International Journal on Emerging Technologies **11**(3): 516-525(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

DLDroid: Feature Selection based Malware Detection Framework for Android Apps developed during COVID-19

Arvind Mahindru[1,2] and A.L. Sangal[3]
[1]Research Scholar, Department of Computer Science and Engineering,
Dr. B.R. Ambedkar National Institute of Technology Jalandhar-144001 (Punjab), India.
[2]Assistant Professor, Department of Computer Science and Applications,
DAV University, Jalandhar 144012 (Punjab), India.
[3]Professor, Department of Computer Science and Engineering,
Dr. B.R. Ambedkar National Institute of Technology Jalandhar-144001 (Punjab), India.

(Corresponding author: Arvind Mahindru)
(Received 20 April 2020, Revised 14 May 2020, Accepted 17 May 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: COVID-19 acted as a window of opportunity for cyber criminals to develop malware-infected apps. During this lockdown period, everyone is sitting at homes and interacting with others mostly through smartphones. With an exponential increase in Android apps and hence in Android malware, it has become really challenging that how to secure user's privacy. For this purpose, a number of academicians and researchers have proposed various signature-based and machine learning approaches to detect Android malware. Signature-based approaches can detect only known malware whose signature definitions are already present in its database. On the other hand, machine learning approaches, which were proposed in the literature were developed either with irrelevant features or not able to detect malware which are developed during COVID-19 pandemic. To overcome these issues it becomes highly essential to develop an effective and efficient Android malware detection model. Therefore, in this research paper, 11,000 distinct Android apps are collected, that belong to twelve different categories of Android apps. A total of 1844 unique features from these gathered Android apps are extracted and using ten distinct feature selection approaches irrelevant features have been removed. After that, an Android malware detection framework is developed by using significant features as input and Deep Neural Network (DNN) as machine learning technique. The experiment results reveal that the model developed by using rough set analysis as feature selection approach along with DNN can detect 97.9% malware from real-world apps.

Keywords: Android apps, Permissions model, API calls, Deep Neural Network (DNN), Feature selection, Intrusion-detection, Cyber security, smartphone.

I. INTRODUCTION

COVID-19 is a global calamity that started in December 2019, in Wuhan, Hubei, China [1], on an unbelievable scale, with devastating consequences. It has not only paid effect on health industry rather it paid effect on the other sectors too, like Education, Banking, IT and Business. To fight with this novel disease, public health officials and local communities suggest "contact tracing" smartphone apps. Indian government released "AarogyaSetu" [2], WHO released MyHealth [3], Italy government launched "Immuni" [4], Singaporean government released "TraceTOgether" [5]. These smartphone apps demand permissions related to approximate location, precise location, bluetooth and data sharing. The proper functioning of an Android app depends upon the permission model. Therefore, permissions play a vital role in the study of smartphone security, as cybercriminals use these permissions to steal the sensitive or personal information of the users from their smartphones.

Growth of Android malware has become a serious threat for user's sensitive information and privacy. According to the report published by GDATA [6], cyber crooks made more than 10,000 malware-infected apps on daily basis. It means that in every 8 seconds a malware-infected app is developed. Google introduced Google Bouncer [7] in the year 2012 for scanning the existing and new apps in its official play store. But Google bouncer has a number of limitations [8] and has failed to achieve a better detection rate. Later on, Google introduced Google play protect in play store for scanning the Android apps at the time of downloading and installation. According to the report published by MacAfee [9], in the first quarter of 2020; 1,000,000 new malware detected in the Q4 of 2019.

To address this issue, in the literature a number of authors proposed signature-based [10] and machine learning approaches [11-13] for detecting malware from Android devices. Signature-based approaches can identify only those malware whose signature is already present in its database. On the other hand, machine-learning approaches proposed by academicians and researchers are examined on the limited data set. So, to build an effective and efficient Android malware detection model, in this research paper, we collect 11,000 distinct Android apps, which further belong to twelve different categories of Android apps. We extract 1844 unique features from these managed apps and divide them into thirty different feature sets. The performance of the machine-learning algorithm is based on the features by which it is trained. To remove irrelevant features and misclassified errors, in this

research paper, we build and compare the model by using ten different feature selection approaches.

In the past few years, the malware detection model developed by considering Deep Neural Network (DNN) has achieved a better detection rate. DNN has an ability to learn from features and do classification simultaneously to achieve better results. Motivated by this, in this study, we use permissions, API calls, number of the user download the apps, and rating of the app as input features to train with DNN. The main reason for considering permission as one of the features is that by using permission, cybercriminal can easily interact with user's information and steal sensitive information from user's smartphones.

Seeing the current situation, most of the organizations have requested, their workforce to work from home. Several countries such as India, China, Italy, France, Poland, New Zealand and the UK have gone into full lockdown and human beings are forced to stay indoors. So, people are entirely dependent upon the mobile apps for communication, news, entertainment, business, medical, health & fitness, dating, social interactions etc. Therefore, COVID-19 has become a new weapon for cyber attackers to develop a number of malware-infected Android apps in the names of COVID-19 and spreading ransom are, trojan and Adware. So, smartphone security becomes highly important during this time. The unique and novel contributions of this paper are as follows:

– To the best of our knowledge, this is the first research work in which 11,000 distinct Android apps [14] are collected which are developed during COVID-19 pandemic.

– In this study, ten distinct feature selection approaches are used to remove irrelevant features. To build effective and efficient malware detection model we consider Deep Neural Network (DNN) as a machine-learning algorithm.

– Collected apps belong to twelve different categories of Android apps, from which 1844 unique features are extracted to build effective and efficient Android malware model.

– Proposed malware detection approach is able to detect malware in less time when compared to previous distinct anti-virus scanners available in the market.

Rest of the paper is summarized as follows. In section II, we describe the related work that has been done so far in the field of Android malware detection and gaps present in the literature. Section III, represents the formulation of experimental data set and creation of feature sets. Feature selection approach is discussed in section IV. In section V, we discuss about the machine learning technique used in this research paper. Section VI, discusses about the different methods on which we will compare our proposed model. Performance parameters for evaluating our proposed model are discussed in section VII. In section VIII and IX, we discuss about the experimental setup and results of our performed experiment. In section X, we present the conclusion and future work.

II. RELATED WORK

In this section of the paper, we discuss about the previous approaches or frameworks developed for Android malware detection.

Faruki *et al.,* (2013) proposed AndroSimilar that generates an automatic signature that extracts statistically syntactic features, which are used for malware detection [4]. Andrubis [15] is a web-based malware analysis platform in which the user can submit apps through web service, and after analyzing the app behavior, it returns detail app is benign or malware. Aurasium [16] takes control of the execution of apps, by applying arbitrary security policies at run-time. It repackages the Android apps to include code for policy enforcement, and any privacy violations are informed to the user. Aurasium has a limitation; it cannot note the malicious behavior if an app changes its signature. CopperDroid [17] performs call-centric dynamic analysis of Android apps; using Virtual Machine Introspection. Authors experimented with more than 2900 Android malware samples, and the technique proposed by them shows conclusive detection of malware behavior. Mahindru and Singh (2017) extract 123 dynamic permissions from 11,000 distinct Android apps and applied five different machine-learning algorithms, i.e., Naïve Bayes, Random Forest, Simple Logistic, Decision Tree, and k-star. Out of five-implemented machine learning algorithms, Simple logistic perform better in detecting malware from real-world apps [13]. Mahindru and Sangal (2019) proposed "DeepDroid", which works on Deep Neural Network (DNN) and Principal Component Analysis (PCA) as feature selection method. An experiment was performed on 1,20,000 Android apps and achieved the detection rate of 94% [18].

CrowDroid [19] is a behavior-based malware detection system, which works on two components, i.e., a crowd sourcing app which needs to be installed on user devices and second on the remote server for malware detection. CrowDroid with the help of crowd sourcing app sends the behavioral data in the form of log-file to the remote server. At the remote server, the collected behavioral data is processed to create feature vector by using 2- mean clustering algorithm to predict whether the app is malicious or benign. However, it has limitation, CrowDroid app always drain the available device resources. Mahindru and Sangal (2020) proposed "PerbDroid", which can detect limited malware families [12]. Features were selected by implementing six distinct feature-ranking approaches (i.e., Principal Component Analysis (PCA), Gain Ratio, Chi-squared test, Information gain feature evaluation, OneR feature evaluation, and Logistic regression analysis). Further, with selected features, they developed sixty distinct models by using ten discrete machine-learning algorithms. The model developed by using a Deep neural network and PCA achieved a detection rate of 97.8% using 2,00,000 different Android apps. TaintDroid [20] track the privacy-sensitive information leakage in the third- party developer apps. Whenever the sensitive data leave from the smartphone, TaintDroid records the label of the particular data and the app, which referred the data along with its destination address. Mahindru and Sangal (2020) compare the performance of supervised and semi-supervised machine learning algorithms by using feature subset selection approaches [11]. They implemented LLGC as a semi-supervised machine-learning algorithm and achieved a higher detection rate on moderate data set. Classification

algorithms have also achieved the higher prediction rate on disease dataset [21]. Table 1 highlights about the feature selection approaches and data set used by different researchers and academicians in their work. From Table 1, it is seen that researchers had applied limited feature selection approaches on their collected dataset and as it is known that significant features play a major role in developing a malware detection model, therefore, to overcome this, in this study ten distinct feature selection approaches are implemented to select significant features. Now, based on the literature review, we consider the following research questions in this research paper.

A. Research Questions

RQ1. Which feature selection approach is more effective for detecting malware from Android apps?

To examine this question, in this study, we applied ten distinct feature selection approaches and developed models by considering DNN as a machine-learning algorithm. Further, the performance of the developed model is compared with two distinct performance parameters, i.e., F-measure and accuracy.

RQ2. Is the feature selection approaches effect on the outcome of the machine-learning algorithm?

To answer this question, we compare the performance of feature selection approaches with all extracted feature sets.

Table 1: Feature selection approaches implemented in the literature.

Proposed Framework	Feature Selection Technique Used
PerbDroid [12]	Principal Component Analysis(PCA), Gain Ratio, OneR feature, Information gain feature evaluation, Logistic regression analysis evaluation, and Chi-square test
Mahindru and Sangal (2019) [11]	Consistency Sub-set Evaluation Approach, Filtered Sub-set Evaluation, Rough Set Analysis Approach and Approach Based on Correlation
Azmoodeh *et al.*, [22]	Information Gain
Shabtai *et al.,* [23]	Fisherscore, Chi-square and Information Gain
Mas'ud *et al.* [24]	Information gain and Chi-square
MKLDroid [25]	Chi-squared

III. FORMULATION OF EXPERIMENTAL DATA SET AND CREATION OF FEATURE SETS

To reduce the effect of Android malware and for building an effective malware model that is capable to detect malware from real world apps, in this research paper, we collect 11,000 distinct Android apps that belong to twelve different categories of Android apps. We collected Android application packages (.apk) from Google official play store and third party app stores i.e, APKmirror [26] and AllFreeAPK [27]. These apk files are published from December 2019 to April 2020 in these repositories. This data set is available publicly [28].

Table 2: Number of Android apps used in this research work.

ID	Category	Google play	Third-party app store
DS1	Business	329	1750
DS2	Education	135	2000
DS3	Game	100	1820
DS4	Entertainment	183	152
DS5	Social Media	128	760
DS6	Travel & Local	12	32
DS7	Food & Drink	176	65
DS8	Finance	184	250
DS9	Medical	187	560
DS10	Health & Fitness	120	1000
DS11	News & Magazine	139	500
DS12	Dating	185	980

Table 2 shows the category wise number of .apk files considered in this study. Out of collected 11,000 .apk packages, 5,500 are malware infected. Virus-total [29] identify Malware packages.

A. Creation of feature sets

After collecting Android apps from different promise repositories, we extract permissions and API calls, which were demanded by Android apps during its installation and run-time. For extracting features from Android apps, we used Android studio as an emulator and self-written java program to extract features from them mentioned in [13]. We extract 1532 unique permissions and 310 API calls for developing malware detection model. List of extracted permissions and API calls are available for researchers and academicians [30]. A total of 1844-dimensional Boolean vector, where "1" implies that the app requires the feature and "0" implies that the feature is not required. It is very common that benign and normal apps may request a similar set of permissions and API calls for its execution. Permissions overview given by Google [31] is used to describe the behavior of permission i.e., "dangerous" or "normal". After extracting the permissions and API calls, we divide them into thirty different feature sets, which are shown in Table 3. In this research paper, we also consider the rating of an app and number of the user download the app as features. To normalize the data, we used the Min-max approach. This approach is based on the principle of a linear transformation, which bring each data point D_{q_i} of feature Q to a normalized value D_{q_i}, that lie in between 0

The following equation is considered to find the normalized value of D_{q_i}:

$$\text{Normalized}\left(\mathrm{D_{q_i}}\right) = \frac{\mathrm{D_{q_i}} - \min\left(\mathrm{Q}\right)}{\max(\mathrm{Q}) - \min\left(\mathrm{Q}\right)},$$

where $\min(\mathrm{Q})$ & $\max(\mathrm{Q})$ are the minimum and maximum significance of attribute Q, respectively.

Table 3: Formulation of Sets containing (App downloaded by number of users, permissions, API calls and rating of the App) as features.

No.	Description related to	No.	Description related to
FS1	Phone State and Phone Connection	FS2	Audio and Video
FS3	Bundle	FS4	Log File
FS5	Synchronization Data	FS6	Contact Information
FS7	System Settings	FS8	Browser Information
FS9	Calendar Information	FS10	Account Settings
FS11	Location Information	FS12	Widget
FS13	System Tools	FS14	Network Information and Bluetooth Information
FS15	Unique Identifier	FS16	File Information
FS17	Services That Cost You Money	FS18	Phone Calls
FS19	Database Information	FS20	Image
FS21	Contain info. Related to API calls	FS22	Contain info. Related to rating and downloads
FS23	Your Accounts	FS24	Storage File
FS25	SMS MMS	FS26	Read
FS27	Access Action	FS28	Read and Write
FS29	Hardware Controls	FS30	Default group

IV. FEATURE SELECTION APPROACHES

In this paper, we implemented ten distinct types of feature selection approaches on a large collection of 1844 features (divided in to thirty distinct feature sets) to identify the best subset of features which assist us to detect malware detection with better detection rate and also minimize the figure of misclassification errors. Table 4 represents the different feature selection approaches used in this study.

Table 4: Feature selection approaches.

Name of the feature selection approach	Description
Gain-ratio feature selection approach [12]	This approach work on the prediction of the gain-ratio in relation to the class to which the app belong. The "Z" known as the gain-ratio of feature is measured as:- $\text{Gain} - \text{ratio} = \text{ in this Gain}(Z) = I(A) - E(Z)$ here A represents the feature set contains X amount of instances having n distinct classes.
Chi-Square feature selection approach [12]	This test is utilized to investigate the self-determination between two situations, and in our study, ranking of features are based on the significance of its statistic, which is related to the class. Higher the calculated value implies the denial of the outliers and as a result, these selected features can be considered as better relevance in detecting malware infected apps.
Information-gain feature selection approach [12]	In Info-gain, features are selected on its relation with respect to the class, which it belong.
OneR feature selection approach [12]	OneR feature selection approach is utilized for ranking the features. To rank individual features utilizes it the classification mechanism. In it valuable features are considered asconstant ones and partition the set of values into a few dissociate intervals made by straightforward approach. In this study, we consider only features that is having better classification rates.
Principal Component Analysis (PCA) [12]	Reduction of attribute is accomplished by implementing PCA on our collected data set. PCA helps in transforming a high dimension data space into a low dimension data space. Features, which are present in low dimension, have extreme importance in detecting malware.
Logistic regression analysis [12]	For feature ranking, Univariate Logistic Regression (ULR) analysis being considered to verify the degree of importance for every feature sets.
Filtered subset evaluation [11]	Based on the principle to select random subset evaluator from data set that was gained by applying arbitrary filtering approach.
Consistency subset evaluation approach [11]	This technique provides the importance of subset of attributes by their level of consistency appearing in class values, when the training instances are applied on the subset of attributes.
Rough set analysis [11]	This approach is an estimation of conventional set, in terms of a joins of feature sets that provide the upper and the lower estimation of the original data set.
Correlation based feature selection [11]	This approach is based on correlation approach which select a subset of features that are particularly related to the class (i.e., benign or malware).

V. MACHINE LEARNING TECHNIQUE

To develop an effective and efficient Android malware detection model, we consider the Deep Learning Model (i.e. DNN) as a machine learning technique. In the literature, a number of authors proposed the construction of a Deep Learning Model with Convolutional neural networks (CNN) and Deep Belief Networks (DBN) [12, 18].

In the present paper, we consider CNN architecture for building the deep learning model [12]. DNN can be assembled with different deep architecture i.e., Deep Belief Networks (DBN) and Convolutional neural networks (CNN). In the present paper, we select DBN architecture to develop our deep learning model. Fig. 1 demonstrates the architecture of deep learning method. It is divided in to two stages, one is supervised back-

propagation and second stage is unsupervised pre-training. In the early stage of model building, Restricted Boltzmann Machines (RBM), with the deep neural network are used to trained the model. In training step, iterative process is used to build the model with unlabeled Android apps. In the back-propagation stage, pre-trained DBN is fine-tuned with labeled Android apps in a supervised manner. Model build by considering deep learning method use an Android app in both stages of the training process.

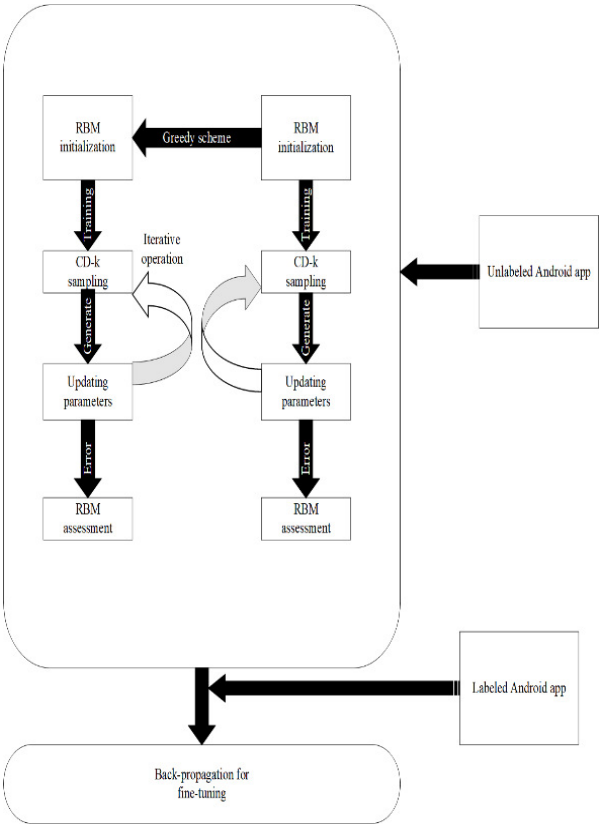


Fig. 1. DNN Model.

VI. COMPARISON OF PROPOSED MODEL WITH DIFFERENT EXISTING TECHNIQUES

To examine that our developed framework is able to achieve a higher detection rate or not, in this research paper, we analyze the outcome of our proposed model with two distinct methods which are mentioned below:

(a) Comparison of results with previously used classifiers: To verify that our developed model is feasible to detect malware as equivalent to previously used classifiers or not, we validate it based on two performance parameters i.e., F-measure and Accuracy.

(b) Comparison of results with different Anti-Virus scanners: To analyze the performance of our model for malware detection, we chose ten available distinct anti-virus scanners and compare their detection rate with the detection rate of the proposed model.

VII. EVALUATION OF PERFORMANCE PARAMETERS

In this section of the paper, we discuss the fundamental definitions of the performance parameters utilized by us while evaluating our proposed model for malware detection. The confusion matrix is used to calculate all these parameters. It consists of information related to actual and detected classification built by detection models. Table 5 demonstrates the confusion matrix for the malware detection model. In this study, two performance parameters namely, F-measure and Accuracy are employed for measuring the performance of malware detection models. Below we yield formulae to evaluate Accuracy and F-measure:

$$\text{Accuracy} = \frac{N_{\text{Benign}\rightarrow\text{Benign}} + N_{\text{Malware}\rightarrow\text{Malware}}}{N_{\text{classes}}}$$

And

$$\text{F} - \text{measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

$$= \frac{2 \times N_{\text{Malware}\rightarrow\text{Malware}}}{2*N_{\text{Malware}\rightarrow\text{Malware}} + N_{\text{Malware}\rightarrow\text{Benign}} + N_{\text{Benign}\rightarrow\text{Malware}}}$$

Table 5: Confusion matrix Used in this study.

	Malware	Benign
Malware	Malware →Malware	Malware→ Malware
Benign	Benign → Malware	Benign →Benign

VIII. EXPERIMENTAL SETUP

In the present section, we introduce the experimental setup done to find the performance of our developed malware detection models. DNN is implemented on 11,000 Android apps, which belong to twelve different categories of android apps mentioned in Table 2. All these data sets have a varying number of benign or malware apps that are adequate to perform our analysis. Fig. 2 demonstrates the framework of DLDroid. The subsequent measures are pursued at the time of either choosing a subset of features to develop the malware detection model that detects that app belongs to benign or malware class. Feature selection approaches are employed on 12 different categories of Android apps. Hence, a total of 132 ((1 selecting all extracted features+ 10 feature selection approaches) × 12 data sets (subsets of different feature sets particular to data sets determined after conducting feature selection) × 1 detection methods) different detection models have been developed in this research paper. The subsets of features obtained from aforementioned procedure are given as an input to machine learning classifiers. To compare the developed models, we use 20-fold cross-validation method. Cross-validation is a statistical learning approach that is utilized to classify and match the models by dividing the data into two different portions. One portion is utilized to train and the remaining portion of data is utilized to verify the build model, on the basis of training. The data is initially separated into *K* same sized segments. K-1 folds are utilized to train the model and the rest one fold is utilized for testing intention.

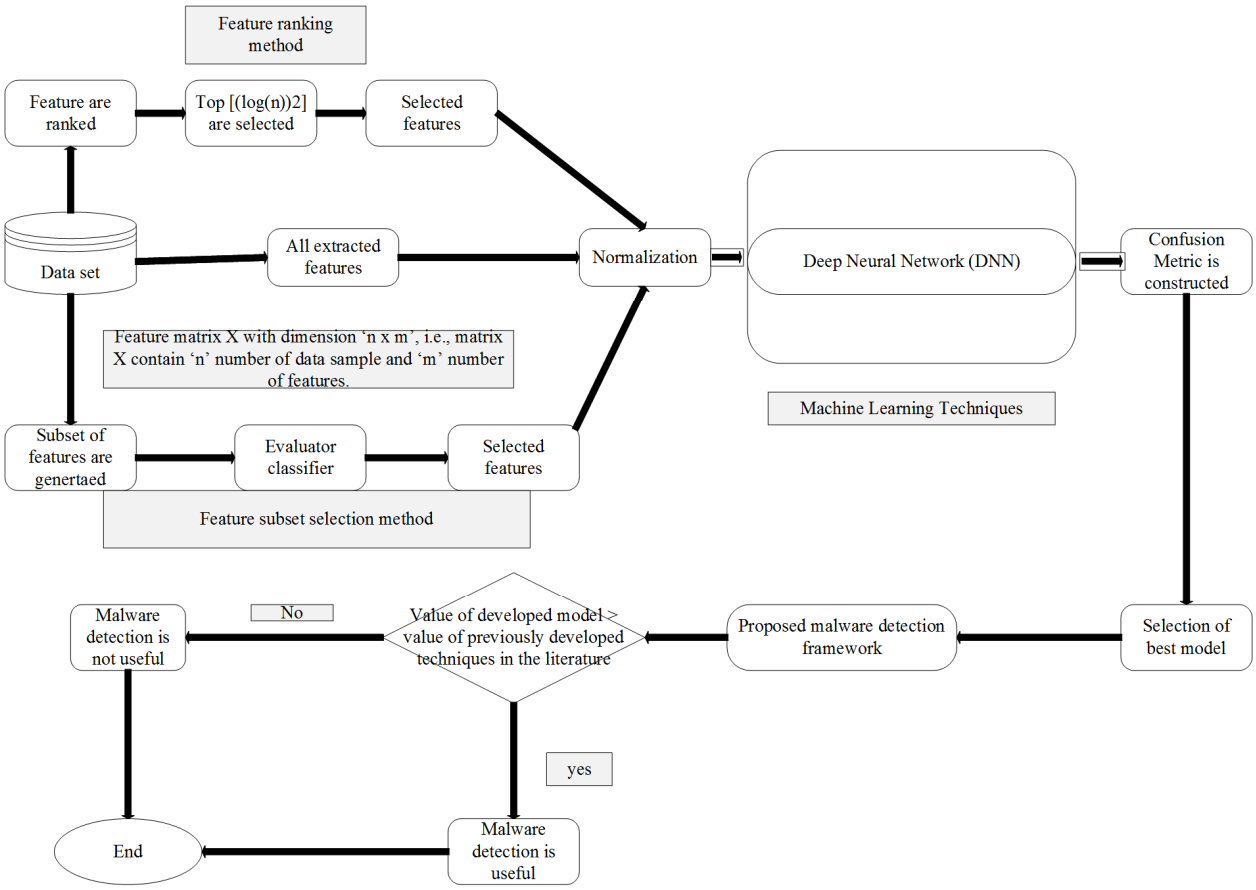


Fig. 2. Framework of DLDRoid.

K-fold cross-validation is having important significance in utilizing the data set for the both testing and training. For this study, 20-fold cross-validation is utilized to analyze the models, i.e., data sets are segregated into 20 portions. The outcomes of all build malware detection models are matched with each other by employing two distinct performance measure parameters: F-measure and Accuracy.

IX. RESULTS OF PERFORMED EXPERIMENT

In the current section of the paper, the relationship among different feature sets and malware detection at the class level is submitted. The set of features is used as an input and presents the ratio of benign and malware apps within an experiment. F-measure and Accuracy are used as performance assessment parameters to match the performance of the Android malware detection model developed by using supervised machine learning algorithms.

A. Feature selection approach

Fig. 3 demonstrates the significant features, which help us to build the malware detection model. Black circle is significant feature set and blank rectangle is insignificant feature set.

B. Machine Learning Techniques

Eleven subsets of features (1 considering all set of extracted features + 10 resulting by implemented feature selection approaches) are used as an input to build a model for malware detection. Hardware used to carry out this study is the Intel Core i9 processor having a secondary memory of 1TB hard disk and primary memory of 16GB. Models are developed by using the MATLAB environment. Further, the performance of each detection model is measured by using two distinct performance parameters i.e., F-measure and Accuracy.

Tables 6 and 7, present the outcomes obtained for distinct data sets by utilizing DNN. Used abbreviations in this study are (FS1: Correlation best Feature Selection, FS2: Classifier Subset Evaluation, FS3: Filtered Subset Evaluation, FS4: Rough Set Analysis (RSA), FR1: Chi Squared test, FR2: Gain Ratio Feature Evaluation, FR3: Filtered Subset Evaluation, FR4: Information Gain Feature Evaluation, FR5: Logistic regression analysis, FR6: Principal.

Component Analysis (PCA) and AF: All Extracted features) From Tables 6 and 7, it may be concluded that:

– Model developed by considering features selected by Rough Set Analysis (FS4) as input is able to detect malware more effectively rather than model developed by using all extracted feature sets.

– From Table 6 and 7, we have seen that feature selection approach paid a serious effect on the outcome of the model developed for malware detection.

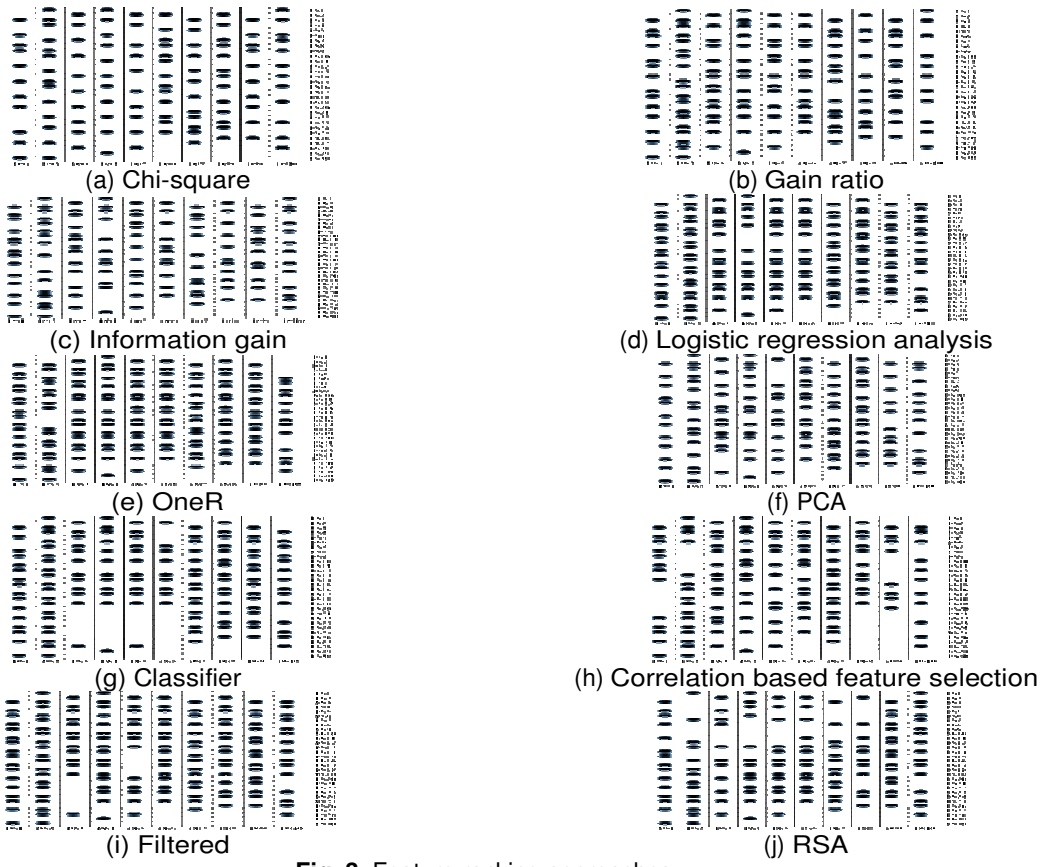


Fig. 3. Feature ranking approaches.

Table 6: Accuracy measured using different feature selection approaches.

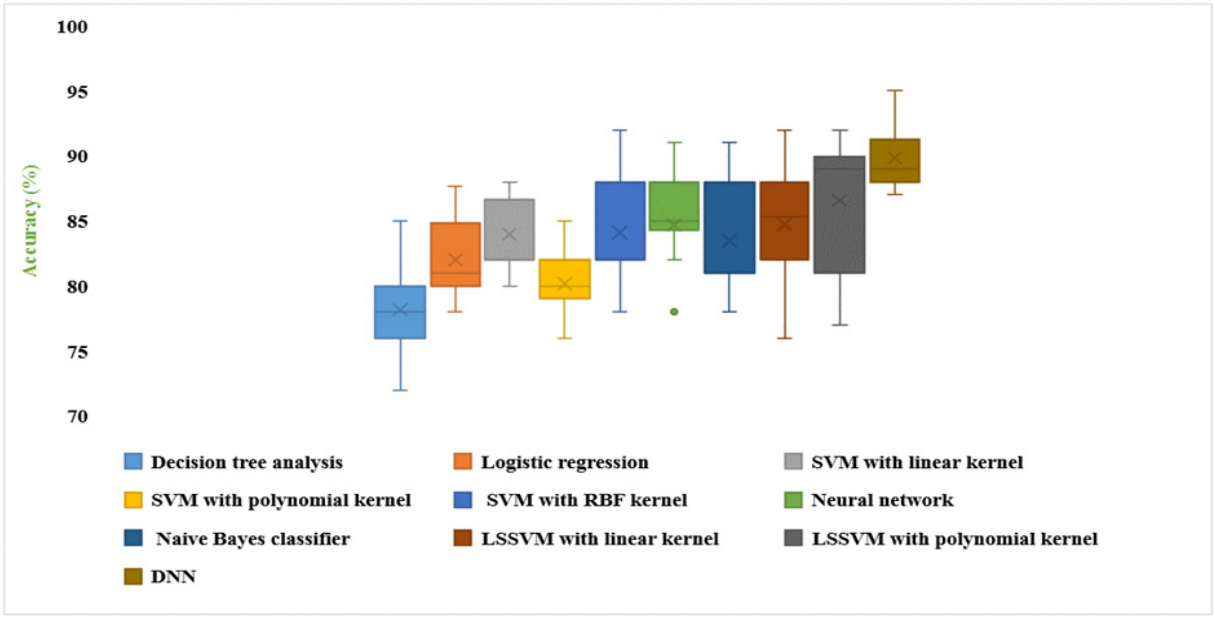
Accuracy											
ID	**AF**	**FR1**	**FR2**	**FR3**	**FR4**	**FR5**	**FR6**	**FS1**	**FS2**	**FS3**	**FS4**
DS1	68.3	81	84	86	86	82	83	85	86	83	89.8
DS2	65	80.8	84	87	86	85	82	85	86	89	91.8
DS3	67	81	84	87	82	85	83	81	84	89	90.8
DS4	62.8	78	81	89	86	83	89	85	87	89	90.7
DS5	68.8	81	83	80	81	86	87	82	83	85	89.8
DS6	67.9	85	87	86	85	84	88	89	92	94	96.7
DS7	78	81	85	88	89	89.6	88.7	86	86.8	89.7	93.8
DS8	65	78	75	78	82	84	85	86	87	88	91
DS9	68	84	87	92	91	83	84	96	95	93	86
DS10	66.8	78	86	88	89	82	89	89	89.8	89.7	97
DS11	79	88	88	86	86	89	89	80	86	88	98
DS12	66.8	82	88	82	86	81	83	88	87	89	90

Table 7: F-measure measured using different feature selection approaches.

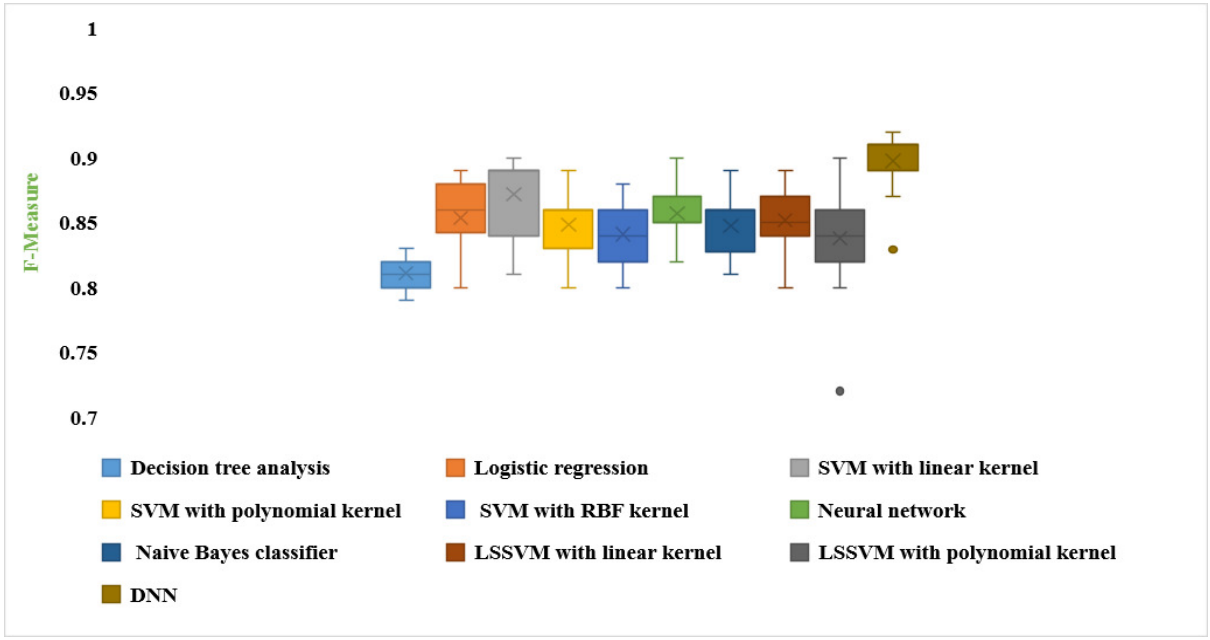
F-measure											
ID	**AF**	**FR1**	**FR2**	**FR3**	**FR4**	**FR5**	**FR6**	**FS1**	**FS2**	**FS3**	**FS4**
DS1	0.79	0.81	0.85	0.83	0.81	0.83	0.85	0.82	0.87	0.81	0.89
DS2	0.75	0.82	0.86	0.85	0.84	0.81	0.85	0.83	0.85	0.81	0.87
DS3	0.78	0.87	0.86	0.85	0.83	0.85	0.86	0.85	0.84	0.87	0.89
DS4	0.72	0.80	0.88	0.84	0.87	0.86	0.86	0.87	0.81	0.86	0.88
DS5	0.67	0.80	0.81	0.82	0.83	0.83	0.84	0.85	0.86	0.87	0.90
DS6	0.69	0.88	0.85	0.86	0.87	0.87	0.85	0.88	0.87	0.88	0.90
DS7	0.70	0.86	0.85	0.84	0.87	0.89	0.86	0.87	0.82	0.81	0.89
DS8	0.67	0.81	0.81	0.88	0.85	0.84	0.83	0.84	0.84	0.88	0.89
DS9	0.78	0.89	0.92	0.94	0.93	0.92	0.96	0.99	0.91	0.92	0.91
DS10	0.70	0.82	0.81	0.88	0.86	0.87	0.85	0.88	0.82	0.88	0.96
DS11	0.72	0.87	0.86	0.86	0.85	0.87	0.85	0.84	0.82	0.85	0.93
DS12	0.75	0.80	0.81	0.82	0.81	0.81	0.82	0.86	0.86	0.84	0.89

C. Evaluation of DLDroid with existing techniques available in the literature

(i) *Comparison of results with previously used classifiers:* In this study, we also makes the comparison with different most often used supervised machine learning approaches present in literature such as SVM with three distinct kernels i.e., linear, polynomial and RBF, Naïve Bayes classifier, Decision tree analysis, Logistic regression and Neural network. Fig. 4 demonstrates the box-plot diagrams for F-measure and Accuracy of commonly utilized classifiers.



(a)



(b)

Fig. 4. Diagram of box-plot showing performance of different classifiers.

On the basis of Fig. 4, we observed that DLDroid (DNN + FS4) has higher median value along with some outliers.

(ii) *Comparison of results with different Anti-Virus scanners:* Although our proposed model gives a better performance as compared to the machine learning technique used in the literature, in the end, it must be comparable with the common anti-virus products available in practice for Android malware detection. For this experiment, we select 10 different anti-viruses that are available in the market and applied them to our collected data set. For this experiment, we consider Android apps whose size is less than 50 MB. The performance of the proposed framework is comparatively better than many of the anti-viruses available in the experiment. Table 8 shows us the results of the experiment with anti-virus scanners.

Table 8: Comparison with distinct anti-virus scanners.

Name of the Anti-virus	Detection rate (in%)	Speed to detect malware in sec
Cyren	82	60
Ikarus	82.68	62
VIPRE	89	40
McAfee	89	30
AVG	90	32
AVware	92.8	30
ESETNOD32	92.9	20
CATQuickHeal	96.9	32
AegisLab	97.1	30
NANOAntivirus	96.2	20
DLDroid(our proposed framework)	97.9	12

The detection rate of the anti-viruses scanners varies considerably. Also, the best antivirus scanners detected 97.1% of the Android malware and certain scanners identified only 82% of the malicious samples, likely do not being specialized in detecting Android malware. By using 11,000 Android apps, DNN gives us the detection rate of 97.9% and outperforms equivalent to different anti-virus scanners. From this, we can say that our proposed framework is more efficient in detecting malware rather than the manually created definition of distinct anti-virus scanners.

(iii) *Experimental findings:* The comprehensive conclusion of our experimental work is presented in this section. The empirical study was performed for twelve distinct categories of Android apps by considering supervised machine learning techniques. Based on the experimental results, this research paper is able to answer the questions mentioned in section II.

RQ1: To address the RQ1, Tables 6 and 7 were analyzed. Here, it is found that the model build by utilizing FS4 is able to detect more malware from Android apps when compared to other approaches.

RQ2: In the present paper, feature selection approach is used to identify the smaller subset of features. By utilizing this, we considered the best possible subsets of the features, which helps to develop a model to identify whether an app is benign or malware. Based on the experimental results mentioned in Tables 6 and 7, it indicates that in number of cases there occurs a reduced subset of features, which are best for building a detection model when compared to all the extracted features.

X. CONCLUSION AND FUTURE SCOPE

This work is emphasized on developing a malware detection framework by using a selected set of features that help us to identify that an Android app belongs to malware class or benign class. The experiment was performed by taking assistance of twelve distinct categories of Android apps.

Our submissions after performing the experiment are the following:

– Empirical results specify that it is feasible to identify a small subset of features. Malware detection model developed by considering a small set of features is able to detect malware and benign apps with the inferior value of misclassified errors and better accuracy.

– Based on experimental findings, we observed that considering feature selection approaches helps to reduce the feature sets. The result of models build by using feature selection approaches perform better when compared to all extracted feature sets.

– Based on the proposed detection framework, it is seen that model build by utilizing FS4 is capable to detect 97.9% unknown malware from real-world apps.

In this research paper, we proposed the malware detection model that detects only whether an app is malware or benign. Further, work can be extended to develop a model for malware detection, which predicts whether a particular feature is capable to detect malware, or not. Moreover, this study can be replicated over other Android apps repository, which utilized soft computing models to attain a better detection rate for malware.

Conflict of Interest. No conflict of interest.

REFERENCES

[1]. https://www.up.ac.za/news/post_2880755-covid-19-why-it-matters-that-scientists-continue-their-search-for-source-of-patient-zeros-infection-

[2]. https://www.mygov.in/aarogya-setu-app/

[3]. https://www.who.int/mediacentre/multimedia/app/en/

[4]. https://www.aa.com.tr/en/europe/italy-to-use-app-to-track-coronavirus-contacts/1808841

[5]. https://www.businessinsider.in/tech/news/singapore-is-using-a-high-tech-surveillance-app-to-track-the-coronavirus-keeping-schools-and-businesses-open-heres-how-it-works-/articleshow/74797714.cms

[6]. https://www.gdatasoftware.co.uk/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware

[7]. https://en.wikipedia.org/wiki/Google_Playhttps://www.gdatasoftware.co.uk/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware

[8]. https://www.eweek.com/security/google-bouncer-vulnerabilities-probed-by-security-researchers

[9]. https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf

[10]. Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., & Bharmal, A. (2013). Andro Similar: robust statistical feature signature for Android malware detection. In *Proceedings of the 6th International Conference on Security of Information and Networks*, 152-159.

[11]. Mahindru, A., & Sangal, A. L. (2020). Feature-Based Semi-supervised Learning to Detect Malware from Android. In *Automated Software Engineering: A Deep Learning-Based Approach*, 93-118.

[12]. Mahindru, A., & Sangal, A. L. (2020). PerbDroid: Effective Malware Detection Model Developed Using Machine Learning Classification Techniques. In *A Journey Towards Bio-inspired Techniques in Software Engineering*, 103-139.
[13]. Mahindru, A., & Singh, P. (2017). Dynamic permissions based android malware detection using machine learning techniques. In *Proceedings of the 10th innovations in Software Engineering Conference*, 202-210.
[14]. http://dx.doi.org/10.17632/k4rt99sfbt.2
[15]. http://anubis.iseclab.org/
[16]. Xu, R., Saïdi, H., & Anderson, R. (2012). Aurasium: Practical policy enforcement for android applications. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)* (pp. 539-552). [17]. http://copperdroid.isg.rhul.ac.uk/copperdroid/index.php
[18]. Mahindru, A., & Sangal, A. L. (2019). DeepDroid: Feature Selection approach to detect Android malware using Deep Learning. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 16-19.
[19]. Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 15-26.
[20]. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., & Sheth, A. N. (2014). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, *32*(2), 1-29.
[21]. Lydia, E. L., Sharmil, N., Shankar, K., & Maseleno, A. (2019). Analysing the Performance of Classification Algorithms on Diseases Datasets. *International Journal on Emerging Technologies*, *10*(3), 224–230.
[22]. 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.
[23]. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, *38*(1), 161-190.
[24]. Mas'ud, M. Z., Sahib, S., Abdollah, M. F., Selamat, S. R., & Yusof, R. (2014, May). Analysis of features selection and machine learning classifier in android malware detection. In *2014 International Conference on Information Science & Applications (ICISA)*, 1-5.
[25]. Narayanan, A., Chandramohan, M., Chen, L., & Liu, Y. (2018). A multi-view context-aware approach to Android malware detection and malicious code localization. *Empirical Software Engineering*, *23*(3), 1222-1274.
[26]. https://www.apkmirror.com/
[27]. https://www.allfreeapk.com/
[28]. http://dx.doi.org/10.17632/k4rt99sfbt.2
[29]. https://www.virustotal.com/gui/home
[30]. http://dx.doi.org/10.17632/b4mxg7ydb7.3
[31]. https://developer.android.com/guide/topics/permissions/overviewr

How to cite this article: Mahindru, A. and Sangal, A. L. (2020). DLDroid: Feature Selection based Malware Detection Framework for Android Apps developed during COVID-19. *International Journal on Emerging Technologies*, *11*(3): 516–525.