Study of Detecting Android Malware Using An Improved Filter Based Technique In Embedded Software

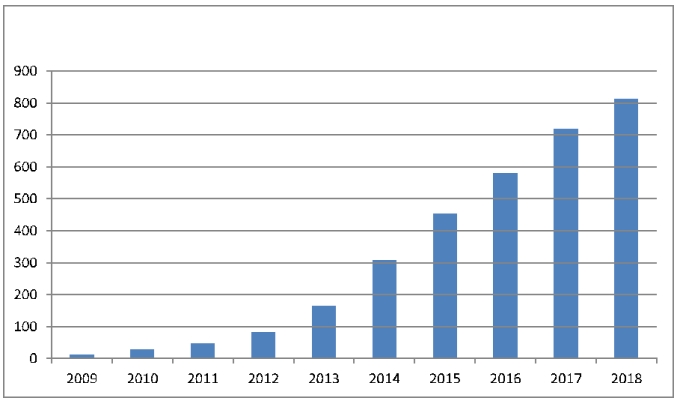
Amol L. Deokate[1], Yogesh A. Pawar[2]
[1, 2] Dept of Information Technology
[1, 2] Sanjivani K.B.P. Polytechnic

Abstract- *The technological advancements have led to evolution of sophisticated devices called smartphones. By providing extensive capabilities, they are becoming more and more popular. The Android based smart-phones are preferred furthermore, due to their open-source nature. This has also led to the development of large number of malwares targeting these smartphones. Thus to protect the devices, some countermeasures are needed. Machine learning methods have gained popularity in detection of malware. This work proposes a malware detection technique in Android devices based on static analysis carried out using the Manifest files extracted from the apk files. The feature selection is performed using the proposed KNN based Relief algorithm and detection of malware is done using the proposed optimized SVM algorithm. The proposed method achieves a True Positive Rate greater than 0.70 and much reduced False Positive Rate values were obtained, with the values of False Positive Rate being very close to zero. The proposed KNN based feature selection is found to select better features in comparison with some popular existing feature selection techniques. The proposed optimized SVM technique achieves a performance that is on par with the performance of Neural Networks.*

Keywords- Android, Malware detection , Permission, Machine learning

I. INTRODUCTION

Smartphones have become the core communication and computing devices today. With higher capabilities, the devices are not just the simple voice-oriented handsets used for communication in the older days. With such increasing capabilities they have become the target for several malware attacks. It was in 2004, the first article appeared on smartphone malwares [1], which reported that the smartphones are the next generation of target for malwares. The Symbian OS was the mobile OS that was affected by malware. Sig-nature based detection are efficient in detecting known malwares, but it fails in detecting new, or unknown malware. This creates a tremendous opportunities for the Attackers. The growth in Android malware every year is shown in Fig. 1.



This work is based on static analysis that uses permission as a feature, which are extracted from the manifest file present inside the apk bundle. Due to availability of rich source of data, machine learning algorithms have gained popularity in recent days. Taking advantage of this fact, the proposed work uses machine learning algorithms to effectively detect malware.

1.1. Machine learning

Machine learning is a subset of Artificial Intelligence, which helps computer systems to perform specific tasks based on the patterns extracted. Machine learning is different from traditional programming in a way that traditional programming involves in-put data which is given to a program to find an output, whereas in machine learning the input data and an optional output is fed into the model and the model generates a program that fits the input data. Fig. 2 provides a diagrammatic representation of that differentiates traditional programming and machine learning.

Machine learning techniques can be categorized into three major categories namely, Supervised Machine Learning, Unsupervised Machine Learning, and Reinforcement Learning. Supervised learning generally involves an input data

and a corresponding output. The process of learning involves finding a function that maps the input data to output data. The two most well-known examples of supervised learning are regression (predicting a continuous value e.g. price of a house, salary of a person etc.) and classification. In unsupervised learning no output information is provided, the ma-chine tries to extract patterns from the input provided and groups the input based on these patterns. In reinforcement learning the system learns using feedbacks it receives. The feedback is in the form of rewards (predicts correctly) and punishments (wrong predictions). The recommendation systems used in online sites are based on this type of machine learning.

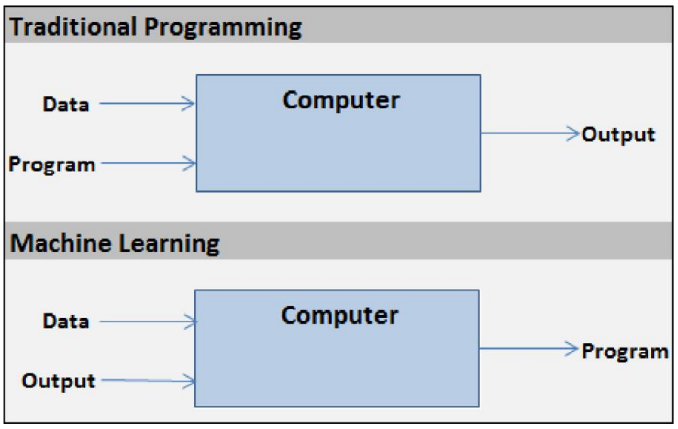


Fig. 2. Traditional Programming vs Machine Learning.

Predictions). The recommendation systems used in online sites are based on this type of machine learning.

1.2. Mobile malware

The first mobile malware was reported in June 2004, almost after 4 years of emergence of smartphone. The Symbian 60 OS was affected with the first mobile malware named Cabir, a worm. It was discovered that the malware was coded using C++ language. This later served as a base for several families of Symbian viruses [24].

Launched in 2008, Android didn't have a large user base to at-tract malware authors in its first two years. Then by 2010, with the emergence of the first Android Trojan in Aug 2010 named, "Androi-dOS.DroidSMS.A", a SMS fraud app, the Android malware started emerging [20].

Google has classified permissions into 3 categories: Normal, Signature and Dangerous. The normal and signature category of permissions are granted automatically without the consent of the user. Only the dangerous permissions require the user's approval.

Initially the experiment was carried out with malware samples from AAGM dataset and normal samples that were collected from Google Play Store. Support Vector Machines (SVM), a well known machine learning algorithm that works by finding a hyperplane that separates the two classes. In this case, one is the malware class and the other is a benign class. The dataset was split into two with 80% samples in spilt one, that is used for training the SVM model and the remaining 20% of data are assumed to be unknown and used to test the model that is built. The obtained results were compared with the original class labels to determine the malware detection efficiency.

Specifically, the contribution of the paper can be summarized as follows:

1. Use of enhanced feature selection algorithm to identify significant features to help improve the detection performance of the machine learning model.
2. To use an optimized version of SVM to detect Android mal-ware, and achieve a good detection results that the existing algorithms

II. **RELATED WORKS**

The existing works use either static or dynamic analysis as a means of extracting the features for the purpose of detecting mal-ware.

Manifest files were used to extract information like permissions [2]. Based on the information extracted a model was constructed using K means clustering to group the malware and benign apps into different clusters.

A dynamic analysis based feature extraction was used to detect malware [3]. The API and system calls extracted during the dynamic analysis helped in the detection of malware the detection was carried out by analysing the frequency of the API and system calls.

Marvin [4] uses a combination of static and dynamic analysis to extract features that are to be used for detection of malware, based on a malice score generated.

DroidDetector [5] used Deep learning methods to detect An-droid malware, with permission, and API features extracted during static analysis and the dynamic behaviour obtained while perform-ing dynamic analysis. The Deep Belief Network algorithm was used to detect malware.

Permissions extracted from manifest file were used to create feature vectors [6] which was then used to create a cluster rep-resenting the benign and malicious permissions, based on which the removal of malicious apps is carried out.

Anomaly based detection of malware [7], was carried out to detect unknown malware using decision tree algorithm. The work claims a detection rate of 90%.

Online detection of malware [8] is carried out using Field programmable gate array. They proposed architecture based on FPGA, which they named as GaurdOL. The dataset used for this work was constituted of 472 malware samples collected from Virusshare and VX Heaven.

Droidminer [9] provides automated detection of Android mal-ware by constructing Control Flow Graph (CGF) for the API calls that were extracted from the classes.dex files.

DAPASA [10] is a graph based detection of malware that calculates the sensitivity of API call using the well-known document mining technique called TF-IDF. The experiment was carried out based on two assumptions, which state how the sensitive API calls were invoked.

Anomaly detection of Android malware was carried out [11], where both static and dynamic analysis was performed to extract the features for detection of malware. The work generated mal-ware and normal pattern sets based on the calling pattern of the malware, and set a threshold to detect malware.

MADAM [12] is a behaviour base malware detection scheme that uses signature based and anomaly based malware detection. It uses a wide range of features to evaluate the application and detect the malware. The work considered four levels of features namely kernel, application, user, and package.

NTPDroid [13], created patterns that occur frequently in mal-ware and benign samples, which were used to fix a threshold that helps in detection of malware. The permission features were extracted from the manifest files and the network traffic was captured using Wireshark.

PMDS [14] provides a client-server based detection of Android malware, where the server decompresses the apk files of the application that were submitted to it to extract the manifest files and the detection of malware is done using several classifiers like C4.5 Decision Tree, K , Ripper, and Naive Bayes and the performance of these classifiers are compared.

The performance of machine learning algorithms like Naive Bayes, Ada Boost, Random Tree, Random Forest, J48 in detection of Android malware was studied [15], on using the permission feature as the input to these algorithms.

ndroid malware detection based on the frequency of the sys-tem call [19], invoked was proposed, where 82 types of system calls invoked were studied.

III. METHODOLOGY

The objective of the research is to improve the True Positive Rate (TPR), and get reduced False Positive Rate (FPR) values using the proposed, KNN based Relief (KNN-R), an improvised version of a filter based feature selection method called Relief. To improve the detection results further, the work proposes an upgraded kernel for SVM, called Optimized SVM (o-SVM).

Android operating system has Linux architecture as its base. The first obstacle for attackers is permissions. Though malicious code can be present inside, the API calls in the code require permissions for execution. Permission protected API is a component of Android OS security. Hence permission serves as an important feature, and thus the work used these permission features for the purpose of malware detection [25].

IV. SVM

SVM is a linear classifier, which identifies a hyperplane to separate the classes. When a non-linear boundary is required to classify the data points, SVM fails. Hence SVM makes use of a mathematical function called “Kernel”, which takes the data points in given dimension as input and projects them to a higher dimensional plane where they’re linearly separable [28].

4.1. Proposed optimized SVM (o-SVM)

The proposed o-SVM uses two kernel functions instead of one kernel function as suggested by the traditional SVM classifier. The kernel function of the proposed o-SVM is defined Lemma:

Lemma −1:

$$K(x, y) = l.k1\ (x, y) + (1 - l).k2\ (x, y) \qquad (1)$$

Where k1 and k2 are two different kernel functions. The two kernel function can be one of the three functions, namely; Polynomial Kernel, Radial Basis Kernel, and Sigmoid Kernel. λ is a constant, whose value lies between 0 and 1, λ can be thought as a weight factor, that implies the importance of the kernel, and if λ becomes equal to 0 or 1, then the o-SVM reduces to traditional SVM.

4.2. Proposed KNN – R based relief (KNN-R)

The proposed KNN based Relief (KNN-R) based feature selection technique uses, k neighbours instead of one neighbour as suggested by the original Relief algorithm. The existing Relief algorithm calculates the weight of each feature using its nearest neigh-bour from the same class, called Nearest Hit (NH), and the nearest neighbour from the opposite class, called Nearest Miss (NM). The proposed KNN-R uses 'k' NH, and 'k' NM, to calculate the weight of the feature. The reason behind using k nearest neighbour is that the existing Relief algorithm doesn't address the problem of noisy data, whereas the proposed KNN-R considers 'k' neighbours which help to eliminate the noise in the data.

Relief algorithm has several advantages like robustness to feature interactions, noise resilient, independent of machine learning algorithms and independent of heuristics [21,26], hence the work used Relief based feature selection.

The weight of the feature is calculated as per the below

Algorithm:

```
procedure KNN-R
  for a:= 1 to P do
    Update W[a]:= 0.0
  end for
  for i:= 1 to N do
    Pick a random Application A
  Find k-nearest miss(NM), and k-nearest hit(NH)
  of A Procedure
  WeightUpdate(N,k,NM[],NH[],A,P)
 end for
 Calculate the relevance Score
```

$$Relevance = \frac{1}{N} W \quad (2)$$

```
end procedure
 procedure WEIGHTUPDATE(N,k,NM[],NH[],A,P)
  for j:= 1 to P do
  Calculate the weight of each feature as:
```

$$W[j] = W[j] - \left(\frac{1}{k} \sum_{l=1}^{k} diff(A, NH_k) + \frac{1}{k} \sum_{l=1}^{k} diff(A, NM_k) \right) \quad (3)$$

```
  end for
```

where,

P - the total number of permission features
W - the weight associated with each feature
A - the sample instance under consideration and the distance metric used to calculate the neighbour is the "Hamming Distance".

The work can be categorized into five modules namely; analysis, feature extraction, feature vector generation, feature selection and detection. The overall flow of the proposed work is shown in Fig. 3 . Analysis of Android application can be performed either using static or dynamic analysis. the proposed work uses static analysis. The static analysis involves the apk file without executing it. Feature extraction is the process of drawing out those features that are required. The feature used in this work is permissions re-quested in manifest file. Feature vector generation is the process of converting the features to a form that is understandable by ma- chine learning algorithms. Feature selection is used to identify the significant features that help to detect the malware and remove those ones that cause ambiguity. The detection process is used to identify the app as either malware or benign. To perform static analysis of the samples a tool called AP- KParser is used, which helps to extract the AndroidManifest.xml from the apk bundle. Once the manifest file is extracted XML DOM parser is used to parse the contents of the manifest file and extract the permissions that were requested. The permission extracted were in the form text, but since the machine learning algorithms require the input to be in the form of numeric vector, the text was converted into numeric values using two widely used techniques: One-hot encoding (Feature vector 1) and the second method of feature construction used a text mining scheme called term frequency - inverse document frequency (TF-IDF), to construct a feature vector (Feature vector 2). Construction of feature using one-hot encoding is simple and easy. By using this technique a binary feature vector is generated that has '1' for the permissions present in it and '0' for those per- missions that are not present in the apk file, with the class label (Malware or Benign) present at the end.

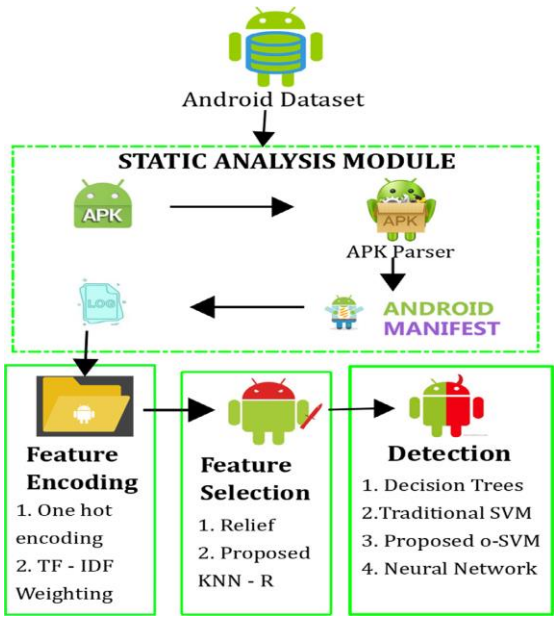


Fig. 3. Block diagram of the proposed work.

The permission feature extracted from Android is a text. The machine learning algorithms re capable of understanding numeric data, and one simple way of encoding them into categorical data is direct binary conversion of the text data (adding 1 for features that are present and 0 for those features that are not present), and the most desired and accepted technique for encoding text features to numeric features is TF-IDF technique, as a result of the work used these 2 techniques to encode the features [27]. The Fig. 4 shows an example of feature vector that is generated using this technique.

Construction of feature vector using TF-IDF is done using as be-low:

The term frequency is the ratio of number of times a particular term occurs in the document to the total number of terms in that document.

$$TF_{i,j} = \frac{No.\ of\ times\ a\ permission\ appears\ in\ the\ application}{Total\ no.\ of\ permissions\ in\ that\ application} \quad (4)$$

Where 'i' represents the total number of samples in the dataset and 'j' represent the total number of permissions present in these samples.

The inverse document frequency is used to find the significance of the term across all the documents. It is calculated as:

$$IDF_j = \ln \frac{Total\ no.o\ f\ applications}{No.o\ f\ applications\ containing\ the\ permission\ p} \quad (5)$$

The feature vector is then generated by computing the product of values of TF matrix and IDF vector as:

$$FV_{i,j} = TF_{i,j}\ \ IDF_j \quad (6)$$

The feature vectors constructed using the above two schemes were used for detection of malware after selection of Significant features using few filter based approaches like Chi-Square, Relief, and the proposed KNN-R. The reason behind choosing the filter based schemes is due to their simplicity. The detection was performed using several existing machine learning algorithms like SVM, Decision Tree and Back Propagation Neural Networks and their performance was compared with the proposed Optimized SVM algorithm.

V. RESULTS AND DISCUSSSIONS

The experiment was carried out using two different malware datasets: AAGM [16] and AMD [17 , 18] datasets. The benign dataset was constructed by downloading the applications from Google Play Store. Since the Play Store is not completely free from malicious application, the downloaded applications were uploaded to a on- line anti-malware checker site called Virus Total, where the application is scanned using more than 50 anti-virus software. Once the checking of the application gets completed, it returns whether the application is clean or not. The applications that were certified as clean by all the software is alone considered for building of benign dataset. The dataset 1 represents malware samples from AAGM dataset and dataset 2 represents malware samples from AMD dataset. The experiment is carried out by parsing the apk file, using permission extraction algorithm and a list of all permission are obtained. Then the permissions are encoded into feature vectors using one hot encoding and TF-IDF encoding techniques [22] and detection is carried out using machine learning algorithms, with support of packages from Python.

Table 1 Dataset Description.

	No. of Samples		
	Normal	Malware	Total
DATASET 1	1005	290	1295
DATASET 2	1005	1592	2597

The True Positive Rate represents the percentage of samples that are correctly classified as malware and is calculated using

$$TPR = \frac{TP}{TP + FN} \quad (7)$$

TPR is the ratio of correctly classified malware samples to the total number of malware samples. This ratio is also called as Sensitivity or Recall or Hit ratio

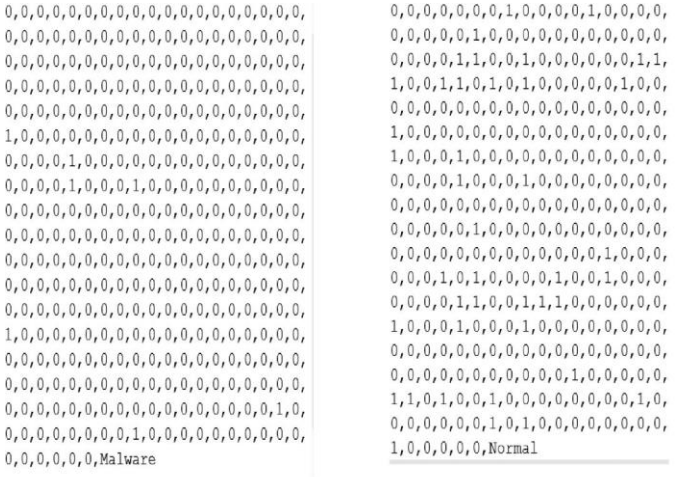
The False Positive Rate represents the percentage of samples that are mistakenly classified as malware and is calculated using

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

FPR is the ratio of normal samples that are misguided as mal-ware to the total number of benign samples in the dataset. This is also called as fall-out rate.

The benign samples present in the two datasets are the same. The number of malware and benign samples used for this research work are given in Table 1.

The initial feature vector generation is done using one-hot encoding technique. Feature selection is performed using 3 different popular filter based algorithms namely, Chi-Square, Relief and the proposed KNN-Relief and the performance of proposed Optimized



(a) Malware Feature Vector

(b) Normal Feature Vector

Fig. 4. Feature Vector.

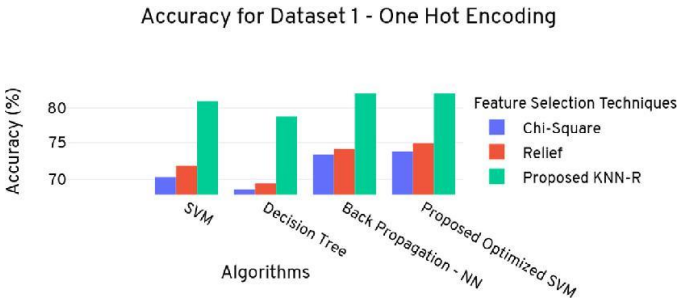


Fig. 5. Performance of machine learning algorithms with different feature selection techniques.

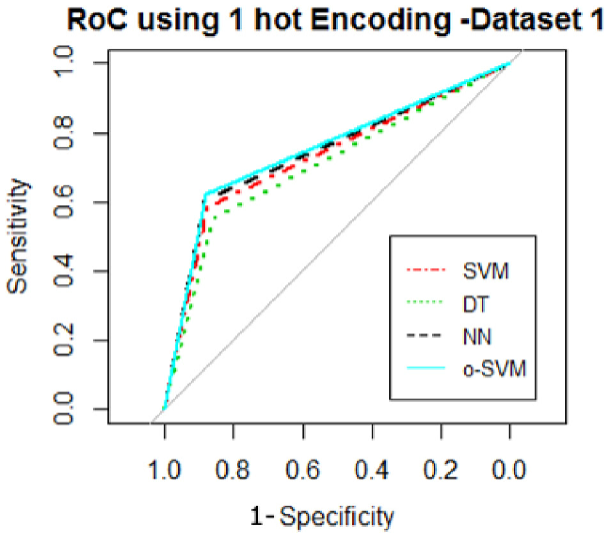


Fig. 6. RoC Comparison of different detection algorithms for proposed KNN-R - Dataset 1.

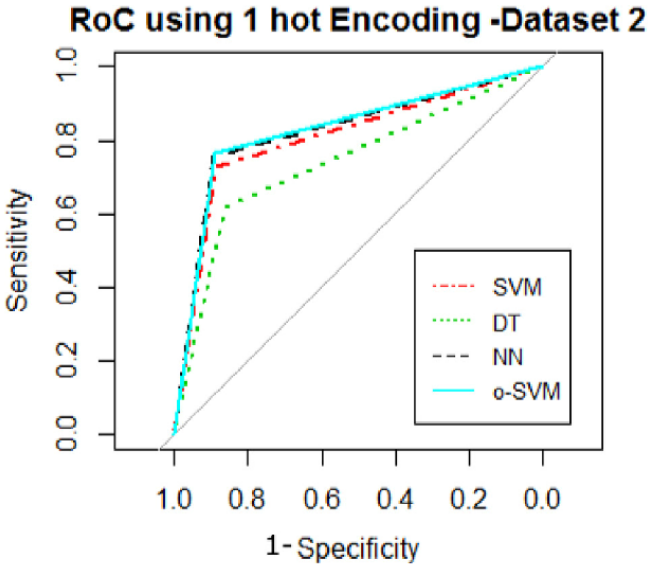


Fig. 7. RoC Comparison of SVM vs o-SVM for KNN-R feature selection - Dataset 2.

Table 1 Dataset Description.

	No. of Samples		
	Normal	Malware	Total
DATASET 1	1005	290	1295
DATASET 2	1005	1592	2597

The True Positive Rate represents the percentage of samples that are correctly classified as malware and is calculated using

TPR is the ratio of correctly classified malware samples to the total number of malware samples. This ratio is also called as Sensitivity or Recall or Hit ratio

The False Positive Rate represents the percentage of samples that are mistakenly classified as malware and is calculated using

$$TPR = \frac{TP}{TP + FN} \quad (7)$$

FPR is the ratio of normal samples that are misguided as mal-ware to the total number of benign samples in the dataset. This is also called as fall-out rate.

The benign samples present in the two datasets are the same. The number of malware and benign samples used for this research work are given in Table 1.

The initial feature vector generation is done using one-hot en-coding technique. Feature selection is performed using 3 different popular filter based algorithms namely, Chi-Square, Relief and the proposed KNN-Relief and the performance of proposed Optimized

Table 2 Performance of Dataset 1 using One Hot Encoding.

	Chi - Square		Relief		Proposed KNN-R	
	TPR	FPR	TPR	FPR	TPR	FPR
SVM	0.466	0.229	0.500	0.219	0.586	0.124
Decision Tree	0.431	0.239	0.448	0.234	0.552	0.144
Back Propagation -NN	0.517	0.204	0.534	0.199	0.603	0.114
Proposed Optimized SVM	0.534	0.204	0.552	0.194	0.621	0.119

Table 3 Performance of Dataset 2 using One Hot Encoding.

	Chi - Square		Relief		Proposed KNN-R	
	TPR	FPR	TPR	FPR	TPR	FPR
SVM	0.522	0.234	0.528	0.224	0.726	0.114
Decision Tree	0.506	0.254	0.513	0.244	0.619	0.139
Back Propagation -NN	0.535	0.219	0.538	0.209	0.755	0.104
Proposed Optimized SVM	0.538	0.214	0.541	0.204	0.764	0.109

SVM (o-SVM) is compared with that of the several existing ma-chine learning algorithms, which is shown in Table 2.

Chi-Square technique [23] is one popular and widely used technique in the literature for ranking features in Android malware analysis, hence the proposed work is compared with Chi-Square technique to analyse its performance.

Table 4 Performance of Dataset 1 using TF-IDF.

	Chi - Square		Relief		Proposed KNN-R	
	TPR	FPR	TPR	FPR	TPR	FPR
SVM	0.552	0.149	0.586	0.129	0.621	0.119
Decision Tree	0.500	0.199	0.534	0.179	0.569	0.164
Back Propagation -NN	0.569	0.129	0.603	0.114	0.638	0.114
Proposed Optimized SVM	0.586	0.124	0.621	0.114	0.655	0.109

From Table 2, it can be seen that the proposed Optimized SVM has a performance that is nearly equivalent to that of a BP-NN. The Fig. 5 shows the comparison of accuracy of these classifiers on using Relief based Feature Selection.

The RoC curve of the proposed KNN-R feature selection for different machine learning algorithm is shown in Fig. 6.

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

FPR is the ratio of normal samples that are misguided as mal-ware to the total number of benign samples in the dataset. This is also called as fall-out rate.

The benign samples present in the two datasets are the same. The number of malware and benign samples used for this research work are given in Table 1.

The initial feature vector generation is done using one-hot en-coding technique. Feature selection is performed using 3 different popular filter based algorithms namely, Chi-Square, Relief and the proposed KNN-Relief and the performance of proposed Optimized

It can be noted from Fig. 6, that the proposed o-SVM achieves a performance that is mostly equal to the performance of Back Propagation Neural Networks.

To verify whether the proposed Optimized SVM algorithm works better than the traditional SVM algorithm on all datasets, the experiment was carried out on another set of samples from Dataset 2, whose results are shown in Table 3.

From Table 3 it can be concluded that the proposed Optimized SVM performs better on the second dataset as well. The RoC comparison for standard SVM and the proposed Optimized SVM on using the proposed KNN-R technique for feature selection is shown in Fig. 7.

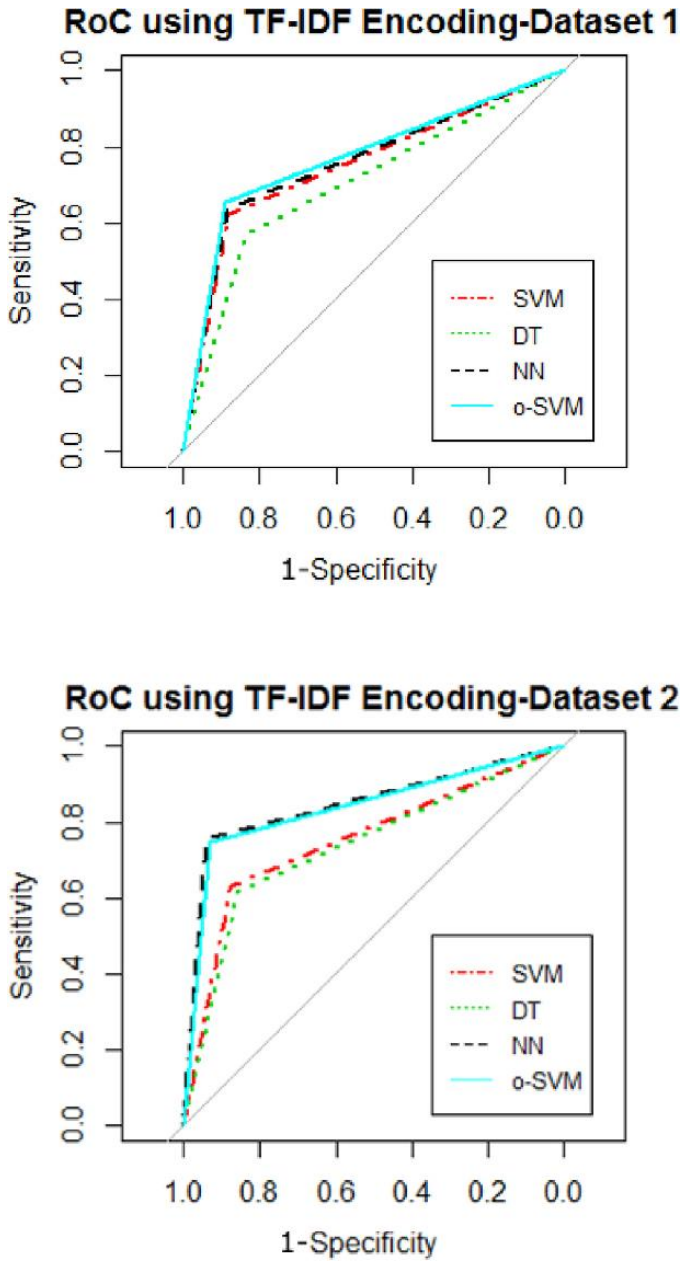


Fig. 9. Performance of SVM vs o-SVM with KNN-R technique.

The performance of the classifiers on using TF-IDF as technique for feature vector generation was also studied. The performance of Dataset 1 was given in Table 4 . From Table 4 , it can be seen that on using TF-IDF the performance of most of the classifiers improved in comparison with one-hot encoding technique. This is because that on using one- hot encoding technique, the permission present in a particular apk doesn't affect other files, whereas on using TF-IDF, the permissions that occur in different files are also taken into account, which helps to identify permissions that occur frequently in malware and those that occur frequently in benign samples, hence the improvement in performance. The RoC performance of the algorithms on using different machine learning classifiers with proposed KNN-R based feature se- lection for the first dataset is given in Fig. 8 . The performance of Dataset 2 for different feature selection and machine learning algorithms is given in Table 5 . Tables 4 and 5 conclude that the proposed KNN-R based feature selection and the proposed Optimized SVM algorithm shows a better performance. The improvement in detection performance on using the proposed o-SVM is shown in Fig. 9 .

Table 5 Performance of Dataset 2 using TF-IDF.

	Chi - Square		Relief		Proposed KNN-R	
	TPR	FPR	TPR	FPR	TPR	FPR
SVM	0.597	0.154	0.613	0.139	0.632	0.124
Decision Tree	0.591	0.169	0.601	0.159	0.619	0.144
Back Propagation - NN	0.733	0.090	0.742	0.080	0.755	0.060
Proposed Optimized SVM	0.726	0.100	0.736	0.085	0.748	0.070

From the results obtained it is clear that the proposed o-SVM technique achieves a better performance than the traditional SVM model and the KNN-R based feature selection also helps in improvement of the detection of malware.

VI. CONCLUSION AND FUTURE WORK

This paper Describe a static analysis by considering the permissions that are required by the application. The process of feature selection is important as it helps to identify the significance features that help in detection of malware. The work utilized some popular feature selection techniques like Chi-Square and Relief. The proposed KNN-R algorithm was found to perform better than the other two algorithms.

The static analysis performed in this work considers only per-mission, however there are chances that the malware application may request only the commonly used by any application but would invoke some unwanted functions. To monitor these behaviors dynamic analysis can be performed to

extract features like system calls, network traffic etc. Further deep learning models can be used in detection of malware instead of machine learning algorithms.

VII. ACKNOWLEDGEMENTS

It is with the greatest pride that we publish this paper. At this moment, it would be unfair to neglect all those who helped me in the successful completion of this paper. I would also like to thank all the faculties who have cleared all the major concepts that were involved in the understanding of techniques behind my paper.

REFERENCES

[1] D. Dagon, T. Martin, T. Starner, Mobile phones as computing devices: the viruses are coming!, IEEE Pervasive Comput. 3 (4) (2004) 11–15.

[2] S.B. Almin, M. Chatterjee, A novel approach to detect android malware, Proce-dia Comput. Sci. 45 (2015) 407–417.

[3] V.M. Afonso, M.F. de Amorim, A.R.A. Grigio, G.B. Junquera, P.L. de Geus, Iden-tifying android malware using dynamically obtained features, J. Comput. Virol. Hacking Tech. 11 (1) (2015) 9–17.

[4] Lindorfer, M., et al., 'Marvin: efficient and comprehensive mobile app classifi-cation through static and dynamic analysis', Proceedings of 39th Annual Computer Software and Applications Conference, pp. 422–433.

[5] Z. Yuan, et al., 'DroidDetector: android malware characterization and detection using deep learning, Tsinghua Sci. Technol. J. 21 (1) (2016) 114–123.

[6] S.B. Almin, M. Chatterjee, A novel approach to detect android malware, Proce-dia Comput. Sci. 45 (2015) 407–417.

[7] W. Yu, H. Zhang, G. Xu, A study of malware detection on smart mobile devices, in: Proceedings of SPIE 8757, Cyber Sensing, 2013.

[8] S. Das, Y. Liu, W. Zhang, M. Chandramohan, Semantics-based online malware detection: towards efficient real-time protection against malware, IEEE Trans. Inf. Forensic Secur. 11 (2016) 289–302.

[9] C. Yang, 'DroidMiner: automated mining and characterization of fine-grained malicious behaviors in android applications, in: Proceedings of European Symposium on Research in Computer Security, 2014, pp. 163–182.

[10] M. Fan, J. Liu, W. Wang, H. Li, Z. Tian, T. Liu, DAPASA:detecting android pig-gybacked apps through sensitive subgraph analysis, IEEE Trans. Inf. Forensics Secur. (2020) In press.

[11] F. Tong, Z. Yan, A hybrid approach of mobile malware detection in android, J. Parallel Distrib. Comput. 103 (2017) (2016) 22–31 In press, doi:10.1016/j.jpdc. 2016.10.012.

[12] D.S. Saracino, G. Dini, F. Martinelli, Madam: effective and efficient behav-ior-based android malware detection and prevention, IEEE Trans. Depend. Se-cure Comput. 99 (2016) 1 PP.

[13] A. Arora; S.K. Peddoju, "NTPDroid: a hybrid android malware detector using network traffic and system permissions", in Proceedings of 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Com-munications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).

[14] P. Rovelli, et al., PMDS: permission-based malware detection system, Int. Conf. Inf. Syst. Secur. (2014) 338–357.

[15] S. Gupta, Permission driven malware detection using machine learning, Int. Res. J. Eng. Technol. 4 (12) (2017) 993–996.

[16] A.H. Lashkari, A.F.A. Kadir, H. Gonzalez, K.F. Mbah, A.A. Ghorbani, Towards a network-based framework for android malware detection and characterization, the proceeding of the 15th International Conference on Privacy, Security and Trust, PST, 2017.

[17] Y. Li, J. Jiyong, X. Hu, X. Ou, Android malware clustering through malicious payload mining, the 20th International Symposium on Research on Attacks, Intrusions and Defenses (RAID 2017), 2017 September 18-20.

[18] F. Wei, L. Yuping, R. Sankardas, O. Xinming, Z. Wu, Deep ground truth analysis of current android malware, the 14th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2017), 2017 July.

[19] S. Malik, et al., Android system call analysis for malicious application detection, Int. J. Comput. Sci. Eng. 5 (11) (2018) 105–108.

[20] M.O. Topgul, E.I. Tatli, The past and future of mobile malwares, 7th Interna-tional Conference on Information Security and Cryptology, 2014.

[21] Ryan J. Urbanowicz, Relief-based feature selection: introduction and review, J. Biomed. Inform. (2018) 189–203.