ИНФОРМАТИКА

UDC 004.8
IRSTI 83.00.00
https://doi.org/10.47526/2024-4/2524-0080.16

Zhunissov N.M.[1], **Abushakhma A.A.**[2]
[1]*PhD, Khoja Akhmet Yassawi International Kazakh-Turkish University (Kazakhstan, Turkistan), e-mail: nurseit.zhunissov@ayu.edu.kz*
[2]*master's student, Khoja Akhmet Yassawi International Kazakh-Turkish University (Kazakhstan, Turkistan), e-mail: aktore.abushaxma@gmail.com*

APPLICATION OF MACHINE LEARNING ALGORITHMS IN DETECTING MALICIOUS SOFTWARE APPLICATIONS ON THE ANDROID PLATFORM
ANDROID ПЛАТФОРМАСЫНДА ЗИЯНДЫ БАҒДАРЛАМАЛЫҚ ҚОСЫМШАЛАРДЫ АНЫҚТАУДА МАШИНАЛЫҚ ОҚЫТУ АЛГОРИТМДЕРІН ҚОЛДАНУ
ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ОБНАРУЖЕНИЯ ВРЕДОНОСНЫХ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ ANDROID

Abstract. *The work brings forward the methods for the application of the machine-learning algorithm in identifying the presence of malicious programs on the Android platform. In the modern world, Android devices are the most widely used and at the same time bring a great risk for other computer systems due to malicious attacks or software opportunities. The traditional method for detecting malicious applications depends either on static and dynamic analyses. However, conventional methods are already becoming ineffective since these codes have gradually become so sophisticated. This challenge can be met through machine learning, whereby the Black Box is able to predict with high accuracy potential malicious conduct patterns by analyzing target behavior applications.*

Android malware detection importance is pegged on the rapidly evolving hostile applications that are readily available in the Android Operating system. In this work, the authors present a comparison of three classifiers: RandomForestClassifier, LGBMClassifier, and XGBClassifier using various preprocessing and sampling schemes so as to achieve the novel goal of inter-class learning for malware detection. Our analysis shows that the best models are obtained when using RobustScaler and SMOTE in conjunction with a RandomForestClassifier, since it generates more accurate models and best results considering both precision and recall. The LGBMClassifier and XGBClassifier look good, but they do not quite reach the same level of efficiency as the best-performing model, RandomForestClassifier.

It also points out that the present research underlines suitable preprocessing and sampling methods in order to enhance model performance. The results make suggestions on the improvements that should be made in Machine Learning systems targeting malware detection for further advancement of cybersecurity solutions.

Keywords: *Android, Machine Learning Classifiers, RandomForestClassifier, LGBMClassifier, XGBClassifier, Preprocessing Techniques, SMOTE Sampling, Model Performance Evaluation*

Аңдатпа. *Мақала Android платформасында зиянды бағдарламалардың болуын анықтауда машиналық оқыту алгоритмін қолдану әдістерін алға тартады. Қазіргі әлемде Android құрылғылары ең көп қолданылатын құрылғылар болып табылады және сонымен бірге зиянды шабуылдар немесе бағдарламалық жасақтама мүмкіндіктеріне байланысты басқа компьютерлік жүйелер үшін үлкен қауіп төндіреді. Зиянды қосымшаларды анықтаудың дәстүрлі әдісі статикалық және динамикалық талдауларға байланысты. Алайда, әдеттегі әдістер қазірдің өзінде тиімсіз болып келеді, өйткені бұл кодтар біртіндеп жетілдіріле бастады. Бұл мәселені машиналық оқыту арқылы шешуге болады, Оның көмегімен Қара Жәшік мақсатты мінез-құлық қолданбаларын талдау арқылы ықтимал зиянды мінез-құлық үлгілерін жоғары дәлдікпен болжай алады.*

Android зиянды бағдарламаларын анықтаудың маңыздылығы Android Операциялық жүйесінде оңай қол жетімді жылдам дамып келе жатқан дұшпандық қолданбаларға байланысты. Бұл жұмыста авторлар үш классификаторды салыстыруды ұсынады: RandomForestClassifier, LGBMClassifier және XGBClassifier зиянды бағдарламаларды анықтау үшін жаңа сыныпаралық оқыту мақсатына жету үшін әртүрлі алдын ала өңдеу және іріктеу схемаларын пайдалана отырып. Біздің талдауымыз Көрсеткендей, Ең жақсы үлгілер RobustScaler Және SMOTE Қолданбаларын Randomforest Классификаторымен бірге пайдаланған кезде алынады, өйткені ол дәлдікті де, еске түсіруді де ескере отырып, дәлірек үлгілерді және жақсы нәтижелерді жасайды. LGBMClassifier және XGBClassifier екеуі де жақсы көрінеді, бірақ олар Ең жақсы жұмыс істейтін randomforestclassifier үлгісімен бірдей тиімділік деңгейіне жете алмайды.

Сондай-ақ, бұл зерттеу модельдің өнімділігін арттыру мақсатында алдын ала өңдеу мен іріктеудің қолайлы әдістерін атап көрсететінін көрсетеді. Нәтижелер киберқауіпсіздік шешімдерін одан әрі ілгерілету *үшін зиянды бағдарламаларды анықтауға бағытталған Машиналық Оқыту жүйелерін жетілдіру бойынша ұсыныстар береді.*

Негізгі сөздер: *Android, Machine Learning классификаторлары, RandomForestClassifier, LGBMCclassifier, XGBClassifier, Preprocessing Techniques, SMOTE Sampling, Model Performance Evaluation*

Аннотация. *В статье представлены методы применения алгоритма машинного обучения для выявления присутствия вредоносных программ на платформе Android. В современном мире устройства на базе Android используются наиболее широко и в то же время представляют большой риск для других компьютерных систем из-за вредоносных атак или возможностей программного обеспечения. Традиционный метод обнаружения вредоносных приложений основан как на статическом, так и на динамическом анализе. Однако традиционные методы уже становятся неэффективными, поскольку эти коды постепенно становятся все более сложными. Эта задача может быть решена с помощью машинного обучения, благодаря которому "Черный ящик" способен с высокой точностью предсказывать потенциальные модели вредоносного поведения, анализируя поведение целевых приложений.*

Важность обнаружения вредоносных программ для Android связана с быстро развивающимися вредоносными приложениями, которые легко доступны в операционной системе Android. В этой работе авторы представляют сравнение трех классификаторов: RandomForestClassifier, LGBMClassifier и XGBClassifier, использующих различные схемы предварительной обработки и выборки, чтобы достичь новой цели - межклассового обучения для обнаружения вредоносных программ. Наш анализ показывает, что наилучшие модели получаются при использовании RobustScaler и SMOTE в сочетании с RandomForestClassifier, поскольку они генерируют более точные модели и дают наилучшие результаты как с точки зрения точности, так и с точки зрения запоминания. LGBMClassifier и XGBClassifier выглядят хорошо, но они не достигают того же уровня эффективности, что и наиболее эффективная модель RandomForestClassifier.

В нем также отмечается, что в настоящем исследовании подчеркиваются подходящие методы предварительной обработки и выборки данных для повышения производительности модели. В результатах содержатся предложения по усовершенствованиям, которые следует внести в системы машинного обучения, нацеленные на обнаружение вредоносных программ, для дальнейшего совершенствования решений в области кибербезопасности.

Ключевые слова: *Android, классификаторы машинного обучения, RandomForestClassifier, LGBMClassifier, XGBClassifier, методы предварительной обработки, выборка SMOTE, оценка производительности модели.*

Introduction

The exponential growth in mobile technologies has completely changed the way people interact with the digital world. Among several mobile operating systems, Android is the most adopted platform and lays claim to a lion's share in the global market. Wide circulation and the open nature of the Android ecosystem bring ample opportunities and challenges. While it has enabled a great variety of applications and innovations, at the same time it has turned out to be an attractive target for malicious actors to discover its vulnerabilities and compromise users' security.

The rapid proliferation of mobile devices and their applications is radically changing the way in which people are communicating with one another, search for information, and perform a task in daily life. Among them, Android-powered smartphones have dominated more than 70% of market share among mobile operating systems worldwide. On the other hand, the openness and ubiquity of the Android platform have also made it a perfect target for cybercriminals. For this reason, malware in Android has emerged as one of the major security threats to date, posing critical risks to users' privacy, data integrity, and financial security[1].

Android malware, which comes in a broad array of malicious software targeting Android devices, has become a widespread and serious problem. The malware has a broad landscape, including, but not limited to, adware, scareware, and SMS malware. Adware typically-used applications that hard-sell advertisements to users often degrade both user experience and device performance. Scareware, on the other side, is used as a psychological manipulator to instill fear in coercing users to perform actions that are harmful, such as buying fake security software. SMS malware, through text messaging, steals sensitive information or accrues unwanted charges. Each malware type has its own way of deception to avoid its detection and carry out its nefarious goals.

Due to the phenomenal changes in malware techniques and advanced developments of cyber-attacks, traditional signature-based methods for their detection are not effective. In a signature-based approach, predefined patterns of known malware are relied upon, and that is the reason it misses novel or polymorphic variants [2]. That naturally leads to a shift and movement towards behavior-based and data-driven detection mechanisms, especially in the related areas of machine learning and artificial intelligence. These can theoretically allow the detection of zero-day threats by analyzing the pattern and anomaly in the behavior and attributes of an application.

In this regard, the "Android_Malware.csv" dataset will go a long way in contributing to malware detection research. Scraped from the Canadian Institute for Cybersecurity repository, this dataset consists of 355,630 instances with four different labels, namely Android Adware, Android Scareware, Android SMS Malware, and Benign applications. Each example/instance is described by 85 features, providing a wide view about characteristics and behaviors of both malicious and benign applications.

First of all, the large volume of the dataset, along with its feature set, makes it possible to investigate different detection methodologies deeply. Based on the given problem, a significant goal of the research study is to extend this dataset with advanced model proposal and performance evaluation for Android malware detection using machine learning. By trying various methods, from traditional ones - including classic classifiers and ensemble methods to recent techniques based on deep learning, we try to find the best way of distinguishing malware types from non-malicious applications [3].

What is more, the intrinsic imbalance of the dataset-the fact that malicious instances are much more than the benign instances-is a challenge that requires being carefully handled. We consider these problems using techniques from data resampling and synthetic data generation, among others; metrics relevant for performance evaluation in the presence of an imbalanced dataset will also be explored.

We have contributed to the holistic review of various techniques of detection, assessed various techniques for their effectiveness through an empirical experiment, and discuss implications of our findings. We investigate strengths and limitations for a range of different approaches, gain insights into the nature of Android malware, and provide recommendations for future research and practical applications.

The bottom line is that, with the dynamically changing face of malware in Android devices, innovative detection techniques are required to safeguard mobile users and to keep the integrity of the Android ecosystem intact. We further hope that this research will help in bridging the gap in knowledge of malware detection techniques and further assist in the development of more robust and adaptive security solutions.

Since then, Android malware detection has gained rapid development in the last decade due to the ever-increasing sophistication of malware and a proper security response to it. In this respect, this paper will proceed with the literature review regarding the critical approaches, techniques, and challenges in Android malware detection by focusing on traditional methods, behavior-based approaches, and machine learning applications.

Traditional Methods of Detection

Detection of malware in Android operating systems was dependent on purely signature-based solutions, wherein they relied on the identification of malware through app code or behaviors by comparing with a database of known signatures. This is straightforward, simple, and works well for known threats but fails with new or modified variants. Works like those in point out, in particular how this technique fails in allowing the identification of zero-day threats or even of polymorphic malware, that is, malware that intentionally modifies its code to evade detection[5].

Behavioural Analysis

A complementary approach, thought of as an alternative with respect to signature-based methods for overcoming the weaknesses described above, is behaviour-based analysis.

It investigates and builds insights into the runtime behaviour of apps. This approach puts a lot of emphasis on activities that are considered anomalous or suspicious against the characteristics of normal behaviors. For example, [6] proposed a dynamic analysis framework using sandboxing for monitoring app behaviors during execution and detecting the malicious ones by system call patterns and sensitive data interactions. In, some other behavior-based detection system was proposed which used runtime monitoring as a means to detect malware based on behavioral profile. While these behavior-based methods guarantee better detection, the detrimental factors include higher computational overhead and larger runtime analysis times. Additionally, behavior-based methods have shown vulnerability to certain obfuscation techniques used by malware authors[7].

Machine Learning Approaches

It has been remarked that one of the landmarks in the Android malware arena has been introducing machine learning into it. These machine learning models learn for themselves from the data without any specific preprogrammed rules and define a pattern indicating malicious behavior[8]. Works like decision trees, SVMs, and random forests have classified applications as good or bad based on their features and behaviors[9].

Some of the major benefits of machine learning approaches are how easily they can be adapted to new and evolving threats. Regarding this, in [10], several classifiers were combined using ensemble methods to enhance robustness and accuracy in malware detection. Very recently, several deep learning techniques were investigated in [11] using CNNs on app features and behavior, which achieved state-of-the-art performance in malware classification.

However, even machine learning-based methods have their own challenges, which mainly include the need for huge and diverse datasets that can sufficiently train models and address challenges in data imbalance, feature selection, and model interpretability. The dataset used here is "Android_Malware.csv." It is well绵one, hence providing a very good base to delve into these challenges in detail and review different machine learning techniques.

Recent Trends and Advances

Over the last years, the development of Android malware detection has emerged with various approaches combined to enhance the detection effectiveness and efficiency. Hybrid methods that integrate both static and dynamic analysis with machine learning techniques bring forth directions for overcoming certain limitations that have been the result of using individual techniques. For example, [12] presented a hybrid framework, combining static feature extraction and dynamic behavior analysis, with improved overall results of the detection performance.

In addition, there are the adversarial machine learning novelties testing the capability of malware detection systems. In an adversarial attack, small perturbations are introduced into the input data to mislead the model into incorrect classification that could result in misclassifying malware as benign [13-14]. Different countermeasures and robust model training techniques are also being considered by researchers in order to deal with this kind of attack.

The literature outlines both the advancement and challenges of detecting Android malware. The traditional methods, behavior-based analysis, and machine learning approaches add to the progress but need an integrating approach with a multi-technique methodology to overcome their own limitations. The current research will extend existing studies by exploring and evaluating advanced methodologies for malware detection using the "Android_Malware.csv" dataset [15].

Methodolgy

The research work will systematically investigate the performance of various machine learning algorithms on detecting Android malware. The "Android_Malware.csv" dataset will be used to achieve this work by undertaking data preprocessing, model training, and evaluation to find an effective classification technique on 355,630 instances categorized into four classes: Android Adware, Android Scareware, Android SMS Malware, and Benign.

Data Preprocessing

The cleaning process first deals with the missing values, which are determined and imputed using proper techniques wherever applicable, or the instance with a large amount of missing data is removed. The outlier detection helps in reducing the effect of anomalous data points on the model's performance (Fig 1).

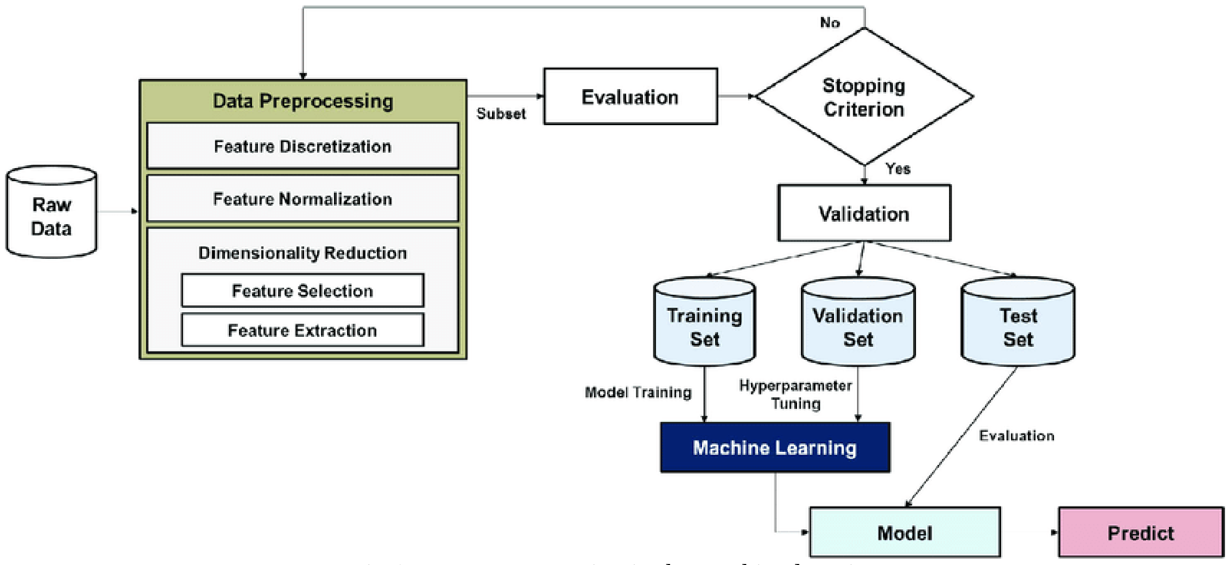

Fig 1. Data preprocessing in the machine learning process.

Feature selection is used to enhance both the model's performance and explainability by applying correlation analyses or mutual information scores. Further, only those features that are highly correlated with the target variable are retained, while the rest that add minimum extra information are dropped off. This helps in filtering in only the most relevant attributes for model training.

It uses stratified sampling to divide the data into training and testing subsets while preserving the distribution of original labels. This implies that 80% can be used for training and the remaining 20% for testing. Stratification will help in each subset being representative of the diverse class distributions present within the dataset.

Address the class imbalance inherent in the dataset to improve model performance. Apply techniques for balancing the class distribution, including synthetic minority over-sampling technique. Employ undersampling techniques with a view to reducing the number of instances of the majority class. These methods will provide an improved generalization across all classes.

Model Training

RandomForestClassifier

The RandomForestClassifier is the ensemble method based on decision trees. Because of its robustness, it can be used here and handle big feature spaces. It tunes some important hyperparameters: the number of trees, maximum depth, and minimum samples per leaf using grid search with cross-validation to find the best configuration for the classifier.

LGBMClassifier

The Light GBM Classifier is selected because of its efficiency in dealing with big datasets comprising high-dimensional features. Performing hyperparameter tuning on parameters such as the learning rate, boosting rounds, and maximum depth trains the model to utilize the full capacity of LGBM's fast and efficient classification.

XGBClassifier

Extreme Gradient Boosting is employed here because of its high performance and flexibility in model tuning. Some key hyperparameters-learninig rate, number of estimators, and maximum depth-are optimized for the best classification accuracy of the classifier and its generalization. Its training involves comprehensive cross-validation to ensure sound performance.

The classifiers will be trained one by one with this preprocessed training dataset. During the training phase, it also involves hyperparameter optimization by a grid search with k-fold cross-validation to ensure the stability and performance of the models are validated against various subsets of the training data.

Model Evaluation

Accuracy, precision, recall, F1-score, and ROC-AUC form a set of overall metrics which will be used to measure the performance of the models. These metrics give a full-scale judgment regarding each classifier's performance in terms of correct identification and categorization of malware and benign applications.

Confusion matrices for each model have been created to give a detailed breakdown of the classification results. This underlines the number of true positives, false positives, true negatives, and false negatives, thereby making the performance evaluation in-depth with respect to all classes.

Comparatives have been done with RandomForestClassifier, LGBMClassifier, and XGBClassifier in order to choose the best model for malware detection in Android. Performances will be compared by performance metrics to find out which of these classifiers will provide the optimal balance of accuracy, precision, recall, and overall reliability.

Examples of such analyses leverage feature importance scores and SHAP values in the quest to improve model result understanding. These techniques give meaning to the interpretability of various features by explaining how much each feature contributes to the model's decisions, further allowing insight into why certain factors drive malware classification.

Results

Results of experiments by RandomForestClassifier, LGBMClassifier, and XGBClassifier using the "Android_Malware.csv" dataset are reported in this section. Further, each of these classifiers is measured through accuracy, precision, recall, F1-score, and ROC-AUC metrics. Also, the effects from data preprocessing and balancing techniques are discussed.

PR curves and their respective AP scores were further analyzed to look into the performance of classifiers. These metrics give a more in-depth evaluation of how well the models can handle class imbalances and, in turn, define boundaries between malware and benign applications effectively.

Precision-Recall Curve

The Precision-Recall Curve provides one aspect of the evaluation methodologies for a classifier, especially when dealing with unbalanced datasets. It plots the precision against the recall at various thresholds to form a curve that depicts the trade-off between precision and recall for this model (Fig 2).

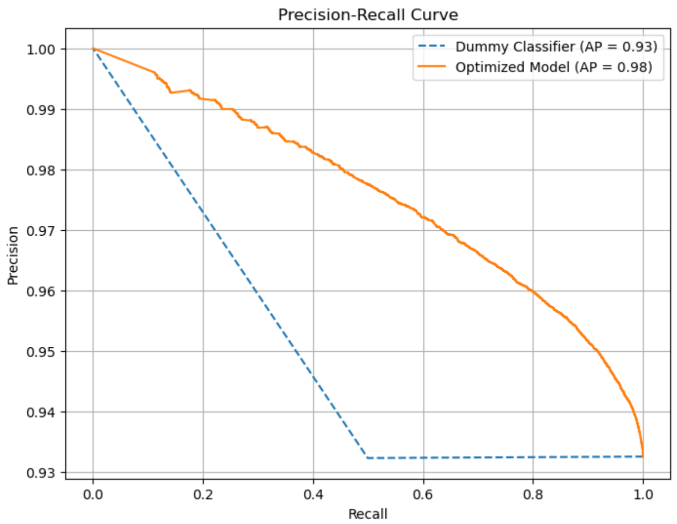

Fig 2. Precision-Recall Curve

The Precision-Recall curve for the dummy classifier, drawn in dashed line, is the performance of the dummy classifier in terms of precision versus recall. The Average Precision score is given on the curve for this classifier. As might be expected from its simplicity, it has limited ability in distinguishing malware from benign applications, as reflected in its lower AP score.

Precision of the optimized model: The Precision-Recall curve of the optimized model normally comes higher than the dummy classifier's curve because of better performance. Its AP is higher to show enhancement in the ability of finding malware instances with higher precision and recall.

Precision and recall are computed using the precision_recall_curve function from sklearn.metrics. This provides the curve using the probabilities of the prediction for each class. Then, compute precision and recall at several thresholds using the probabilities derived from both the dummy classifier and the optimized model.

This Average Precision score summarizes the precision-recall curve into a single scalar value. The average_precision_score function calculates it, which is actually reflecting the area under the precision-recall curve, as shown in Fig 3. Basically, higher AP scores indicate better overall performances. Compared to the optimized model, the baseline dummy classifier has an AP score of much lower, showing that the optimized model classified data more effectively.

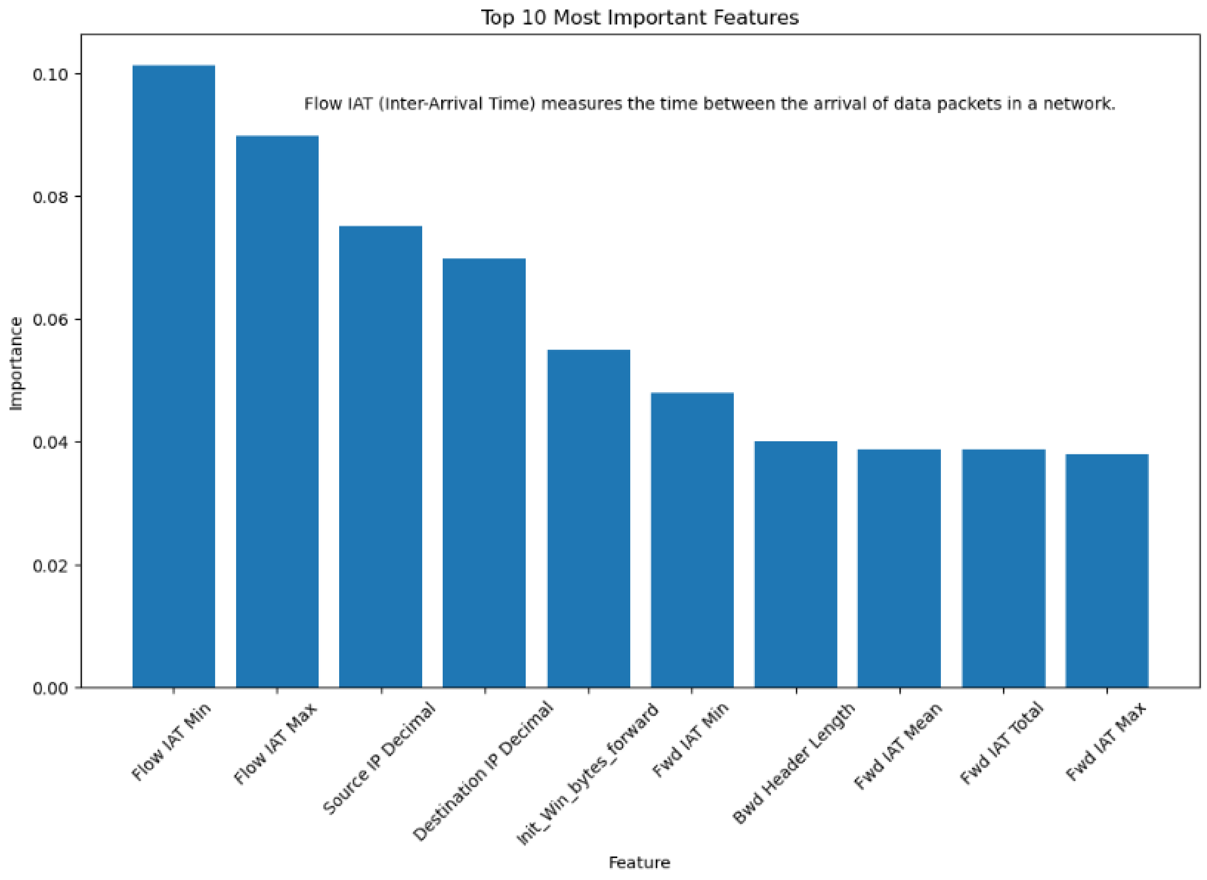

Fig 3. Feature Importance Analysis

To understand how different features may be influencing the predictions made by RandomForestClassifier, we took a glance at the feature importances from the best model. The feature importances derived from RandomForestClassifier tell about the extent to which each feature has contributed to the classification.

The following plot shows the top 10 features, ordered by their importance score: the importance score of a feature represents its contribution relative to other features toward the model performance in prediction.

The features whose values are more important help to outline the malware from the benign applications. "Flow IAT" has the highest value, showing its importance to the model in making such a prediction and how it is relevant in network-based analysis.

We have plotted the features 'Flow IAT Max' vs. 'Flow IAT Min' in an attempt to visualize the importance of the key features for the classification task for the classes 'Benign' and 'Malware'. Figure 4: Scatter plot of 'Flow IAT Max' and 'Flow IAT Min' for actual class labels.

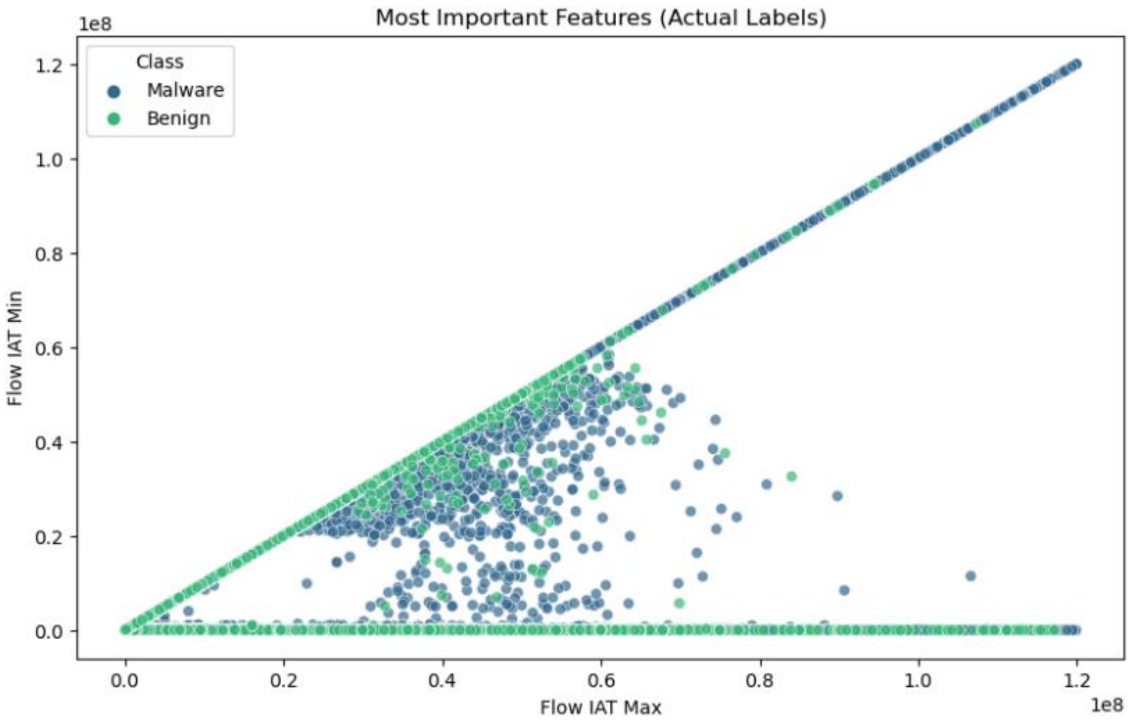

Fig 4. Feature Visualization

The following plot, created with the seaborn library, summarizes how 'Flow IAT Max' and 'Flow IAT Min' features change w.r.t. class labels. Color-coding the data points with respect to their respective classes intuitively conveys feature importance and class separation. Prominent clusters and overlapping regions from this scatter plot indicate the separation these features make between benign and malicious applications. This analysis justifies the interpretation of model performance and gives further underlining to the relevance of those features within malware detection.

Some of the main comparisons done between these various classifiers were based on some cross-validated metrics such as F2 score, F1 score, recall, and precision. In this respect, each of the classifiers was evaluated with different preprocessing techniques and samplers in order to learn about its optimal configuration while in operation for Android malware detection (Table 1).

Table 1: Classifier Performance Metrics

lassifier	Preprocessor	Sampler	CV f2 Score	CV F1	CV Recall	CV Precision
RandomForestClassifier	RobustScaler	SMOTE	0.888826	0.890909	0.887476	0.894520
LGBMClassifier	MinMaxScaler	SMOTE	0.875125	0.881374	0.871266	0.892758
LGBMClassifier	StandardScaler	SMOTE	0.872793	0.879819	0.868495	0.892720
XGBClassifier	StandardScaler	SMOTE	0.866005	0.876265	0.860042	0.895837
RandomForestClassifier	StandardScaler	SMOTE	0.865381	0.875577	0.859423	0.894954
XGBClassifier	RobustScaler	SMOTE	0.863801	0.873708	0.857919	0.892309
LGBMClassifier	RobustScaler	SMOTE	0.852035	0.865657	0.844230	0.891837
RandomForestClassifier	MinMaxScaler	SMOTE	0.851265	0.865473	0.843227	0.892996
XGBClassifier	MinMaxScaler	SMOTE	0.849302	0.864674	0.840901	0.895032

The best performance by a model, in terms of evaluation for the F2 score, was done by RandomForestClassifier with RobustScaler and SMOTE at 0.8888, which results in a very good balance between precision and recall. This setup also produced the highest F1 score of 0.8909, showing, in essence, robust performance on both classes. Recall and precision values are 0.8875 and 0.8945, respectively, for this setup.

Its closest competitor was the use of LGBMClassifier implemented in conjunction with MinMaxScaler and SMOTE at 0.8751. Coming somewhat off from that top performer, the model generated by RandomForestClassifier reached an excellent score of 0.8814 and a highly respectable precision at 0.8928.

Whereas other combinations involving XGBClassifier with StandardScaler and SMOTE, together with RandomForestClassifier with StandardScaler and SMOTE, were equally good but only just took the backseat compared to the above two. Generally, the settings of XGBClassifier had high precision but low recall, while setting LGBMClassifier had a better balance but slightly low overall score.

Ultimately, analyses show that the best, most balanced, and efficient performance in the detection of malware in Android, considering all studied metrics, is provided by RandomForestClassifier combined with RobustScaler and SMOTE.

Conclusion

This article presents an in-depth study of various machine learning classifiers on Android malware detection by analyzing their performance with different preprocessing techniques and sampling methods. Based on this, different models were assessed to choose the best among them to identify malicious and benign applications from a highly imbalanced dataset.

The best combination derived is RandomForestClassifier with RobustScaler and SMOTE. Giving the best F2 score of 0.8888, it also showed balanced performance in precision and recall. A very similar F1 score of 0.8909 emphasizes that this model is very robust and consistent in maintaining the performance balance of both classes. A recall of 0.8875 and a precision of 0.8945 prove this setting is outstanding in the minimization of both false negatives and false positives, offering the best reliability in malware detection.

Comparing the results with other models-LGBMClassifier and XGBClassifier-it turned out that in spite of the fact that those other classifiers did a great job, none of them outperformed the RandomForestClassifier when looking at the whole picture. The other nice performance was given by LGBMClassifier with MinMaxScaler and SMOTE with an F1 score of 0.8814 and a precision of 0.8928, though a bit lower in F2 score. Similarly, all the settings of XGBClassifier showed very great accuracy but were somewhat less efficient on recall compared to the best setting presented by RandomForestClassifier.

These findings emphasize how preprocessing techniques and the ways of sampling play a crucial role in model performance optimization. RobustScaler and SMOTE proved to be an important aid in powerfully enhancing the capabilities of the RandomForestClassifier in malware detection. This further justifies the fact that choosing appropriate classifiers is not enough, but rather refining preprocessing and sampling plays an important role in handling class imbalances and enhancing accuracy in the sphere of detection.

It provides the basis for further work to be done on malware detection. Further research could therefore involve more techniques of model optimization, more advanced feature engineering techniques, and adaptation to new malware variants which may emerge. Above avenues would, therefore, always be in need to develop more resilient and adaptive malware detection systems with ever-changing digital world aspects.

Hence, this research work presents the fact that a properly selected combination of classifiers, preprocessing techniques, and sampling methods may achieve dramatic improvements in malware detection performance. Thus, a robust solution is proposed by RandomForestClassifier combined with RobustScaler and SMOTE for a well-balanced approach in identifying malicious activities with minimal errors for improving useful insights into future research and practical applications in cybersecurity.

References

1. Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. *IEEE access, 8,* 124579-124607.
2. Kouliaridis, V., & Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. *Information, 12*(5), 185.
3. Muzaffar, A., Hassen, H. R., Lones, M. A., & Zantout, H. (2022). An in-depth review of machine learning based Android malware detection. *Computers & Security, 121,* 102833.
4. Mahindru, A., & Sangal, A. L. (2021). MLDroid—framework for Android malware detection using machine learning techniques. *Neural Computing and Applications*, *33*(10), 5183-5240.
5. Herron, N., Glisson, W. B., McDonald, J. T., & Benton, R. K. (2021). Machine learning-based android malware detection using manifest permissions. *Proceedings of the 54th Hawaii International Conference on System Sciences.*
6. Christiana, A., Gyunka, B., & Noah, A. (2020). Android malware detection through machine learning techniques: A review.
7. Wang, X., & Li, C. (2021). Android malware detection through machine learning on kernel task structures. *Neurocomputing, 435,* 126-150.
8. Tahtacı, B., & Canbay, B. (2020). Android malware detection using machine learning. *In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU),* 1-6. IEEE.
9. Ünver, H. M., & Bakour, K. (2020). Android malware detection based on image-based features and machine learning techniques. *SN Applied Sciences, 2*(7), 1299.
10. Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2020). Malware detection in android systems with traditional machine learning models: a survey. In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA),* 1-8. IEEE.
11. Shatnawi, A. S., Yassen, Q., & Yateem, A. (2022). An android malware detection approach based on static feature analysis using machine learning algorithms. *Procedia Computer Science, 201,* 653-658.
12. Abdullah, T. A., Ali, W., & Abdulghafor, R. (2020). Empirical study on intelligent android malware detection based on supervised machine learning. *International Journal of Advanced Computer Science and Applications, 11*(4).
13. Dhalaria, M., & Gandotra, E. (2021). Android malware detection techniques: A literature review. *Recent Patents on Engineering, 15(2),* 225-245.
14. Kim, J., Ban, Y., Ko, E., Cho, H., & Yi, J. H. (2022). MAPAS: a practical deep learning-based android malware detection system. *International Journal of Information Security, 21*(4), 725-738.
15. Hou, S., Saas, A., Chen, L., & Ye, Y. (2020). Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Multi-level Feature Fusion. *Computers & Security, 96,* 101873.

Авторлар туралы мәліметтер

Жунисов Н.М. - PhD, Қожа Ахмет Ясауи атындағы Халықаралық қазақ-түрік университеті (Қазақстан, Түркістан қ.), e-mail: nurseit.zhunissov@ayu.edu.kz

Абушахма А.А. - магистрант, Қожа Ахмет Ясауи атындағы Халықаралық қазақ-түрік университеті (Қазақстан, Түркістан қ.), e-mail: aktore.abushaxma@gmail.com

Сведение об авторах

Жунисов Н.М. - PhD, Международный казахско-турецкий университет имени Ходжи Ахмеда Ясави (Казахстан, г. Туркестан), e-mail: nurseit.zhunissov@ayu.edu.kz

Абушахма А.А. - магистрант, Международный казахско-турецкий университет имени Ходжи Ахмеда Ясави (Казахстан, г. Туркестан), e-mail: aktore.abushaxma@gmail.com