MACHINE LEARNING BASED ENSEMBLE CLASSIFIER FOR ANDROID MALWARE DETECTION

P Sumalatha[1] and G.S. Mahalakshmi[2]

[1]Associate Professor, Bhoj Reddy Engineering College for Women,
Hyderabad, and Telangana, India.
[2]Associate Professor, Anna University, Chennai, Tamilnadu, India.

ABSTRACT

Malware problem has infiltrated into every aspect of cyber space including Android mobiles. Due to proliferation of Android applications and widespread usage of smartphones, malware problem is causing significant damage to mobile users and application vendors. With the emergence of Artificial Intelligence (AI), machine learning (ML) models are widely used for detection of Android malware. However, many of the existing methods focused on static or dynamic data to train classifiers for malware detection. In this paper, we propose an ensemble model with intelligent methods that are empirically selected. Only the malware detection models with highest accuracy are chosen to be part of stacking ensemble model. An algorithm named Stacking Ensemble for Automatic Android Malware Detection (SE-AAMD)is proposed and implemented. We made three experiments with the same algorithm but three different datasets reflecting features obtained through different modus operandi. Each dataset is found to have influence on the performance of the models. However, in all experiments, the ensemble approach showed highest performance. The proposed method can be used in improving security for Android devices and applications.

KEYWORDS

Artificial Intelligence, Machine Learning, Anomaly Detection, Android Malware Detection

1. INTRODUCTION

Malware is malicious program that has capacity to infect systems and networks. Adversaries are spreading malware for monetary and other benefits. This malware propagation is spread to applications in all walks of life in cyberspace. Android platform is no exception. Of late ML models have proved to be efficient due to their learning process in malware detection. However, it is important to have a methodology that considers different aspects of the detection mechanism [1]. Traditionally there are methods that concentrated on either static or dynamically extracted information from Android applications for understanding and analysing towards detection of presence of malicious programs. However, with the emergence of AI-based techniques such as ML models, researchers attracted towards this learning based phenomena to have more scalable approach in detection of such malicious programs and their propagation in Android platform.

There are many existing methods such as [5], [8], [13] that focused on ML models for malware detection in Android environments. Milosevic *et al*. [5] focused on learning based approach with feature extraction and ML models to automate malware detection process. Naser *et al*. [8] proposed a method for the malware detection in Android devices by considering API calls and also permissions along with ML models. Since malware influences permissions and causes API calls, their method assumes significance. Kaijun *et al*. [13] studied different malware detection

methods that involve in learning-based techniques and found that they could perform better than traditional methods. Some researchers proposed tool, as explored in [16], [22], [24] for automatic detection of malware using ML. Sercan *et al*. [16] proposed a detection a tool known as And MFC that is meant for automatically finding malware and classify it in Android platform. Xiao *et al*. [22] proposed and implemented a tool named AndroidHIV which is meant for malware repackaging experiments that were meant for evading malware detection process. Fauzia *et al*. [24] proposed a tool named PIndroid that is a system for detection of malware exploits ensemble models in ML. Few hybrid approaches are also found in literature as studied in [7], [10] and [17]. Importantly different ensemble methods are covered in the literature as in [11], [24] and [31]. However, the novelty of our method in this paper is that the constituent selection is made with a threshold-based approach. From the literature it is found that many of the existing methods focused on static or dynamic data to train classifiers for malware detection. Moreover, the ensemble models could improve performance but there is room for improving accuracy in detection of malware. Our contributions in the paper are as follows.

1. We proposed an ML-based ensemble framework that could improve malware detection performance in Android platform.
2. We proposed an algorithm named Stacking Ensemble for Automatic Android Malware Detection (SE-AAMD). The ensemble method is based on carefully chosen and best performing classifiers.
3. We made empirical study to evaluate performance of SE-AAMD and compared with existing models. We found that our algorithm outperforms all constituent ML models. The remainder of the paper is structured as follows. Section 2 reviews existing methods for finding presence of malware in Android platform. Section 3 presents our stacking ensemble methodology which has novel approach in constituent classifier selection. Section 4 presents experimental results and Section 5 concludes our work.

2. RELATED WORK

This section reviews literature on different existing ML-based methods for malware detection. Yerima and Khan [1] investigated on different ML models and their performance in malware detection process. Potha *et al*. [2] proposed an ensemble method for improving efficiency in malware detection. It is based on random ensemble phenomenon which exploits extrinsic training instances along with many pre-determined instances. Their algorithm has mechanisms to find final malware score for each instance through different methods and ensemble the classifiers. Yerima *et al*. [3] proposed an APK analyser method to extract features and use them to train ML classifier towards malware detection process. They found that Random Forest is an ensemble approach showed better performance. Chen *et al*. [4] proposed a system known as Secure Droid for improving security of mobile applications form malware problems. It extracts features and performs ensemble learning approach to determine class labels. Milosevic *et al*. [5] focused on learning based approach with feature extraction and ML models to automate malware detection process. Kouliaridis *et al*. [6] investigated on different ML techniques and found that they are feasible in detection of malware with good quality in detection results. Lilian *et al*. [7] followed a hybrid approach that includes feature engineering and also an ensemble classification method for detecting Android malware. They observed significance of feature engineering in improving accuracy. Naser *et al*. [8] proposed a method for malware detection in Android devices by considering API calls and also permissions. Since malware influences permissions and causes API calls, their method assumes significance. Zhuo *et al*. [9] focused on malware detection in Android with the help of malware related control flow graphs and also algorithms that learn from given historical data. Alejandro *et al*. [10] incorporated novelty in their research by exploiting a hybrid feature selection and fusion besides considering ensemble approach in detection of malware.

Pengbin *et al*. [11] proposed a dynamic approach in malware detection. It has provision for dynamically extracted features and then use them to train ensemble classifiers towards improving detection accuracy. Suleiman *et al*. [12] proposed a malware detection system for Android. It is known as DroidFusion which has a classifier fusion approach at multiple levels towards improving focus on malware detection process. Kaijun *et al*. [13] studied different malware detection methods that involve in learning-based techniques and found that they could perform better than traditional methods. Shahzeb *et al*. [14] focused on SDN based systems in distributed environment. In such environment, their focus was on building an ensemble classifier based on deep learning to detect anomalies. Dehkordy *et al*. [15] investigated on different aspects of datasets including imbalanced datasets that are to be treated differently. With such imbalanced data, they proposed a ML based method for malware detection. Sercan *et al*. [16] proposed a detection tool known as AndMFC that is meant for automatically finding malware and classify it in Android platform. Surendran *et al*. [17] proposed a TAN-assisted approach with hybrid methodology to detect malware. Their methodology has revealed the significance of TAN usage in the process of malware detection. Rana *et al*. [18] studied ML models with the view of malware detection in mind. Their methodology showed that it could have improved detecting performance. Daoudi *et al*. [19] investigated on possible reproducibility of existing ML models that are used for detecting malware. Their research could find that the models can be reused proper environment and datasets. Zhenxiang *et al*. [20] investigated network traffics that reflect high imbalances towards detection of malware. Their methodology used ML models in such datasets.

Xin *et al*. [21] focused on the notion of information fusion in order to improve malware detection probability using ML techniques. Their empirical study is made on parallel processing environment a found the improvement in faster processing of data. Xiao *et al*. [22] proposed and implemented a tool named AndroidHIV which is meant for malware repackaging experiments that were meant for evading malware detection process. Zainab *et al*. [23] focused on ML models for malware detection and performed quantification of evasion attacks from adversaries. Fauzia *et al*. [24] proposed a tool named PIndroid that is a system for detection of malware exploits ensemble models in ML. Mohammed *et al*. [25] defined a methodology named as DL-Droid that makes use of deep learning techniques to detect malware automatically in Android platform. Other important research contributions include Anastasia tool [26], pragmatic detection function [27], learning-based framework [28], Tree-based ensemble [29], ensemble optimization [30] and stacking ensemble [31]. In [33] and [34], it is observed that deep learning based approaches do have their impact on solving problems in various domains. From the literature it is found that many of the existing methods focused on static or dynamic data to train classifiers for malware detection. Moreover, the ensemble models could improve performance but there is room for improving accuracy in detection of malware.

3. MATERIALS AND METHODS

We proposed a ML-based methodology for malware detection. It is an intelligent approach that has novel constituent selection procedure for ensemble classifiers. Android malware dataset is used for the empirical study with three kinds of focus on the data.

3.1. The Framework

We proposed a framework for automatic detection of malware with ensemble learning approach. The outline of the proposed framework, known as Stacked Ensemble Framework (SEF), is provided in Figure 1 and its complete architecture is illustrated in Figure 2. It is based on ML models along with ensemble and threshold-based constituent selection method to improve

detection performance. The framework emphasizes more accurate means of making ensemble with stacking approach after choosing best constituent classifiers.

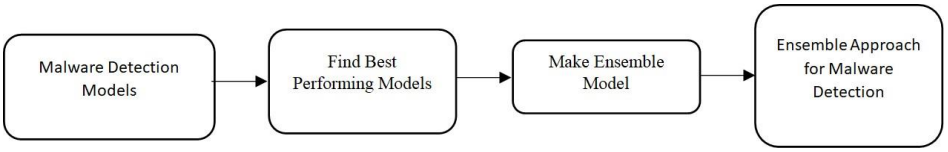

Figure 1. Outline of our methodology

We have considered many ML classifiers for the empirical study. It is data-driven approach where supervised learning plays crucial role in learning from training data and perform detection process based on the learned knowhow. We have set a threshold of 90% accuracy to be qualified for classifiers to participate in ensemble model. This is meant for ensuring higher level of accuracy in detection process. The stacking ensemble model is used as it could lead to better performance in malware detection.

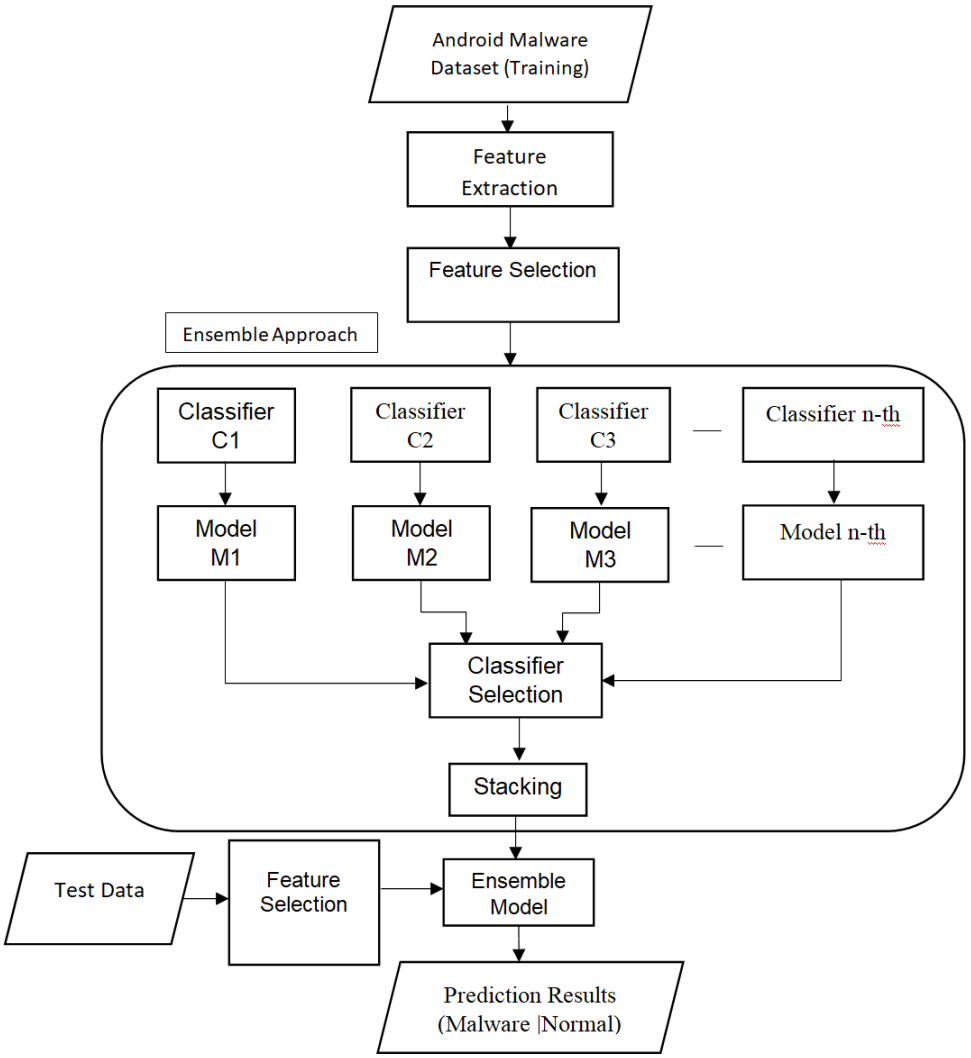

Figure 2. Overview of proposed Stacked Ensemble Framework (SEF)

Our method has merits in terms of choosing constituent classifiers and making a stacking ensemble to realize SEF. There is feature selection process that is based on three kinds of data used for three different experiments. With all the three experiments, feature selection is made in order to improve quality in the learning process. Only selected features are used to train the prediction models. While making ensemble of different classifiers, they are filtered based on their performance with the given threshold that is >90%. In Figure 2, classifier refers to an algorithm while model refers to a trained or learned model. Since the training data is labelled data, the model has discriminative power. The classification results given by constituent models are finally determined with stacking ensemble approach for more efficient detection of malware. Finally, our framework assigns class label for each test instance as either NORMAL or MALWARE based on the intelligence the proposed model gains.

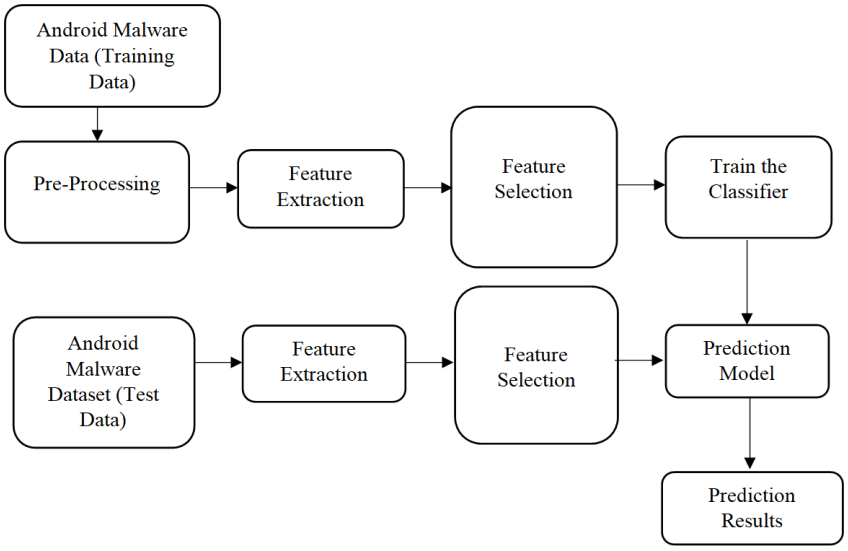

Figure 3. Training and Testing process

As illustrated in Figure 3, the given Android malware data (training data) is undergoing pre-process to improve quality of data. Then it has feature extraction and feature selection that is crucial for leveraging learning accuracy. Then the training is given to classifier based on the selected features only. With regard to testing phase, the given test data contains no class labels. In other words, it is the data to be used by the proposed system to detect presence of Android malware. In other words, class labels are to be predicted by the system. A learned model is used to work on the test data to classify given samples into MALWARE or NORMAL.

3.2. Dataset Description

Datasets used in this research are obtained from [32]. The data is named **CICAndMal2017**. It has both benign and malware samples. The total number of samples available in the online resource is 10854. The dataset has 215 features. Some of the important features are shown in Table 1.

Table.1 some of the important features in the dataset

S.no	Feature
1	Transact,
2	android.os.Binder,
3	SEND_SMS,
4	Ljava.lang.Class.getCanonicalName,
5	Ljava.lang.Class.getMethods,
6	Ljava.lang.Class.cast,
7	READ_CONTACTS,
8	DEVICE_POWER,
9	HARDWARE_TEST,
10	ACCESS_WIFI_STATE,
11	WRITE_EXTERNAL_STORAGE,
12	ACCESS_FINE_LOCATION,
13	SET_WALLPAPER_HINTS,
14	SET_PREFERRED_APPLICATIONS,
15	WRITE_SECURE_SETTINGS,
16	class

3.3. Feature Selection

Featue selection plays crucial role in improving learninig quality in supervised models. In this paper we applies feature extraction procedure based on the given kind of dataset for each experiment. Then from the chosen features, class label prediction contribution is used to analyze importance of each feature. Based on the feature importance, highly contributing features are selected for training classifiers.

Table.2 The selected features after performing feature selection

S.no	Feature
1	Transact,
2	android.os.Binder,
3	SEND_SMS,
4	Ljava.lang.Class.getMethods,
5	Ljava.lang.Class.cast,
6	READ_CONTACTS,
7	ACCESS_WIFI_STATE,
8	ACCESS_FINE_LOCATION,
9	SET_WALLPAPER_HINTS,
10	WRITE_SECURE_SETTINGS,

The selected feature are taken as input to the models and the proposed models will be trained with respect to classified features and class label. The prediction will be done based on the model which shows better prformance after features selection methods had implemented.

3.4. Stacking Ensemble Approach

In our framework SEF, stacking ensemble approach is used. It is an approach that has provision to combine many constituent ML models or regression models with the help of a meta-regressor. The constituent ML models are trained with entire training data as per feature selection process. Then the feature outcomes of those models are used to train the meta-regressor in order to enhance detection accuracy. It mainly involves combining the predictions from multiple machine

learning models on the same dataset, like bagging and boosting. As we import the module called stacking CVC classifier from the mlxtend package which represents extensions for ensemble techniques in data science tasks. The main benefit of stacking approach is that it can harness the capabilities of range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble.

3.5. Algorithm Design

We proposed an algorithm known as Stacking Ensemble for Automatic Android Malware Detection (SE-AAMD) is proposed and implemented. We made three experiments with the same algorithm but three different datasets reflecting features obtained through different modus operandi. Each dataset is found to have its influence on the performance of the models. Our approach to constituent models selection is illustrated in Figure 4.

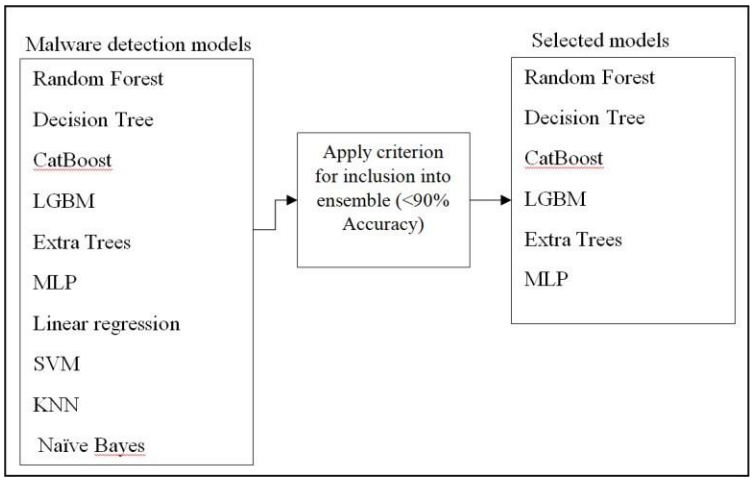

Figure 4. Shows the baseline models chosen based on given criterion

There are different ML models considered as input to the selection process. However, with empirical study, each model is evaluated and if the model performs with accuracy >=90%, only such models are chosen for ensemble.

Algorithm: Stacking Ensemble for Automatic Android Malware Detection (SE-AAMD)
Inputs:
Android malware dataset *D*
Intelligent prediction models in pipeline *P*
Output:
Malware detection results *R'*

1. Begin
2. Initialize models map *M*
3. Initialize ensemble map *E*
4. *F*←FeatureExtraction(*D*)
5. (*T1*, *T2*)←DataPreProcess(*D*)

Training ML Models

6. For each constituent model *p* in *P*
7. *t*←Train *p* using *F*
8. Add p and *t* to *M*
9. End For

Testing ML Models

10. For each map entry*m* in *M*
11. Test model for *T2* with *F*
12. Add model results to *R*
13. Compute confusion matrix
14. Updated *E* with model and *R*
15. End For

Ensemble Decision

16. *R'*←ApplyStackingModel(*E*)
17. Display *R'*
18. End

Algorithm 1. Hybrid Ensemble and Feature Engineering for Stroke Prediction (HEFE-SP) algorithm

As presented in Algorithm 1, it takes patient brain stroke dataset D and selected stroke prediction classifiers B as pipeline and produces output in the form of stroke classification results R'. The given dataset is split into 80% and 20% training set and testing set respectively. Both training and test data are subjected to feature engineering using our proposed algorithm HMA-FE. The selected baseline models are trained and then they are used as ensemble model. The ensemble model results in final predictions based on weighted majority voting approach. Stacking ensemble used in SE-AAMD harnesses capabilities of models while making final predictions. It also helps in improving prediction accuracy. Its limitation is that prediction accuracy is based on capabilities of constituent models used in the ensemble. Moreover the constituent models we chose for ensemble or carefully selected with some criteria could improve its performance when comapared with the state of the art.

3.6. Evaluation Methodology

Performance of different prediction models is evaluated using confusion matrix-based measures like precision, recall, F-measure and accuracy. These measures are based on the predictions of the models in terms of number of true positives (TP), number of true negatives (TN), number of false positives (FP) and number of false negatives (FN). A performance metric known as AUC (Area Under Curve) is used for evaluation of performance. It is based on two metrics known as True Positive Rate (TPR) and False Positive Rate (FPR).

$$\text{TPR} = \frac{TP}{TP+FN} \quad (1)$$

$$FPR = \frac{FP}{FP+TN} \quad (2)$$

Computation of performance metrics is carried out as in Eq. 1 and Eq. 2. These metrics hold a value between 0.0 and 1.0 reflecting the least and highest performance. More in value for TPR or FPR indicates higher performance.

4. RESULTS AND DISCUSSION

This section presents results of our three experiments. Each experiment is given with different data but includes constituent methods and stacking ensemble as well. Each experiment has shown that the stacking ensemble approach is performing better than existing models.

4.1. Results of First Experiment

This section shows results of first experiment made for automatic android malware detection process.

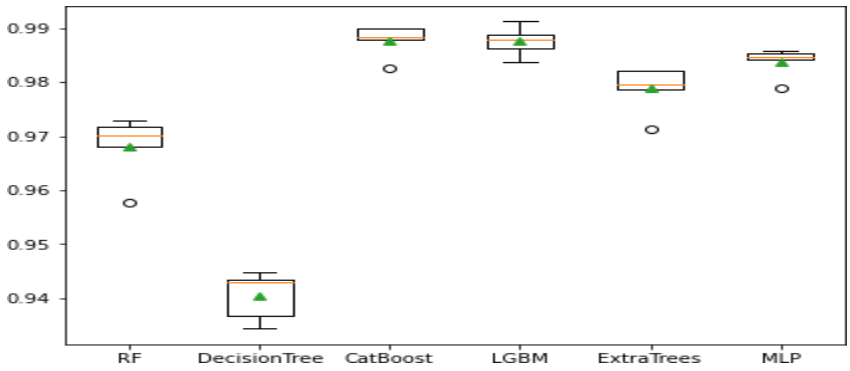

Figure 5. Shows AUC of constituent models

As presented in Figure 5, the AUC of different constituent models in the first experiment is provided. Higher in AUC indicates better performance.

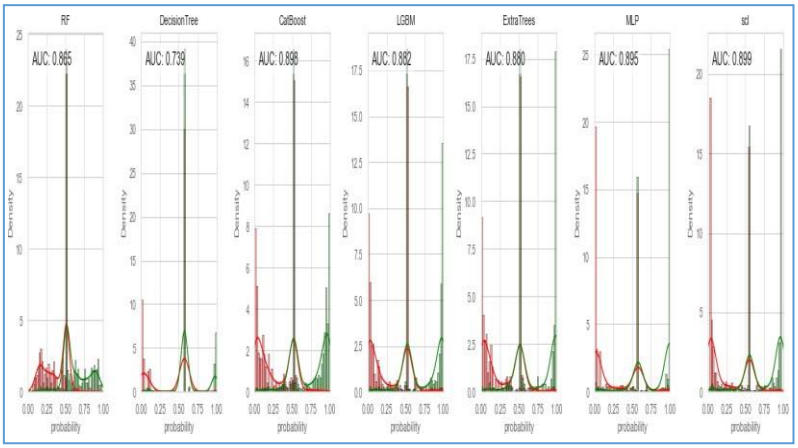

Figure 6. Show AUC analysis for each constituent model and stacking ensemble model

As presented in Figure 6, the AUC analysis of different constituent models in the first experiment is provided along with stacking ensemble model.

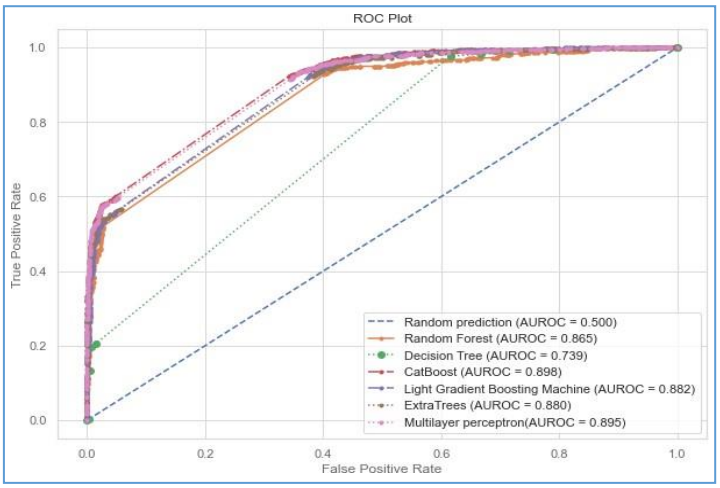

Figure 7. ROC performance of constituent models

As presented in Figure 7, ROC plot that makes use of TPR and FPR values for experiment 1 is provided to show performance comparison. Higher ROC indicates better performance.

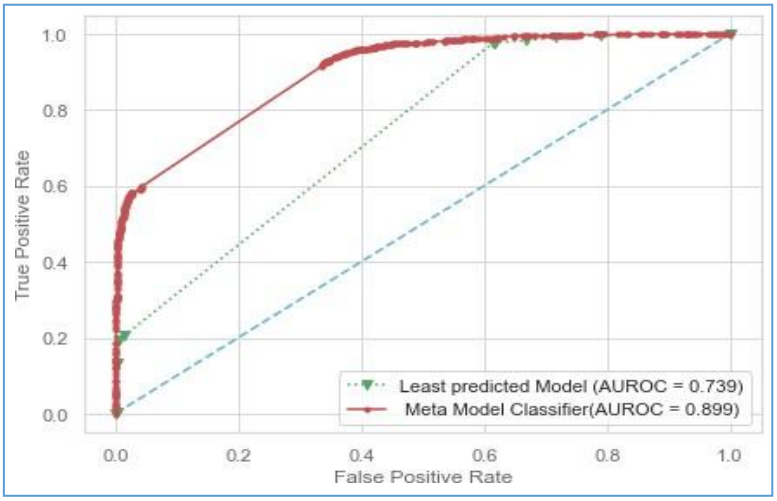

Figure 8. Shows ROC of least predicted model and the proposed meta-regressor model

As presented in Figure 8, the proposed meta-regressor model showed the highest performance in terms of ROC in the first experiment.

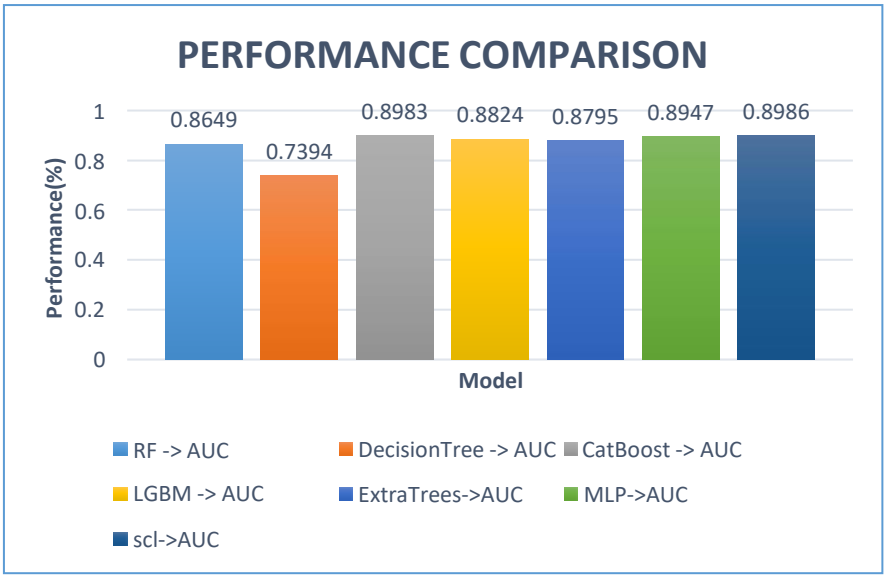

Figure 9. Performance comparison between constituent models and ensemble model in first experiment

As presented in Figure 9, the AUC performance which reflects malware detection accuracy is compared among all constituent models and ensemble model. The least AUC is exhibited by DT with 73.94%. RF showed 86.49%, CatBoost 89.83,LGBM 88.24%, ExtraTrees 87.95% and MLP 89.7%. However, the highest accuracy is achieved by stacking ensemble model proposed with 89.86% accuracy. The performance improvement of stacking ensemble model is due to the fact that it exploits knowledge of constituent models to leverage prediction performance.

4.2. Results of Second Experiment

This section shows results of second experiment made for automatic android malware detection process.

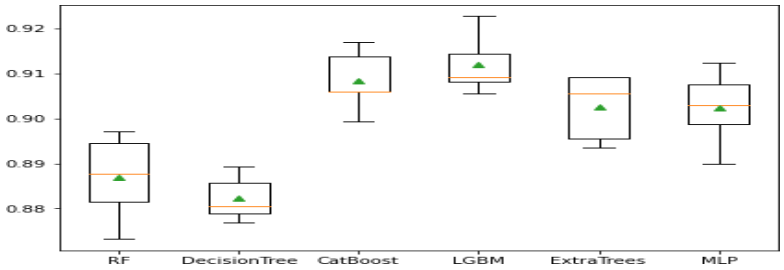

Figure 10. Shows AUC of constituent models

As presented in Figure 10, the AUC of different constituent models in the second experiment is provided.

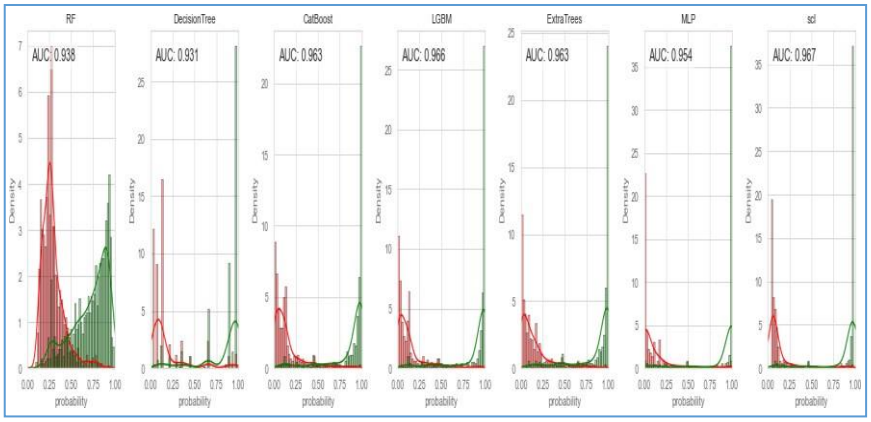

Figure 11. Show AUC analysis for each constituent model and stacking ensemble model

As presented in Figure 11, the AUC analysis of different constituent models in the second experiment is provided along with stacking ensemble model.

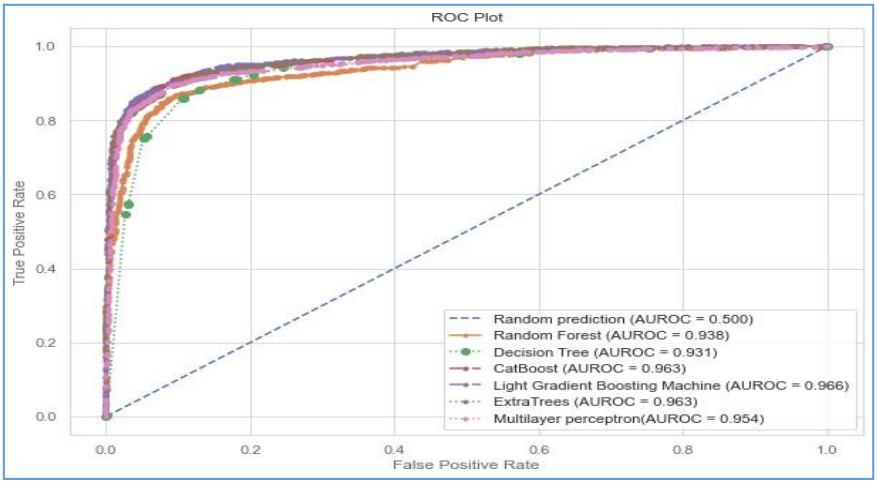

Figure 12. ROC performance of constituent models

As presented in Figure 12,ROC plot that makes use of TPR and FPR values for experiment 2 is provided to show performance comparison.

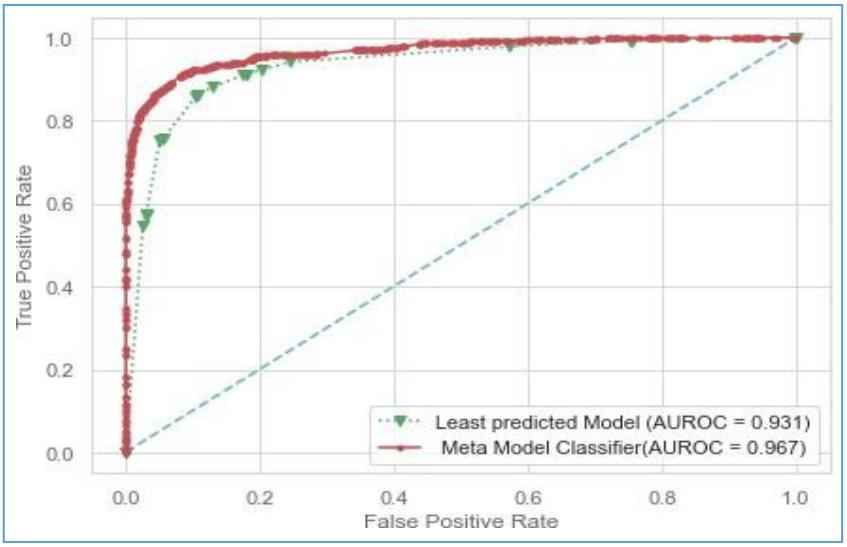

Figure 13. Shows ROC of least predicted model and the proposed meta-regressor model

As presented in Figure 13, the proposed meta-regressor model showed the highest performance in terms of ROC in the second experiment.

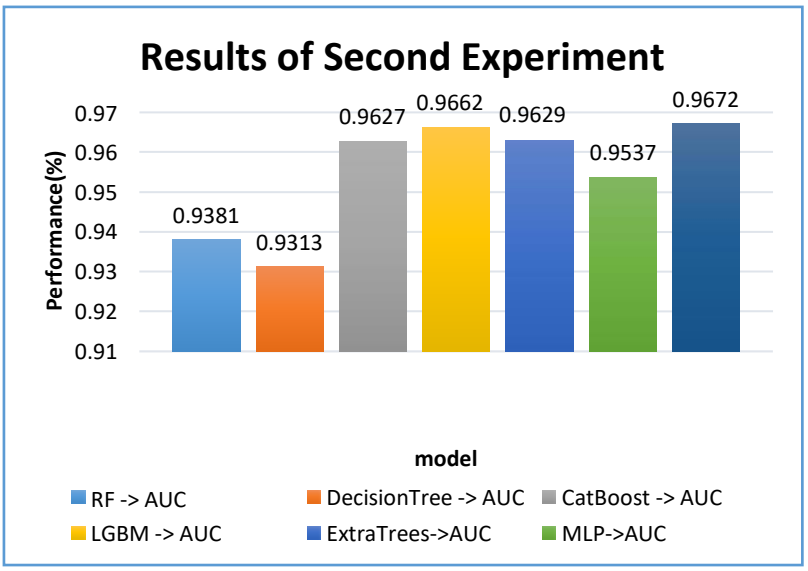

Figure 14. Performance comparison between constituent models and ensemble model in second experiment

As presented in Figure 14, the AUC performance which reflects malware detection accuracy is compared among all constituent models and ensemble model. The least AUC is exhibited by DT with 93.13%. RF showed 93.81%, CatBoost 96.27, LGBM 96.62%, ExtraTrees 96.29% and MLP 95.37%. However, the highest accuracy is achieved by stacking ensemble model proposed with 96.72% accuracy. The performance improvement of stacking ensemble model is due to the fact that it exploits knowledge of constituent models to leverage prediction performance.

4.3. Results of Third Experiment

This section shows results of third experiment made for automatic android malware detection process.

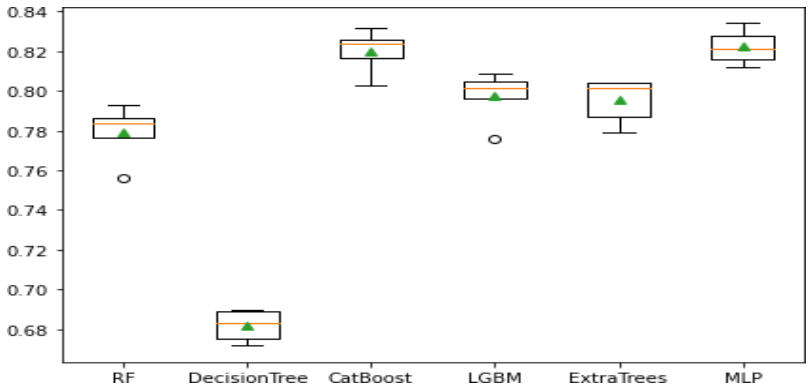

Figure 15. Shows AUC of constituent models

As presented in Figure 15, the AUC of different constituent models in the third experiment is provided.

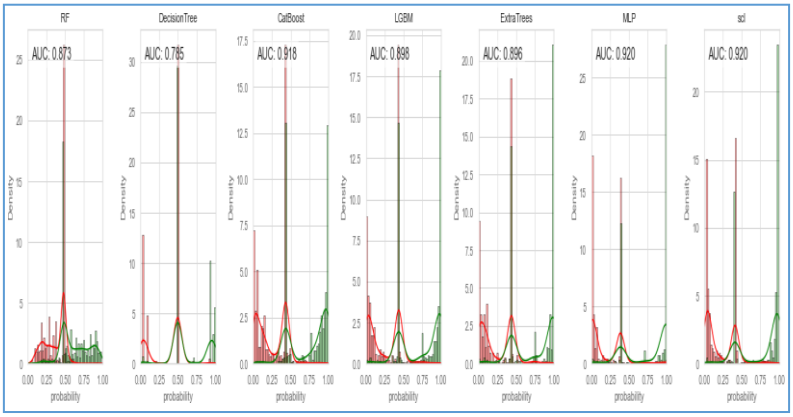

Figure 16. Show AUC analysis for each constituent model and stacking ensemble model

As presented in Figure 16, the AUC analysis of different constituent models in the third experiment is provided along with stacking ensemble model.

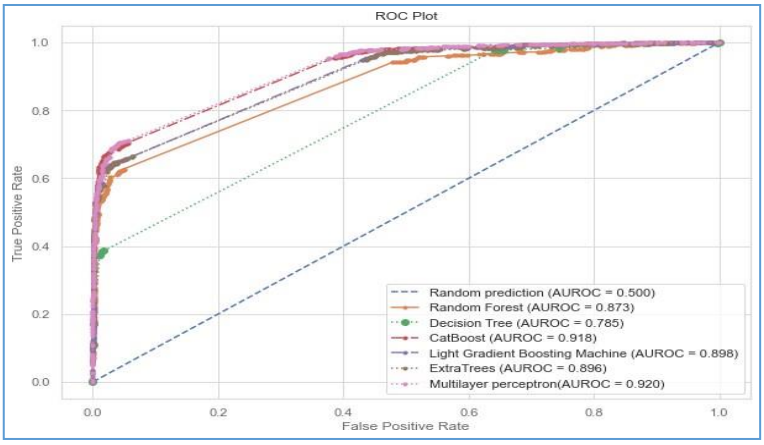

Figure 17. ROC performance of constituent models

As presented in Figure 17, ROC plot that makes use of TPR and FPR values for experiment 3 is provided to show performance comparison.

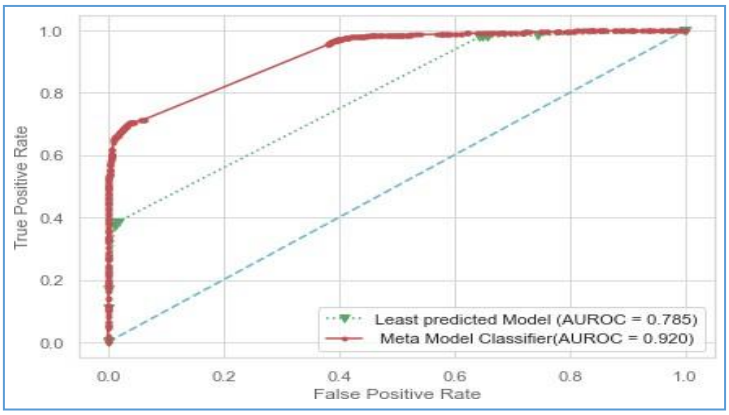

Figure 18. Shows ROC of least predicted model and the proposed meta-regressor model

As presented in Figure 18, the proposed meta-regressor model showed the highest performance in terms of ROC in the third experiment.

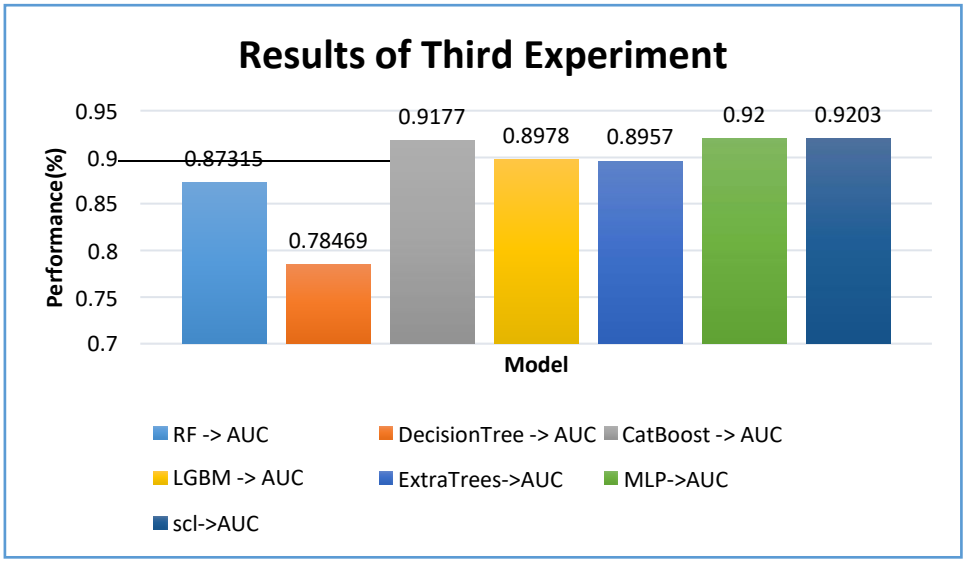

Figure 19. Performance comparison between constituent models and ensemble model in third experiment

As presented in Figure 19, the AUC performance which reflects malware detection accuracy is compared among all constituent models and ensemble model. Least AUC is exhibited by DT with 73.94%. RF showed 96.49%, CatBoost 89.83, LGBM 88.24%, ExtraTrees 87.95% and MLP 89.7%. However, the highest accuracy is achieved by stacking ensemble model proposed with 89.86% accuracy. The performance improvement of stacking ensemble model is due to the fact that it exploits knowledge of constituent models to leverage prediction performance.

5. CONCLUSION

In this paper, we propose an ensemble model with intelligent methods that are empirically selected. Only the malware detection models with the highest accuracy are chosen to be part of stacking ensemble model. An algorithm named Stacking Ensemble for Automatic Android Malware Detection (SE-AAMD) is proposed and implemented. We made three experiments with the same algorithm but three different datasets reflecting features obtained through different modus operandi. Each dataset is found to have its influence on the performance of the models. However, in all experiments, the ensemble approach showed the highest performance. The proposed ensemble model exhibited highest accuracy with 96.72% which is better than many ML models such as RF, DT, CatBoost, LGBM, ExtraTrees and MLP. The proposed method can be used in improving security for Android devices and applications. In future we intend to investigate on deep learning models for Android malware detection.

REFERENCES

[1] Yerima, Suleiman Y.; Khan, Sarmadullah (2019). [IEEE 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security) - Oxford, United Kingdom (2019.6.3-2019.6.4)] Longitudinal performance analysis of machine learning-based Android malware detectors. , p1–8.

[2] Potha, Nektaria; Kouliaridis, V.; Kambourakis, G. (2020). An extrinsic random-based ensemble approach for android malware detection. Connection Science, p1–17.

[3] Muttik, Igor; Yerima, Suleiman Y.; Sezer, Sakir (2015). High accuracy android malware detection using ensemble learning. IET Information Security, 9(6), p313–320.

[4] Chen, Lingwei; Hou, Shifu; Ye, Yanfang (2017). [ACM Press the 33rd Annual Computer Security Applications Conference - Orlando, FL, USA] Proceedings of the 33rd Annual Computer Security Applications Conference on - ACSAC 2017 - SecureDroid. , p362–372.

[5] Milosevic, Nikola; Dehghantanha, Ali; Choo, Kim-Kwang Raymond (2017). Machine learning aided Android malware classification. Computers & Electrical Engineering, p1-9.

[6] Vasileios Kouliaridis; Georgios Kambourakis; (2021). A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection . Information, p1-12.

[7] Coronado-De-Alba, Lilian D.; Rodriguez-Mota, Abraham; Ambrosio, Ponciano J. Escamilla- (2016). [IEEE 2016 8th IEEE Latin-American Conference on Communications (LATINCOM) - Medellin, Colombia (2016.11.15-2016.11.17)] Feature selection and ensemble of classifiers for Android malware detection. , p1–6.

[8] Peiravian, Naser; Zhu, Xingquan (2013). [IEEE 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI) - Herndon, VA, USA (2013.11.4-2013.11.6)] Machine Learning for Android Malware Detection Using Permission and API Calls. , p300–305.

[9] Ma, Zhuo; Ge, Haoran; Liu, Yang; Zhao, Meng; Ma, Jianfeng (2019). A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. IEEE Access, p1–11.

[10] Martín, Alejandro; Lara-Cabrera, Raúl; Camacho, David (2018). Android malware detection through hybrid features fusion and ensemble classifiers: the AndroPyTool framework and the OmniDroid dataset. Information Fusion, p1-19.

[11] Feng, Pengbin; Ma, Jianfeng; Sun, Cong; Xu, Xinpeng; Ma, Yuwan (2018). A Novel Dynamic Android Malware Detection System With Ensemble Learning. IEEE Access, p1–16.

[12] Yerima, Suleiman Y.; Sezer, Sakir (2018). DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. IEEE Transactions on Cybernetics, p1–14.

[13] Liu, Kaijun; Xu, Shengwei; Xu, Guoai; Zhang, Miao; Sun, Dawei; Liu, Haifeng (2020). A Review of Android Malware Detection Approaches based on Machine Learning. IEEE Access, p1–30.

[14] Haider, Shahzeb; Akhunzada, Adnan; Ahmed, Ghufran; Raza, Mohsin (2019). [IEEE 2019 UK/ China Emerging Technologies (UCET) - Glasgow, United Kingdom (2019.8.21-2019.8.22)] Deep Learning based Ensemble Convolutional Neural Network Solution for Distributed Denial of Service Detection in SDNs. , p1–4.

[15] Diyana Tehrany Dehkordy;Abbas Rasoolzadegan; (2021). A new machine learning-based method for android malware detection on imbalanced dataset . Multimedia Tools and Applications, p1-22.

[16] Turker, Sercan; Can, Ahmet Burak (2019). [IEEE 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops) - Istanbul, Turkey (2019.9.8-2019.9.8)] AndMFC: Android Malware Family Classification Framework, p1–6.

[17] Surendran, R., Thomas, T., & Emmanuel, S. (2020). A TAN based hybrid model for android malware detection. Journal of Information Security and Applications, 54, 102483. P1-11.

[18] Rana, Md. Shohel; Gudla, Charan; Sung, Andrew H. (2018). [ACM Press the 2018 VII International Conference - Taipei City, Taiwan (2018.12.14-2018.12.16)] Evaluating Machine Learning Models for Android Malware Detection. , p17–21.

[19] Nadia Daoudi;Kevin Allix;Tegawendé F. Bissyandé;Jacques Klein; (2021). Lessons Learnt on Reproducibility in Machine Learning Based Android Malware Detection . Empirical Software Engineering, p1-53.

[20] Chen, Zhenxiang; Yan, Qiben; Han, Hongbo; Wang, Shanshan; Peng, Lizhi; Wang, Lin; Yang, Bo (2017). Machine Learning Based Mobile Malware Detection Using Highly Imbalanced Network Traffic. Information Sciences, p1-23.

[21] Wang, Xin; Zhang, Dafang; Su, Xin; Li, Wenjia (2017). Mlifdect: Android Malware Detection Based on Parallel Machine Learning and Information Fusion. Security and Communication Networks, 2017, p1–14.

[22] Chen, Xiao; Li, Chaoran; Wang, Derui; Wen, Sheng; Zhang, Jun; Nepal, Surya; Xiang, Yang; Ren, Kui (2019). Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. IEEE Transactions on Information Forensics and Security, p1–15.

[23] Abaid, Zainab; Kaafar, Mohamed Ali; Jha, Sanjay (2017). [IEEE 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA) - Cambridge, MA (2017.10.30-2017.11.1)] Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers. , p1–10.

[24] Idrees, Fauzia; Rajarajan, Muttukrishnan; Conti, Mauro; Chen, Thomas M.; Rahulamathavan, Yogachandran (2017). PIndroid: A novel Android malware detection system using ensemble learning methods. Computers & Security, 68, p36–46.
[25] [25] Alzaylaee, Mohammed K.; Yerima, Suleiman Y.; Sezer, Sakir (2019). DL-Droid: Deep Learning Based Android Malware Detection Using Real Devices. Computers & Security, p1-28.
[26] Fereidooni, Hossein; Conti, Mauro; Yao, Danfeng; Sperduti, Alessandro (2016). [IEEE 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS) - Larnaca, Cyprus (2016.11.21-2016.11.23)] ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications. , p1–5.
[27] Palumbo, Paolo; Sayfullina, Luiza; Komashinskiy, Dmitriy; Eirola, Emil; Karhunen, Juha (2017). A pragmatic android malware detection procedure. Computers & Security, p1-27.
[28] Uzoma rita alo, henry friday nweke2 , sylvester i. Ele. (2021). Machine learning-based framework for automatic malware detection using android traffic data. Journal of theoretical and applied information technology. 99(15), pp.1-19.
[29] Atluri, Venkata (2019). [IEEE SoutheastCon 2019 - Huntsville, AL, USA (2019.4.11-2019.4.14)] 2019 SoutheastCon - Malware Classification of Portable Executables using Tree-Based Ensemble Machine Learning. , p1–6.
[30] Christianah, Abikoye Oluwakemi; Gyunka, Benjamin Aruwa; Oluwatobi, Akande Noah (2020). Optimizing Android Malware Detection Via Ensemble Learning. International Journal of Interactive Mobile Technologies (iJIM), 14(09), p1-18.
[31] Zhu, H., Li, Y., Li, R., Li, J., You, Z., & Song, H. (2021). SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection.IEEE Transactions on Network Science and Engineering, 8(2), p984–994.
[32] Android Malware Dataset (CIC-AndMal2017). Retreived from https://www.unb.ca/cic/datasets/andmal2017.html
[33] Sultan, M. T., Sayed, H. E. and Khan, M. A. (2023). An Intrusion Detection Mechanism for MANETs based On Deep Learning Artificial Neural Networks (ANNS). International Journal of Computer Networks & Communications (IJCNC) Vol.15, No.1, January 2023.
[34] Kassa, L., Deng, J., Davis, M, and Cai, J. (2022). Machine Learning Based Frame Size Optimization in WLAN Downlink MU-MIMO Channel with The Least Cost of Delay. International Journal of Computer Networks & Communications (IJCNC) Vol.14, No.6, November 2022.

AUTHORS

Sumalatha Potteti is working as Assistant Professor at Bhoj Reddy Engineering College for Women, Hyderabad, Telangana (State), India. She has received B.Tech (CSE), M.Tech (CSE) Degree in Computer Science and Engineering. Her main research interest includes Machine Learning and Intrusion Detection.

Dr. G.S. Mahalakshmi is working as Associate Professor at Anna University, Guindy, Chennai, Tamilnadu (State), India. She has received B.E(CSE), M.E(CSE), Ph.D. Degree in Computer Science and Engineering. Her main research interest includes Artificial Intelligence, Machine Learning, Natural Language Processing knowledge representation and Text Mining.