Journal of Computer Science

Original Research Paper

Preliminary Analysis of Malware Detection in Opcode Sequences within IoT Environment

[1]Firas Shihab Ahmed, [1]Norwati Mustapha,
[2]Aida Mustapha, [3]Mohsen Kakavand and [2]Cik Feresa Mohd Foozy

[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
[3]School of Science and Technology, Sunway University, Bandar Sunway, Selangor, Malaysia

Article history
Received: 28-06-2020
Revised: 25-09-2020
Accepted: 02-10-2020

Corresponding Author:
Norwati Mustapha
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia,
Selangor, Malaysia
Email: norwati@upm.edu.my

Abstract: With the technological development and means of communication, the Internet of Things (IoT) has become an essential role in providing many services in daily life through millions of heterogeneous but interconnected devices and nodes. This development is opening to many security and privacy challenges that can cause complete network breakdown, bypassed access control or the loss of critical data. This paper attempts to provide a preliminary analysis for malware detection within data generated by IoT-based devices and services in the form of operational codes (Opcode) sequences. Three machine learning algorithms are evaluated and compared for accuracy, precision, recall and F-measure. The results showed that the Random Forest (RF) achieved the best accuracy of 98%, followed by SVM and *k*-NN, both with 91%. The results are further analyzed based on the Receiver Operating Characteristic (ROC) curve and Precision-Recall curve to further illustrate the difference in performance of all three algorithms when dealing with IoT data.

Keywords: Machine Learning, Malware Detection, Operation Codes

Introduction

Today, the Internet of Things (IoT) has offered many services through interconnection of huge number of sensor devices, embedded systems or services (Mosenia and Jha, 2016; Azmoodeh *et al*., 2018). IoT has become a driving technology in many domain such as smart city, intelligent transportation, as well as health and energy systems (D'Orazio *et al*., 2016; Patel *et al*., 2012). The massive expansion of IoT applications has resulted in surge of data, hence opening to many security and privacy challenges such as the malware attacks (Tankard, 2015; D'Orazio *et al*., 2016; Watson and Dehghantanha, 2016). The core reason of these challenges is simply because any network is subject to threat and penetration from devices that are connected to the network (Yang *et al*., 2016; Li *et al*., 2019; Wazid *et al*., 2019).

Malware is the collective name for different types of malicious software, including viruses, ransomware and spyware. The main issue with malware detection lies in the ineffective methods used for signing and monitoring the suspected code for known security changes. This has led to many investigation on formulating new methods and techniques that can overcome different attack vectors (Burguera *et al*., 2011).

Machine learning is a popular method used to detect attacks and malware, as the concept of self-learning by extracting data features and training them is able to identify features of other data that have not been trained before (Rehman *et al*., 2018). In 2016, feature selection methods have been investigated in anomaly detection systems using the Principle Component Analysis (PCA) and Guttman-Kaiser (Kakavand *et al*., 2016). However, the study was not limited to reducing the dimensions of the features but rather preserving the information that is important in classifying the anomalies. The results showed a high intrusion rate of 97% with a false positive rate of 1.2%.

Research by Milosevic *et al*. (2017) studied malware detection that targeted android systems. This research used permissions and source code analysis through the use of the bag-of-words representation model and features implemented using a privacy and security protection application for Android devices called OWASP Seraphim droid. The results showed that the classification accuracy achieved was 89% and further increased to 95% with source code analysis.

Subsequent research by Kakavand *et al.* (2018) applied two machine learning algorithms, which are Support Vector Machines (SVM) and *K*-Nearest Neighbors (*k*-NN), through the supervised learning process in order to classify malware and benign. This research focused on android application data and reported 79.1 and 80.5% accuracy percentage for SVM and *k*-NN, respectively.

In a more recent work, Kumar *et al*. (2019) presented a new method that combined machine learning methods and blockchain technology to improve the performance of malware detection model in Android devices. The proposal was implemented using a sequencing approach that combined clustering and classification in blockchain technology, as well as extracting information about malware and storing it back in blockchain. The main purpose was to develop a malware database, thus easily detecting other malicious in the future that do not exist previously.

Deep learning approaches have also been explored to classify data based on the dynamic approach to malware detection. A new method has been introduced to extract features in order to analyze the dynamic behavior and build a model of repeated neural networks to extract the abstract features (Xiaofeng *et al*., 2018). This research also studied many of the serial data processing to get rid of redundant data. The results showed that combining the two methods had better results and it was 99.3% where the classification performance was proven to be higher when merging machine learning and deep learning methods as compared to using the models separately.

In general, malware detection is an important and fundamental matter in providing security in IoT-based applications such as smart devices. According to the Kaspersky Lab, in 2016 most of the Internet devices were unsafe and most of the devices had a default password or security glitches that were not processed, which lead to easy penetration of these devices (Kolias *et al*., 2017; HaddadPajouh *et al*., 2018; Goyal *et al.*, 2019). Security experts have warned the dangers to which the Internet of Things (IoT) can cause, specifically the malware due to the widespread dependence on devices connected to the Internet. Organizations are in need of a mechanism that has the ability to discover malware and suspicious bugs when their devices and services are connected to the Internet (Mahindru and Singh, 2017; Meidan *et al*., 2017).

In detecting malware within IoT environment, Bragen (2015) investigated both supervised and non-supervised machine learning approach to detect attacks on IoT-generated data such as spoofing attack eavesdropping and jamming. HaddadPajouh *et al*. (2018) used three different Long Short Term Memory (LSTM), a type of Recurrent Neural Network (RNN) machine learning architecture. The results showed that second configurations with two-layer neurons achieved the highest accuracy of 98.2%. Although various machine learning and deep learning approaches have been used in malware detection, the literature has shown that the domain has evolved from email to mobile devices and most recently, to IoT devices (Lu *et al*., 2003).

In order to address the gap in providing adequate protection systems among IoT-based applications and smart devices, this research is set to provide a preliminary analysis of malware detection for operation code (Opcode) sequences within IoT environment as benchmark performance for future works. The remaining of this paper is organized as follows. Section 2 presents the materials and methods along with validation methods and algorithms. Section 3 presents the results, Section 4 discusses the results and finally section 5 concludes with future work.

Materials and Methods

In detecting malware within IoT-based applications, a classification methodology is adopted to predict categorical class labels (malware vs. benign) from the operational codes (Opcode) sequence dataset. The classification experiments will be carried out based on training and testing dataset to classify newly available data (Allahyari *et al*., 2017). The classification methodology is shown in Figure 1. The sub-sections will detail out the dataset, pre-processing, model validation, algorithms and the evaluation metrics.

Dataset

This research focuses on malware detection within data generated by IoT-based applications. With Raspberry Pie II, it is worth noting that AMD processors have been widely used in cloud edge devices, hence qualifying Raspberry Pi II as an IoT cloud edge device. The dataset used in this research was sourced from the Linux Debian package repositories (https://pkgs.org/).

The dataset is based on 32-bit ARM-based malware within the Virus Total Threat Intelligence platform as of 30 September 2017 in the form of Executable and Linkable Format (ELF). The ELF is used because it consideres the structure for binaries, libraries and core files, as well as roles in the process of linking program and execution. Since ELF features are considered static features, higher accuracy in malware detection is expected. Analyzing ELF is also important as it gives generic understanding of how an operating system works during software development.

Following HaddadPajouh *et al*. (2018), a Linux bash script for the Opcode samples in the dataset was written to extract the sequence of Opcodes in each sample. After extracting the ELF files using the Debian bundle, the dataset provided 280 malware and 270 benign

DOI: 10.3844/jcssp.2020.1306.1318

programs samples. Next, Object-Dump tool was used to decompile all samples to extract Opcode sequences in each sample. Fig. 2 and Fig. 3 show the excerpts of malware and benign samples.

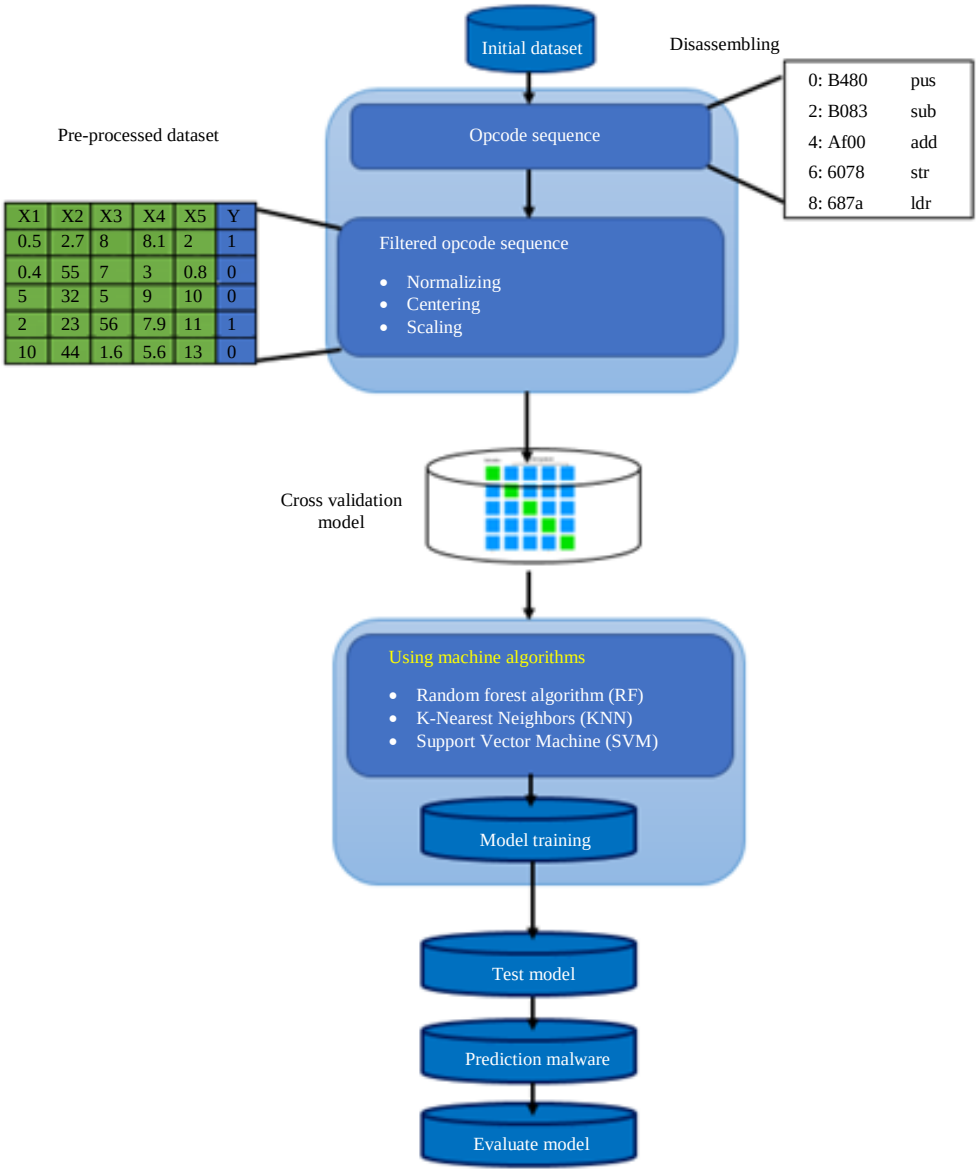

Fig. 1: Research methodology

	w	l	k	9	(	v	4	8	2		...	5	p	3	)	e	_	\n	-	s	Class
0	17	731	8	0	245	228	0	0	0	5599	...	0	990	2	245	533	0	0	0	713	Attack
1	44	4462	155	0	1542	1062	0	0	0	28252	...	0	5476	10	1542	3224	0	0	0	3310	Attack
2	184	3192	80	0	448	1891	0	0	4	19741	...	0	5420	31	448	2395	0	0	1	5174	Attack
3	44	5684	163	0	1547	1156	0	0	1	38833	...	0	7660	23	1547	4218	0	0	0	5235	Attack
4	56	4036	1	0	988	4294	0	0	0	22813	...	0	5519	3	988	2644	0	0	0	4565	Attack

Fig. 2: Excerpt of attack (malware) samples

	w	l	k	9	(	v	4	8	2		...	5	p	3	)	e	_	\n	-	s	Class
0	3	485	8	0	97	82	0	0	0	3298	...	0	628	1	97	378	0	0	0	534	Normal
1	8	1023	11	0	240	245	0	0	0	6612	...	0	1376	5	240	618	0	0	0	944	Normal
2	15	484	5	0	160	107	0	0	0	3674	...	0	745	2	160	286	0	0	0	440	Normal
3	14	1854	32	0	403	449	0	0	0	9797	...	0	2338	4	403	1336	0	0	0	1135	Normal
4	62	4197	75	0	945	1077	0	0	0	22394	...	0	5368	13	945	2825	0	0	0	2661	Normal

Fig. 3: Excerpt of normal (benign) samples

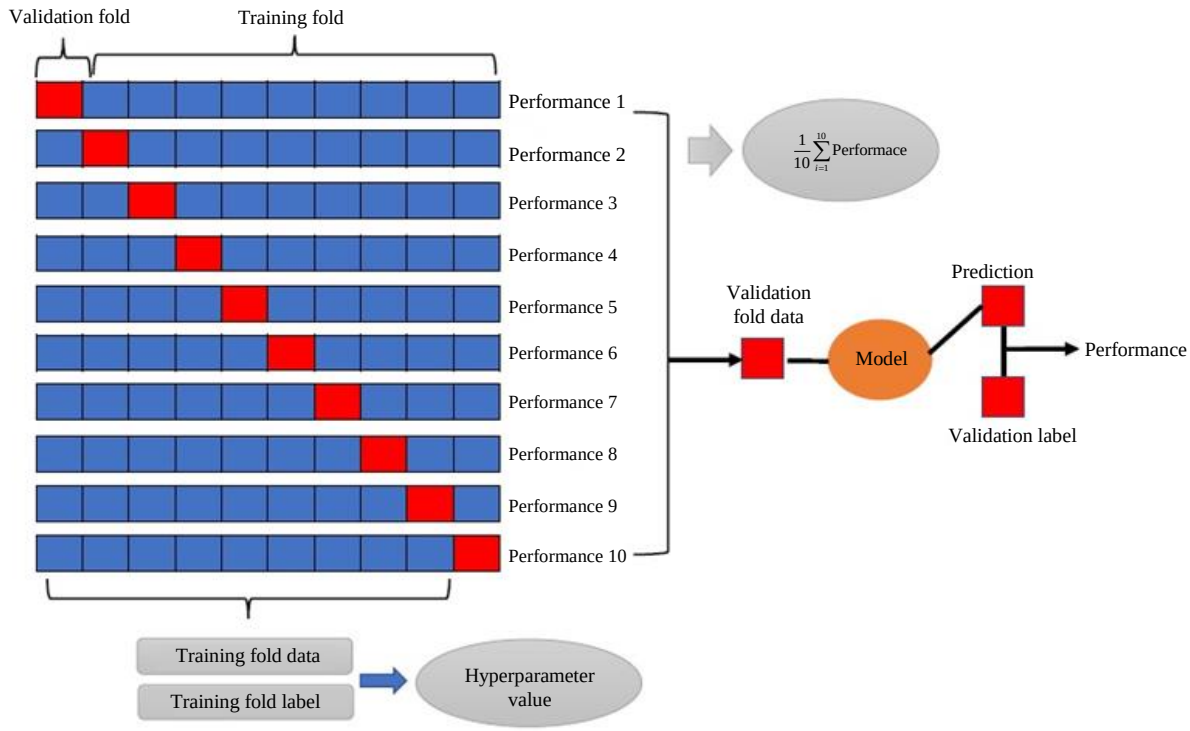

Fig. 4: Validation methodology

Pre-Processing

After disassembling, the opcode sequences extracted will be pre-processed through various pre-processing steps, which include normalizing, centering and scaling. A Phyton code is used to convert the opcodes into Excel file with rows of opcodes and columns of features before they are ready for splitting into training and testing set. Normalizing can give us several meanings, is used informally in statistics, it is the ability to remove the unit's measurement of data, which allows us to compare data from different places with greater ease.

For many types of data, centering and scaling are intertwined. Centering corresponds to a subtraction of a reference vector (often represented by the mean values of the variables or the settings of the setpoint). Scaling corresponds to a multiplication by a vector. The choice of scaling vector is crucial (Bro and Smilde, 2003).

Model Validation

The anomaly detection or malware classification experiments were carried out using the *k*-fold validation method for training and testing as shown in Fig. 4. In the ten-fold validation setting, eight times was used for training, one for documentation and the rest for testing. Following (Davis and Goadrich, 2006). A confusion matrix was derived summarized from 10 experiments together, analyzed and reported. Based on this figure, the *k*-fold cross-validation method divides *n* samples into *k* groups, whereby validation uses *n*/*k* a sample in each group at a time. When a group is chosen for a test, the group of *k*-1 and the other is used for training, after

which the training is switched to the test group every time (cycle). In this way, the performance of the classifier can be determined by calculating the average error of *k* in each cycle (Varoquaux, 2018; Zhang *et al*., 2016).

Algorithms

Three machine learning algorithms are used in the preliminary experiments, which are *k*-Nearest Neighbor (*k*-NN), Support Vector Machines (SVM) and Random Forest. All the algorithms were implemented using the Anaconda Navigator, TensorFlow, scikit-learn machine learning, Jupyter note-book, as well as tools in Phyton. All three algorithms have served as benchmark algorithms under the machine learning approach in many malware or anomaly detection problems in Internet-of-Things (IoT) devices (Hasan *et al*., 2019; Nakhodchi *et al*., 2020; Darabian *et al*., 2020), networks (Kumar and Lim, 2019) and services (Ham *et al*., 2014; Sethi *et al*., 2017; Tien *et al*., 2020). The other reason is that these algorithms are more efficient with a small data set if compared to other methods such as deep learning methods that need big data since the data set of this research is considered small in size (Gislason *et al*., 2003; Noi and Kappas, 2018; Wang *et al*., 2018a).

k-Nearest Neighbor (k-NN)

k-NN is an algorithm that determine a class of *k* nearest training samples through finding the most frequent class available in the feature space (Gupta and Mittal, 2018; Wang *et al*., 2018b). Given a set of features and classes $(x_1, y_1),\ldots, (x_n, y_n)$, where features $x_1 \in R^d$ and classes $y_1 \in y$, then for a given *i*, *k*-NN rates the neighbors of a test sequence among the training sample and use the class labels for the nearest neighbors in order to expect the test vector class (Allahyari *et al*., 2017). Therefore, *k* takes the new *k*-points and ranks them according to the majority of votes obtained for the closest *k*. This algorithm uses the Euclidean distance to measure the resemblance between two vectors points (Aburomman and Reaz, 2016). The formula for *k*-NN is shown in Equation 1:

$$d^2\left(x_i, x_j\right) = \left\|x_i, x_i\right\| = \sum_{k=1}^{d}\left(x_{ik} - x_{jk}\right)^2$$
$$\left(x_i, x_j\right) \in R^d, x_i = \left(x_{i1}, x_{i2}, \ldots\ldots\ldots x_{id}\right) \quad (1)$$

Fig. 5 illustrates the concept of k-NN with two classes; when $k = 3$ and $k = 7$.

Support Vector Machines (SVM)

SVM is a group classifier models that is considered one of the effective methods with high popularization ability in practice (Huang *et al*., 2018). In contrast to statistical methods that are based on reducing experimental risks, SVM is based on reducing structural risks, this indicates the ability of this algorithm to avoid overrun. The way the algorithm works is to create hyperlinked decision plans that are divided into two categories with the maximal margin in the Fig. 6. These decisions are known as hyperplane as defined in Equation 2 where *w* is the weight vector, *x* is the input feature vector and *b* is the bias:

$$wx^T + b = 0 \quad (2)$$

The objective of SVM is to find decision boundaries between two classes that allow predictions of labels from one or more features, in a way that it separates the data and maximum the margin $1/\|w\|^2$, making them as close as possible to data points from each of these categories is called close points (Apostolidis-Afentoulis and Lioufi, 2015).

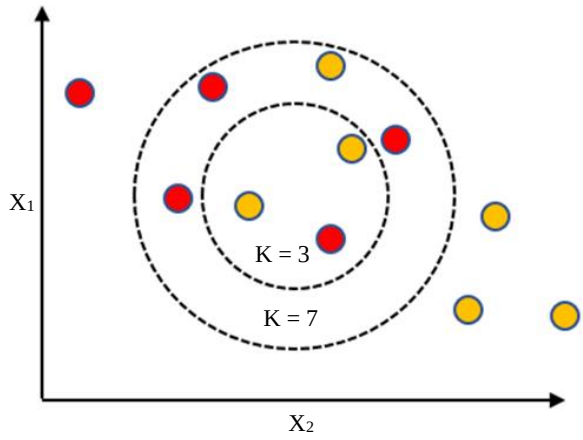

Fig. 5: K-NN algorithm concept

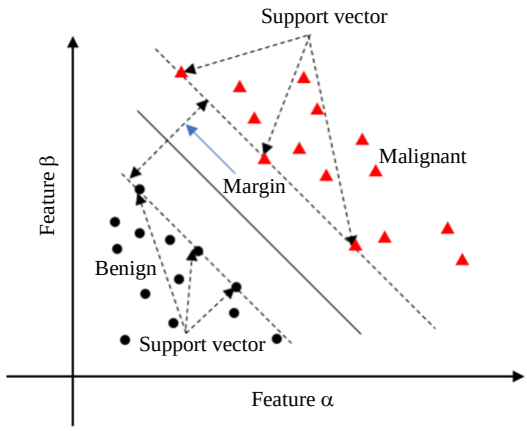

Fig. 6: Support Vector Machine (SVM)

Random Forests (RF)

RF is one of the groups of classification algorithms that depend on decision trees. This algorithm consists of different subsets of training data taken from the original data set using the method bootstrap sampling approach, after that creating the decision tree k through training the sub-groups, in the end building a random forest of decision trees as shown in Fig. 7 (Chen *et al*., 2016). RF has the least error in classifying data if compared with other traditional tree-based methods. The number of trees, the minimum node size and the number of features that are used to divide each node have several advantages such as, after completing the random tree forest can be referred to in the future, RF has the ability to overcome the over fitting (Farnaaz and Jabbar, 2016). An RF algorithm can be formalized as Equation 3:

$$S = \left\{\left(x_i, y_i\right)\right\}, \tag{3}$$

where x is the sample, y is the feature variable of s, n is the number of samples, m is the feature variable for each sample, $i = 1,2,\ldots, n$ and $j = 1,2,\ldots, m$.

Evaluation Metrics

Following Nikam (2015), the evaluation metrics used in the experiments are accuracy, precision, recall and F-measure. The percentage is calculated based on confusion matrix, where the rows in the matrix represent instances of the actual class and each column represent instances of the predicted class. A confusion matrix is implemented based on the results illustrated in Table 1. The correct forecasts are distributed with the number of values distributed for each category given the total expected results after classification (Powers, 2011).

Based on Table 1, a TP means the instance originally labeled as benign is correctly predicted as benign. A TN means the instance originally labeled as malware is correctly predicted as malware. An FP means the instance originally labeled as malware is incorrectly predicted as benign. Finally, an FN means the instance originally labeled as benign is incorrectly predicted as malware. Equation 4-7 show the formula for calculating the evaluation metrics.

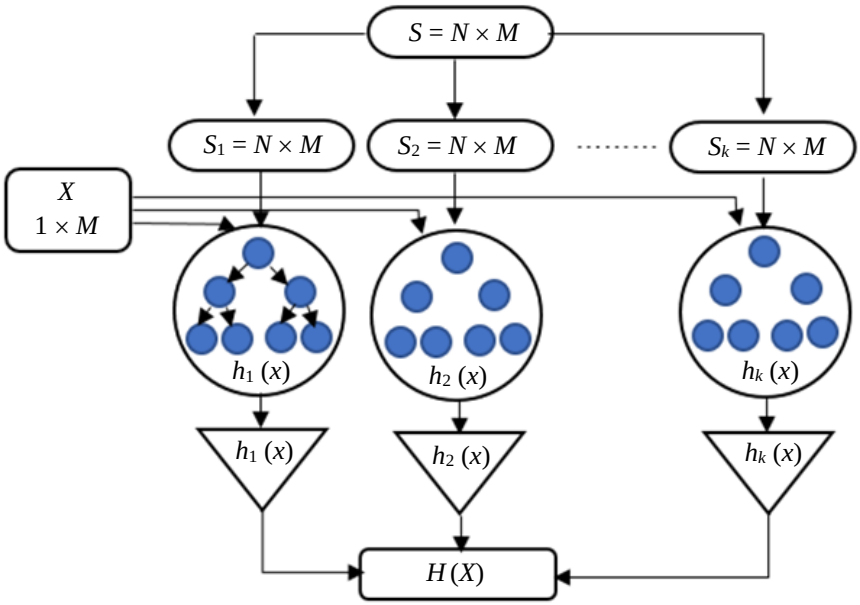

Fig. 7: Construction process of RF algorithm

Table 1: Confusion matrix

	Predicted	
Actual	Positive class (Benign)	Negative class (Malware)
Positive class (Benign)	True Positive (TP)	False Negative (FN)
Negative class (Malware)	False Positive (FP)	True Negative (TN)

Accuracy

Accuracy is number of correct predictions from all predictions made. The formula for calculating accuracy is shown in Equation 4:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Precision

Precision is the ratio of predicted malware that are correctly labeled a malware. The formula for calculating accuracy is shown in Equation 5:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

Recall

Recall or detection rate is the ratio of malware samples that are correctly predicted. The formula for calculating accuracy is shown in Equation 6:

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

F-Measure

F-Measure is the harmonic mean of precision and recall, which is a very useful measure of success of prediction when the classes are imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned (Sabharwal and Sedghi, 2017). The formula for calculating accuracy is shown in Equation 7:

$$F - measure = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (7)$$

Precision, recall and F-measure is measured because accuracy alone can be misleading. The Confusion Matrix as a way of describing the breakdown of errors in predictions for an unseen dataset. Precision will give exactness of a model while recall gives completeness the model. Finally, F-measure or F1 score gives the balance between the two.

Results

The purpose of the experiments is to compare the performance of three algorithms, which are *k*-Nearest Neighbor (*k*-NN), Support Vector Machines (SVM) and Random Forest (RF). The full results of accuracy, precision, recall and F-measure are shown in Table 2. Next, the results in Table 2 are analyzed based on Receiver Operating Characteristic (ROC) curve and Precision-Recall (PR) curve. Both types of curves played a fundamental role in understanding the technique of the various systems in the presence of uncertainty. These curves were used in several areas such as radiology, electrical engineering and several other arenas to education the performance of a binary forecast system as a function of a control parameter. As the control parameter, it is possible to increase the accuracy and reduce the false positive rate of the system according to the lower recall, which is the true positive rate or sensitivity (Pavlick *et al*., 2015; Ekelund, 2017).

The Area Under Curve (AUC) will be used as a summary of the model skill. The model skill will be compared against a no-skill classifier, which is the one that cannot discriminate between the classes and would predict a random class or a constant class in all cases. A model with no-skill is represented at the point (0.5, 0.5). A model with no-skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.5. Table 3 summarizes the results of AUC for both ROC and PR curves across all three algorithms.

Receiver Operating Characteristic (ROC) Curve

A Receiver Operating Characteristic (ROC) curve summarizes the trade-off between TP rate and FP rate for a predictive model using different probability thresholds. It has two dimensions where the *x*-axis indicates the False Positive (FP) rate and the *y*-axis indicates the True Positive (TP) rate (Grau *et al*., 2015). Fig. 8 shows the ROC curve for *k*-Nearest Neighbor (*k*-NN). The ROC AUC is 0.959 with no-skill AUC at 0.500.

Table 2: Comparison of results

Algorithm	Accuracy Rate	Precision Rate	Recall Rate	F- Measure
k-Nearest Neighbor (*k*-NN)	0.91	0.91	0.91	0.864
Support Vector Machines (SVM)	0.91	0.91	0.91	0.905
Random Forest (RF)	0.98	0.98	0.98	0.980

Table 3: Area Under Curve (AUC) for both ROC and PR

Algorithm	ROC AUC	PR AUC
k-Nearest Neighbor (*k*-NN)	0.959	0.960
Support Vector Machines (SVM)	0.888	0.885
Random Forest (RF)	0.981	0.983

DOI: 10.3844/jcssp.2020.1306.1318

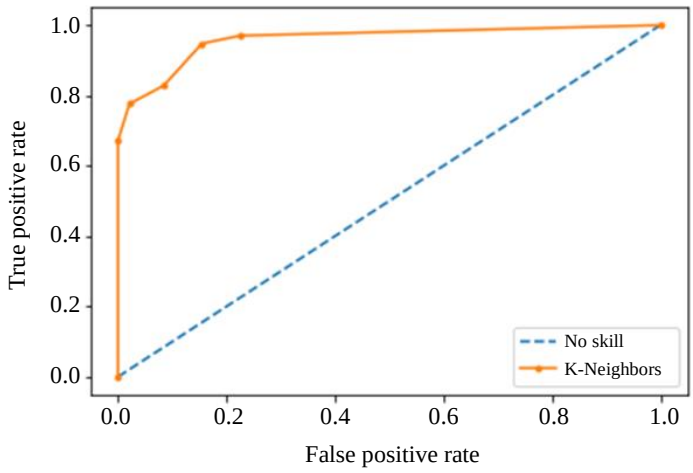

Fig. 8: ROC curve for k-Nearest Neighbor (*k*-NN)

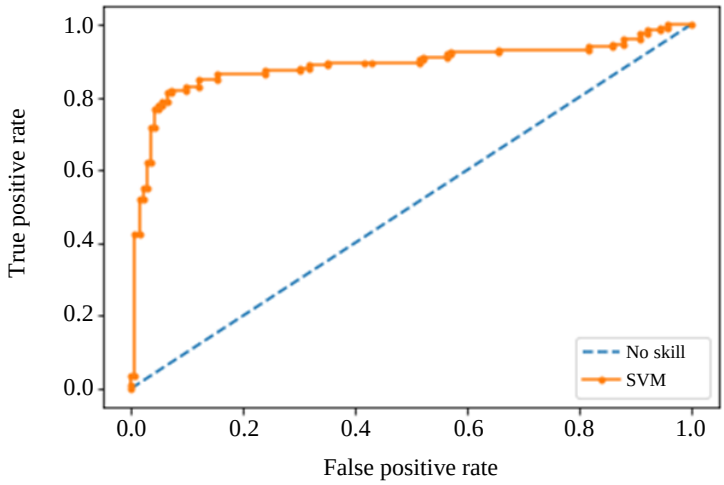

Fig. 9: ROC curve for Support Vector Machines (SVM)

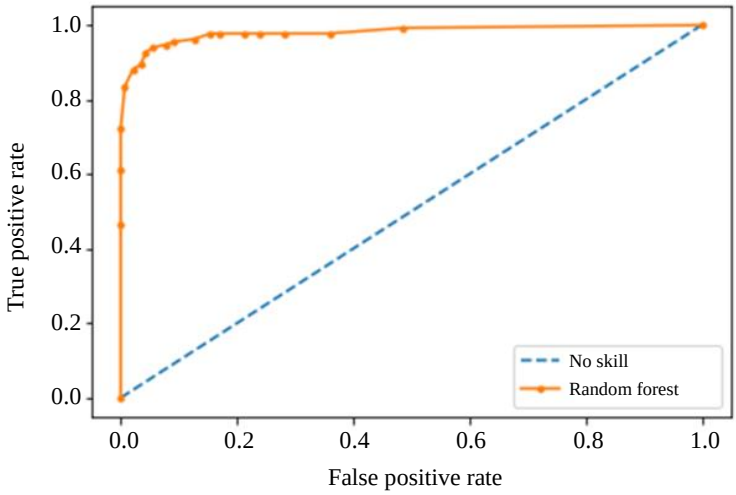

Fig. 10: ROC curve for Random Forest (RF)

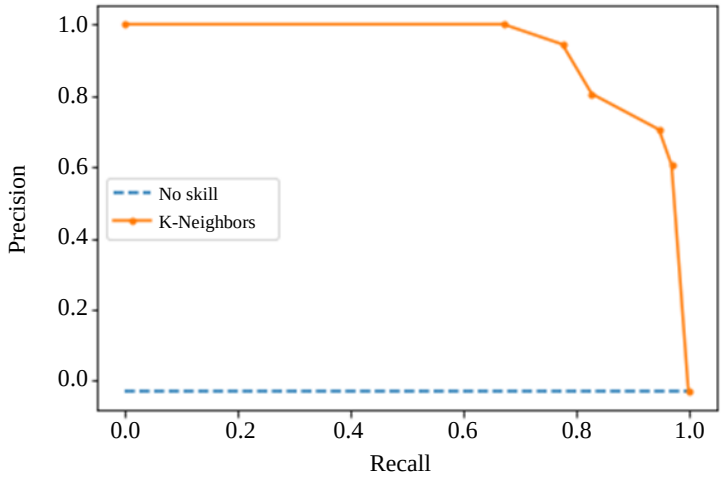

Fig. 11: PR curve for *k*-Nearest Neighbor (*k*-NN)

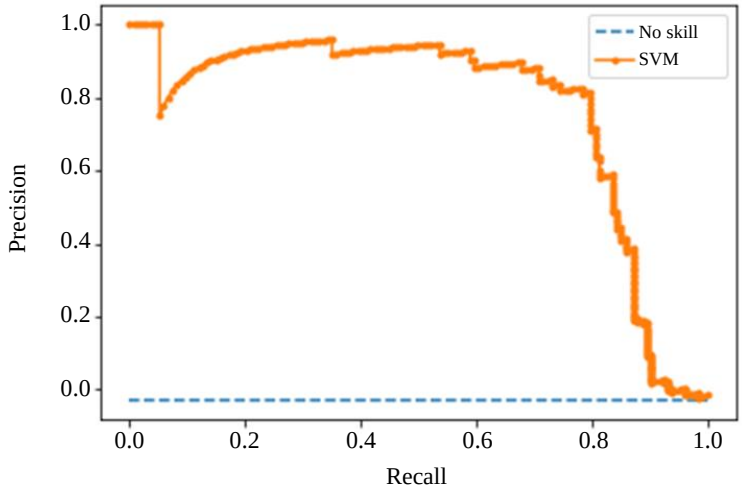

Fig. 12: PR curve for Support Vector Machines (SVM)

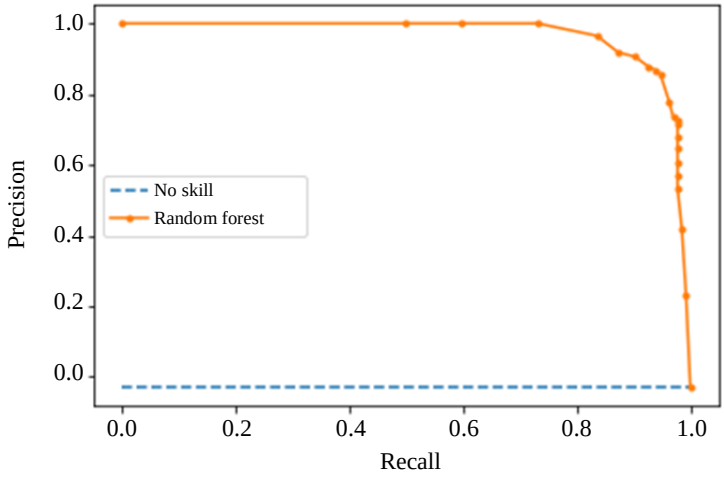

Fig. 13: PR curve for Random Forest (RF)

Fig. 9 shows the ROC curve for Support Vector Machines (SVM). The ROC AUC is 0.888 with no-skill AUC at 0.500. Meanwhile, Fig. 10 shows the ROC curve for Random Forest (RF). The ROC AUC is 0.981 with no-skill AUC at 0.500.

Precision-Recall Curve

A Precision-Recall (PR) curve summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds. PR curve is a plot of the precision in the *y*-axis and the recall in the *x*-axis for different probability thresholds. Basically, it is the plot of Recall (*x*) vs. Precision (*y*). Fig. 11 shows the PR curve for *k*-Nearest Neighbor (*k*-NN). The F-measure is 0.864 and PR AUC is 0.960. Fig. 12 shows the PR curve for Support Vector Machines (SVM). The F-measure is 0.802 and PR AUC is 0.885. Finally, Fig. 13 shows the PR curve for Random Forest (RF). The F-measure is 0.925 and PR AUC is 0.983.

Discussion

The preliminary analysis was carried out based on Area Under Curve (AUC) of two curves; Receiver Operating Characteristic (ROC) curve and Precision-Recall (PR) curve. AUC is the best measurement as AUC does not have errors in the prediction, so ideal classifiers can an ideal that classifies data into two classes, which means the model succeeded in being without any false positives. Another benefit of using the ROC and PR curves together is to find points that are close or shared to give the best evaluation of the models used in this research as shown in Table 3.

In both ROC and AUC, the threshold was used to apply to the cut-off point in probability between the positive and negative classes where the threshold is chosen by default for any classifier at 0.5 in the middle area of the outputs (0 and 1). The classifier that does not have the ability to distinguish between positive and negative class will be the diagonal line between the false rate of 0 and the true positive rate of zero (0, 0) and in the case of predicting all negative class to the false positive rate 1 or the true positive rate (1, 1) or expect all positive class. So, the line represented by the points below is the inability predictability of and there is no-skill in distinguishing between positive and negative class. So, the perfect classifier when the value between (0.0 and 1.0).

The performance of the perfect model for the malware detection for the dataset depends on the choice of the appropriate model for the dataset. In *k*-Nearest Neighbor model, it can be seen that the ROC AUC for *k*-NN model on the synthetic dataset is about 0.903, which is much better than a no-skill classifier with a score of about 0.500. In SVM, it can be seen that the ROC AUC model on the synthetic dataset is about 0.903, which is much better than a no-skill classifier with a score of about 0.500. Finally, in RF, it can be seen that the ROC AUC model on the synthetic dataset is about 0.981, which is much better than a no-skill classifier with a score of about 0.500.

The results showed that operational code (Opcode) sequence dataset generated from IoT sensors are highly useful in developing a malware detection model within the Internet of Things environment. The accuracy rates are considerably high and this indicates the possibility of developing and using machine learning methods with real-data from the Internet of things. The challenges facing the operational code sequence (Opcode) dataset is that not every sample consisting of all cipher codes in its vector feature, hence the features may have a zero value. Therefore, using word embedding technology to convert each sample into a digital sequence representation is possibly required (Puthal *et al*., 2016).

Conclusion

This paper presented a preliminary analysis of malware detection models within the scope of Internet-of-Things (IoT) applications. The dataset used is in the form of operational codes (Opcodes) sequences generated from IoT-based devices (HaddadPajouh *et al*., 2018). Three machine learning algorithms were constructed and compared, which are *k*-Nearest Neighbor (*k*-NN), Support Vector Machines (SVM) and Random Forest. The experimental results showed that RF outperformed both *k*-NN and SVM with 98% of detection accuracy as compared to 91% for both *k*-NN and SVM. These results are supported by analysis of Receiver Operating Characteristic (ROC) curve and Precision-Recall (PR) curve, which showed that the best methods used in this study is Random Forest, with highest accuracy of 0.98 and supported by ROC/PR curves 0.983.

The results from this preliminary analysis will be used as benchmark results for exploring deep learning methods with the same or similar dataset from IoT environment. It is hoped that these detection models will be embedded in the IoT application in order to secure the systems from malware attacks.

Acknowledgement

This research is supported by Universiti Putra Malaysia.

Author's Contributions

Firas Shihab Ahmed: Conceived the original idea, carried out experiments, collected the results and drafted the manuscript.

Norwati Mustapha: Supervised the project, advised on structure of manuscript, in charge of overall direction and planning.

Aida Mustapha: Conceived the original idea, worked on the manuscript.

Mohsen Kakavand: Processed the dataset, helped with the experiments, and interpreted the results.

Cik Feresa Mohd Foozy: Worked on the literature reviews and revising the manuscript.

Ethics

All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.

References

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. arXiv preprint arXiv:1707.02919.

Apostolidis-Afentoulis, V., & Lioufi, K. I. (2015). Svm classification with linear and rbf kernels. July): 0-7. http://www. academia. edu/13811676/SVM_Classification_with_Linear_and_RBF_kernels.[21]

Aburomman, A. A., & Reaz, M. B. I. (2016). A novel SVM-kNN-PSO ensemble method for intrusion detection system. Applied Soft Computing, 38, 360-372.

Azmoodeh, A., Dehghantanha, A., & Choo, K. K. R. (2018). Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning. IEEE Transactions on Sustainable Computing, 4(1), 88-95.

Bragen, S. R. (2015). Malware detection through opcode sequence analysis using machine learning (Master's thesis).

Bro, R., & Smilde, A. K. (2003). Centering and scaling in component analysis. Journal of Chemometrics, 17(1), 16-33.

Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011, October). Crowdroid: behavior-based malware detection system for android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 15-26).

Chen, J., Li, K., Tang, Z., Bilal, K., Yu, S., Weng, C., & Li, K. (2016). A parallel random forest algorithm for big data in a spark cloud computing environment. IEEE Transactions on Parallel and Distributed Systems, 28(4), 919-933.

D'Orazio, C. J., Choo, K. K. R., & Yang, L. T. (2016). Data exfiltration from Internet of Things devices: iOS devices as case studies. IEEE Internet of Things Journal, 4(2), 524-535.

Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., & Choo, K. K. R. (2020). An opcode-based technique for polymorphic Internet of Things malware detection. Concurrency and Computation: Practice and Experience, 32(6), e5173.

Davis, J., & Goadrich, M. (2006, June). The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning (pp. 233-240).

Ekelund, S. (2017). Precision-recall curves–what are they and how are they used?. acutecaretesting. org.

Farnaaz, N., & Jabbar, M. A. (2016). Random forest modeling for network intrusion detection system. Procedia Computer Science, 89(1), 213-217.

Gislason, P. O., Benediktsson, J. A., & Sveinsson, J. R. (2003). Random forests for land cover classification. Pattern Recognition Letters, 27(4), 294-300.

Goyal, M., Sahoo, I., & Geethakumari, G. (2019). HTTP Botnet Detection in IOT Devices using Network Traffic Analysis. In 2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC) (pp. 1-6). IEEE.

Grau, J., Grosse, I., & Keilwagen, J. (2015). PRROC: computing and visualizing precision-recall and receiver operating characteristic curves in R. Bioinformatics, 31(15), 2595-2597.

Gupta, V., & Mittal, M. (2018). KNN and PCA classifier with autoregressive modelling during different ECG signal interpretation. Procedia Computer Science, 125, 18-24.

HaddadPajouh, H., Dehghantanha, A., Khayami, R., & Choo, K. K. R. (2018). A deep recurrent neural network based approach for internet of things malware threat hunting. Future Generation Computer Systems, 85, 88-96.

Ham, H. S., Kim, H. H., Kim, M. S., & Choi, M. J. (2014). Linear SVM-based android malware detection for reliable IoT services. Journal of Applied Mathematics, 2014.

Hasan, M., Islam, M. M., Zarif, M. I. I., & Hashem, M. M. A. (2019). Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. Internet of Things, 7, 100059.

Huang, S., Cai, N., Pacheco, P. P., Narrandes, S., Wang, Y., & Xu, W. (2018). Applications of support vector machine (SVM) learning in cancer genomics. Cancer Genomics-Proteomics, 15(1), 41-51.

Kakavand, M., Dabbagh, M., & Dehghantanha, A. (2018, November). Application of machine learning algorithms for Android malware detection. In Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems (pp. 32-36).

Kakavand, M., Mustapha, N., Mustapha, A., & Abdullah, M. T. (2016). Effective Dimensionality Reduction of Payload-Based Anomaly Detection in TMAD Model for HTTP Payload. TIIS, 10(8), 3884-3910.

Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. Computer, 50(7), 80-84.

Kumar, A., & Lim, T. J. (2019, April). EDIMA: Early detection of IoT malware network activity using machine learning techniques. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (pp. 289-294). IEEE.

Kumar, R., Zhang, X., Wang, W., Khan, R. U., Kumar, J., & Sharif, A. (2019). A multimodal malware detection technique for Android IoT devices using various features. IEEE Access, 7, 64411-64430.

Li, W., Tug, S., Meng, W., & Wang, Y. (2019). Designing collaborative blockchained signature-based intrusion detection in IoT environments. Future Generation Computer Systems, 96, 481-489.

Lu, J., Yu, C. S., Liu, C., & Yao, J. E. (2003). Technology acceptance model for wireless Internet. Internet research.

Mahindru, A., & Singh, P. (2017, February). Dynamic permissions based android malware detection using machine learning techniques. In Proceedings of the 10th innovations in software engineering conference (pp. 202-210).

Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N. O., Guarnizo, J. D., & Elovici, Y. (2017). Detection of unauthorized iot devices using machine learning techniques. arXiv preprint arXiv:1709.04647.

Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). Machine learning aided Android malware classification. Computers & Electrical Engineering, 61, 266-274.

Mosenia, A., & Jha, N. K. (2016). A comprehensive study of security of internet-of-things. IEEE Transactions on Emerging Topics in Computing, 5(4), 586-602.

Nakhodchi, S., Upadhyay, A., & Dehghantanha, A. (2020). A comparison between different machine learning models for IoT malware detection. In Security of Cyber-Physical Systems (pp. 195-202). Springer, Cham.

Nikam, S. S. (2015). A comparative study of classification techniques in data mining algorithms. Oriental journal of computer science & technology, 8(1), 13-19.

Noi, P. T., & Kappas, M. (2018). Comparison of random forest, k-nearest neighbor and support vector machine classifiers for land cover classification using Sentinel-2 imagery. Sensors, 18(1), 18.

Patel, S., Park, H., Bonato, P., Chan, L., & Rodgers, M. (2012). A review of wearable sensors and systems with application in rehabilitation. Journal of neuroengineering and rehabilitation, 9(1), 1-17.

Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2015, July). PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings and style classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 425-430).

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

Puthal, D., Nepal, S., Ranjan, R., & Chen, J. (2016). Threats to networking cloud and edge datacenters in the Internet of Things. IEEE Cloud Computing, 3(3), 64-71.

Rehman, Z. U., Khan, S. N., Muhammad, K., Lee, J. W., Lv, Z., Baik, S. W., ... & Mehmood, I. (2018). Machine learning-assisted signature and heuristic-based detection of malwares in Android devices. Computers & Electrical Engineering, 69, 828-841.

Sabharwal, A., & Sedghi, H. (2017). How Good Are My Predictions? Efficiently Approximating Precision-Recall Curves for Massive Datasets. In UAI.

Sethi, K., Chaudhary, S. K., Tripathy, B. K., & Bera, P. (2017, October). A novel malware analysis for malware detection and classification using machine learning algorithms. In Proceedings of the 10th International Conference on Security of Information and Networks (pp. 107-113).

Tankard, C. (2015). The security issues of the Internet of Things. Computer Fraud & Security, 2015(9), 11-14.

Tien, C. W., Chen, S. W., Ban, T., & Kuo, S. Y. (2020). Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features. Digital Threats: Research and Practice, 1(1), 1-19.

Varoquaux, G. (2018). Cross-validation failure: small sample sizes lead to large error bars. Neuroimage, 180, 68-77.

Wang, F., Zhen, Z., Wang, B., & Mi, Z. (2018a). Comparative study on KNN and SVM based weather classification models for day ahead short term solar PV power forecasting. Applied Sciences, 8(1), 28.

Wang, H., Zheng, B., Yoon, S. W., & Ko, H. S. (2018b). A support vector machine-based ensemble algorithm for breast cancer diagnosis. European Journal of Operational Research, 267(2), 687-699.

Watson, S., & Dehghantanha, A. (2016). Digital forensics: the missing piece of the Internet of Things promise. Computer Fraud & Security, 2016(6), 5-8.

Wazid, M., Das, A. K., Rodrigues, J. J., Shetty, S., & Park, Y. (2019). IoMT malware detection approaches: Analysis and research challenges. IEEE Access, 7, 182459-182476.

Xiaofeng, L., Xiao, Z., Fangshuo, J., Shengwei, Y., & Jing, S. (2018). ASSCA: API based sequence and statistics features combined malware detection architecture. Procedia Computer Science, 129, 248-256.

Yang, K., Hicks, M., Dong, Q., Austin, T., & Sylvester, D. (2016, May). A2: Analog malicious hardware. In 2016 IEEE symposium on security and privacy (SP) (pp. 18-37). IEEE.

Zhang, Y. D., Yang, Z. J., Lu, H. M., Zhou, X. X., Phillips, P., Liu, Q. M., & Wang, S. H. (2016). Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine and stratified cross validation. IEEE Access, 4, 8375-8385.