is not conceptually as simple to comprehend as Accuracy, the F1 Score is often more helpful than Accuracy, particularly if any dataset has unbalanced class labels. The most excellent Accuracy is achieved when the costs of FP and FN are comparable. Including both Precision and Recall is preferable if the costs of FP and FN vary significantly. Equation 5 illustrates the mathematical formula for Accuracy.

$$F1 - Score = \frac{2\times(Recall \times Precision)}{Recall+Precision} \qquad (5)$$

5.2 Evaluation of Methodology

Data pre-processing is an important stage before insertion into a machine learning algorithm. The data was first stored in a data frame using Python code. We have used the Hybrid Ensemble-based Feature Selection (HEFS) technique for feature selection. A hybrid ensemble-based feature selection technique automatically chooses the cut-off rank value from filtered measures evaluated by the Information Gain filter method. The pseudocode for this feature selection technique is mentioned in detail in [36]. The cut-off rank value determined automatically by the CDF-g is 0.159262.

The feature subset chosen by HEFS is enlisted in Table 3. The dataset contains 84 features which are considered a full feature set. After applying HEFS, the features are reduced to 13. Features with information gain value less than 0.159262 are discarded.

Table 3. Feature subset after applying HEFS.

Number of features in Full Feature Set	**Number of features in feature subset after HEFS**	**Features in the feature subset**
84	13	' Source Port', ' Flow Duration', 'Flow Bytes/s', ' Flow Packets/s', ' Flow IAT Mean', ' Flow IAT Std', ' Flow IAT Max', ' Flow IAT Min', 'Fwd IAT Total', ' Fwd IAT Mean', ' Fwd IAT Max', 'Fwd Packets/s', ' Bwd Packets/s'

Table 4 illustrates the parameter lists and suitable parameters determined by GridSearch and their corresponding mean accuracy from 5-fold cross-validation.

Table 4. Optimal hyperparameter values for the classification algorithm.

Classification Algorithms	**Hyperparameter Name**	**Hyperparameter Values**	**Optimal Hyperparameter value**	**Mean Accuracy on 5-fold Cross Validation**
Random Forest	n_estimators	20, 50, 100, 110, 120, 130, 140, 150	110	0.9524
	criterion	'gini', 'entropy'	‘gini’	
LightGBM	learning_rate	0.01, 0.1, 0.3, 0.4, 0.5, 0.6, 0.8, 0.9	0.9	0.7128
	n_estimators	20, 50, 100, 120, 130, 150, 160, 180	130	
CatBoost	depth	4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 17	15	0.9135
	learning_rate	0.001, 0.01, 0.02, 0.03, 0.04, 0.1, 0.3, 0.5, 0.7, 0.9	0.9	

6 Research Results

This section described the performance of the machine learning methods using both the full feature set and selected features. The accuracy, precision, re-call, F1-score, and AUC score of algorithms are also compared in this section. The K-fold cross-validation technique was used to train and test machine learning classification algorithms like Random Forest, LightGBM, and CatBoost. The GridSearch method was used to find the classification algorithm's optimal hy-perparameters. The effectiveness of various machine learning methods is covered in the next sections.

6.1 Results of Random Forest

Random Forest provided the following results for selected features and the full feature set. Table 5 and Fig. 6 show the detailed performance of Random Forest both on selected features and the full feature set:

Table 5. Detailed performance results of random forest on both selected features and full feature set.

Classifier	Features	Accuracy	Precision	Recall	F1-Score	AUC Score
Random Forest	Selected Features	0.9612	0.9608	0.9588	0.9598	0.990
	Full Feature Set	0.9771	0.9771	0.9755	0.9763	0.996

Table 5 shows the evaluation metrics for the Random Forest classifier using two different feature sets: "Selected Features" and "Full Feature Set".

For the "Selected Features" set, the Random Forest classifier achieved an accuracy of 0.9612, a precision of 0.9608, a recall of 0.9588, an F1-score of 0.9598, and an AUC score of 0.990. For the "Full Feature Set", the Random Forest classifier achieved a higher accuracy of 0.9771, a precision of 0.9771, a recall of 0.9755, an F1-score of 0.9763, and a higher AUC score of 0.996. It appears that using the "Full Feature Set" resulted in better performance for the Random Forest classifier in this context. However, it's important to note that this may not always be the case, as the optimal feature set can depend on the specific dataset and classification task at hand.

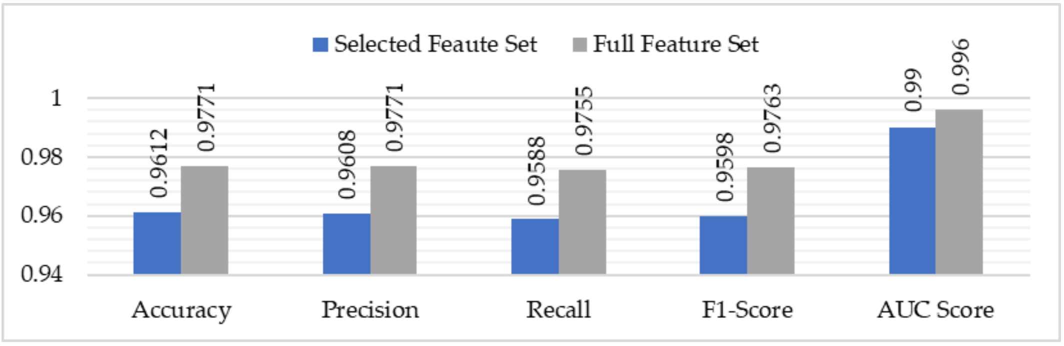


Figure 6. Detailed performance result graph of random forest on both selected features and full feature set.

When comparing selected and full features on Random Forest, the selected features only experienced a minimal accuracy deterioration of 1.59% while achieving a massive 84.5% reduction in feature space. In short, the evaluation results have validated the effectiveness of using the Hybrid Ensemble Feature Selection technique.

6.2 Results of LightGBM

The hyperparameters for the LightGBM classifier are likewise specified in the 'sklearn' module, and LightGBM was implemented in Python. The confusion matrix obtained from the LightGBM classification method is shown in the figure with and without selected features. The results from LightGBM for both the full feature set and the selected features are presented in Figures 5.6 and 5.7 and will be considered for analysis.

Table 6 and Fig. 7 show the detailed performance of LightGBM both on selected features and the full feature set:

Table 6. Detailed performance results of LightGBM on both selected Features and the full features set.

Classifier	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC Score**
LightGBM	Selected Features	0.7134	0.7112	0.6829	0.6867	0.768
	Full Feature Set	0.7136	0.7461	0.6658	0.6638	0.777

Table 6 shows the evaluation metrics for the LightGBM classifier using two different feature sets: "Selected Features" and "Full Feature Set". For the "Selected Features" set, the LightGBM classifier achieved an accuracy of 0.7134, a precision of 0.7112, a recall of 0.6829, an F1-score of 0.6867, and an AUC score of 0.768. For the "Full Feature Set", the LightGBM classifier achieved a slightly higher accuracy of 0.7136, a higher precision of 0.7461, a lower recall of 0.6658, a lower F1-score of 0.6638, and a higher AUC score of 0.777.

It appears that the two feature sets resulted in similar overall performance for the LightGBM classifier, with the "Full Feature Set" performing slightly better in terms of precision and AUC score, but worse in terms of recall and F1-score. As with the previous example, the optimal feature set can depend on the specific dataset and classification task at hand.

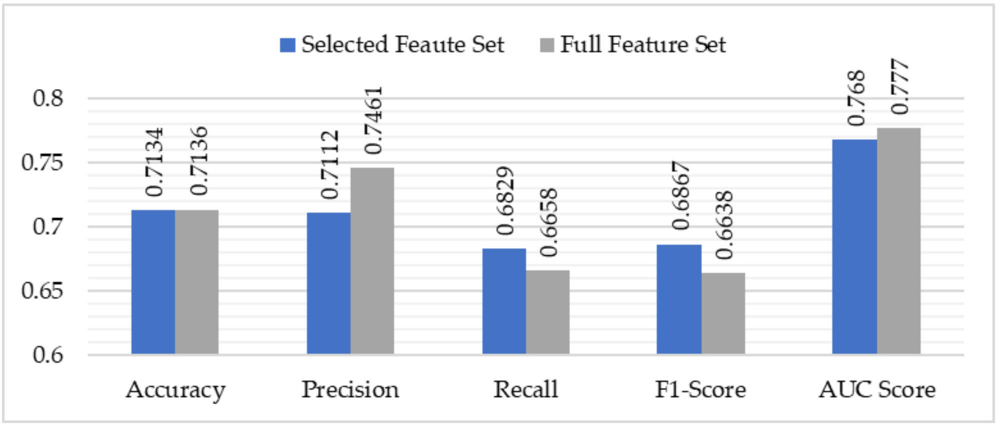


Figure 7. Detailed performance results graph of LightGBM on both selected features set and full feature set.

Selected features on LightGBM achieved a staggering 84.5% reduction in feature space while only suffering a modest accuracy degradation of 0.02% compared to complete features. The usefulness of applying the Hybrid Ensemble Feature Selection approach has, in brief, been proven by the assessment outcomes.

6.3 Results of CatBoost

Table 7 and Fig. 8 show the detailed performance of CatBoost both on selected features and the full feature set:

Table 7. Detailed Performance Results of CatBoost on both Selected Features and the Full Feature Set.

Classifier	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC Score**
CatBoost	Selected	0.9251	0.9250	0.9197	0.9221	0.970

	Features					
	Full Feature Set	0.7691	0.7837	0.7375	0.7449	0.846

Table 7 shows the evaluation metrics for the CatBoost classifier using two different feature sets: "Selected Features" and "Full Feature Set". For the "Selected Features" set, the CatBoost classifier achieved a higher accuracy of 0.9251, a higher precision of 0.9250, a higher recall of 0.9197, a higher F1-score of 0.9221, and a higher AUC score of 0.970. For the "Full Feature Set", the CatBoost classifier achieved a lower accuracy of 0.7691, a lower precision of 0.7837, a lower recall of 0.7375, a lower F1-score of 0.7449, and a lower AUC score of 0.846. It appears that using the "Selected Features" set resulted in much better performance for the CatBoost classifier in this context, with higher scores across all evaluation metrics. However, as always, the optimal feature set can depend on the specific dataset and classification task at hand.

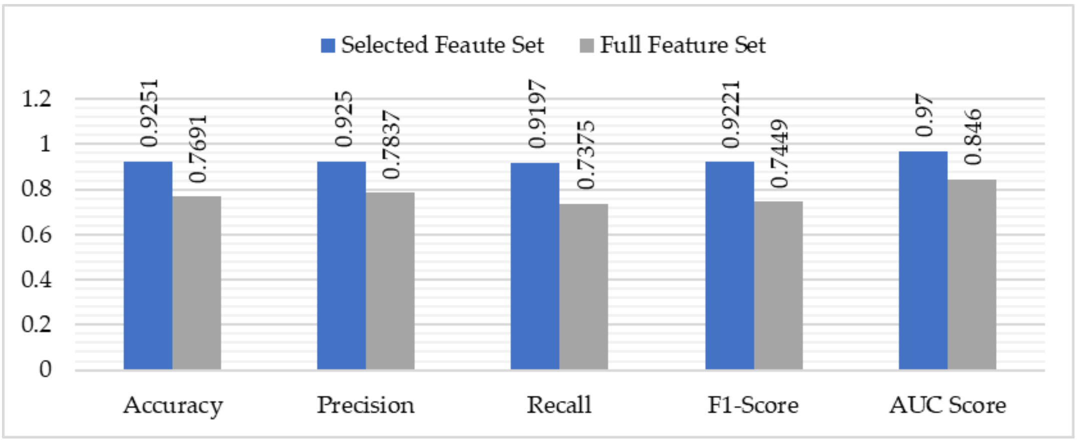


Figure 8. Detailed performance result graph of CatBoost on both selected features set and full feature set.

On CatBoost, while comparing selected and full features, the selected features gained a maximum accuracy boost of 15.6% despite massively reducing the feature space by 84.5%. In other words, the evaluation's findings have shown that the hyperparameter tuning algorithm and Hybrid Ensemble Feature Selection approach is successful.

6.4 Comparative Analysis of Machine Learning Algorithms

Table 8 shows the combined results of the performance of all three classifiers on the selected and full features set.

Table 8. Performance comparison of all machine learning algorithms on the selected feature set and full feature set.

Classifier	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC Score**
Random Forest	Selected Features	0.9612	0.9608	0.9588	0.9598	0.990
	Full Feature Set	0.9771	0.9771	0.9755	0.9763	0.996

LightGBM	Selected Features	0.7134	0.7112	0.6829	0.6867	0.768
	Full Feature Set	0.7136	0.7461	0.6658	0.6638	0.777
CatBoost	Selected Features	0.9251	0.9250	0.9197	0.9221	0.970
	Full Feature Set	0.7691	0.7837	0.7375	0.7449	0.846

Table 8 shows that the Random Forest classifier has the highest accuracy and AUC score among the three models. When using the full feature set, it achieves an accuracy of 0.9771 and an AUC score of 0.996. The precision, recall, and F1-score are also quite high, indicating that the model is performing well in both identifying positive and negative cases.

On the other hand, the LightGBM model appears to be underperforming, with an accuracy of only 0.7134 when using the selected features and 0.7136 when using the full feature set. Its precision, recall, and F1-score are also lower than the other two models, suggesting that it may not be as effective in identifying positive and negative cases.

The CatBoost model is performing reasonably well, with an accuracy of 0.9251 when using the selected features and 0.7691 when using the full feature set. While its performance is not as high as the Random Forest model, it is still better than the LightGBM model. The precision, recall, and F1-score of CatBoost are also relatively high, indicating that it is performing well in identifying both positive and negative cases.

6.4.1 Comparison of Precision Metric

Precision refers to showing the learning ability of a machine learning classification algorithm and how correctly it can detect positive instances. It shows the ratio between all positives and true positive (TP) instances. Fig. 9 shows that Random Forest achieved the highest 96.08% precision value for the selected feature set compared to the other two classifiers. When comparing selected and full features on Random Forest, the selected features only experienced a minimal precision deterioration of only 1.03%. The weakest performer among all the classifiers for both the selected and full feature sets was LightGBM. CatBoost achieved quite a better result (approximately 14.13% better) for the selected feature compared to the full feature set. This improvement is mostly due to optimal hyperparameters searched for by GridSearch.

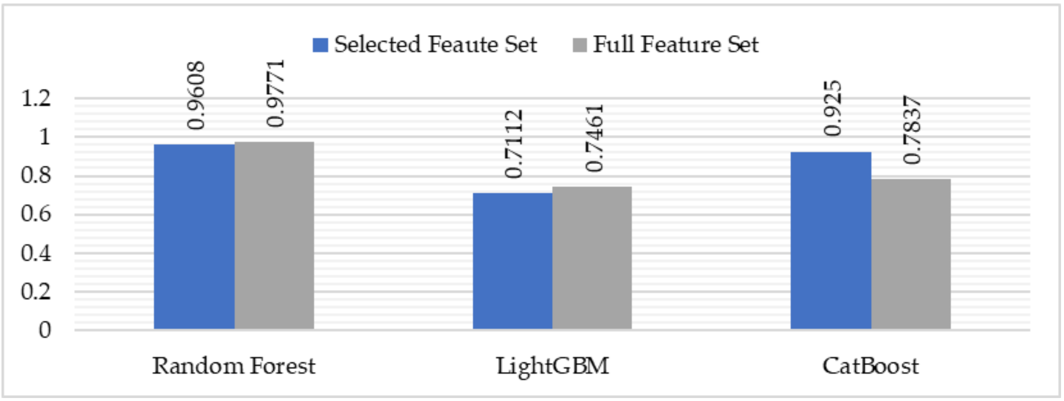


Figure 9. Precision comparison on both selected features set and full feature set.

6.4.2 Comparison of Accuracy Metric

Accuracy refers to the learning ability of the machine learning classification algorithm and how correctly it can detect the benign and keylogger instances out of the total instances. Fig. 10 shows that Random Forest achieved the highest 96.12% accuracy value for the selected feature set compared to the other two classifiers. When comparing selected and full features on Random Forest, the selected features only experienced a minimal accuracy deterioration of only 1.59%. The weakest performer among all the classifiers for both the selected and full feature sets was LightGBM. CatBoost achieved quite a better result (approximately 14.14% better) for the selected feature compared to the full feature set. This improvement is mostly due to optimal hyperparameters searched for by GridSearch.

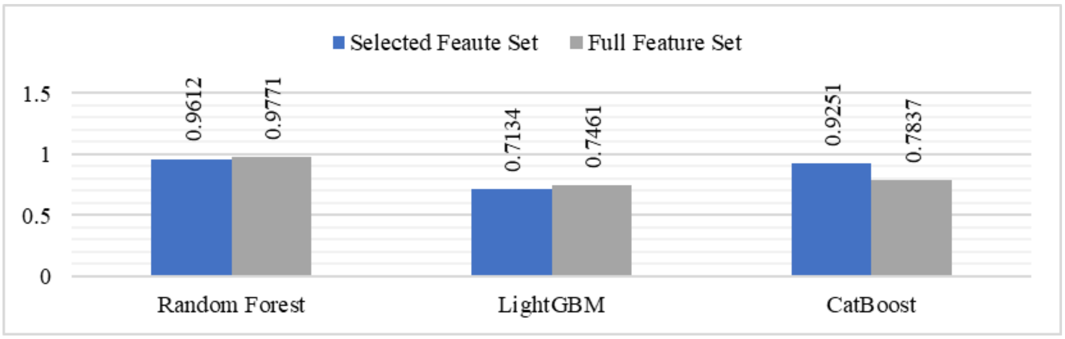


Figure 10. Accuracy comparison on both selected features set and full feature set.

6.4.3 Comparison of Recall or Sensitivity Metric

The ratio between positively predicted instances to all instances in the true positives class is known as Recall. The recall question is how many keylogger instances have the machine learning system properly labeled. Compared to the other two classifiers, Random Forest obtained a recall value of 95.88% for the selected feature set, as shown in Fig. 11. On Random Forest, the difference between recall degradation for the full and selected features was just 1.67%. CatBoost significantly outperformed the full feature set in terms of performance (by around 18.22% better) for the selected feature. This improvement is mostly attributable to GridSearch's use of optimal hyperparameters. LightGBM had the worst performance of all the classifiers for the entire feature set and the selected feature set.

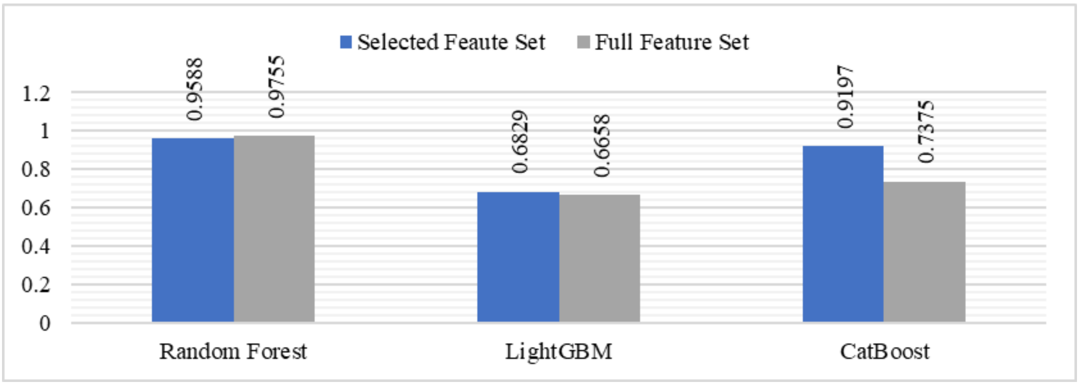


Figure 11. Recall comparison on both the selected features set and the full feature set.

6.4.4 Comparison of F1-Score

The harmonic mean between Precision and Recall is considered an F1-Score. Hence, this metric accounts for false negatives (FN) and false positives (FP). Fig. 12 shows that Random Forest achieved the highest 95.98% F1-score for the selected feature set compared to the other two classifiers. When comparing selected and full features on Random Forest, the selected features only experienced a minimal f1-score deterioration of only 1.65%. CatBoost achieved quite a better result (approximately 17.72% better) for the selected feature compared to the full feature set. This improvement is mostly due to optimal hyperparameters searched for by GridSearch. The weakest performer among all the classifiers for both selected and full feature sets was LightGBM.

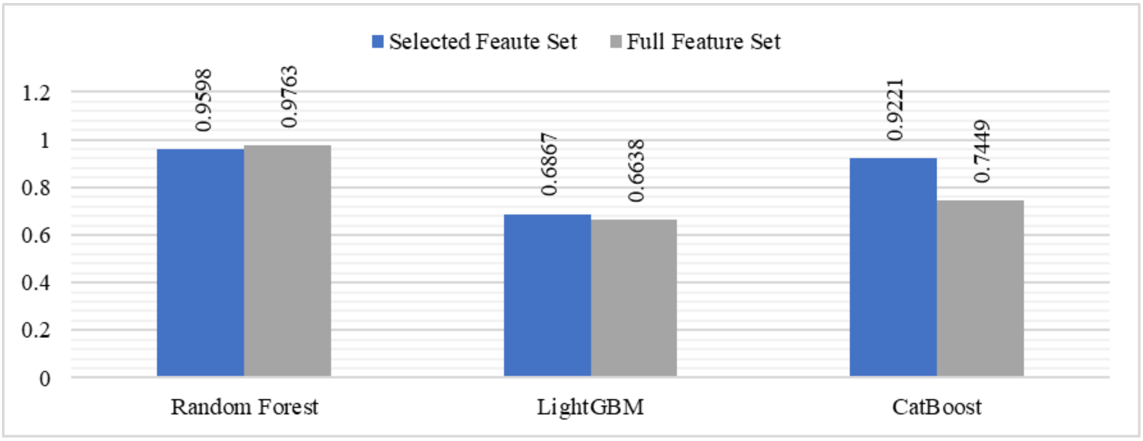


Figure 12. F1-score on both selected features set and full feature set.

6.4.5 Comparison of AUC-Score

Area Under the Curve (AUC) quantifies a classifier's capacity to differentiate between target variables. The better the model distinguishes between positive and negative classifications, the greater the Area Under the Curve score. Fig. 13 shows that Random Forest achieved the highest 99% AUC score for the selected feature set compared to the other two classifiers. When comparing selected and full features on Random Forest, the selected features only experienced a minimal AUC score deterioration of only 0.06%. CatBoost achieved quite a better result (approximately 12.4% better) for the selected feature compared to the full feature set. This improvement is mostly due to optimal hyperparameters searched for by GridSearch. The weakest performer among all the classifiers for both selected and full feature sets was LightGBM.

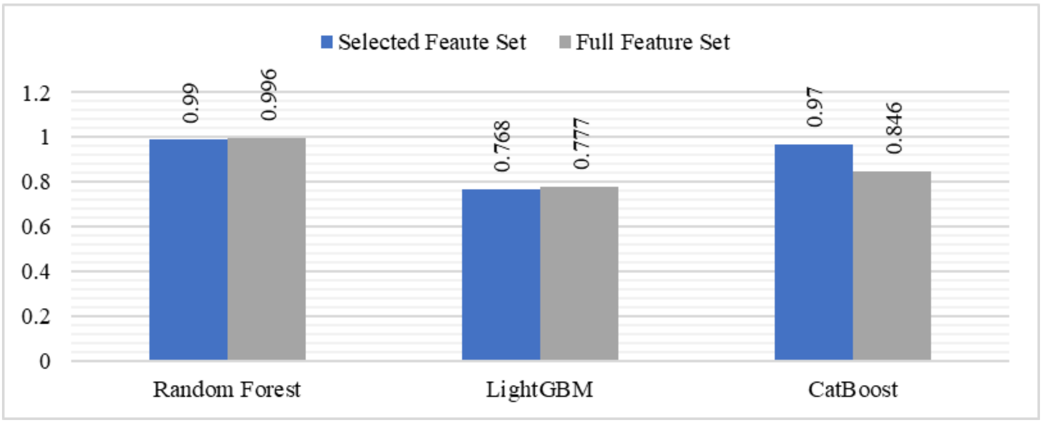


Figure 13. AUC Score on both selected features set and full feature set.

6.5 Summary of Comparative Analysis

After comparing different metrics and their corresponding results derived from classification algorithms, it has been learned that Random Forest outperformed the other two classification algorithms in every classification metric. The performance of the CatBoost classifier increased significantly due to the use of GridSearch for tuning its hyperparameters. CatBoost's performance increased by 12-18% in some metrics. When comparing selected and full features on Random Forest, the selected features only experienced an overall accuracy deterioration of less than 1.7% while achieving a massive 84.5% reduction in feature space. The weakest classifier was LightGBM. Therefore, the Random Forest classifier with Hybrid Ensemble Feature Selection technique is recommended for the keylogger detection task.

7. Conclusion and Future Work

The usage of keyloggers, which deceive users into doing activities that give attackers a chance to victimize the user and steal desired information from him, is growing daily. The attacker gains access to sensitive information by breaking into the organization's systems. In cyber security attacks, keyloggers become vital. This study assessed a hybrid ensemble feature selection technique with machine learning algorithms with a publicly available keylogger detection dataset. The dataset was processed and cleaned using Python in the Google Colab platform. Machine learning classifiers such as Random Forest, LightGBM, and CatBoost were trained and tested with the cross-validated data. Random Forest was found to be the best in the performance. CatBoost was found to be moderately good, CatBoost's performance increased by 12-18% in some metrics, but LightGBM performed worst. Random Forest experienced a minimal accuracy deterioration of 1.59% while achieving a massive 84.5% reduction in feature space.

Exploration in these fields is necessary for future developments to improve the efficiency of keylogger detection in real time using neural networks and deep learning developed on multiple datasets. A study of adaptive feature selection by machine learning techniques is necessary since preset features may not always function as intended, and attackers may use exploits to evade them. Additionally, research is needed for continual learning so that the model can continuously learn, evolve, and classify keyloggers based on their divergent actual characteristics. Depending on the situation and surroundings, the number of features should be changed or added. To perform keylogger detection using machine learning and a hybrid ensemble-based feature selection approach, this work contributes to cybersecurity.

References

[1] Javaheri D; Hosseinzadeh M; Rahmani AM. (2018). Detection and elimination of spyware and ransomware by intercepting kernel-level system routines. *IEEE Access.* 78321-32.

[2] Oz H; Aris A; Levi A; Uluagac AS. (2022,). A survey on ransomware: Evolution, taxonomy, and defence solutions. *ACM Computing Surveys (CSUR).* 1-37.

[3] Chatterjee R; Doerfler P; Orgad H; Havron S; Palmer J; Freed D; Levy K; Dell N; McCoy D; Ristenpart T. (2018). The spyware used in intimate partner violence. *In 2018 IEEE Symposium on Security and Privacy (SP).* pp. 441-458.

[4] Zou D; Zhao J; Li W; Wu Y; Qiang W; Jin H; Wu Y; Yang Y. (2018). A multigranularity forensics and analysis method on privacy leakage in cloud environment. *IEEE Internet of Things Journal*. 1484-94.

[5] Thakur KK; Nair NR; Sharma M. (2022). Keylogger: A Boon or a Bane. *Trinity Journal of Management; IT & Media (TJMITM).* 145-53.

[6] Rahim R; Nurdiyanto H; Abdullah D; Hartama D; Napitupulu D. (2018). Keylogger application to monitoring users activity with exact string matching algorithm. In Journal of Physics: Conference Series. Vol. 954, No. 1, p. 012008.

[7] Monaco JV. (2018). Sok: Keylogging side channels. *In 2018 IEEE Symposium on Security and Privacy (SP).* pp. 211-228.

[8] Singh A; Choudhary P. (2021). Keylogger detection and prevention. *In Journal of Physics: Conference Series*, Vol. 2007, No. 1, p. 012005.

[9] Barankova II; Mikhailova UV; Lukyanov GI. (2020). Software development and hardware means of hidden usb-keylogger devices identification. *InJournal of Physics: Conference Series.* Vol. 1441, No. 1, p. 012032.

[10] Kumar A; Dubey KK; Gupta H; Memoria M; Joshi K. (2022). Keylogger Awareness and Use in Cyber Forensics. *InRising Threats in Expert Applications and Solutions*. pp. 719-725.

[11] Bhardwaj A; Goundar S. (2020). Keyloggers: silent cyber security weapons. *Network Security*. 14-9.

[12] Sbai H; Goldsmith M; Meftali S; Happa J. (2018). A survey of keylogger and screenlogger attacks in the banking sector and countermeasures to them. *International Symposium on Cyberspace Safety and Security*. pp. 18-32.

[13] Ahmed YA; Maarof MA; Hassan FM; Abshir MM. (2014). Survey of Keylogger technologies. *International journal of computer science and telecommunications*. 5(2).

[14] Pillai D; Siddavatam I. (2019). A modified framework to detect keyloggers using machine learning algorithm. *International Journal of Information Technology*. 707-12.

[15] Solairaj A; Prabanand SC; Mathalairaj J; Prathap C; Vignesh LS. (2016). Keyloggers software detection tech-niques. *In 2016 10th International Conference on Intelligent Systems and Control (ISCO).* pp. 1-6.

[16] Brown J; Anwar M; Dozier G. (2017). An artificial immunity approach to malware detection in a mobile platform. *EURASIP Journal on Information Security*. 1-0.

[17] Wen L; Yu H. (2017). An Android malware detection system based on machine learning. *AIP conference proceedings*. (Vol. 1864, No. 1, p. 020136).

[18] Hatcher WG; Maloney D; Yu W. (2016). Machine learning-based mobile threat monitoring and detection. *In 2016 IEEE 14th International Conference on Software Engineering Research; Management and Applications (SERA).* pp. 67-73.

[19] Qabalin MK; Naser M; Alkasassbeh M. (2022). Android Spyware Detection Using Machine Learning: A Novel Dataset. *Sensors.* 22(15):5765.

[20] Usman N; Usman S; Khan F; Jan MA; Sajid A; Alazab M; Watters P. (2021). Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. *Future Generation Computer Systems*. 118:124-41.

[21] Rathore H; Agarwal S; Sahay SK; Sewak M. (2018). Malware detection using machine learning and deep learning. *International Conference on Big Data Analytics*. pp. 402-411.

[22] Ma Z; Ge H; Liu Y; Zhao M; Ma J. (2019). A combination method for Android malware detection based on control flow graphs and machine learning algorithms. *IEEE access*. 21235-45.

[23] Aafer Y; Du W; Yin H. (2013). Droidapiminer: Mining API-level features for robust malware detection in Android. *International conference on security and privacy in communication systems*. pp. 86-103.

[24] Chiew KL; Tan CL; Wong K; Yong KS; Tiong WK. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences*. 153-66.

[25] Dorogush AV; Ershov V; Gulin A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

[26] Fan J; Wang X; Zhang F; Ma X; Wu L. (2020). Predicting daily diffuse horizontal solar radiation in various climatic regions of China using support vector machine and tree-based soft computing models with local and extrinsic climatic data. Journal of Cleaner Production. 248, 119264.

[27] Breiman L. (2001) Random forests. Machine learning., 5-32.

[28] Keylogger Detection. (2021, September 17). Kaggle. Retrieved October 12, 2022, from https://www.kaggle.com/datasets/subhajournal/keylogger-detection