Received: May 14, 2023. Revised: June 29, 2023.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybridized Least Absolute Shrinkage Selection Operator and Cross-Validation Algorithm for Classification of Malware

Chandini Shivaramu Banumathi[1]* **Ajjipura Basavegowda Rajendra[1]**

[1]*Department of Information Science, Vidyavardhaka College of Engineering, Mysuru, India*
* Corresponding author's Email: tchannie@gmail.com

Abstract: In recent years, the growth of malicious software is becoming high and led to a huge number of security threats. The malware exploits the system information and uses the important information of the user without any intimation. Moreover, the malware furtively directs that information to the servers which are organized by the attackers. In recent years, researchers and scientists discovered anti-malware products to identify known malware. But, these methods are not robust to detect obfuscated and packed malware. The fore mentioned issues rely on the existing approaches can be rectified using the hybridized least absolute shrinkage selection operator and cross validation (LASSO-CV) algorithm are proposed. The LASSO is used in the process of selecting the stochastic features and CV visualize the feature using weighted co-efficient approach. The effective features are selected by the LASSO-CV algorithm which eases the process of classifying the malware. This research also utilized long-short term memory (LSTM)-softsign classifier to classify the malware. The malware samples are collected from the VXHeavens and android malware dataset (AMD) dataset which consists of malware samples from various software. The classification results obtained from the LSTM-softsign classifier have better classification accuracy of 97.85% for VX heavens dataset and 99.24% for AMD.

Keywords: Least absolute shrinkage selection operator, Cross validation, Long-short term memory, Malware classification, Softsign function, VXHeavens.

1. Introduction

Malicious software is generally termed as malware which affects the computer systems and conciliates with the security of the user. A code or application is noted as malware when it performs against the path of computer users and takes part in malware activities [1]. In other words, malware is known as software that manipulates the sensitive data or confidential information of internet users [2]. Malware is an ideal platform for attackers or hackers who utilize harmful software and attacking strategies to corrupt the user's security and gives threats [3]. There is an increase in the count of various applications and android devices in recent years and this leads to an increase in the number of users. It is considered a revolution in technology, but at the same time, it leads to increased security issues and privacy concerns for users [4, 5]. These issues are due to the exchange of data, interaction among social networks, digital transactions, etc [6]. The Malware creators employed various methodologies to create fresh constraints of malware consisting of instruction variation, variation in the codes of software, and inserting non-operative conditions.

Malware analysis is a method that is based on detecting [7] and understanding the behavior of the malware. Generally, the analysis of malware takes place by extraction of opcodes, APIs, system calls and traces of the network [8] The malware is detected based on two approaches such as signature-based and heuristic based detection. The detection method based on a signature based approach does not produce false positives since it is designed by a malware analyst. However, the signature based approach can detect only the known malware and leave the users in unsecured conditions. So, in recent days most of the research is based on heuristic-based approaches [9,

10]. The accuracy in detecting the malware is based on identifying the corrupted files and separating them from the malware group [11, 12]. Research using machine learning techniques in the detection of malware is increasing rapidly. Machine learning techniques are utilized in the analysis and selection of features to provide better classification performance [13-15]. To overcome the mentioned problems, the deep learning approach is utilized in this research.

The major contributions of this research are listed as follows:

1. This research proposed a hybrid LASSO-CV algorithm for feature selection. The features selected using the hybrid LASSO-CV make the classification process easier by providing better accuracy and classifying the malware from the software applications.
2. The LSTM-Softsign classifier utilized in this research maps with a large probability value and obtains better classification performance.

The remaining paper is organized as follows, section 2 represents the related works of the paper. The proposed method is discussed in section 3. The results and discussion are provided in section 4. Finally, section 5 represents the overall summary of the paper.

2. Related works

Ashik [16] have introduced a methodology to detect malicious files utilizing various datasets which have samples related to real-time malware. The methodology was based on machine learning and deep learning techniques. The machine learning modules were trained based on the selected features obtained from maximum relevance and variance analysis. The deep learning modules are utilized in the classification of unknown samples. The performance was evaluated based on the results obtained from API and system calls. The features are selected from the factorial criterion of Analysis of Variance (ANOVA). The proposed methodology utilized an obfuscation technique to detect malicious files and classifies the unknown samples effectively. However, the proposed methodology is vulnerable to attacks based on cyber security.

Moti [17] have presented a framework for generative adversarial network (GAN) to detect the malware samples present in the IoT layer. The high level features were selected using convolutional neural network (CNN) and GAN was utilized for creating new samples of malware. The attention-based model is a combination of CNN and long short term memory (LSTM) which was utilized to improve attention to extract the features. The proposed framework was effective and efficient to detect the malware present in the opcode or other data. However, the training of the proposed framework with limited data leads it to remain recessive for more data.

Li [18] introduced a framework based on incremental malware classification (IMC) that was comprised of opcode sequence extractor to extract the opcodes from the samples of malware and then transform them into n-gram sequences. The features were selected using the opcode sequence selector to select the representative sequence and the classification is performed using the IMC-support vector machine (IMC-SVM) classifier. The SVM classifies the malware and updates it to the IMC. The input data were obtained from the VX Heavens dataset where the 600 samples of malware are categorized as 500 for training and 100 samples for testing. The IMC can learn new class knowledge without disremembering the old class knowledge. However, the performance of the IMC-SVM classifier was poor when the samples became less than 1000.

Talal and Zagrouba [19] have introduced a robust malware anomaly detection system (RoMADS) which was based on the deep learning technique that can detect renewable malware. The deep learning method was utilized to progress the efficiency of MADS in the IoT environment. The RoMADS consist of the architecture of fog computation and the architecture of IoT. The architecture of fog computation and IoT process the sensors and gateways before storing them in the cloud which is considered the time reduction process. However, regular updates must be provided in the RoMADS system to ensure security.

Kim [20] have introduced a static automated method to classify malicious code using the machine learning method. The classification system was based on a static automated method that categorizes the malware by using the portable executable (PE) structure. The pre-processing was performed to extract the hash values and PE data to proceed with image conversion. The image was created using the image create module based on CNN and converted to the image data. When the obtained code possessed similar results, the classification was repeated. If again the code possessed similar results, it is considered malicious code. The automated classification system can detect the response and transmission of real-time data. However, the automated method only classifies the malicious codes and not their types.

XianWei Gao [21] have introduced a MaliCage

framework to categorize the malware using deep neural network (DNN) and generative adversarial network (GAN). The MaliCage is comprised with three modules such as packer detector, malware classifier and a packer. After distinguishing the packed samples, the sandbox is used to extract the features. Finally, GAN was used to create fake malware samples which improvise the classification process performed using DNN. The packer detector used in the MaliCage framework opcode the features effectively which helps in the process of classifying the malwares. However, the suggested approach faced issues while detecting the family of packed malwares.

Xusheng Wang [22] have introduced a stacked ensemble learning framework refereed as MFDroid for detection of android malware. The feature selection was performed using seven algorithms to select the API calls and opcodes. Moreover, logical regression was used as a meta classifier. At last, each and every individual features were used to detect the difference among the malicious and benign applications. However, the usage of single learning algorithm in the suggested approach was affected by data skew and computational burden.

Hairen Gui [23] have introduced aligned assembly pre-training function embedding (AAPFE) model for analysis of malware. The suggested approach effectively performs data augmentation and the triplet network structure is embedded to the trained model. Every network mines the instruction sequence with the help of self-attention mechanism and graph convolutional neural network. However, the issues related to binary obfuscation occurred while extracting block level information.

3. Hybridized least absolute shrinkage selection operator and cross validation (Lasso-CV) algorithm for feature selection

The classification of malware in software codes and software applications is not an easy task in the interpretation of malware classification. This research proposed a hybrid of the LASSO-CV algorithm which is utilized in feature selection. The features selected using the hybrid LASSO-CV make the classification process easier by providing better accuracy and classifying the malware from the software applications. The overall process involved in the classification process is diagrammatically represented in Fig. 1 as follows:

3.1 Dataset

This research collected malware samples from the dataset known as VXHeavens [24] which consists of 23000 sample files of which 932 are malware samples and the benign software samples are gathered from Windows XP system directory and 346 PE format EXE files such as multimedia software, graphics software, etc. The malware samples from benign software are identified from 265 anti-virus software applications. The VXHeavens website has 586 samples which include viruses, worms, Trojans etc. Secondly, android malware dataset (AMD) [25] with 40000 applications are taken into consideration to evaluate the efficiency of the proposed approach. Among 4000, 2000 samples are selected from Derbin project and 2000 legitimate apps were downloaded from Google Play store.

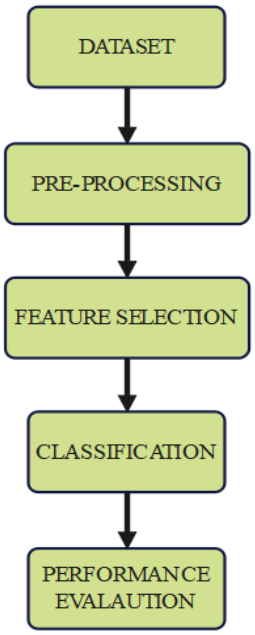

Figure. 1 Process involved in the classification of malware

3.2 Pre-processing

After collecting the data from the datasets, pre-processing is necessary to be performed. Since the malware files consist of redundant data and unnecessary files they should be removed from them. The malware files obtained from the VXHeavens dataset are pre-processed to select the required features from the dataset. Here, pre-processing is performed to obtain the selected features from the dataset.

3.3 Feature selection using hybrid least absolute shrinkage selection operator and cross validation (LASSO-CV) algorithm

The output from the pre-processing is provided as

the input for the feature selection process using hybridized LASSO-CV algorithm. Feature selection is an essential process that ease the process of classification and provides better results. Feature selection is described as the selection of variables or attributes where the subsets of appropriate features are selected to construct the model. Initially, the feature selection process is utilized in the field of statistical learning and it is now applicable in various fields such as signal processing, classification of text, image classification, etc. The previous research utilized stepwise regression and CV to overcome the Malware problems, but the outcomes of these methodologies don't provide enough classification results. To overcome these issues, the hybrid of the LASSO-CV algorithm is introduced which selects the stochastic features and provides better classification accuracy than the existing methodologies. Based on each statistic, a specific feature is visualized and weighted by co-efficient $a = (a_1, \dots, a_r)^T$. The basis matrix is represented as $\varphi_{N\times P} = (\varphi^1, \dots, \varphi^P)$ where the sample collection is represented as N which has the feature P. The specific feature is represented in a column and the realization is represented in rows. The realization of the output field is denoted as $h = (h_1, \dots, h_N)^T$ and the prediction of the model is denoted as $\hat{h} = \varphi_a = (\hat{h}_1, \dots, \hat{h}_N)^T$.

The LASSO works based on Least Angle Regression (LAR) to remove the unnecessary features using a constraint co-efficient which is represented in eq. (1) as follows:

$$a^{LASSO} = arg\,min\{\|h - \hat{h}_2\|\}_2 \text{ , s.t. } \|a\|_1 \le t \quad (1)$$

Where the penalty term of the L1 norm is denoted as $\|a\|_1$ whose value is $\sum_{j=1}^{P}|a_j| \le t$.

The corresponding Langrangian formulation of Eq. (1) is denoted in Eq. (2) as follows:

$$a^{LASSO} = arg\,min\left\{\|h - \hat{h}_2\|_2 + \lambda\|a\|_1\right\} \quad (2)$$

Where the Langrangian factor is represented as $\lambda \in [0, +\infty)$, a one-to-one mapping is performed among t and.

LASSO combined with LAR can overcome the problem related to complex computations by using the L1 norm. So, it is effective to be used in feature selection due to its shrinkage methodology. However, LASSO is expensive and lacks in performance when it comes to large features. To overcome these issues, this research utilized LAR to perform with LASSO. Since LAR is a heuristic search algorithm that aids better feature selection.

The hybrid of LASSO-LAR and CV is efficient and deals better with high dimensional issues when $N \ll P$. The steps involved in LASSO-CV for an efficient feature selection are represented as follows:

(i) During the initialization process, each feature present in the vector column is normalized and target values are centralized based on Eq. (3) as follows:

$$\frac{1}{N}\sum_{i=1}^{N}\varphi_{ij} = 0,\ \frac{1}{N}\sum_{i=1}^{N}\varphi_{ij}^2 = 1,\ j = 1, \dots, P \text{ and } \frac{1}{N}\sum_{i=1}^{N}h_i = 0, \quad (3)$$

Initialization takes place with residual $r = h$ and the value of $a_{1,\dots,}a_P = 0$

(ii) The feature vector φ_j which is linked with r is identified.
(iii) The co-efficient a_j is transferred as least-square co-efficient when another competitor φ_k is correlated with the current residual φ_j .
(iv) The subset $\{a_j, a_k\}$ moves in a defined direction of least square co-efficient until another subset occurs in the direction of current residuals.
(v) when the non-zero co-efficient reaches 0, the features are dropped and computation occurs in the direction of the least square value.
(vi) This feature is continued till the optimal feature subsets are selected.

The algorithm is known as the least angle because utilized the smallest and equal angle with an active subset of the feature $A_{k.}$ With step 5 being the primary change, the LASSO-CV algorithm is relatively similar to the LAR algorithm. Even if the chosen characteristics were incorrectly identified in earlier phases, they are not dropped in LAR. The features in LASSO-LAR are evaluated based on the behavior of the coefficients. When the active feature subset A_k is determined, the hybrid of LASSO-LAR is obtained which is represented in Eq. (4) as follows:

$$a_{A_k}^{hyb} = arg\,min\left\{\|h - \varphi_{A_k}a\|_2\right\} = (\varphi_{A_k}^T\varphi_{A_k})^{-1}\varphi_{A_k}^T h \quad (4)$$

The LASSO-LAR effectively offers an ideal path to directly deal with the optimization problems and it has an active feature subset that is constructed throughout each iteration of the algorithm. Because LASSO-LAR only supplies a few ratings to the feature subsets, ratings will be established using

specific criteria, and the best subset will then be used for model creation.

3.3.1.Cross validation

The LAR utilized in the previous section helps in providing better results with LASSO but it is not capable to rectify the issues regarding the selection of constraints. To overcome this issue, this research utilized a cross-validation approach using random sub-sampling replication to select the best feature value. The replication method was used in cross-validation to overcome the problem related to uncertainty and predictive ability in LASSO. At each replication, the reference features are categorized into two sub–features as training sample and the validation sample. The training sample estimates SNP using LASSO-LAR and the second is based on a validation sample which is used to validate the extracted features.

Cross-validation takes place through three various approaches to classify the data into training and validation sets.

(i) In the first case, the individual features are allocated using a random splitting method where the features are assigned to train the sample or cross-validate the sample.
(ii) In the second case, the individual features are allotted to validation samples or training samples by segregating them within the overall features which take part in the process of cross-validation.
(iii) In the final case, the validated sample is split among the features which take part in the previous iteration. i.e. the validated sample takes place in the overall feature or is present in the training sample.

Lastly, cross-validation takes place in two categories such as training and validation. The validation occurs across the allotted features where each feature is trained and validated randomly. On other hand, the validation takes place within the feature set to train and validate the best features. Thus two cross-validation techniques are utilized to predict and select the relevant features which eases the process of classification.

3.4 Classification using LSTM soft sign activation function

The final stage is the classification of the selected features to classify the malware. This research utilizes a long short-term memory network (LSTM) with a soft sign activation function. LSTM is a special type of recurrent neural network (RNN), which can learn long-term dependence. The structure of LSTM is represented in Fig. 2.

Cell state is the central component of LSTM, which can add or remove information from cells and selectively permit information to pass through the door mechanism to accomplish this. The forget gate, input gate, and output gate make up an LSTM. The input gate chooses what information to add to the cell state after the forget gate has decided which information to remove from the cell state. The cell state can be updated once these two points have been established. The output gate, in the end, determines the network's ultimate output. The process of the node present in LSTM is described in Eqs. (5-10) as follows:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{5}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{6}$$

$$\tilde{C}_t = softsign(W_C.[h_{t-1}, x_t] + b_C) \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

$$o_t = \sigma(W_0.[h_{t-1}, x_t] + b_0) \tag{9}$$

$$h_t = o_t * soft\ sign(C_t) \tag{10}$$

Where, the hidden state of the prior layer is denoted as h_{t-1}, input for the current layer is denoted as x_t. The weight and biased state are denoted as W and b respectively. The sigmoid function is denoted as σ and the output of the forget gate is denoted as f_t. The output from the input gate is represented as i_t and the intermediate temporary state is denoted as $\tilde{C}_t$. The state of the cell present in the prior layer id denoted as C_{t-1} and the state of the cell present in the next layer is denoted as C_t. The output from the output gate and the hidden state of the succeeding layer is denoted as o_t and h_t respectively.

Computing the output of the input and output gate individually doesn't provide better performance so, the output from the input and output gate can be distinguished using the factor $1 - f_t$. This helps to improve the cell state for the next layer in the input and output gate, it is represented in Eq. (11) as follows:

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t \tag{11}$$

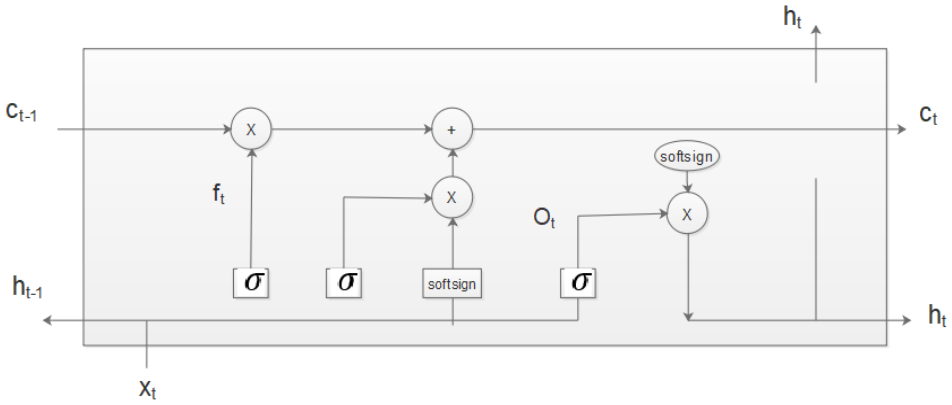

Figure. 2 Structure of LSTM

3.4.1. Softsign activation function

Softsign is a type of activation function that can be utilized in LSTM. The features obtained using hybrid LASSO-LAR are combined with the LSTM and the features are sent to the Softsign classifier in a fully connected manner to perform the classification of malware. Softsign activation function maps with a large probability value and obtain better classification performance. The Softsign is a quadratic polynomial which is represented using the Eq. (12) as follows:

$$f(x) = \left(\frac{x}{|x|+1}\right) \tag{12}$$

Where the absolute input value is denoted as $|x|$.

The major variation in tanh and Softsign functions is the rate of convergence, in tanh it takes place in the exponential form and in Softsign it takes place in the form of polynomial.

4. Results and analysis

This section provides the results and analysis of this research. The result portion is classified into performance analysis and comparative analysis which are represented in the following sections. Moreover, the performance of the proposed approach is evaluated by considering the performance metrics such as accuracy, sensitivity, recall, F-1 score and Area Under Curve (AUC).

Accuracy: It is calculated by dividing correct number of predictions to the total number of predicted values. The formula to evaluate accuracy is represented in Eq. (13).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{13}$$

Sensitivity: It is defined as the proportion of total number of positive cases which is predicted as positive and it is evaluated using the Eq. (14).

$$Sensitivity = \frac{TP}{TP+FN} \tag{14}$$

Recall: It is defined as the total number of actual positive labels identified by the model.

F-1 score: It is evaluated by measuring the harmonic mean of precision and recall, it is evaluated using the Eq. (15) as follows:

$$F1\ score = 2 \times \frac{Precision \times recall}{precision+recall} \tag{15}$$

Where TP is known as true positives, TN represents the true negatives, FP represents the false positives and FN represents false negatives.

4.1 Performance analysis

The performance of LSTM-Softsign with the existing classifiers is evaluated in this section. The analysis is performed and the results are evaluated with and without the feature selection for two datasets such as VX Heavens and AMD. The performance of existing classifiers such as Iterative Dichotomiser 3 (ID3), random forest (RF), Adaboost, Bagging and LSTM. The performance of the classifier without hybrid LASSO-LAR (Feature selector) is represented in Table 1 for VX Heavens and AMD dataset. The performance is evaluated based on accuracy, sensitivity, recall, F-1 score, and Area Under Curve (AUC).

Results from table 1 show that the LSTM-Softsign classifier attained lesser value in classification accuracy (90.81%) when compared with the existing classifiers such as LSTM (90.85%) and IDF (94.61%) for VX Heavens dataset. The absence of LASSO-LAR leads to reduce the performance of the LSTM-Softsign classifier. Without LASSO-LAR the redundant features are directly involved in the process of classification and affect the performance of the classifier. For, AMD dataset, the LSTM-soft sign classifier has achieved

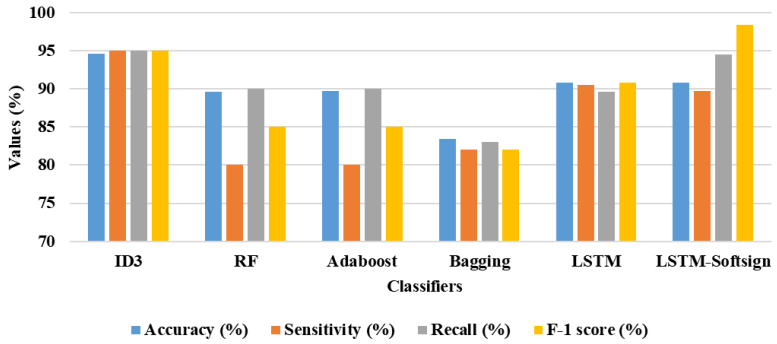

Figure. 3 Graphical representations for the performance of classifiers without LASSO-LAR for VX Heavens dataset

Table 1. Performance of the classifier without feature selection for VX Heavens and AMD datasets

Dataset	Classifier	Accuracy (%)	Sensitivity (%)	Recall (%)	F-1 score (%)	AUC
VX Heavens	ID3	94.61	95.00	95.00	95.00	84.36
	RF	89.65	80.00	90.00	85.00	49.97
	Adaboost	89.71	80.00	90.00	85.00	49.96
	Bagging	83.46	82.00	83.00	82.00	49.96
	LSTM	90.85	90.49	89.63	90.81	87.76
	LSTM-Softsign	90.81	89.70	94.54	98.40	49.97
AMD	ID3	75.50	83.00	75.00	76.00	79.77
	RF	81.44	78.00	78.00	78.00	74.27
	Adaboost	79.28	78.00	78.00	78.00	72.17
	Bagging	84.70	86.00	85.00	84.00	80.14
	LSTM	97.17	97.23	94.80	96.03	96.65
	LSTM-Softsign	99.00	98.61	98.79	98.79	98.68

Table 2. Performance of the classifier with feature selection for VX Heavens and AMD datasets

Dataset	Classifier	Accuracy (%)	Sensitivity(%)	Recall (%)	F-1 score (%)	AUC
VX Heavens	ID3	96.32	96.00	96.00	96.00	84.65
	RF	89.71	80.00	90.00	85.00	50.00
	Adaboost	96.32	96.00	96.00	96.00	84.65
	Bagging	93.87	94.00	94.00	94.00	87.18
	LSTM	90.85	90.49	89.63	90.81	87.76
	LSTM-Softsign	97.85	97.61	99.46	98.53	89.11
AMD	ID3	93.51	94.00	94.00	94.00	93.09
	RF	94.57	95.00	95.00	95.00	93.81
	Adaboost	89.06	89.00	89.00	89.00	88.50
	Bagging	86.17	87.00	86.00	85.00	81.63
	LSTM	99.14	98.61	98.97	98.79	96.84
	LSTM-Softsign	99.24	99.06	98.98	98.93	98.87

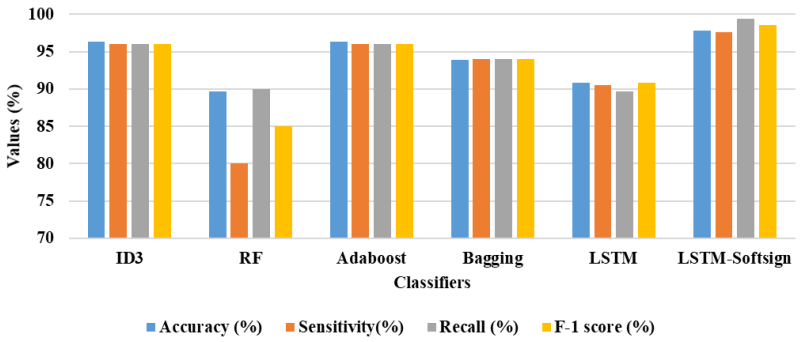

Figure. 4 Graphical representations for the performance of classifiers with LASSO-LAR for VX Heavens dataset

Table 3. Different feature selection approaches

Dataset	Feature selection	Accuracy (%)	Sensitivity (%)	Recall (%)	F1 score (%)	AUC
VX Heavens	Chi-Square	90.17	89.63	99.07	94.36	50.00
	Recursive Feature Elimination	90.17	89.63	99.07	94.36	49.96
	Variance Threshold	90.19	89.89	99.36	94.51	49.95
	Linear Regression	90.81	89.70	99.40	94.54	50.0
	LASSO-LAR	97.85	97.61	99.46	98.53	89.11
AMD	Chi-Square	98.00	98.61	98.97	98.79	98.05
	Recursive Feature Elimination	97.03	97.12	94.44	97.19	96.64
	Variance Threshold	99.07	98.70	98.70	98.70	98.82
	Linear Regression	97.17	97.23	94.80	96.03	96.65
	LASSO-LAR	99.24	99.06	98.79	98.93	98.87

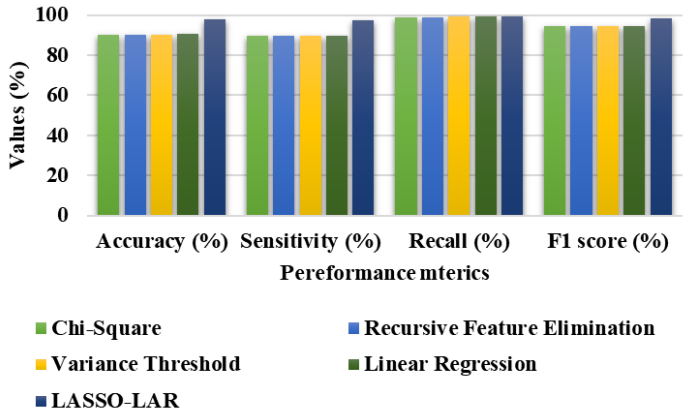

Figure. 5 Performance of different feature selection methods for VX Heavens dataset

better classification accuracy of 99.0% which is comparatively higher than the existing classifiers presented in Table 1. The Fig. 3 depicted below graphical representations for the performance of classifiers without LASSO-LAR for VX Heavens dataset.

The overall results from Table 2 show that the performance of the LSTM-Softsign function is higher than other classifiers by obtaining better metrics in accuracy, sensitivity, recall and F-1 score. The combination of LSTM with soft sign function mapped the large probability values to obtain better classification performance. Moreover, the Softsign classifier is designed in a fully connected way which effectively classifies the malware without any loopholes. The Fig. 4 depicted below graphical representations for the performance of classifiers with LASSO-LAR for AMD dataset.

The performance of the LASSO-LAR is compared with the existing feature selection methods such as Chi-square, recursive feature elimination, variance threshold and linear regression is represented in Table 3 and the Fig. 5 shows the graphical representation for various feature selection approach for VX Heavens dataset.

Results from Table 3 shows that the hybrid LASSO-LAR achieved better performance when compared with different feature selection methodologies. LASSO-LAR achieved better accuracy of 97.85% for VX Heavens dataset and 99.24% for LASSO-LAR dataset. These obtained results are comparatively higher than the existing feature selection methods. The LASSO-LAR achieved better results because it provides ratings to the feature subsets, ratings will be established using specific criteria, and the best subset will then be used for the creation of the classification model.

4.2 Comparative analysis

The comparative analysis of the proposed LSTM-Softsign classifier is performed with two datasets obtained from VX Heavens and Android Malware dataset. The existing approaches such as MalGan (with CNN features), MFDroid, and AAPFE are used to evaluate the performance of the proposed method. The comparative table of the proposed LSTM-Softsign with the existing classifiers are represented in Table 4.

Table 4. Comparative analysis of the proposed method for different classifiers

Methodologies	Dataset	Accuracy (%)	Sensitivity(%)	Recall (%)	F-1 score (%)
MalGan (with CNN features) [17]	VX Heavens	97.5	96.5	92.81	97.61
MFDroid [22]	AMD	96.21	92.43	93.78	95.34
AAPFE [23]	AMD	94.86	95.21	95.32	93.33
LSTM-Softsign	VX Heavens	97.85	97.61	99.46	98.53
	AMD	99.24	99.06	98.98	98.93

From the above comparative table and it is shown that the proposed LSTM-Softsign classifier achieved better performance when compared with the existing classifiers such as MalGan (with CNN features), MFDroid and AAPFE. The selected features from hybrid LASSO-LAR are classified using the LSTM-Softsign classifier. The LSTM- Softsign classifier provides better classification accuracy compared with other classifiers. This is due to the Softsign layer and the performance of the LSTM-Softsign function is higher than other classifiers by obtaining better accuracy, sensitivity, Recall and F1 score. The combination of LSTM with soft sign function mapped the large probability values to obtain better classification performance. The LSTM-Softsign classifier attained better classification accuracy of 97.85% than MalGan (97.5%) for VX Heavens dataset. For AMD dataset, the proposed approach achieved classification accuracy of 99.24% which is comparatively higher than the MFDroid and AAPFE with 96.21% and 94.86% respectively.

5. Conclusion

The development of the internet and software in the modern world leads to increased threats and a lack of security due to malicious software (malware). Malware corrupts and exploits user information which may lead to a ransomware attack or other security threats. To overcome these issues, the LASSO-LAR algorithm is utilized in which the effective features are selected and ease the process of classifying the malware. This research utilized the LSTM-Softsign classifier for classifying the malware and the VXHeavens, AMD datasets are utilized to obtain the malware samples. The LSTM-Softsign classifier attained better classification accuracy of 97.85% than MalGan (97.5%) for VX Heavens dataset. For AMD dataset, the proposed approach achieved classification accuracy of 99.24% which is comparatively higher than the MFDroid and AAPFE with 96.21% and 94.86% respectively. In future, the proposed method can be improvised to handle the inability of temporal dependencies such as computational time.

Nomenclature

Parameter	Description
$\|\|a\|\|_1$	Penalty term of L1 norm
N	Collection of sample
λ	Langrangian factor
h	The realization of output field
φ_j	Feature vector
h_{t-1}	Hidden state of prior layer in LSTM
A_k	Active subset of the feature
x_t	Input for the current layer
σ	Sigmoid function
f_t	Output of the forget gate of LSTM
$\tilde{C}_t$	Intermediate temporary state of LSTM
h_t	Hidden state of the succeeding layer
$\|x\|$	Absolute input value of Softsign function

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st. The supervision and project administration, have been done by 2nd author.

References

[1] L. N. Vu and S. Jung, "AdMat: A CNN-on-matrix approach to Android malware detection and classification", *IEEE Access*, Vol. 9, pp. 39680-39694, 2021.

[2] F. O. Catak, A. F. Yazı, O. Elezaj, and J. Ahmed, "Deep learning based Sequential model for malware analysis using Windows exe API Calls", *PeerJ Computer Science*, Vol. 6, p. e285, 2020.

[3] S. Riaz, S. Latif, S. M. Usman, S. S. Ullah, A. D. Algarni, A. Yasin, A. Anwar, H. Elmannai, and S. Hussain, "Malware Detection in Internet of

Things (IoT) Devices Using Deep Learning", *Sensors*, Vol. 22, No. 23, p. 9305, 2022.

[4] H. Ghanei, F. Manavi, and A. Hamzeh, "A novel method for malware detection based on hardware events using deep neural networks", *Journal of Computer Virology and Hacking Techniques*, Vol. 17, No. 4, p. 319-331, 2021.

[5] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning", *Ad Hoc Networks*, Vol. 101, p. 102098, 2020.

[6] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices", *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 1563-1578, 2021.

[7] S. I. Imtiaz, S. U. Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network", *Future Generation Computer Systems*, Vol. 115, pp. 844-856, 2021.

[8] Q. Li, J. Mi, W. Li, J. Wang, and M. Cheng, "CNN-based malware variants detection method for Internet of Things", *IEEE Internet of Things Journal*, Vol. 8, No. 23, pp. 16946-16962, 2021.

[9] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning", *Journal of Signal Processing Systems*, Vol. 93, No. 2, pp. 265-273, 2021.

[10] Y. Lin, "A novel DL approach to PE malware detection: exploring Glove vectorization, MCC_RCNN and feature fusion", arXiv preprint arXiv:2101.08969, 2021.

[11] M. A. Kasassbeh, S. Mohammed, M. Alauthman, and A. Almomani, "Feature selection using a machine learning to classify a malware", *Handbook of Computer Networks and Cyber Security*, Springer, Cham, pp. 889–904, 2020.

[12] A. Alazab, A. Khraisat, M. Alazab, and S. Singh, "Detection of Obfuscated Malicious JavaScript Code", *Future Internet*, Vol. 14, No. 8, p. 217, 2022.

[13] N. Zhang, Y. Tan, C. Yang, and Y. Li, "Deep learning feature exploration for android malware detection", *Applied Soft Computing*, Vol. 102, p. 107069, 2021.

[14] H. Wang, H. Long, A. Wang, T. Liu, and H. Fu, "Deep Learning and Regularization Algorithms for Malicious Code Classification", *IEEE Access*, Vol. 9, pp. 91512-91523, 2021.

[15] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based android malware detection and variant classification", *Computational Intelligence*, Vol. 38, No. 5, pp. 1748-1771, 2022.

[16] M. Ashik, A. Jyothish, S. Anandaram, P. Vinod, F. Mercaldo, F. Martinelli, and A. Santone, "Detection of malicious software by analyzing distinct artifacts using machine learning and deep learning algorithms", *Electronics*, Vol. 10, No. 14, p. 1694, 2021.

[17] Z. Moti, S. Hashemi, H. Karimipour, A. Dehghantanha, A. N. Jahromi, L. Abdi, and F. Alavi, "Generative adversarial network to detect unseen internet of things malware", *Ad Hoc Networks*, Vol. 122, p. 102591, 2021.

[18] J. Li, D. Xue, W. Wu, and J. Wang, "Incremental learning for malware classification in small datasets", *Security and Communication Networks*, Vol. 2020, p. 6309243, 2020.

[19] H. Talal and R. Zagrouba, "A Robust Framework for MADS Based on DL Techniques on the IoT", *Electronics*, Vol. 10, No. 21, p. 2723, 2021.

[20] S. Kim, S. Yeom, H. Oh, D. Shin, and D. Shin, "Automatic malicious code classification system through static analysis using machine learning", *Symmetry*, Vol. 13, No. 1, p. 35, 2021.

[21] X. Gao, C. Hu, C. Shan, and W. Han, "MaliCage: A packed malware family classification framework based on DNN and GAN", *Journal of Information Security and Applications*, Vol. 68, p.103267, 2022.

[22] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, "MFDroid: A stacking ensemble learning framework for Android malware detection", *Sensors*, Vol. 22, No. 7, p.2597, 2022.

[23] H. Gui, K. Tang, Z. Shan, M. Qiao, C. Zhang, Y. Huang, and F. Liu, "AAPFE: Aligned Assembly Pre-Training Function Embedding for Malware Analysis", *Electronics*, Vol. 11, No. 6, p. 940, 2022.

[24] Link for VXHeavens dataset: http://vxheaven.org/

[25] Link for AMD dataset: https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning