An Efficient Machine Learning-Based Malware Prediction and Classification Architecture: A Cybersecurity Case Study

[1]B.Swathi,
[1]Assistant Professor, Vignan Institute of Technology and Sciences, Hyderabad
[2]G.Anitha
[2]Assistant Professor, CMR Institute of Technology, Hyderabad
[3]G.Himabindu
[3]AssistantProfessor, Geethanjali College of Engineering and Technology, Hyderabad

Abstract –
Malware is a threat to information security and poses a security threat to harm networks or computers. Not only the effects of malware can generate damage to systems, they can also destroy a country when for example, its defense system is affected by malware. This paper takes a look at different machine learning techniques that can be used to predict a system's probability of getting hit by various families of malware, based on different properties of that system. Given a dataset of these properties and the machine infections, the proposed solution is to use a fusion framework, namely LightGBM +MSVM, to build a model that predicts whether a system will soon be hit with malware.The evaluation is carried out through classification evaluation indexes such as accuracy, precision, recall, F1-score.

Index Terms: **LightGBM; feature selection, LightGBM; feature selection**
DOI Number: 10.48047/nq.2022.20.19.NQ99297 **NeuroQuantology2022;20(19): 3350-3358**

1. INTRODUCTION

With the widespread use of automatic generation tools, a large number of new variants of malicious code has been generated rapidly. According to the reports published by software security groups in 2019, there is 25% increment in the usage of destructive malware compared to last year, enterprise infections were up by 12%, and financial Trojans were up by 4 percent in comparison with the previous year [1]. Malware causes a lot of harm to users, such as stealing personal information and using too much battery or CPU. The majority of adversaries can be involved in targeted attacks: corporations, cybercriminals, hacktivists, online social hackers, nation states, cyber terrorists, cyber fighters, employees and script kiddies [2, 3].There are many types of malwares that are currently available on the Internet but worm, Trojan, backdoor, virus and botnet are the most common types of malwares to be considered as the many dangerous threats for Internet users. Therefore, the most malware studies were aimed to predict these types of malwares due to causing more harms and defects to the network and operating systems in comparison to other Malwares. Security researchers combat vulnerabilities in operating systems and computer applications by designingantivirus applications and anti-malware which are used to detect malware [4, 5].The prediction of malware attacks remains one of the most challenging problems for industry and academia. Traditional security solutions cannot keep pace with the ever-evolving threats that cause damage to critical systems, leading to loss of money, sensitive information, and reputation. The malware threat continues to grow along with the drastic rise in the number of victims due to the growing number of users in cyberspace, financial gains, seeking increased computational power for further attacks (botnets), availability of malware scripts, etc. Different machine learning methods have been proposed to address the problem of predicting malware attacks. Light Gradient Boosted Machine (LGBM) is the most popular classification technique currently used in detecting malware. Some of the benefits of LGBM are that it is easy to create, easy to understand, and reduces complexity [6]. In addition, with the increase of malware threats, a lot of big companies use AutoAI to help protect their systems. Automated Artificial Intelligence (AutoAI) is a variant of automated machine learning technology that automates the entire life cycle of machine learning [7]. Automation evaluates a number of tuning choices to obtain the best possible outcome then ranks model-candidates.

However, the high dimensionality and complex data types of malware attacks data make industry and academia a challenging task. Many machine learning algorithms, such as SVM [8], logistic regression [9], and XGBoost [10], have been used to develop intrusion detection models and have achieved good results. In recent years, deep learning has received extensive attention due to its ability to mine high-dimensional and large-scale data.In this paper, we propose a new method by giving a dataset consisting of machine properties and whether the machine was hit by malware, a model to predict whether a new machine given its properties will be hit by malware soon, can be built. There are various machine learning techniques that can be used to model the patterns associated with malware attacks in computer systems. The ones explored in this paper are: LightGBM, SVM and LSTM. Taking into consideration the given dataset, LGBM+MSVM is settled on as the best approach to the proposed problem statement.

The rest of the paper is organized as follows. We first briefly review the related literature in Section 2. Section 3 presents our method to improve malware prediction analysis framework. Section 4 presents experimental results which show that our method stands out as a state-of-the-art technique. Finally, we present a discussion and conclude the paper in Section 5.

2. RELATED WORK

In this section, we discuss previous state-of-art works which uses machine learning for malware analysis.

In [11], the authors aimed for classification of malware using a deep learning model to obtain an accurate and efficient performance. The system proposed in this study extracts a number of features and trains the long short-term memory (LSTM) model. The study utilizes hyper-parameter tuning to improve the accuracy and efficiency of the LSTM model. The findings revealed 99.65% accuracy using sigmoid function that outperforms other activation function. This work can be helpful in malware detection to improve security posture.

In [12], the authors have explored different algorithms to obtain the best algorithm for malware prediction and to obtain the best set of features that will help us in predicting malware efficiently. From our analysis, they have seen that ensemble methods are better than traditional machine leaning algorithms for predicting malware. They have reduced the number of features from 215 to 100 achieving an accuracy of 99.5% using Light GBM. In addition, they have obtained an accuracy of 99.1% using Random Forest having only 55 features.

In [13], the authors applied machine learning algorithms to predict the malware infection rates of computers based on its features. They used supervised machine learning algorithms and gradient boosting algorithms. They have collected a publicly available dataset, which was divided into two parts, one being the training set, and the other will be the testing set. After conducting four different experiments using the aforementioned algorithms, it has been discovered that

LightGBM is the best model with an AUC Score of 0.73926.

In [14], the authors proposed an integrative feature extraction algorithm based on simhash, which combines the static information e.g., API (Application Programming Interface) calls and dynamic information (such as file, registry and network behaviors) of malicious samples to form integrative features. The experiment extracts the integrative features of some static information and dynamic information, and then compares the classification, time and obfuscated-detection performance of the static, dynamic and integrated features, respectively, by using several common machine learning algorithms. The results show that the integrative features have better time performance than the static features, and better classification performance than the dynamic features, and almost the same obfuscated-detection performance as the dynamic features. This algorithm can provide some support for feature extraction of malware detection.

In [15], the authors proposed a control flow-based feature extraction dynamic programming algorithm for fast extraction of control flow-based features with polynomial time. From the experimental results, it is demonstrated that the proposed algorithm is more efficient and effective in detecting malware than the existing ones. Applying the proposed algorithm to an Internet of Things dataset gives better results on three measures: Accuracy = 99.05%, False Positive Rate = 1.31% and False Negative Rate = 0.66%.

In [16], the authors represented malware as opcode sequences and detect it using a deep belief network (DBN). Compared with traditional shallow neural networks, DBNs can use unlabeled data to pretrain a multi-layer generative model, which can better represent the characteristics of data samples. They compared the performance of DBNs with that of three baseline malware detection models, which use support vector machines, decision trees, and the k-nearest neighbor algorithm as classifiers. The experiments demonstrate that the DBN model provides more accurate detection than the baseline models. When additional unlabeled data are used for DBN pretraining, the DBNs perform better than the other detection models. They also used the DBNs as an autoencoder to extract the feature vectors of executables. The experiments indicate that the autoencoder can effectively model the underlying structure of input data and significantly reduce the dimensions of feature vectors.

3. METHODOLOGY

For classification task, LightGBM is fast, efficient and difficult to over fit, especially for high-dimensional data, but it can only give label classification. SVM algorithm with linear kernel function can give hyperplane representing malware detection, but its effect depends on the quality of feature selection. Taking the advantages of the two algorithms, this paper proposes a malware detection algorithm based on LightGBM + SVM fusion model, shown in Figure 1.Firstly, the training set is sent to LightGBM for training. In the training process, features importance used for feature selection are determined by two parts, i.e., times being used and their gain to final classification results. Finally, selected features are sent to SVM for training to solve the malware detection boundary.

Dataset:

The dataset for the proposed work chosen was the Microsoft Malware Prediction dataset from Kaggle which consists of system properties and machine infections of systems running on Windows [17]. The problem which we tried to solve was to predict if a machine will soon be hit with malware. There are totally around 9 million rows and 84 columns in the training dataset and there are another 9 million rows and columns 83 in the test dataset. Each row in the test and train dataset corresponds to a system, uniquely identified by a ’MachineIdentifier’. ’HasDetections’ is the target and indicates whether Malware was detected on the system. ’HasDetections’ is missing in the test dataset and must be predicted using the train dataset.The goal of this competition is to predict a Windows machine’s probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender. Each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier. HasDetections is the ground truth and indicates that Malware was detected on the machine. Using the information and labels in train.csv, you must predict the value for HasDetections for each machine in test.csv.

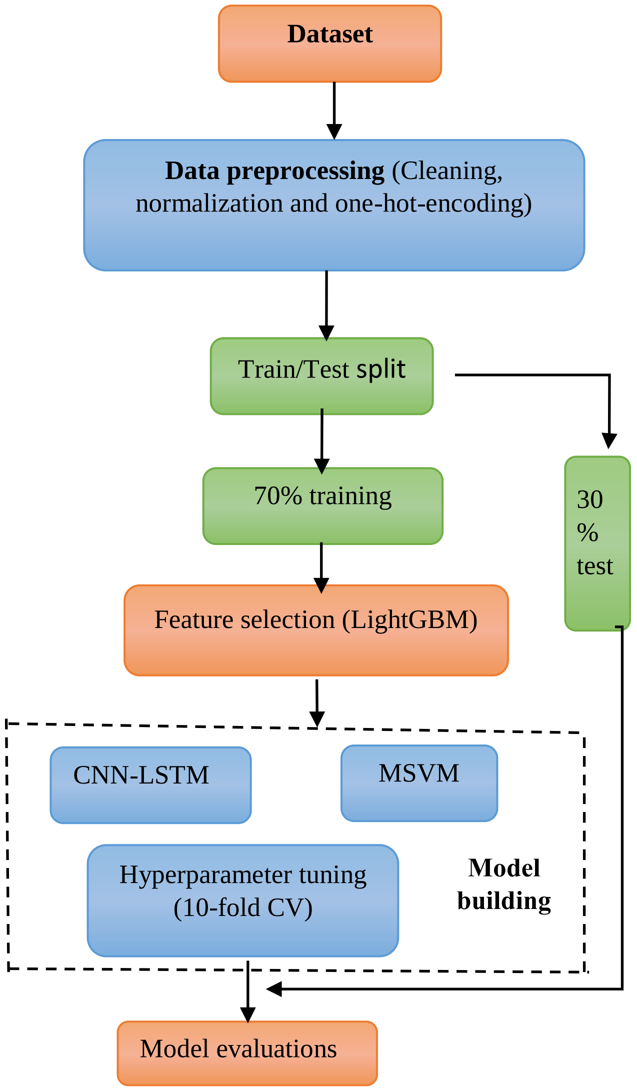
Dataset
Data preprocessing (Cleaning, normalization and one-hot-encoding)
Train/Test split
70% training
30 % test
Feature selection (LightGBM)
CNN-LSTM
MSVM
Hyperparameter tuning (10-fold CV)
Model building
Model evaluations

Figure 1: Scheme of malware prediction algorithm with LightGBM + MSVM fusion

Data Preprocessing

Data preprocessing is a necessary step before training the model. It includes three parts: data Cleaning, data normalization and one-hot-encoding.

Data Cleaning: Before the model was trained, some data cleaning had to be done. There were 5 columns in the dataset which had above 70% empty values. They were dropped. The columns were PuaMode, CensusProcessorClass, DefaultBrowsersIdentifier, CensusIsFlightingInternal, and CensusInternalBatteryType.A column named SmartScreen had some inconsistent values. There were values like 'promt', 'promprt', etc for the value 'prompt','of' in place of 'off', 'enabled' for 'on','00000' for '0' and so on. Those inconsistent values too had to be replaced by the proper values. We also encountered many NaN values for many rows. So, we replaced the NaN values with the median of the column to which it belonged to.

Data Normalization: The min-max normalization method is adopted to scale the value $x_{i,j}$into the numeric range [0,1], according to:

$$x_{fj} = \frac{x_{fi} - min(x_f)}{(x_f) - min(x_f)} \quad (1)$$

Among them, max(x_f) and min(x_f) represent the maximum and minimum value of the f_{th} (numerical) feature x_f ; whereas x_{fj} is the normalized feature value ranged between [0,1].

One-Hot-Encoding: Also looking at the test dataset, many category values for certain categorical attributes that were present in the training data were missing in test data. Hence, they will not be of much use in predicting for test data. Such category values were replaced with a uniform dummy value prior to encoding. For all categorical variables sorted frequency encoding was done.

Feature selection with LightGBM:

Feature selection, the process of finding and selecting the most useful features in a dataset, is a crucial step of the machine learning pipeline. Unnecessary features decrease training speed, decrease model interpretability, and, most importantly, decrease generalization performance on the test set. The goal is to limit the number of features used in the final model based on features' importance and correlation with others. The averaged importance score for each feature was calculated by using lightGBM. Next, we calculate the correlation within those features. Then, we select features which have high importance scores (higher than an importance score threshold) and low correlation (lower than a correlation threshold). For the features which are highly correlated, the one with highest importance score will be chosen from correlated features sets. The feature selection is done in the training set.

The training instances are arranged in descending order according to the absolute value of their gradients, and the first a% of the instances with larger gradients are retained to form the instance subset *A*. For the residual set A^c formed by the (1 − a)% of instances with smaller gradients, a subset *B* of size b*| A^c | is randomly formed. Finally, the instance is split according to the estimated variance gain $V_j^*(d)$ on the subset $A \cup B$.

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

where $A_l = \{x_i \epsilon A: x_{ij} \le d\}$, $A_r = \{x_i \epsilon A: x_{ij} > d\}$, $B_l = \{x_i \epsilon B: x_{ij} \le d\}$, $B_r = \{x_i \epsilon B: x_{ij} > d\}$, d is the point in the data where the split is calculated to find the best gain invariance, and the coefficient $\frac{1-a}{b}$ is used to normalize the gradient sum over B back to the size of A^c.

The trees in the LightGBM model are constructed based on the above steps. Let the feature set, x_i be x_1, x_2,. . . , x_m where *i* = 1, 2, . . . , *m*. Then, according to the number of times each feature is used to split the training data across all trees, the feature importance score FIS_i is calculated. Therefore, the feature importance score set is represented as:

$$FIS_i = \{s | a = w_i x_i\} \quad (3)$$

where w_i represents the weight of each feature, and xi represents the feature set.

Feature Engineering

To capture the sampling rate, we engineer a new feature 'WeekNumber' which is the number of weeks that had passed since 1st Jan 2018 when the observation was sampled. If an observation was sampled before 1st Jan 2018, this attribute will have a negative value. The newly engineered feature is now representative the sampling rate at the time at which the observation was sampled. This new feature is important for predicting whether the observation is of a system with a malware detection.

An observation with 'WeekNumber' value equal to the most

frequent value of 'WeekNumber' will have been sampled from the time period when sampling rate was very high. When the sampling rate is high detection rate is high as seen in Fig. 5. Thus, an observation with a common 'WeekNumber' value is more likely to have malware detection. We later apply sorted frequency encoding on the attribute as the sorted frequency of 'WeekNumber' is essentially sampling rate, and sampling rate directly correlates with the target which we are trying to predict (Malware detection rate).

Classification with Multi-Support Vector Machine (MSVM): We analyze it with a classification-regression machine learning system. Unlike artificial neural networks, SVM has many generalizations that prevent overfitting. With a suitable kernel, the SVM can handle nonlinear data for regression and classification. *C* and *γ*are SVM hyper-parameters. Before model training, we should tweak hyper-parameters. We trained the SVM with the overall parameters. We checked the validation collection model by testing the model using the reference dataset. As SVM kernels and decision functions, we employed linear, polynomial, and RBF (ovo). A comparative examination of *C,γ* kernel parameters. In training, validation, and testing, practically all classifiers are the same. Because SVM was created for binary classification, multi-class classification is difficult. ovo means binary versus one. SVM may not be suitable for this purpose, though. The result displays inaccurate data. Through exploration, we uncovered three multi-class problem-solving systems. ovr, *M* (*M*-1)/2, and SVM formula expansion. The adapted classifier has the following form.

$$f_{tar}(x) = \sum_{k=1}^{M} \tau^k f_{src}^k(x) + \Delta f(x) \tag{4}$$

where, $\tau^k \in (0,1)$ is the weight of each base classifier $f_{src}^k(x)$, $\Delta f(x)$ is the perturbation function that is learnt from a small set of labelled target-domain data in D_{tar}^l. As shown in [27] it has the form:

$$\Delta f(x) = w^T \phi(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) \tag{5}$$

$$\min_{w,\tau,\xi} \frac{1}{2} w^T w + \frac{1}{2} B(\tau)^T \tau + C \sum_{i=1}^{N} \xi_i$$

$$\text{s.t. } y_i \sum_{k=1}^{M} \tau^k f_{src}^k(x) + y_i w^T \phi(x_i) \geq 1 - \xi_i$$

$$\xi_i^m \geq 0, \forall \left(x_i, y_i\right) \in D_{src} \tag{6}$$

$$w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i) \; \tau^k = \frac{1}{B} \sum_{i=1}^{N} \alpha_i y_i f_{src}^k(x_i) \tag{7}$$

where τ^k a weighted sum $y_i f_{src}^k(xi)$ and the classification performance of the target domain. Consequently, if we classify the labelled destination domain info well, we have allocated more massive base classifiers. With the new decision function provided (4), (5) and (7) now:

$$f_{tar}(x) = \frac{1}{B} \sum_{k=1}^{M} \sum_{i=1}^{N} \alpha_i y_i f_{src}^k(x_i) f_{src}^k(x) + \Delta f(x)$$

$$= \sum_{i=1}^{N} \alpha_i y_i \left(K(x_i, x) + \frac{1}{B} \sum_{k=1}^{M} f_{src}^k(x_i) f_{src}^k(x) \right) \tag{8}$$

Compared with (8) a regular SVM model$f(x) = \sum_{i=1} \quad a_i y_i K(x_i, x)$, this multi-classification adaptation model may be interpreted as applying additional features to the projected labels of the basic classifiers in the target domain. The scalar *B* combines the influence of the initial characteristics and more features according to this interpretation.

4. RESULTS AND DISCUSSIONS

In this section, we will present the performance of our proposed malware analysis framework with extensive experimental results. Scikit-Learn library produced results with detailed accuracy of each class and confusion matrix for both binary classifier and multi-class classifier. For malware analysis, we divided them into training and testing set. The training set contains 70% of malware samples, and the testing set contains 30% of malware samples.We have used five different performance metrics, i.e., Accuracy, Precision, Recall, F-measure and AUC (Area under ROC Curve) to evaluate the performance of our proposed malware analysis framework. The performance metrics can be expressed in terms of TP (True Positive), FP (False Positive), TN (True Negative) and FN (False Negative).

Exploratory Data Analysis:

Dataset consists of about 8 million rows and 83 columns describing several attributes of Microsoft devices under consideration. An exploration of the time series aspect of the problem – As mentioned previously, malware detection could be viewed as a time series problem. To visualize this, plots sampling density over time were drawn for both train and test data as seen in Fig 2. and Fig 3. respectively. The green line depicts sampling density over time and the black line represents the malware detection rate (only for training data) over time. The following observations can be made:

• A majority of the train data is from August to October 2018.

• A majority of the test data is from October to December 2018.

• It is evident that there exists a relation between detection rate and sampling rate as the detection rate roughly increases and decreases with sampling rate for the training data.

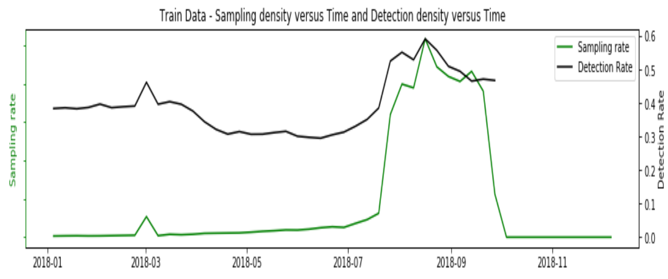

Figure 2: Train Data - Sampling density versus Time and Detection density versus Time

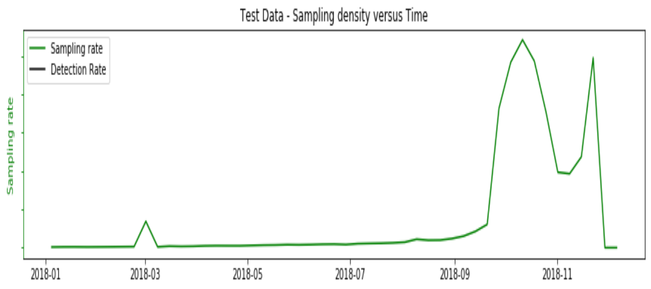

Figure 3: Test Data - Sampling density versus Time

The last point indicates that it would be useful to engineer a feature that can capture the sampling rate at the time the observation was sampled, as a machine is more likely to have malware if it is sampled from a period with a high sampling rate.

Target variable is whether the malware is detected or not. Figure 4 shows that data is balanced with respect to target attribute hence no need of up sampling any class.

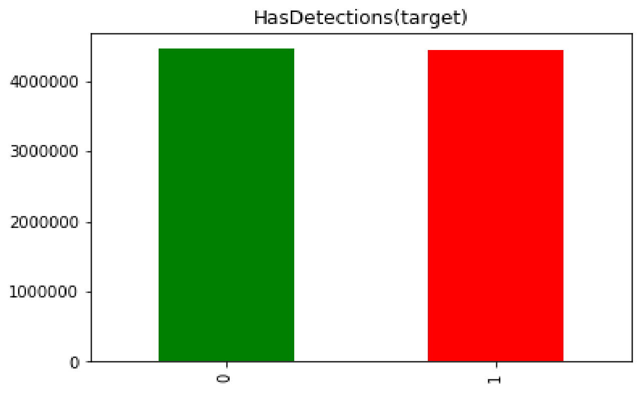

Figure 4: Distribution of Target attribute across the dataset

Figure 5 shows that Detachable devices usually have SSD, desktop PC have HDD and SDD and Notebooks usually have HDD.

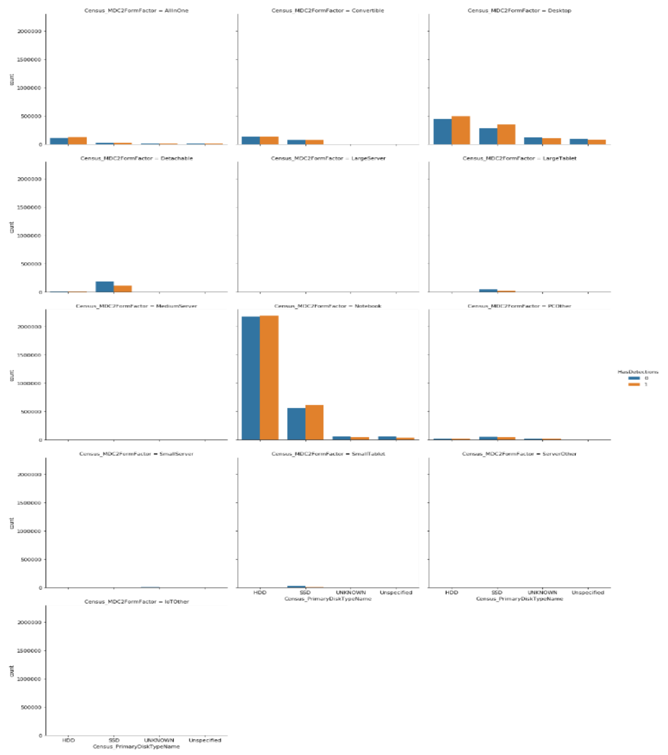

Figure 5: Correlation between CensusPrimaryDiskTypeName and CensusMDC2FormFactor

We can see several interesting things here:

PuaMode and Census_ProcessorClass have 99%+ missing values, which means that these columns are useless and should be dropped; In Default Browsers Identifier column 95% values belong to one category, so we think this column is also useless; There are 26 columns in total in which one category contains 90% values. we think that these imbalanced columns should be removed from the dataset.

Fig 6. depicts the feature importance of 20 of the most important features of the train data for the model. Note that newly engineered feature 'WeekNo' is one of the most important attributes for predicting malware.

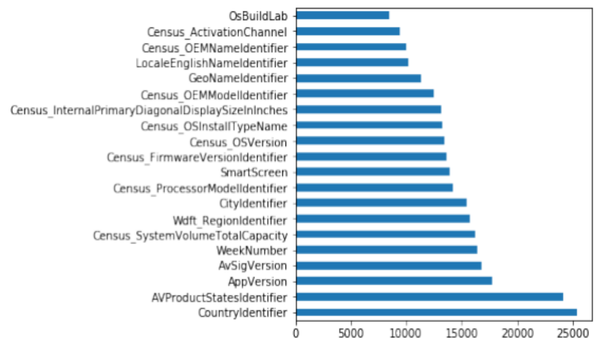

Figure 6: Feature importance of the 20 most important features- sourced from the trained LGBM model

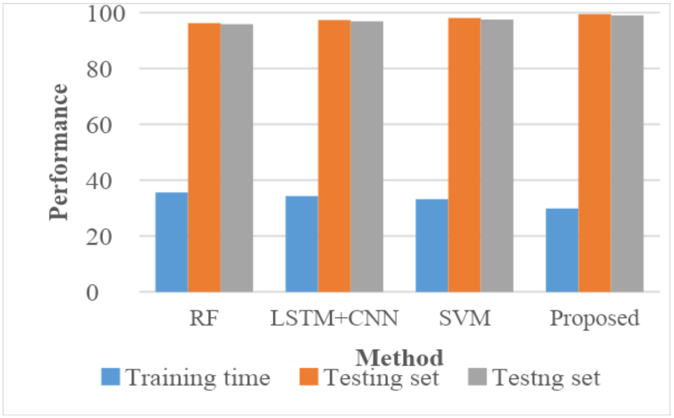

Figure 7: Comparison of the classification accuracy with a RF, LSTM+CNN, SVM and proposed work.

Table 1: Evaluation results of training data and test data with a baseline and proposed method

Evaluation metric	LSTM+CNN Baseline		LightGBM+ SVM Proposed	
	Training Data	Test Data	Training Data	Test Data
ROC AUC	0.8909	0.9175	0.9211	0.9322
Time (msec)	250.109	167.68	189.21	145.71

Based on the Table 1, the proposed model area under the curve for the receiver operating characteristic curve (ROC) is 93.22% for test data and testing time is 145msec. The evaluation of the metrics table and ROC curve visualization gives a summary of how the model performs in general. Figure 8 shows that our method produces less training and testing time of 189.21msec and 145.71msec as related to baseline model.The computational time of our model is around 60 sec for training on 300 samples and less than 50 ms for testing on one surface.

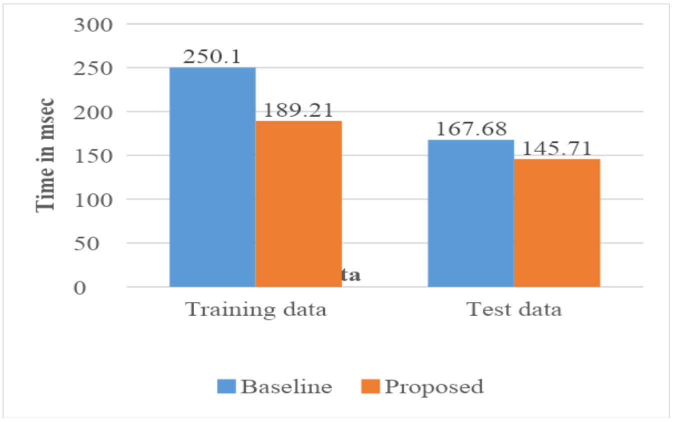

Figure 8: Comparison of the training and testing time using baseline and proposed model

5. CONCLUSION

In this paper, we present a novel malware analysis framework which can detect and classify malware efficiently. Our proposed approach uses two different feature selection algorithms in order to extract most relevant features which reduces training time and increases detection and classification accuracy. Given the relatively good performance of our model we can conclude that using machine learning techniques to predict whether a system will soon be hit with malware could potentially be an excellent preventative measure in addition to existing security tools. In our proposed model, we select features based on the feature importance score generated by the LightBGM model. From the comparison results, the classification accuracy of the model proposed (LightGBM+MSVM) in this paper is higher, reaching 99.12%, which is more advantageous than other models. The experimental results show that the method has a good detection effect on network intrusion detection.

REFERENCES

[1] Canfora, Gerardo &Mercaldo, Francesco &Visaggio, Corrado Aaron. (2016). An HMM and structural entropy-based detector for Android malware: An empirical study. Computers & Security. 61. 10.1016/j.cose.2016.04.009.

[2] Canfora, G., Mercaldo, F., &Visaggio, C. A. (2016). An HMM and structural entropy-based detector for Android malware: An empirical study. Computers & Security, 61, 1-18.

[3] Talha, K. A., Alper, D. I., & Aydin, C. (2015). APK Auditor: Permission-based Android malware detection system. Digital Investigation, 13, 1-14.

[4] Lee, Kyungho&Seo, Dongkyun. (2017). Comparing security vulnerability by operating system environment. International Journal of Services Technology and Management. 23. 154. 10.1504/IJSTM.2017.10002715.

[5] Murugan, S. &Kuppusamy, K.. (2011). Malware And Operating Systems. i-manager's Journal on Electronics Engineering. 1. 1-4. 10.26634/jele.1.2.1365.

[6] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., and Venkatraman, S. (2019). Robust Intelligent Malware Detection Using Deep Learning. IEEE Access, 7:46717–46738.

[7] Wangoo, D. P. (2018). Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design. In 2018 4th International Conference on Computing Communication and Automation (ICCCA), pages 1–4.

[8] Liu, W.; Ci, L.; Liu, L. A New Method of Fuzzy Support Vector Machine Algorithm for Intrusion Detection. Appl. Sci. 2020, 10, 1065.

[9] Maalouf, M.; Homouz, D.; Trafalis, T.B. Logistic regression in large rare events and imbalanced data: A performance comparison of prior correction and weighting methods. Comput. Intell. 2018, 34, 161–174.

[10]Bhattacharya, S.; Krishnan, S.S.R.; Maddikunta, P.K.R.; Kaluri, R.; Singh, S.; Gadekallu, T.R.; Alazab, M.; Tariq, U. A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU. Electronics 2020, 9, 219.

[11]Iqbal, Saba & Ullah, Abrar &Adlan, Shiemaa&Soobhany, A. (2022). Malware Prediction Using LSTM Networks. 10.1007/978-981-16-7618-5_51.

[12]Sarah, Neamat& Rifat, Fahmida & Hossain, Md Shohrab&Narman, Husnu. (2021). An Efficient Android Malware Prediction Using Ensemble machine learning algorithm. Procedia Computer Science. 191. 184-191. 10.1016/j.procs.2021.07.023.

[13]Zawad, Safir& Evan, Nahian& Mansur, Raiyan& Asad, Ashub& Hossain, Muhammad Iqbal. (2020). Analysis of Malware Prediction Based on Infection Rate Using Machine Learning Techniques. 10.1109/TENSYMP50017.2020.9230624.

[14]Li, Y., Liu, F., Du, Z., & Zhang, D. (2018). A Simhash-Based Integrative Features Extraction Algorithm for Malware Detection. Algorithms, 11(8), 124. https://doi-org.proxy1.library.eiu.edu/10.3390/a11080124.

[15]Phu, T. N., Tho, N. D., Hoang, L. H., Toan, N. N., & Binh, N. N. (2021). An Efficient Algorithm to Extract Control Flow-Based Features for IoT Malware Detection. Computer Journal, 64(4), 599–609. https://doi-org.proxy1.library.eiu.edu/10.1093/comjnl/bxaa087.

[16]Yuxin, D., & Siyi, Z. (2019). Malware detection based on deep learning algorithm. Neural Computing & Applications, 31(2), 461–472. https://doi-org.proxy1.library.eiu.edu/10.1007/s00521-017-3077-6.

[17]Microsoft, Microsoft Malware Prediction Data Description, Kaggle. Accessed on: Oct. 11, 2019. [Online]. Available: https://www.kaggle.com/c/microsoft-malware-prediction/data.