2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE) | 979-8-3503-8828-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAEEE62219.2024.10561751

Feature Selection-Based Machine Learning Approaches for Detecting Android Malware with Explainable AI

1st Mahbuba Habib
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology
Rajshahi-6204, Bangladesh
mahbubahabib26@gmail.com

2nd Hafsa Binte Kibria
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology
Rajshahi-6204, Bangladesh
hafsabintekibria@gmail.com

***Abstract*—With the involvement of technology, its negative sides are showing out drastically. As Android is an open-source platform that has nice customization, control and accessibility features cost-effectively, it attracts middle-class device users. As of 2023, Android has a 70.29% global market share. So, its insecurity has a tremendous effect on the mobile industry. This study represents how to efficiently detect Android malware in machine learning with various feature selection techniques like Correlation, Mutual Information, Chi-Square test and Recursive Feature Elimination to serve the optimization of the huge amount of information collected from different apps. Besides, these criteria are tested with different under-sampling and oversampling techniques. This research evaluates some more widely used algorithms: Decision Tree, Random Forest, and XG Boost. Among them, Random Forest surpasses all other algorithms. The output is evaluated with another dataset. Also, Explainable AI is used for the transparency of the models. This implies that Smote NC-based Mutual Information gives the highest outcome(98.5%) with an F1 score of 98% using a 10-fold cross-validation (CV) in Android malware detection.**

***Index Terms*—Android malware, Feature Selection, Sampling Technique, Random Forest, XgBoost, Malware Analysis, Explainable AI**

I. Introduction

A. Security in Android

Android is a Linux kernel-based operating system owned by Google giving a real-life experience to the end-user. Due to its high usability, it is a common source of attack for cybercriminals. As per the latest Stat counter report, Android has a total worldwide hold of 69.74 percent whereas iOS holds a 29.58 percent share of the global mobile operating system market [1]. Both play a vital role bearing more than 99 percent share of the total market. Along with Android users, its security vulnerability also increases every year. The security enhancements in Android were clarified by Enck, Ong Tang, and McDaniel. Some of them are the public vs. private components, the permission for broadcast intent, content provision, secure APIs, pending intents, permission protection level, and URIs permission. Many approaches are used to identify and neutralize harmful apps and threats to ensure Android users' privacy. The latest research signifies that Machine learning and Deep learning have shown tremendous improvement in this field. [2].

B. Trends in Android malware

From the latest survey by the National Institute of Standards and Technology (NIST), a drastic increase in the malware detection mechanism in the year 2020 [3]. The new trend shows that 70-90 percent of attacks were specific organization-based. Recently, the high emergence of polymorphic malware has been used to evade the antivirus detection approach [2]. Besides, Evasion techniques have been used to hide malware presence like dead code insertion, packing, anti-emulation techniques, etc [2]. Above all of that, in Android devices, some malware has been found that is spying and collecting sensitive information about a country's political and defense matters [2]. The infected Android application also contains an additional library to download malicious plugins.

This paper mainly focuses on handling data efficiently and finding malicious and benign Android app characteristics. Proper handling of data helps to

- Assist in performing faster training and inference.
- Employ ten fold cross-validation to ensure the reliability and generalizability of the model.
- Integrate explainable AI techniques to enhance the transparency and interpretability of the model's decision-making process.

The first section introduces the research topic and its importance. The second section makes a comparison of other studies regarding this topic. In the next level, some basic terms are presented to make a clear concept in this arena. Then, the implementation of our feature selection method is covered. In the last part, our model's performance is evaluated visualizing a comparative analysis.

II. LITERATURE REVIEW

Efficient and effective continuous malware detection process is an urgent need in this era. As the Android malware attack process is changing continuously, giving a specific solution is quite challenging.

The paper [4] presented deep learning approaches for detecting Android malware [5]. It outlined the recent work from 2018-2021 and tried to increase the detection process accuracy. This implied Long Short-Term Memory (LSTM) and Long Short-Term Memory (LSTM) with Convolutional Neural Network (CNN) in static and dynamic analysis. It had achieved a satisfactory result.

The author of the study [6] focused on various machine learning models and their applications in different datasets. It compared the performance of all evaluation metrics utilized to evaluate their efficiency. In [7]the COL-COM 2016 dataset was used and they implemented K-Nearest Neighbor, Multi-Layer Perception, and Naive Bias. Multi-layer perception gave the highest accuracy but took a significant amount of time. It implied improving the model with the proper time management.

In [8], Synthetic Minority Oversampling Technique (SMOTE) was applied and PCA was used as feature selection technique. Besides, it developed a Light Gradient Boosting Model to identify Android malware and classify them [8]. Different machine learning algorithms were analyzed in [9] and found Random Forest as a good option for the specific dataset. Further, it encouraged more implementation of the deep learning model.

Another study [10] used semi-supervised learning. They used a few labeled data and more unlabeled data to train a model [10]. Here, semi-supervised learning with auto-encoders for feature extraction was performed and PF, SVC, and CNN were applied. The model got a good accuracy. Here [11] researchers created their own data set and implemented KNN, Random Forest, and Naive Bias in semi-supervised learning. It got the highest accuracy in Naive Bias.

This research paper [12] conducted a comparison of various machine learning algorithms and recommended the adoption of additional methods to enhance the effectiveness of Android hardware security. Various gradient boosting algorithms were applied in [13] and it recommended using feature selection techniques with it and verifying the result. The paper [14] described the co-existence approach using dynamic features. It tried out a Decision Tree, Support Vector Machine, and Random Forest and got an acceptable result.

In the existing literature, a variety of algorithms have been utilized, achieving commendable accuracy. However, there is still scope for improvement. It was observed that many studies did not incorporate k-fold cross-validation, and explainable AI was not used in any of those studies which is a major drawback. Our research focuses on addressing these drawbacks, enhancing reliability and accuracy, and ensuring the transparency of the results, especially through the use of explainable AI.

III. Background

A. Analysis Criteria

The research is focused on static analysis-based Android malware detection. Android malware can be detected in several processes. Some are:

- Static Analysis: Features extracted without executing the app in real-time.
- Dynamic Analysis: Features extracted running the app in the virtual machine.
- Hybrid Analysis: A consolidation of static and dynamic analysis [3].

In this paper, operations are performed on the information extracted from Static Analysis.

B. Performance Matrix

Accuracy is measured in different metrics like Precision, Recall, and F1 score to show how the model performs in different types of data.

- Precision: Describes the percentage of positive predictions that were correct (True Positive/(True Positive+False Positive))
- Recall: Describes the percentage of positive cases the model matched. (True Positive/(True Positive +False Negative))
- F1 Score: It is the weighted average of both Precision and Recall

C. Feature Selection Techniques

Different Feature selection techniques are available now. In this study, we mainly performed filter-based feature selection techniques. Besides, the Wrapper method was introduced. Wrapper base accuracy is greater than Filter base feature selection, but it is computationally expensive and prone to over-fitting.

- Mutual Information: Mutual information is a fast, model-neutral feature selection technique. It measures the entropy drops under the condition of the target value. A higher value indicates a higher dependency between variables. Thus, the features that have high mutual information with the target variable are selected for further process.
- Pearson Correlation: Pearson Correlation finds how each column is correlated with each other. Then according to an input percentage, the columns that are highly correlated with each other, either of them are dropped. Thus, the dataset becomes more robust and straightforward.
- Chi-Square Test: In chi-square distribution, it first determines that the given distributions are independent. For each feature, the chi-square value is calculated, and the values are sorted in descending order. The higher value resembles it is more dependent on the output label. Thus, selecting a specific range, important features are selected.
- Recursive Feature Elimination: RFE recursively removes the least important features till the preferred features are found. It uses different learning algorithms on the original data and selects relevant features based on the performance of the learning algorithm. It is often used to improve performance and reduce data over-fitting.

D. Sampling Techniques

Sampling is a process of picking out a few individuals from a large group of informational data. Sampling can be of different types. In this research, we have focused on various types of SMOTE and the ADASYN techniques. But ADASYN cannot give satisfactory results. So, SMOTE variations are focused.

- SMOTE: Synthetic Minority Over-Sampling Technique is a re-sampling process where synthetic instances of the minority class are created. Identifying the minority class instances, it selects one random neighbor to generate a synthetic instance along the line segment connecting the chosen neighbor and the original instances. It is generally performed in a continuous data structure.
- SMOTE-NC: SMOTE-NC is SMOTE especially used for both categorical and nominal data. In SMOTE a sample is created based on the nearest samples of the minority class. In the case of categorical data, generally, it contains integer values of classes but the sample result could be float and so on. Then a little error is generated. SMOTE-NC helps to overcome this barrier by handling datasets with a mix of both nominal (categorical) and continuous (numeric) data.
- Border Line SMOTE: In SMOTE, when the minority class has outlying data that appears in the majority class, it sometimes creates a barrier by creating a line with the larger part of the class. In case all its neighbors are from the majority class, Border Line classifies the outlier minority points as noise and ignores it. Then taking a few points as border points that have both classes as neighborhoods, it generates synthetic samples. Border Line Smote can be of 2 types. One oversamples the closest data to the borderline. Whereas another only oversamples the minority class.
- Tomek Links: Tomek Links is an under-sampling technique, a modification from the Condensed Nearest Neighbors under-sampling technique. In CNN, a random sample is taken but the Tomek Links method uses the rule to select the pair of observations. It is a cleaning process to remove the majority class that was overlapping with the other minority class.
- Smote-Tomek: Smote-Tomek is the integration of both the SMOTE and Tomek Links. Its main target is to improve the synthetic generation sample quality by addressing potential noise and ambiguous cases in the minority class. Thus, it contributes to better model generalization and performance in imbalanced classification tasks.
- Adasyn: ADASYN is a more generic framework, that tries to oversample the minority data based on the data density. The density function determines the no. of synthetic instances generated for the samples that are difficult to learn. It is the main difference from SMOTE. Thus it helps to adaptively change the decision boundaries based on the samples difficult to learn.

E. Machine Learning Models

For Android malware detection, numerous supervised, unsupervised, and deep learning models are proposed by the researchers. Some most widely used models are:

- Decision Tree: DT is a tree-base supervised Algorithm. It has mainly 3 parts.
  1. Internal node- Represents features
  2. Branch- Presents decision rules
  3. Leaf node- Represents a class or value

  It can detect the target variables for both discrete and categorical data.
- Random Tree: A random forest is an ensemble of decision trees, generally trained via the bagging method. It has all the hyperparameters of decision tree classifiers as well as bagging classifiers to control the ensemble itself. It searches for the best attributes among a random subset of features, which trades a higher bias for a lower variance. Thus, it generally helps to yield an overall better model.
- XGBoost: XGBoost symbolizes Extreme Gradient Boosting. It is an extended, more generalized version of a gradient-boosting algorithm. Some key advantages of the XG Boost algorithm are the implementation of parallel processing, built-in standard to handle missing values, split the tree up to a maximum depth, etc.

F. Explainable AI

Explainable AI is a bundle of the set of processes and methods that aim to provide a clearer and human-understandable explanation for the decisions generated by various machine and deep learning models. There are two broad categories of model explainability: model-specific methods and model-agnostic methods. The second one focuses on analyzing the features' input-output pair. The two widely used are LIME and SHAP.

- LIME: Local Interpretable Model-agnostic Explanations focuses on explaining the model's prediction for individual instances.
- SHAP: SHapley Additive exPlanations aims to explain the prediction of an observation by computing the contribution of each feature to the prediction.

IV. Methodology

While many researchers are engaged in implementing more advanced and high-powered models, our main target is to bring out a solution with a feature selection technique for a large dataset with transparent models. In this study, interpretable Machine learning models are used to evaluate the model's accuracy. The whole procedure has been displayed in Figure 1. A brief explanation of our model is proposed here:

A. Android Malware Dataset Selection

The dataset used for training and validation in this paper was created by Yerima and Suleiman (2018) [15]. It has 215 features with a malware record of 15,036 occurrences.It is a binary classification problem. Then the model is evaluated

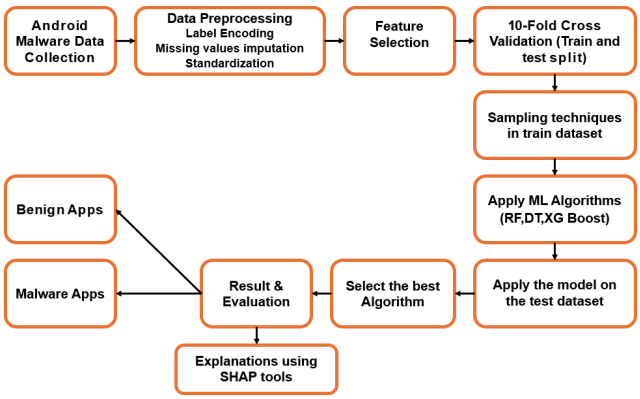

Fig. 1: Architecture of the proposed model

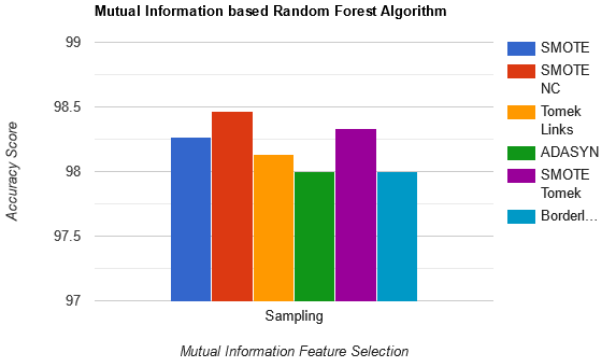

Fig. 2: Random Forest Algorithm in the dataset-I

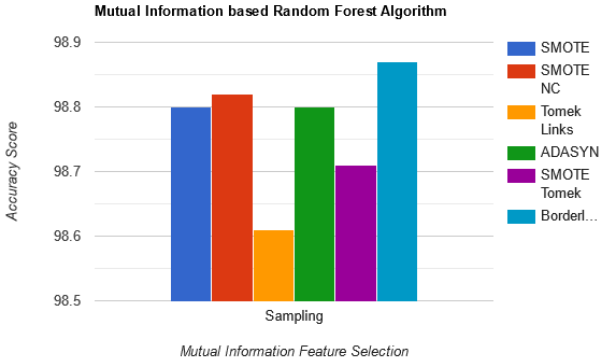

Fig. 3: Random Forest Algorithm in the dataset-II

with the asm files of BIG Malware Dataset from Microsoft Dataset [16] which classifies 9 different classes of malware.

B. Data pre-processing

First, the features of the dataset were studied. Then any null value and the types of the data were checked. Upon non-numerical objects, the Label Encoder operation was performed. Scaling the dataset, the nest steps were performed.

C. Feature Selection

Among different feature selection techniques, PCA, Mutual Information, Pearson correlation, and RFE were performed.

D. Sampling in the training dataset

In the training dataset, SMOTE, SMOTE-NC, Borderline SMOTE, Tomek Links, SMOTE-Tomek, and ADASYN were analyzed. Then, test data were evaluated with the realtime data values.

E. Fit into ML model

Some widely used models Decision Trees, Random Forest, and XG Boost were chosen to train and evaluate the model. Among them, Random Forest and XGBoost give the satisfactory result.

F. Evaluate with cross-validation

10-fold cross-validation is performed here for the binary and multi-class classification.

G.Implement Explainable AI

Model-agnostic method SHAP is implemented with Random Forest output in both datasets.

V. Results & Discussions

Feature selection technique helps reduce the larger dataset's complexity without any significant data loss. Table 1 shows the output of different Machine Learning algorithms without any feature selection technique. We performed all combinations of feature selection and sampling techniques with the machine learning algorithms. In the case of different sampling techniques, SMOTENC and Tomek-Links work best. Besides, Feature selection-based RFE gives a satisfactory result but it requires more time than the others.

Table 2-4 shows the outcome of implementing Mutual information-based feature selection with different sampling techniques in our two datasets. Among them, Random Forest gives the highest accuracy. Though accuracy is not improved much, the dataset is reduced from 215 attributes to 150 attributes. This accuracy along with other metrics was performed with 10-fold cross-validation.

Figure 2 shows the comparative output of various sampling techniques in Mutual Information-based Random Forest in the dataset-I. After successfully implementing it in the dataset-I, these models are evaluated on the dataset of the Microsoft Malware Classification Challenge (BIG 2015) [16]. It is a multiclass dataset having 9 categories of malware data. In this case, also, Mutual Information-based SMOTENC and Borderline SMOTE give satisfactory result.

Figure 3 reveals the Mutual information-based accuracy in the second dataset as it works better than others. The notable point is that in the second case, Borderline SMOTE also performs well in the case of multi-class classification. Thus, we can say that Mutual Information-based SMOTENC feature selection in Random Forest can be a good option for Android Malware Detection.

Figure 4 and 5 presents the summary plot with SHAP XAI of the 2 datasets.Here, each dot represents the impact on a specific class of a specific feature for the given instance. The color of the dot represents the magnitude of

TABLE I: Machine Learning accuracy without any feature selection technique

ML Algorithm	Dataset- 1				Dataset-2			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.97	0.9555	0.9658	0.9606	0.9714	0.9308	0.9177	0.9228
Random Forest	0.9813	0.9835	0.9658	0.9746	0.988	0.9866	0.9344	0.9510
XG Boost	0.9654	0.9700	0.9317	0.9505	0.9652	0.9832	0.9315	0.9478

TABLE II: Mutual Information-based Decision Tree Performance

Sampling Techniques	Dataset- 1				Dataset-2			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
SMOTE	0.9693	0.9554	0.9622	0.9588	0.9723	0.968	0.9173	0.9338
SMOTENC	0.9654	0.9468	0.9604	0.9536	0.976	0.9749	0.9466	0.9588
Tomek Links	0.968	0.952	0.9622	0.9571	0.9732	0.8554	0.8632	0.9588
ADASYN	0.964	0.9358	0.9694	0.9523	0.9696	0.9576	0.9445	0.9485
Smote Tomek	0.972	0.9573	0.9676	0.9624	0.9742	0.9699	0.9442	0.955
BorderLine Smote	0.9693	0.9537	0.964	0.9589	0.97145	0.9421	0.9419	0.9419

TABLE III: Mutual Information-based Random Forest Performance

Sampling Techniques	Dataset-1				Dataset-2			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
SMOTE	0.9827	0.9836	0.9694	0.9764	0.988	0.9867	0.9609	0.9716
SMOTENC	**0.98469**	**0.9872**	**0.9712**	**0.9791**	0.988	0.986	0.9613	0.9714
Tomek Links	0.9813	0.9835	0.9658	0.9746	0.9861	0.983	0.932	0.9479
ADASYN	0.98	0.9786	0.9676	0.9729	0.988	0.986	0.9613	0.9643
Smote Tomek	0.9833	0.9836	0.9712	0.9774	0.9871	0.9848	0.9598	0.9699
BorderLine Smote	0.98	0.9713	0.9748	0.9731	**0.9887**	**0.987**	**0.9622**	**0.9724**

TABLE IV: Mutual Information-based XG Boost Performance

Sampling Techniques	Dataset-1				Dataset-2			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
SMOTE	0.9614	0.956	0.9388	0.9474	0.9834	0.979	0.9575	0.966
SMOTENC	0.9694	0.956	0.9388	0.9474	0.9861	0.978	0.9575	0.9652
Tomek Links	0.9654	0.9737	0.9317	0.9522	0.9843	0.98	0.9577	0.9668
ADASYN	0.9494	0.896	0.9766	0.9346	0.9843	0.9807	0.9574	0.9667
Smote Tomek	0.9614	0.9544	0.946	0.9475	0.9806	0.9738	0.9546	0.9617
BorderLine Smote	0.9467	0.8889	0.9784	0.9315	0.9825	0.978	0.9575	0.9655

the contribution to the model impact, with red denoting a high value and blue denoting a low value. Besides, the higher distance from the zero position means a high impact on the classification. In the first figure, the summary plot of the malware class is highlighted. It is shown that SEND_SMS, READ_PHONE_STATE, TelephonyManager.getLine1Number have positive effects for the prediction of Android malware. In the second dataset, asm_commands_dd,size_asm,line_count_asm are negatively correlated for finding the first sample malware. In both cases, the first 20 features are considered.

A lot of research has been done on both datasets. Some related research work has been highlighted in Table V. In those papers, cross-validation was not implemented.Also, they did not have transparency in their models. We tried to perform with 10-fold cross-validation to judge the output more efficiently. The proposed model gives an accuracy of 98.5% in the first dataset and 98.8% in the second dataset. At last, the output is evaluated with Explainable AI.

VI. Conclusions & Future Work

The experiment conducted in this paper aims to focus on light feature selection techniques to bring out a successful way that detect Android malware within a minimal amount of time. In our proposed methodology, the model achieves an accuracy of 98.5% by using a Mutual Information-based Random Forest algorithm in the first dataset. We have also used explainable AI for the transparency of our model. In future work, some advanced feature selection techniques like Genetic Algorithms and auto-encoder models will be implemented to bring out a more robust model. Besides, this model can be performed with other larger datasets of Android malware.

References

[1] "Global stats." [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide

TABLE V: Previous work with Dataset - I & II

Paper No	Dataset-1				Dataset-2			
	Author	Year	Approach	Accuracy	Author	Year	Approach	Accuracy
1	[17]	2023	Several XGBoost models	0.99 F1-Score	[18]	2016	PCA based KNN,SVM,ANN	96.6% in SVM
2	[19]	2022	RF,LMT,J48,Forest PA	99.6% in RF	[20]	2017	NB,KNN,SVM,XGBoost	0.963% in SVM

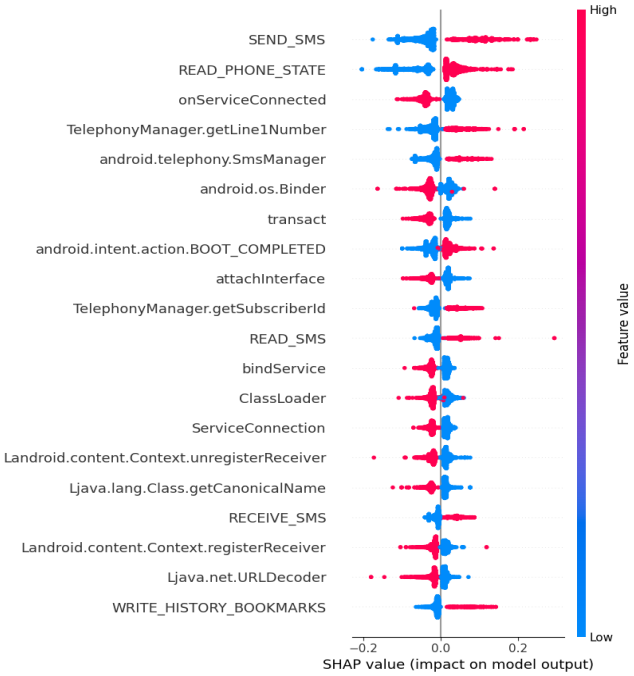

Fig. 4: Summary plot of the first dataset

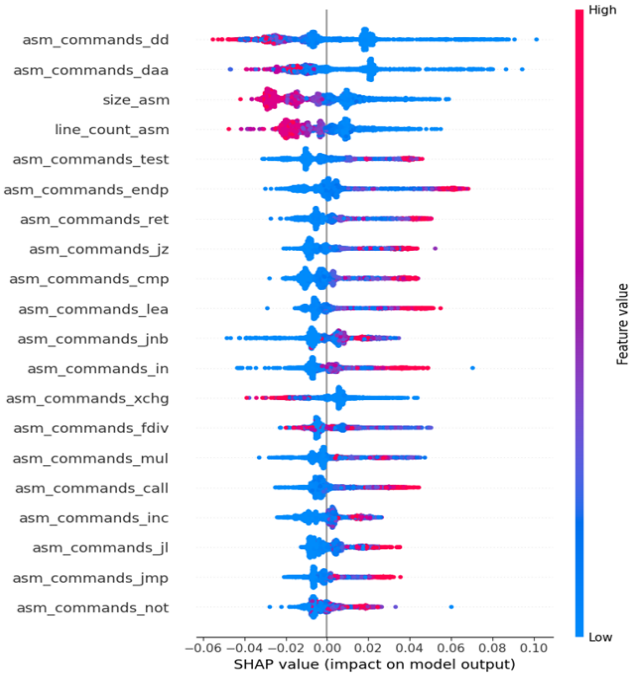

Fig. 5: Summary plot of the second dataset

[2] D. W. Utomo, C. Lim *et al.*, "Image-based malware classification: A systematic literature review," in *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)*. IEEE, 2023, pp. 99–103.

[3] "Nist cybersecurity framework," Nov 2023. [Online]. Available: https://www.nist.gov/itl/smallbusinesscyber/planning-guides/nist-cybersecurity-framework

[4] E. Calik Bayazit, O. Koray Sahingoz, and B. Dogan, "Deep learning-based malware detection for android systems: A comparative analysis," *Tehnički vjesnik*, vol. 30, no. 3, pp. 787–796, 2023.

[5] S. Acharya, U. Rawat, and R. Bhatnagar, "A comprehensive review of android security: Threats, vulnerabilities, malware detection, and analysis," *Security and Communication Networks*, vol. 2022, 2022.

[6] M. N.-U.-R. Chowdhury, A. Haque, H. Soliman, M. S. Hossen, T. Fatima, and I. Ahmed, "Android malware detection using machine learning: A review," *arXiv preprint arXiv:2307.02412*, 2023.

[7] H. H. R. Manzil and S. M. Naik, "Detection approaches for android malware: Taxonomy and review analysis," *Expert Systems with Applications*, p. 122255, 2023.

[8] H. AlOmari, Q. M. Yaseen, and M. A. Al-Betar, "A comparative analysis of machine learning algorithms for android malware detection," *Procedia Computer Science*, vol. 220, pp. 763–768, 2023.

[9] M. S. Akhtar, "Analyzing and comparing the effectiveness of various machine learning algorithms for android malware detection," *Advances in Mobile Learning Educational Research*, vol. 3, no. 1, pp. 570–578, 2023.

[10] X. Li, L. Khan, M. Zamani, S. Wickramasuriya, K. Hamlen, and B. Thuraisingham, "Con2mix: A semi-supervised method for imbalanced tabular security data 1," *Journal of Computer Security*, no. Preprint, pp. 1–22, 2023.

[11] M. Memon, A. A. Unar, S. S. Ahmed, G. H. Daudpoto, and R. Jaffari, "Feature-based semi-supervised learning approach to android malware detection," *Engineering Proceedings*, vol. 32, no. 1, p. 6, 2023.

[12] S. Acharya, U. Rawat, and R. Bhatnagar, "A comprehensive review of android security: Threats, vulnerabilities, malware detection, and analysis," *Security and Communication Networks*, vol. 2022, 2022.

[13] S. Aurangzeb and M. Aleem, "Evaluation and classification of obfuscated android malware through deep learning using ensemble voting mechanism," *Scientific Reports*, vol. 13, no. 1, p. 3093, 2023.

[14] E. Odat and Q. M. Yaseen, "A novel machine learning approach for android malware detection based on the co-existence of features," *IEEE Access*, vol. 11, pp. 15 471–15 484, 2023.

[15] S. Y. Yerima and S. Sezer, "Droidfusion: A novel multilevel classifier fusion approach for android malware detection," *IEEE transactions on cybernetics*, vol. 49, no. 2, pp. 453–466, 2018.

[16] X. Wang, J. Liu, and X. Chen, "Microsoft malware classification challenge (big 2015) first place team: say no to overfitting," *no. Big*, 2015.

[17] A. Kitanovski, H. Mihajloska Trpcheska, and V. Dimitrova, "Detecting malware in android applications using xgboost," 2023.

[18] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for malware classification," in *2016 IEEE national aerospace and electronics conference (NAECON) and ohio innovation summit (OIS)*. IEEE, 2016, pp. 338–342.

[19] O. Olorunshola, A. Oluyomi, C. Jonathan, and O. Awujoola, "Synthetic minority over-sampling technique and resample approach for android malware detection using tree-based classifiers."

[20] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3854–3861.