Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 14, Issue. 4, April 2025, pg.68 – 82

PERFORMANCE ANALYSIS AND COMPARISON OF MALWARE DETECTION FRAMEWORK USING PCA BASED ANN

Dr. Khyati Rami[1]; Dr. Vinod Desai[2]; Ms. Roshni Patel[3]

[1]SAL Institute of Technology & Engineering Research, Ahmedabad, Gujarat Technological University, India
[2]Department of Computer Science, Sardar Patel University, Vallabh Vidhyanagar Gujarat, India
[3]SAL College of Engineering, Ahmedabad, Gujarat Technological University, India

[1] drkhyatirami@gmail.com; [2] vinoddesai123@gmail.com; [3] vinayroshni@gmail.com

DOI: https://doi.org/10.47760/ijcsmc.2025.v14i04.006

Abstract: The Malicious programs, including malware, can pose a threat to computer systems, and the internet is a common medium for spreading such threats. To detect malware, two basic approaches are commonly used: signature-based detection and heuristic-based detection. To improve the accuracy and effectiveness of malware detection, researchers have combined heuristic methods with machine learning techniques. Machine learning algorithms, such as artificial neural networks (ANN), can be trained on large datasets of known malware samples to learn patterns and characteristics that distinguish malware from benign programs. By using machine learning, the detection system can better adapt to new and evolving malware threats. A malware detection framework based on principal component analysis (PCA) and ANN [1] implemented in MATLAB GUI, PCA is a dimensionality reduction technique that can help extract relevant features from the dataset, making it easier for the ANN classifier to differentiate between mal-ware and non-malware samples. In this paper performance analysis and comparison of Malware detection framework is elaborated deeply. This paper presents the performance evaluation and comparison of malware detection frame-work. Finding the right performance indicators is essential for evaluating the detection performance properly. The confusion matrix was used to build and use the following metrics for classifier evaluation. With a processing time of less than 0.2 seconds, the ANN classifier utilized for malware detection displays computational efficiency. In contrast, the computation time for the NB classifier (probably a Naive Bayes classifier) is 0.82 seconds. The computationally efficient ANN classifier for malware detection has a processing time of less than 0.2 seconds. In comparison, the NB classifier (presumably referring to a Naive Bayes classifier) takes 0.82 seconds for computation. This suggests that the ANN-based approach is faster.

Keywords: Android Malware, machine learning, performance evaluation, comparison, Principle component analysis

I. Introduction

Malware and other computer threats are made with the purpose of damaging sensitive data and hurting users. Mal- ware can in fact constitute a serious threat to the security and integrity of computer systems. Malware comprises viruses, worms, Trojan horses, ransom ware, and other malicious software. A rising issue in recent years has been the spread of malware. To avoid detection and take advantage of flaws in computer systems and networks, cybercriminals constantly create new methods and variants. These assaults may be carried out for a variety of reasons, such as monetary gain, the theft of private information, the interruption of services, or just general mayhem. Security incidents have increased in frequency as a result of the growing threat environment. The risk of data breaches, identity theft, financial loss, and other negative outcomes is increasing for both organizations and individuals. These occurrences may have detrimental effects on people, governments, and corporations alike. Strong security measures must be put in place in order to reduce the danger posed by malware and other computer threats. This entails performing routine software and operating system updates, using secure passwords that are both strong and unique, using dependable antivirus and anti-malware software, and adopting safe browsing practices including avoiding dubious websites and not opening email attachments from unknown senders. It's important to remember that cyber security is a field that's continuously changing, and both attackers and defenders are constantly changing how they operate. To effectively defend against harmful activity, it is crucial to keep up with the most recent dangers and security procedures.[2,3]. Malware has the power to spread like a chain reaction, which is risky because there is no centralized control, making it difficult to detect. According to studies, malwares pose a serious threat to computer security[4]. Malware is intentionally designed to be difficult to detect and regularly changes its strategies to transform into undetected harmful code. The process starts with encryption and en- compasses the creation of viruses that possess the characteristics of being oligomorphic, polymorphic, and metamorphic. Research suggests that current detection methods are inadequate. The use of artificial intelligence, machine learning, and data mining has the potential to greatly enhance the detection of malware [5]. Polymorphic malware, which is characterized by its ability to alter its signature, renders signature-based detection useful only for recognized threats but insufficient for novel ones. Heuristic approaches have the capability to identify both familiar and unfamiliar malware, but they are susceptible to a significant number of false positives and false negatives. This emphasizes the necessity for more accurate techniques. The increase in polymorphic malware has necessitated the integration of heuristic methods with machine learning to achieve enhanced precision in detection. Ongoing research and improved solutions are needed in light of these challenges. A research was conducted using an Artificial Neural Network (ANN) model to assess its effectiveness in detecting malware. The model was tested using the CSDMC2019 API dataset and was integrated into a mobile operating system. Additionally, the study involved moving files from the mobile operating system to a desktop operating system [6]. The study conducts a comparative analysis of several models, such as ANN and DS, to assess their effectiveness in detecting malware. The analysis is structured into sections that include an introduction, literature review, performance analysis, results, and conclusions.

II. Literature Review

Table 1 Summery of Literature Review

Narayanan, Annamalai, MahinthanChandramohan, Lihui Chen, and Yang Liu. "A multi-view context-aware approach to Android malware detection and malicious code localization." Empirical Software Engineering (2018): 1-53	MKLDROID is a framework that utilises context- awareness and multiview techniques to identify mal-ware and locate harmful code. MKLDROID use static analysis and graph kernels in its identification and lo-calization workflow to derive five complementarysets of semantic representations of programs. [7].
Nachum, Shay, Assaf Schuster, and OpherEtzion. "Detection in the Dark–Exploiting XSS Vulnerability in C&C Panels to Detect Malwares." In International Sympo-sium on Cyber Security Cryptography and Machine Learning, pp. 227-242. Springer, Cham, 2018.	The researchers developed an innovative method for identifying and a demonstration of a counter- measure against Cross-Site Scripting (XSS) on the command and control (C&C) web interfaces utilisedby malicious software [8].
Mishra, Preeti, Vijay Varadharajan, Emmanuel Pilli, and UdayaTupakula. "VMGuard: A VMI-based Security Architecture for Intrusion Detection in Cloud Environment." IEEE Transactions on Cloud Computing (2018).	The authors suggested an adaptive and efficient se-curity architecture design utilising Virtual Machine Introspection (VMI) for precise monitoring of virtualmachines. This design aims to identify both known threats and their variations at a fine-grained level.[9]
Kumara, Ajay, and C. D. Jaidhar. "Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at VMM." Future Generation Computer Systems	Their proposal involved implementing an advancedAutomated Multilevel Malware Detection System (AMMDS) that makes use of Virtual Machine Monitor (VMM)-based guest assistance. This system in- tegrates Virtual Machine Introspection (VMI)

79 (2018): 431-446.	and Memory Forensic Analysis (MFA) techniques to de- tect initial indications of malware behavior. Itachieves this by revealing hidden processes that aresecretly running on a running guest operating sys- tem.[10]
Xiao, Xi, Shaofeng Zhang, Francesco Mercaldo, Guangwu Hu, and Arun Kumar Sangaiah. "Android malware detection based on system call sequences and LSTM." Multimedia Tools and Applications 78, no. 4 (2019): 3979-3999	A approach for dynamic analysis utilising the Long Short-Term Memory (LSTM) language model hasbeen presented. This method does not need the presence of the application's source code. This technique depends on describing the dynamic behaviours of an application by analysing the sequences of system calls it produces. The system calls in these sequences, which resemble normal language, provide semantic information that may be utilised to distinguish malware from reliable applications. [11]
Ribeiro, José, Firooz B. Saghezchi, Georgios Mantas, Jonathan Rodriguez, Simon J. Shepherd, and Raed A. Abd-Alhameed. "An Autonomous Host-Based Intrusion Detection System for Android Mobile Devices." Mobile Networks and Applications (2019): 1-9.	They came up with a new Host-based IDS (HIDS) that uses statistical and semi-supervised machine learning techniques. There were two good things about their planned IDS. First, it was completely self-contained and ran on the phone without connecting to a computer. Second, as tuning examples,it only needed good ones, though a few bad ones might have been okay too. [12]
Yen, Yao-Saint, and Hung-Min Sun. "An Android mutation malware detection based on deep learning using visualization of importance from codes." Microelectronics Reliability 93 (2019): 109-114.	In this study, a deep learning-based method forfinding malware on Android is demonstrated. The process involves extracting code out of APK files, figuring out how important each word is, and then turning this knowledge into a digital form. [13]
Xu, Yang, Guojun Wang, JuRen, and Yaoxue Zhang. "An adaptive and configurable protection framework against android privilege escalation threats." Future Generation Computer Systems 92 (2019): 210-224.	This study presents a flexible security design for An-droid that aims to stop confused deputy attacks from taking advantage of permission leak vulnerabilities in third-party apps. The framework works bykeeping track of the changing states of apps and controlling who can see what based on rules and abilities. This approach tries to reduce the number of risky interactions between applications. This makes application security more flexible, respon- sive, and varied. [14]
Le, Van-Giap, Huu-Tung Nguyen, Duy-Phuc Pham, Van-On Phung, and Ngoc-Hoa Nguyen. "GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities." In Transactions on Computational Collective Intelligence XXXII, pp. 184-208. Springer, Berlin, Heidelberg, 2019.	In order to identify SQL injection, Cross-site Scripting, and heightened vulnerability detection, they presented E-THAPS, a unique detection technique. The method used to identify malicious web shells inGuruWS relies on the utilisation of taint analysis andpattern matching techniques. [15]
Saracino, Andrea, Daniele Sgandurra, GianlucaDini, and Fabio Martinelli. "Madam: Effective and efficient behavior-based android malware detection and prevention." IEEE Transactions on Dependable and Secure Computing (2016).	This study categorises malware into discrete behavioural classifications, each defined by a precise andrestricted set of dangerous activities. These acts, ormisbehaviours, can be detected by monitoring the characteristics linked to several recent iterations ofthe Android operating system. [16]
Guo, Yonghe, Chee-Wooi Ten, Shiyan Hu, and Wayne W. Weaver. "Preventive maintenance for advanced metering infrastructure against malware propagation." IEEE Transactions on Smart Grid 7, no. 3 (2016): 1314-1328	In this paper they proposed a method which deter-mines the great inspection techniques up to date tally updated at the remark from the existing anom-aly eyecup to daters deployed within the net- work.[17]
Wen, Hui, Weidong Zhang, Yan Hu, Qing Hu, Hongsong Zhu, and Limin Sun. "Lightweight IoT Malware Visualiza-tion Analysis via Two-Bits Networks." In International Conference on Wireless Algorithms, Systems, and Appli-cations, pp. 613-621. Springer, Cham, 2019.	The authors of the research presented a technique that starts by converting deconstructed malware bi-naries into a series of grayscale pictures with a sin- gle channel. Subsequently, they utilised a Two-Bits Convolutional Neural Network (TBN) specifically de-veloped for the purpose of detecting and classifyingIoT malware families. This technique is remarkable for its capacity to represent the weights of networkedges using just two bits. [18]
Rastogi, Vaibhav, Yan Chen, and Xuxian Jiang. "Droidchameleon: evaluating android anti-malware against transformation attacks." In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 329-334. ACM, 2013.	They purposely assess hostile to malware items forAndroid with respect to their protection against different change methods in known malware.[19]
Seo, Seung-Hyun, Aditi Gupta, Asmaa Mohamed Sallam, Elisa Bertino, and KangbinYim. "Detecting mobile malware threats to homeland security through static analysis." Jtheirnal of Network and Computer Applications 38 (2014): 43-53.	They will demonstrate how portable assault situations are relevant to Homeland Security and presentgenuine Risk. Second, they propose an investigationapparatus of Android applications, DroidAnalyzer that recognizes potential vulnerabilities.[20]

III. MALWARE DETECTION FRAMEWORK BY PCA BASED ANN

To find malware on mobile devices, a smart host-based method was created and tested by the researcher. K. Rami & V. Desai[1]. They had created a framework which was made to be light on the system, using the least amount of CPU, memory, and battery power possible. Messaging, phone calls, and application-related functions fall under the application Framework category and were obtained through the framework's APIs; keyboard, touch screen, scheduling, and memory-related capabilities fall under the Linux Kernel category.

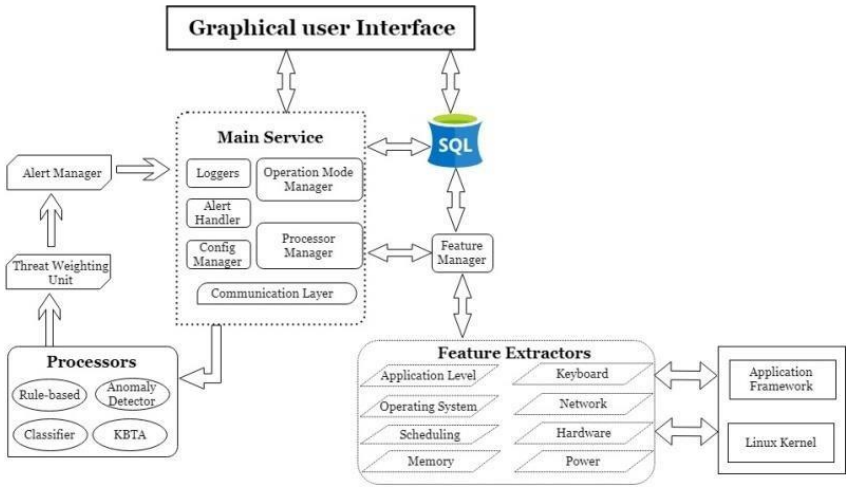

Fig. 1 Malware Detection Framework [1]

The objective of improving the accuracy of malware detection for a wide range of file samples prompted the evaluation of several machine learning classifiers. This project aimed to determine the optimal classifier for accurately detecting mobile malware. The classifiers that were assessed include the Artificial Neural Network (ANN), Bayes network, decision tree (DT) utilizing the J48 method, K-nearest neighbour (KNN), and support vector machine (SVM). The analysis incorporated a sample size of 1000, encompassing 49 unique families and 1,260 instances of Android malware from the MalGenome project. The machine learning approach was organized into three distinct phases: data collection, which entailed the recording of network traffic; feature selection and extraction; and the utilization of machine learning classifiers [1].

IV. PERFORMANCE ANALYSIS & COMPARISON

In this part, we assess the efficacy of our recently devised malware detection technique by comparing it to many established systems [1] . Choosing the appropriate performance measures is essential for obtaining an accurate evaluation. The metrics are obtained from the confusion matrix and are employed to assess the performance of classifiers. True Positives (TP) refer to the exact classification of normal data, whereas True Negatives (TN) indicate the precise identification of malware samples. On the other hand, False Positives (FP) refer to regular samples that are mistakenly identified as malware, while False Negatives (FN) are instances of malware that are inaccurately classed as normal. <text>The assessment metrics consist of True Positive Rate (TPR), False Positive Rate (FPR), Positive Predictive Value (PPV), Negative Predictive Value (NPV), F-Measure, and Accuracy. The TPR, which represents the accuracy of malware predictions, and the FPR, which measures the rate of mislabeling normal data as malware, are highly important. Precision, also known as Positive Predictive Value (PPV), represents the ratio of relevant out- comes, whereas Recall quantifies the ability to identify the majority of relevant findings. The F-Measure is a metric that combines accuracy and recall into a single score. A score of 1.000 represents the highest level of performance.

Table 2 Performance comparison with 25% features

	With 25% Features									
	ACC	**TPR**	**FPR**	**F M**	**MCC**	**MK**	**BM**	**FOR**	**PPV**	**NPV**
ANN	0.997	1	0.005	0.997	0.994	0.994	0.995	0	0.994	1
SVM	0.851	0.83	0.123	0.856	0.705	0.703	0.706	0.180	0.884	0.819
KNN	0.587	1	0.451	0.299	0.310	0.175	0.548	0	0.175	1
NB	0.826	0.818	0.164	0.828	0.653	0.653	0.653	0.185	0.839	0.814

DT	0.859	0.842	0.121	0.862	0.719	0.718	0.720	0.165	0.884	0.834

Table 3 Performance comparison with 50% features

	With 50% Features									
	ACC	**TPR**	**FPR**	**FM**	**MCC**	**MK**	**BM**	**FOR**	**PPV**	**NPV**
ANN	0.989	0.9850	0.005	0.99	0.979	0.979899	0.9799	0.015	0.994	0.984
SVM	0.914	0.9230	0.093	0.913	0.829	0.829146	0.8294	0.075	0.904	0.924
KNN	0.643	1	0.416	0.445	0.408	0.286432	0.5835	0	0.286	1
NB	0.701	0.9651	0.371	0.582	0.488	0.40201	0.5933	0.015	0.417	0.985
DT	0.904	0.8815	0.069	0.907	0.810	0.809045	0.8119	0.125	0.934	0.874

Table 4 Performance comparison with 75% features

	With 75% Features									
	ACC	**TPR**	**FPR**	**FM**	**MCC**	**MK**	**BM**	**FOR**	**PPV**	**NPV**
ANN	0.020	0.980	0	0.990	0.980	0.979	0.980	0.0201	1	0.9798
SVM	0.020	0.979	0.025	0.977	0.954	0.954	0.954	0.0201	0.9748	0.9798
KNN	0	1	0.16386	0.891	0.819	0.804	0.836	0	0.804	1
NB	0.030	0.950	0.30324	0.718	0.595	0.547	0.647	0.0301	0.5778	0.9698
DT	0.055	0.947	0.00529	0.970	0.940	0.939	0.942	0.0552	0.9949	0.9447

Table 5 Performance comparison with 100% features

	With 100% Features									
	ACC	**TPR**	**FPR**	**FM**	**MC**	**MK**	**BM**	**FOR**	**PPV**	**NPV**
ANN	0.997	0.9	0.005	0.987	0.974	0.974	0.9750	0.0201	0.9949	0.9798
SVM	0.851	0.975	0.010	0.982	0.96	0.964	0.9650	0.0251	0.989	0.974
KNN	0.587	1	0.082	0.952	0.913	0.9095	0.9170	0	0.909	1
NB	0.826	0.9	0.094	0.909	0.817	0.819	00.8191	0.085	0.904	0.9145
DT	0.859	0.965	0.02	0.97	0.939	0.9396	0.9397	0.0351	0.974	0.964

Table 2 to 5 shows the performance comparison of our proposed ANN classifier with existing classifiers like SVM, KNN, NB and DT classifier in terms of accuracy, PPV, NPV, BM, MCC, MK, TPR, FPR, FOR and F measure. We have evaluated the performance of above mentioned classifiers with 25% features, 50% features, 75% features and 100%features. Figure 2 to 5 shows the performance graph related to table 2 to 5.

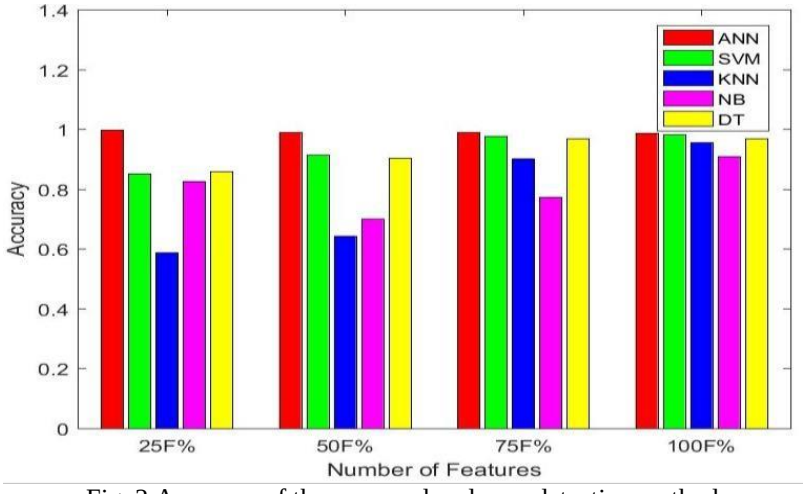

Fig. 2 Accuracy of the proposed malware detection method

Figure 2 shows the comparison of accuracy between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. It also presents the accuracy variation created by PCA as 25% features, 50% features, 75% features and 100% features. The proposed system [1] shows maximum of 99% accuracy and a stable performance over other systems in all the four scenarios. Highest accuracy represents the effectiveness of the proposed system

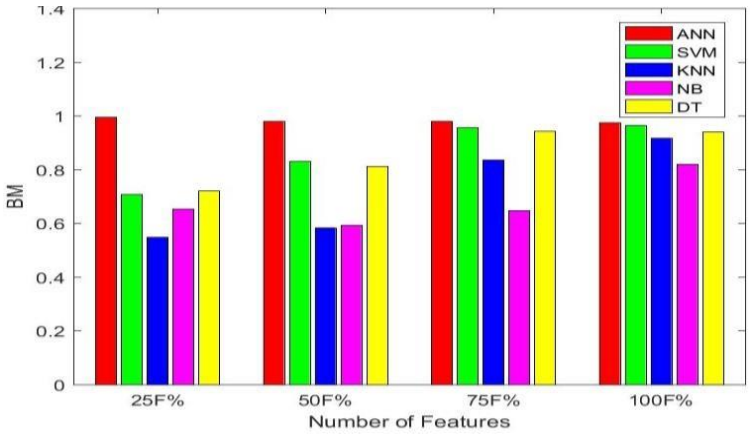

Fig. 3 Bookmaker Informedness of the proposed malware detection method

Figure 3 shows the comparison of bookmaker informedness between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. It also presents the accuracy variation created by PCA as 25% features, 50% features, 75% features and 100% features. The proposed system shows maximum of 0.98 BM which is higher than all the other existing systems.

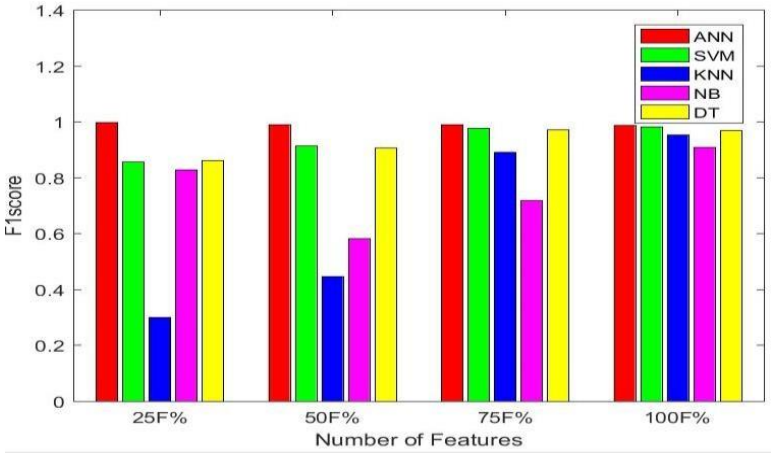

Fig. 4 F measure of the proposed malware detection method

Figure 4 shows the comparison of F measure between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. It also presents the accuracy variation created by PCA as 25% features, 50% features, 75% features and 100% features. From figure 4 it is observed that the proposed system can perform well even without PCAthan the existing system.

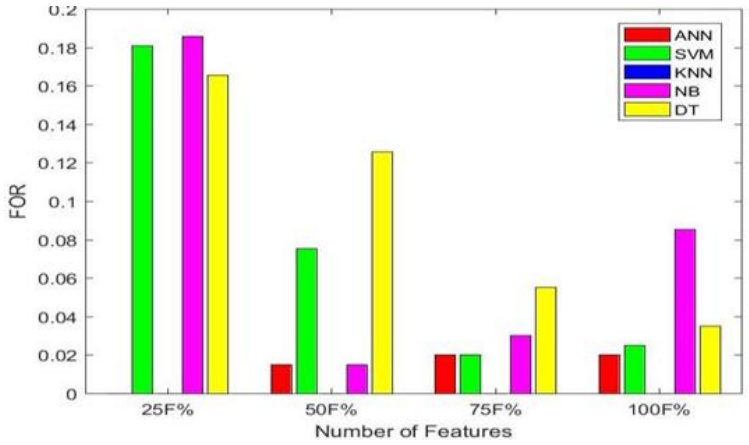

Fig. 5 False Omission Rate of the proposed malware detection method

Figure 5 shows the comparison of FOR between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. FOR should be minimum as possible for an efficient system it is clear from figure 5.

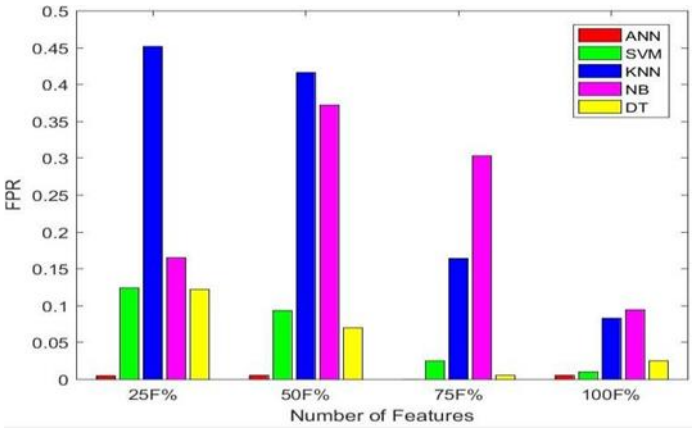

Fig. 3 False Positive Rate of the proposed malware detection method

Figure 6 shows the comparison of FPR between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. FPR should be minimum as possible for an efficient system. Figure 6 show that the proposedsystem serves its purpose.

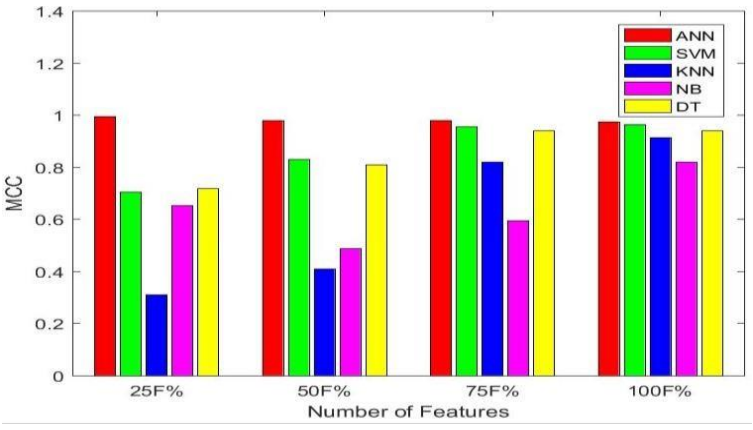

Fig. 4 Matthews Correlation coefficient of the proposed malware detection method

Figure 7 shows the comparison of Matthews Correlation coefficient between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. When compared to other systems the proposed one shows nearlycent% results as maximum. This proves the superiority over the others.

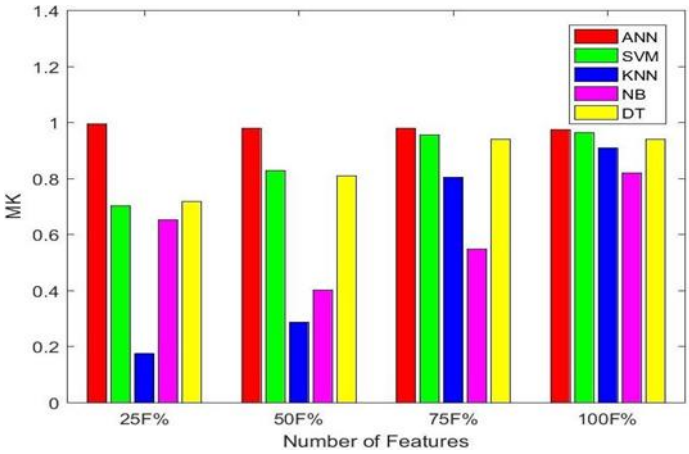

Fig. 5 Markedness of the proposed malware detection method

Figure 8 shows the comparison of markedness between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. Maximum of markedness shows minimum misclassification rate. In that case the proposed system proves its significance via figure 8.

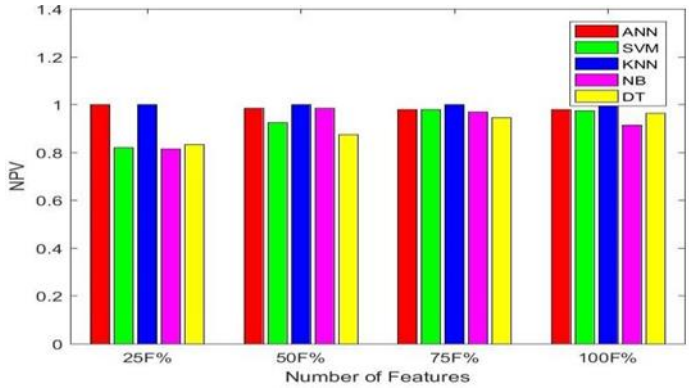

Fig. 6 Negative predictive Rate of the proposed malware detection method

Figure 9 shows the comparison of NPR between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. The proposed system has to classify both cases of positive and negative conditions. So that NPR should be maximum as possible. Figure 9 shows that the proposed system performance is moderate over other systems in terms of NPR

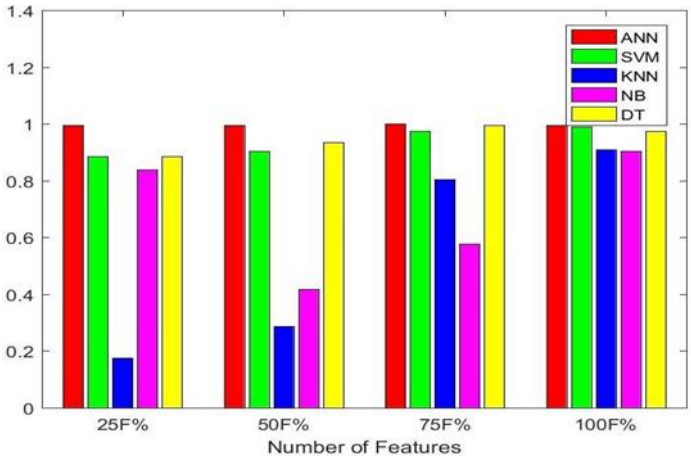

Fig. 7 True Positive Rate of the proposed malware detection method

Figure 10 shows the comparison of TPR between the proposed system and with some existing classifiers namely SVM, KNN, NB and DT. As like NPR, FPR also should be maximum as possible.

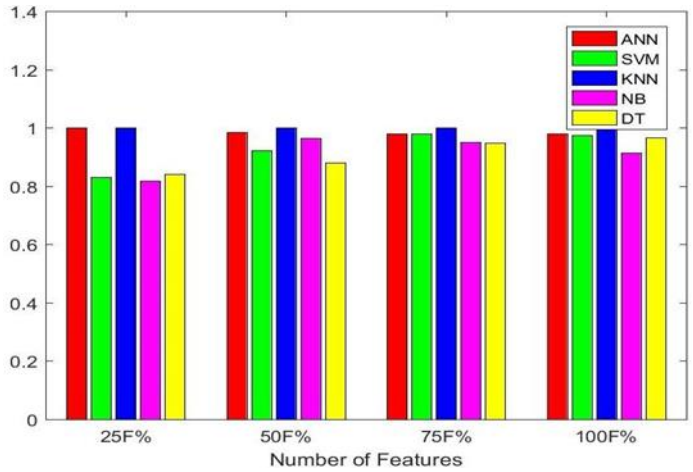

Fig. 8 Positive Predictivity Rate of the proposed malware detection method

From figure 1 to 11 it is evident that the proposed ANN classifier outperforms the other classifiers in all the above scenarios. The proposed ANN classifier shows a stable performance with 25%, 50%, 75% and 100 % features.

In addition to that we use in-built feature selection process using PCA before detecting the malwares. The addition of PCA reduces the computational time of the overall detection process also reducing the computational complexity. Figure 12 to figure 21 shows the comparison of different parameters with and without PCA.

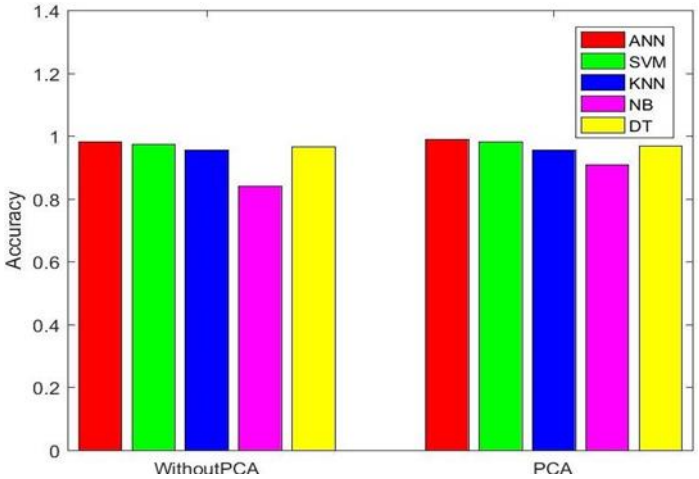

Fig. 9 Accuracy comparison with and without PCA

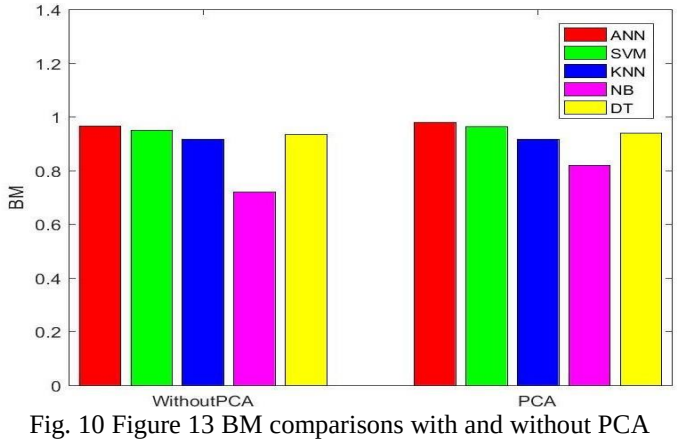

Fig. 10 Figure 13 BM comparisons with and without PCA

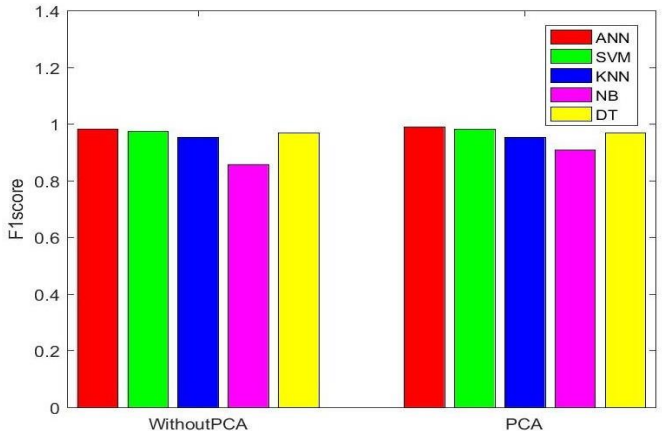

Fig. 11 F measure comparison with and without PCA

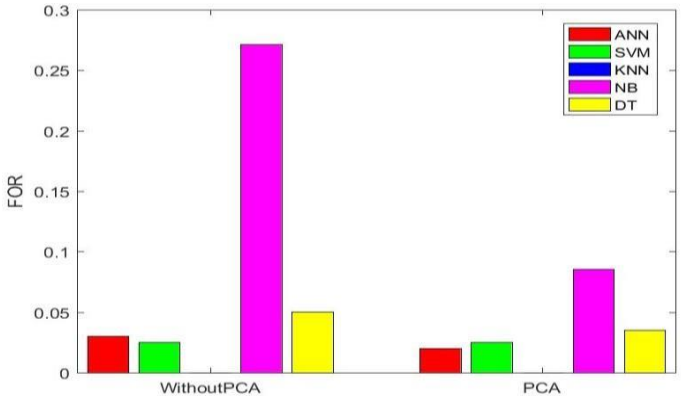

Fig. 12 FOR comparison with and without PCA

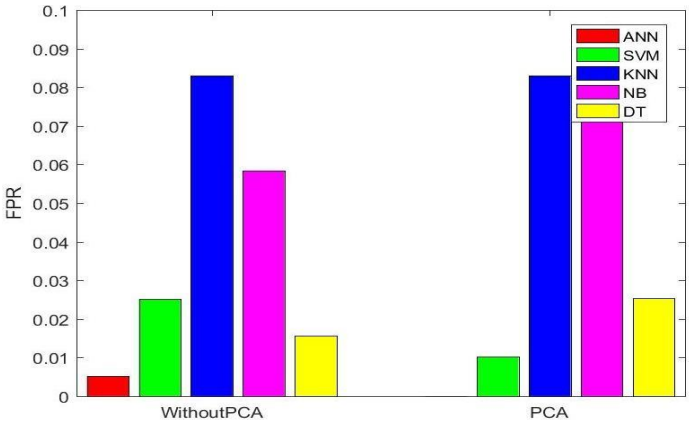

Fig. 13 FPR comparison with and without PCA

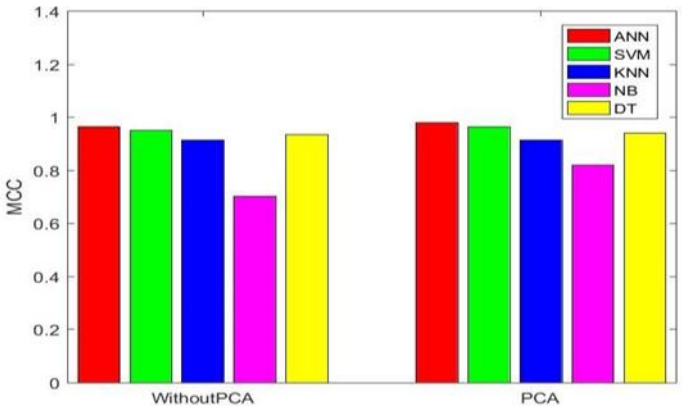

Fig. 14 MCC comparison with and without PCA

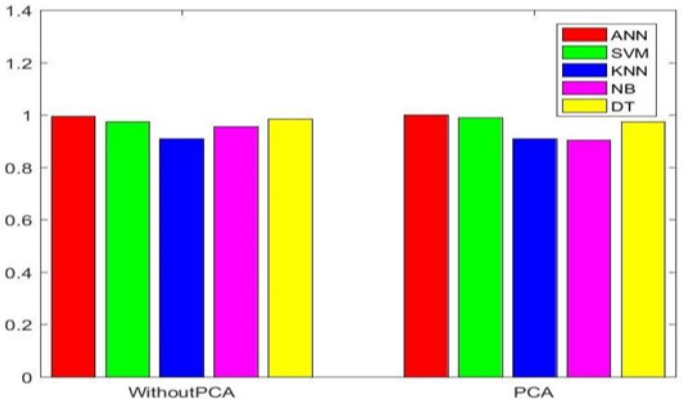

Fig. 15 PPV comparison with and without PCA

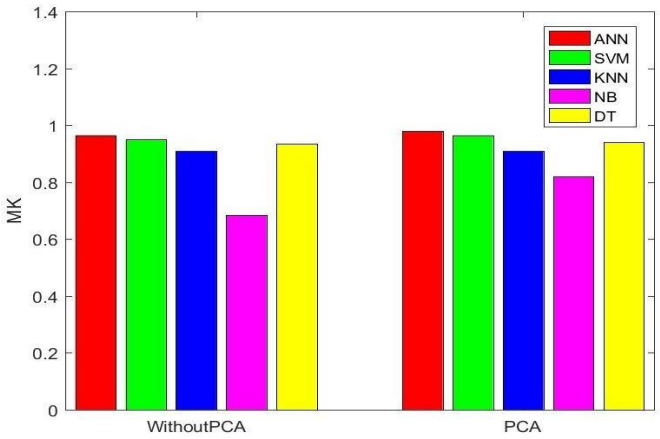

Fig. 16 MK comparison with and without PCA

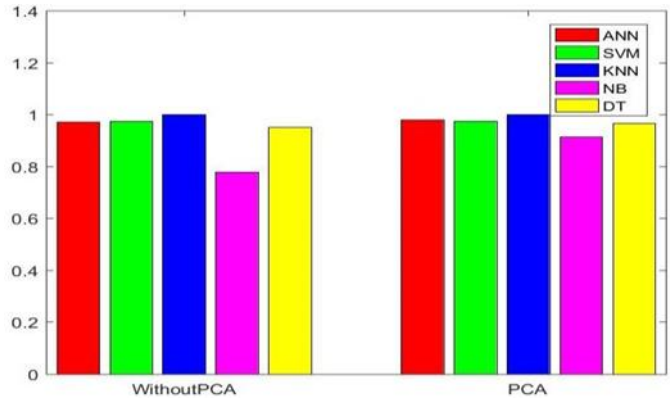

Fig. 17 TPR comparison with and without PCA

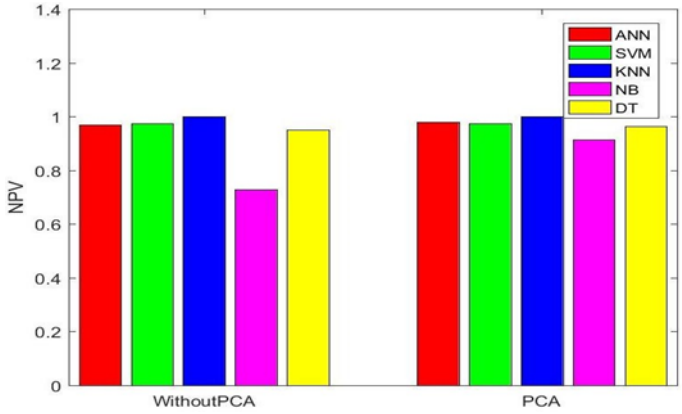

Fig. 18 NPV comparison with and without PCA

Figure 12 to 21 once again proves the superiority of ANN classifier over other classifiers like SVM, NB, KNN and DT. The performance of ANN classifier shows slight variations when it is combined with and without PCA and comparatively when ANN used with PCA performs better.

V. PERFORMANCE ANALYSIS

In addition to the above performance measures we analyze the performance of the proposed malware detectionsystem using ROC curve, computational time and computational complexity.

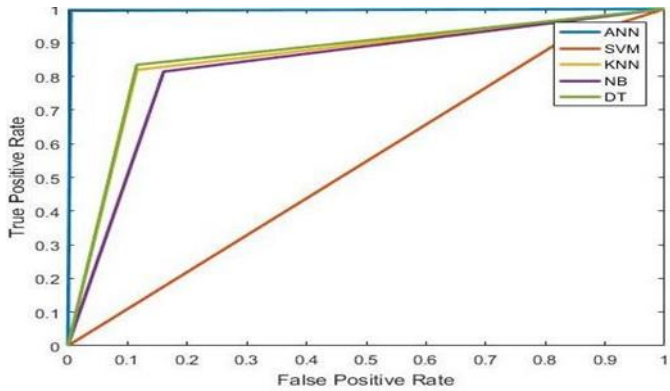

Fig. 19 ROC curve comparison with 25% F

Figure 22 shows the variation TPR with the increase of FPR for the proposed classifiers and existing classifiers in the presence of 25% reduced features.

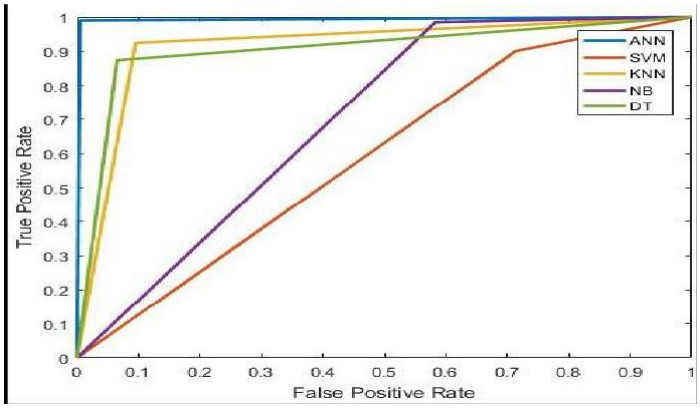

Fig. 20 ROC curve comparison with 50% F

Figure 23 shows the variation TPR with the increase of FPR for the proposed classifiers and existing classifiers in the presence of 50% reduced features

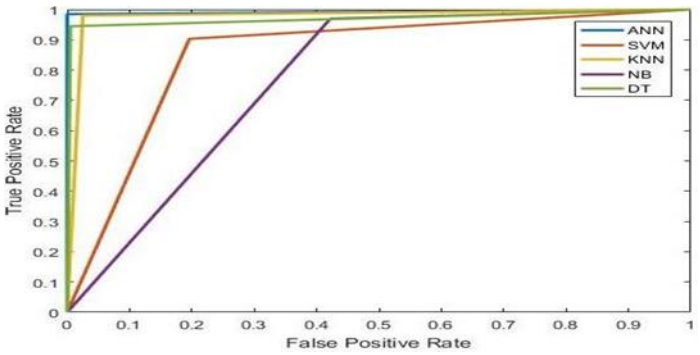

Fig. 21 ROC curve comparison with 75% F

Figure 24 shows the variation TPR with the increase of FPR for the proposed classifiers and existing classifiers inthe presence of 75% reduced features

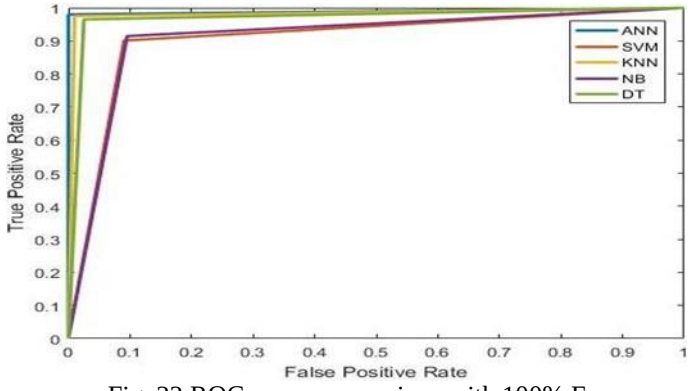

Fig. 22 ROC curve comparison with 100% F

Figure 25 shows the variation TPR with the increase of FPR for the proposed classifiers and existing classifiers in the presence of 100% features.

Figure 22 to 25 shows the relationship between true positive and false positive rate for each classifiers. It is observed that the true positive rate and false positive rate are dependent to each other. With the increase in true positive rate the false positive rate decreases, similarly with the decrement in true positive rate the false positive rate increases. ANN shows more linear results in all scenarios which shows that ANN is always stable and outperforms the other classifiers.

CPU time is the period during which the CPU is actively executing instructions. During program runs, the CPU

frequently sits inactive, awaiting data retrieval from input devices such as the keyboard or disc, or data transmission to output devices. As a result, the amount of time the CPU takes to execute a program is usually substantially less than the overall time it takes for the program to run. Multitasking operating systems enhance this element by allocating the CPU's time across many programs The quantification of CPU times serves several purposes. It plays a crucial role in comparing the efficiency of two different processors, analyzing a program's demand on the CPU, and measuring how processing time is allocated among several programs in a multitasking environment.

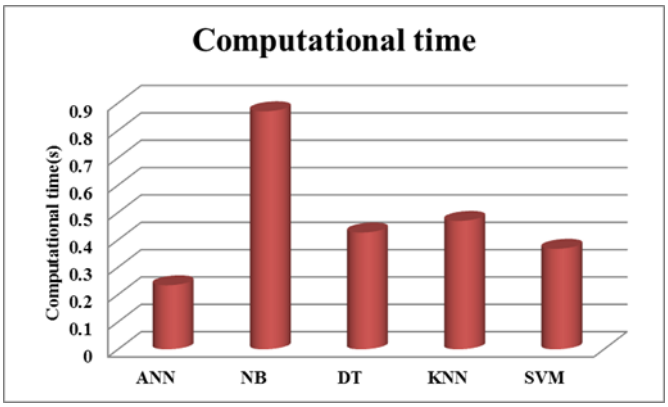

Fig. 23 Computational time comparison[1]

Figure 26 shows that the computational time of the proposed ANN detection classifier is less than 0.2 seconds, 0.4 seconds for DT classifier, 0.45 seconds for KNN classifier, 0.3 seconds for SVM classifier and 0.82 seconds for NB classifier. This shows the computational time of ANN classifier is significantly reduced than the other classifiers.

VI. Conclusion

The main aim of this article was to carry out a comprehensive performance analysis and comparison of the suggested framework [1]. The study aimed to enhance mobile malware detection by employing machine learning classifiers. This was achieved by selecting pertinent network properties for analysis by the classifiers and determining the most successful classifier based on true-positive rate (TPR) values. This study is significant for utilising the most recent and personally collected malware samples. The results and effectiveness of the classifiers were quite impressive. We evaluated various machine learning classifiers to enhance the identification of malware in a comprehensive collection of file samples, with the objective of identifying the most effective classifier for detecting mobile malware. The classifiers we evaluated encompassed Artificial Neural Network (ANN), Bayes network, decision tree (J48), K-nearest neighbor (KNN), and support vector machine (SVM). The experiment utilized samples from the MalGenome Project,

which included 49 distinct families of Android malware. The total number of malware samples was 1,260, out of which 1,000 were utilized in the experiment. The machine learning process consists of three stages: data acquisition to collect net- work traffic, selection and extraction of features, and application of the machine learning classifier. We selected the top 20 free applications from Google Play for the standard dataset. The results indicated that the random forest classifier achieved a detection accuracy rate of 99.99% for the Genome Malware dataset. The high classification percentages of the latest malware detection approach indicate the usefulness of the KNN technology, achieving an accuracy rate of 84.57%. These findings validate the ability of machine learning classifiers to identify modern malware. The increased quantity of the dataset led to longer durations for developing the model, but, it also resulted in enhanced accuracy, recall, and f-measure values. This work establishes a basis for future investigations, whereby game theory and multi- agent systems will be employed to discover and develop a dynamic system that is resistant to conventional bypass techniques.

Future research should prioritize the development of real-time mobile malware detection systems utilizing machine learning classifiers inside cloud environments. The efficacy and potency of machine learning in real-world mobile malware situations have already been demonstrated by this approach. It is advisable to do more research to understand why the chosen characteristics produced significant outcomes. Additionally, it is important to examine the interconnectedness of these characteristics and their influence on the detection system.

REFERENCES

[1]. K.Rami & L.Desai Malware Detection Framework Using PCA Based ANN, Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, 298-313, Springer Singapore

[2]. Adelstein, F., Matthew, S., Dexter, K.: Malicious code detection for open firmware. In: 2002 18th Annual Proceedings of the Computer Security Applications Conference, IEEE (2002)

[3]. Bergeron, J., et al.: Static detection of malicious code in executable programs. Int. J. Req. Eng 79, 184–189 (2001)

[4]. William, S.: Computer Security: Principles And Practice. Pearson Education India, Bengaluru (2008)

[5]. Chumachenko, K.: Machine Learning Methods for Malware Detection and Classification (2017)

[6]. La Polla, M., Martinelli, F., Sgandurra, D.: A survey on security for mobile devices. IEEE Commun. Surv. Tuto- rials, 15(1), First Quarter (2013)

[7]. Narayanan, Annamalai, MahinthanChandramohan, Lihui Chen, and Yang Liu. "A multi-view context-aware approach to Android malware detection and malicious code localization." Empirical Software Engineering (2018): 1-53.

[8]. Nachum, Shay, Assaf Schuster, and OpherEtzion. "Detection in the Dark–Exploiting XSS Vulnerability in C&C Panels to Detect Malwares." In International Symposium on Cyber Security Cryptography and Machine Learning, pp. 227-242. Springer, Cham, 2018.

[9]. Mishra, Preeti, Vijay Varadharajan, Emmanuel Pilli, and UdayaTupakula. "VMGuard: A VMI-based Security Architecture for Intrusion Detection in Cloud Environment." IEEE Transactions on Cloud Computing (2018).

[10]. Kumara, Ajay, and C. D. Jaidhar. "Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at VMM." Future Generation Computer Sys- tems 79 (2018): 431-446.

[11]. Xiao, Xi, Shaofeng Zhang, Francesco Mercaldo, Guangwu Hu, and Arun Kumar Sangaiah. "Android mal- ware detection based on system call sequences and LSTM." Multimedia Tools and Applications 78, no. 4 (2019): 3979-3999.

[12]. Ribeiro, José, Firooz B. Saghezchi, Georgios Mantas, Jonathan Rodriguez, Simon J. Shepherd, and Raed A. Abd-Alhameed. "An Autonomous Host-Based Intrusion Detection System for Android Mobile Devices." Mo- bile Networks and Applications (2019): 1-9.

[13]. Yen, Yao-Saint, and Hung-Min Sun. "An Android mutation malware detection based on deep learning using visualization of importance from codes." Microelectronics Reliability 93 (2019): 109-114.

[14]. Xu, Yang, Guojun Wang, JuRen, and Yaoxue Zhang. "An adaptive and configurable protection framework against android privilege escalation threats." Future Generation Computer Systems 92 (2019): 210-224.

[15]. Le, Van-Giap, Huu-Tung Nguyen, Duy-Phuc Pham, Van-On Phung, and Ngoc-Hoa Nguyen. "GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities." In Transactions on Computational Collective Intelligence XXXII, pp. 184-208. Springer, Berlin, Heidelberg, 2019.

[16]. Saracino, Andrea, Daniele Sgandurra, GianlucaDini, and Fabio Martinelli. "Madam: Effective and efficient behavior-based android malware detection and prevention." IEEE Transactions on Dependable and Secure Com- puting (2016).

[17]. Guo, Yonghe, Chee-Wooi Ten, Shiyan Hu, and Wayne W. Weaver. "Preventive maintenance for advanced metering infrastructure against malware propagation." IEEE Transactions on Smart Grid 7, no. 3 (2016): 1314-1328.

[18]. Wen, Hui, Weidong Zhang, Yan Hu, Qing Hu, Hongsong Zhu, and Limin Sun. "Lightweight IoT Malware Visualization Analysis via Two-Bits Networks." In International Conference on Wireless Algorithms, Systems, and Applications, pp. 613-621. Springer, Cham, 2019.

[19]. Rastogi, Vaibhav, Yan Chen, and Xuxian Jiang. "Droidchameleon: evaluating android anti-malware against transformation attacks." In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 329-334. ACM, 2013.

[20]. Seo, Seung-Hyun, Aditi Gupta, Asmaa Mohamed Sallam, Elisa Bertino, and KangbinYim. "Detecting mobile malware threats to homeland security through static analysis." Jtheirnal of Network and Computer Applications 38 (2014): 43-53.

[21]. Clancey W. J, "Transfer of Rule-Based Expertise through a Tutorial Dialogue", Ph.D. diss., Dept. of Computer Science, Stanford University, 1979.

[22]. Rice J, "Poligon: A System for Parallel Problem Solving", Technical Report, KSL-86-19, Dept. of Computer Science, Stanford University, 1986.

[23]. NSC, "Injury Facts Online. Itasca, Ill.: National Safety Council'. Available at: www.nsc.org. Accessed on 17December 2001.

[24]. Moulton R. K, "Method for on-site cleaning of contaminant filters in livestock housing facilities. U.S. Patent No.32455986, 1992.

[25]. Clancey W. J. and Rennels G. R, "Strategic Explanations in Consultation", *The International Journal of Man-MachineStudies*, Vol. 20, No. 1, pp. 3–19, 1983

