[1] K S Ranadheer Kumar

[2] K S Ranadheer Kumar

[3] Jagadish Gurrala

Enhancing Android Malware Detection through Filter-Based Feature Selection and Machine Learning Classification

Journal of Electrical Systems

Abstract: - The escalating prevalence of Android malware poses a significant threat to cybersecurity. This research explores the effectiveness of filter-based feature selection techniques, specifically Information Gain, Chi-Square Test, and Fisher's Score, in analyzing Android permission patterns for malware detection. A suite of machine learning classifiers, including Decision Trees, K-Nearest Neighbors, Random Forest, Support Vector Machine, and Logistic Regression, were employed to evaluate the performance of these techniques. Results demonstrate that filter-based feature selection utilizing Information Gain and Fisher's Score outperformed the Chi-Square Test in terms of feature reduction, achieving classification accuracies of 91.53% and 91.22% respectively on the high-dimensional CICInvesAndMal2019 dataset. This study highlights the potential of filter-based feature selection methods, particularly Information Gain and Fisher's Score, for enhancing the efficiency and accuracy of Android malware detection.

Keywords: Android malware, feature selection, Information Gain, Chi Square Test, Fisher Score, Machine Learning

1 Introduction

In contemporary times, there has been a notable focus on research pertaining to the security of mobile devices running on the Android operating system [1] [2]. A significant concern within this realm is the issue of undesirable permissions linked to the installation of Android applications, which serve as a primary conduit for Android malware. The pace at which new forms of malware are being uncovered is alarming, with a discovery rate of approximately one instance every 10 seconds [3]. This trend is further underscored by the staggering identification of around 3 million malicious Android applications in the year 2020 alone.

While the Android operating system has incorporated dynamic permission management and user-facing warnings regarding suspicious permissions during app installation, these safeguards are not foolproof. Despite the alerts, users often grant permissions without thorough scrutiny, potentially compromising device security. Moreover, malicious applications can still exploit vulnerabilities to download and execute harmful code after installation, enabling the exfiltration of sensitive data to remote servers. This highlights the ongoing need for robust security mechanisms to counter the evolving tactics employed by malicious actors in the Android ecosystem.

Numerous permissions are requested by Android applications during installation, which users may or may not knowingly grant. The risk of malware infiltration is particularly pronounced in applications sourced from outside the official Play Store [5]. Analyzing the permissions requested by applications can serve as a valuable method for detecting Android malware. Developing an anti-malware system capable of scrutinizing permissions requested during installation could alert Android users to potential threats. Machine learning tools can be employed for the dynamic analysis of Android permissions. These permissions are characterized by continuous evolution and high dimensionality, as evidenced by the comprehensive CICInvesAndMal2019 dataset containing 4,115 features [6]. Therefore, there is a critical need to conduct experiments aimed at identifying a reduced feature set using optimization techniques to minimize false positives and enhance accuracy.

This paper proposes a novel methodology for detecting malicious Android applications by leveraging the distinct permission patterns they exhibit. The approach integrates machine learning algorithms, including Random Forest, K-Nearest Neighbors, Support Vector Machines, Decision Trees, and Logistic Regression, to classify applications based on their permission requests. To enhance the efficiency and accuracy of malware detection, filter-based feature selection techniques such as Information Gain, Chi-Square test, and Fisher's Score are applied.

The paper is structured as follows: Section 2 provides a comprehensive overview of relevant literature, Section 3 details the methodology employed, Section 5 describes the experimental setup, Section 6 presents the performance analysis and experimental results, and Section 7 concludes the study, outlining potential avenues for future research.

2 Related Work

[1] Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Green fields, Guntur, Andhra Pradesh, India. 2202031117@kluniversity.in

[2] Assistant Professor, Department of CSE (Data Science). CVR College of Engineering, Hyderabad. ranadheer.k.s@gmail.com

[3] Associate Professor, Dept. of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Green fields, Guntur, Andhra Pradesh, India. gjagadish@kluniversity.in

Recent research has focused on developing more sophisticated feature selection algorithms to improve the accuracy and efficiency of malware detection, particularly in the Android environment. Hein and Myo [10] pioneered a permission-based malware classification system, extracting data from the Android manifest file and employing dimensionality reduction techniques. While their study demonstrated the potential of permission-based analysis, they also acknowledged its limitations, highlighting the need for a more comprehensive approach. Sanz et al. [11] further explored permission-based malware identification with their PUMA system, achieving a noteworthy 92% accuracy using the Random Forest classifier. However, they emphasized that incorporating dynamic analysis techniques could further enhance the effectiveness of malware detection. These studies highlight the crucial role of permission patterns in identifying malicious applications, while also underscoring the need for integrating dynamic analysis into future malware detection systems.

Aung and Zaw [12] developed a framework for classifying Android applications as either benign or malicious using machine learning techniques. Their approach employed a range of dimensionality reduction methods, including Gain Attribute Evaluator, Relief-F Attribute Evaluator, Cfs Subsystem Evaluator, and Consistency Subsystem Evaluator, to optimize the feature set. They evaluated the performance of several classification algorithms, including J48, Decision Tree, Bayesian classification, Random Forest, Regression Tree, and Sequential Minimal Optimization (SMO), using metrics such as accuracy, true positive rate, false-positive rate, and precision. While their study demonstrated the effectiveness of their approach, it was limited by the evaluation dataset consisting of only 200 samples. Further research with larger and more diverse datasets is necessary to validate the generalizability of their findings.

Further exploring the use of machine learning for Android malware detection, Singh et al. [13] developed a permission-based framework utilizing Principal Component Analysis (PCA) for feature selection and a Support Vector Machine (SVM) classifier, achieving a 90.08% accuracy rate. Suleiman et al. [14] expanded on this approach, extracting 58 features, including API calls and system commands, using APK Analyzer. Through experimentation with a Bayesian classifier, they determined that a reduced set of 15-20 features was sufficient for optimal performance, outperforming traditional signature-based antivirus software. However, both studies acknowledge the need for further validation with larger and more diverse datasets to confirm their findings and ensure the robustness of their models against evolving malware tactics.

Zhang et al. [15] propose a novel method for not only detecting Android malware but also classifying it into specific malware families. Their approach utilizes n-gram analysis and feature hashing to extract multi-level fingerprints from applications. These fingerprints are then fed into an online classifier trained on the Android Malware Genome Project (AMGP) dataset. Utilizing open-source tools like xxd and aapt, they generate n-grams for each sub-fingerprint, which are then hashed into bit-vectors using the Scikit-learn toolkit. Notably, their method achieved impressive accuracies of 99.2% and 98.8% in identifying malware families within the Derbin Dataset. Despite this high performance, the authors acknowledge that incorporating dynamic runtime analysis into the sub-fingerprint extraction process could further improve the model's ability to detect and classify evolving malware strains.

Oktay and Ibrahim [16] investigated the efficacy of using a Genetic Algorithm (GA) for feature selection in Android malware detection. Their approach involved training three classifiers—Naive Bayes, Decision Trees, and Support Vector Machines—each utilizing a unique subset of features selected by the GA. Comparing the performance of these classifiers, they found that the SVM classifier achieved the highest accuracy (98.45%) when applied to the 16 features identified by the GA. However, their study utilized the AMGP dataset, which is known for its outdated nature. Therefore, further research employing more recent datasets with permissions from higher API levels is crucial to validate the effectiveness and generalizability of their proposed method.

Allix et al. [17] undertook a comprehensive forensic analysis of Android malware, examining a vast dataset comprising 500,000 applications, both benign and malicious. Their investigation focused not only on malware identification through antivirus solutions but also on understanding the methodologies employed by malware authors. Significantly, their analysis revealed a critical vulnerability stemming from a lack of cryptography knowledge among malware writers. This deficiency frequently leads to the reuse of digital certificates across both benign and malicious applications, as current mitigation techniques fail to adequately verify their authenticity. This finding underscores the need for improved security measures that can effectively detect and prevent such certificate misuse.

Shifting from static analysis to behavioral analysis, Zaman et al. [18] developed a malware detection method based on traffic analysis and system call monitoring. Their approach involved creating a blacklist of known malicious applications and implementing a logging mechanism to record communications between apps and remote servers. By comparing the blacklisted domains with the logged URLs, they were able to identify potentially malicious behavior. Patel [19] provided a comprehensive overview of various Android malware detection models, including DroidNative, DREBIN, ICCDetector, APK Auditor, DMDAM, AndroDialysis, and API-based systems, analyzing the techniques employed by each model. Further comparative analyses of related works are presented in Table 1.

Table 1. Related work comparison

Paper, Year	Dataset	Classifier	Feature Selection	No. of Features	Accuracy
[23], 2018	MODroid	Linear discriminant analysis (LDA), RF, SVM	-	134	85.50%
[24], 2018	Own Dataset	RF, SVM, KNN, NB, LR	NA	696	81%
[25], 2019	MODroid	NB, RF, SVM	NA	134	86.06%
[26], 2014	Google Play Store	Dalvik Virtual Machine	NA	-	85%
[27], 2018	Own Dataset	NB	cfsSusetEval + Random Search, PCS+Ranker, ClassifierSubsetEval + RandomSearch	11	80%

This study proposes a novel approach for Android malware classification by employing filter-based feature selection techniques. Specifically, Information Gain, Chi-Square test, and Fisher Score are utilized to identify and select a reduced subset of highly relevant features from high-dimensional datasets, thereby enhancing the efficiency and accuracy of subsequent classification models.

3 Methodology

Feature selection plays a critical role in optimizing machine learning models by identifying and retaining only the most salient features while eliminating irrelevant or redundant ones. This process not only enhances the model's efficiency but also improves its accuracy. This study focuses on a filter-based feature selection technique, employing algorithms such as Information Gain, Chi-Square test, and Fisher's Score to effectively discern the most informative features within high-dimensional datasets. By reducing the feature space to the most discriminative attributes, these methods aim to enhance the performance of subsequent classification models.

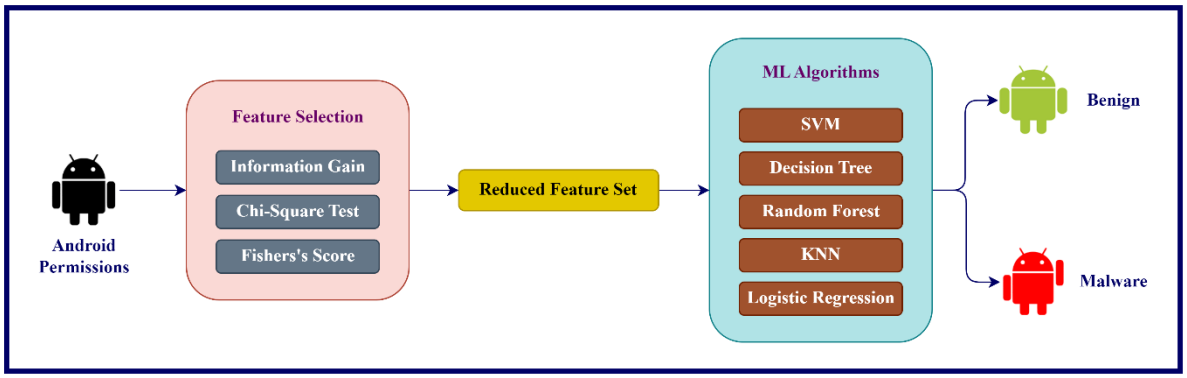


Fig 1. System Architecture

The proposed model follows a systematic process for Android malware classification. Initially, the complete feature set of the dataset is considered. Subsequently, a refined subset of features is carefully selected through the application of filter-based feature selection techniques. This reduced set of highly relevant features is then employed for intrusion detection, leveraging a chosen machine-learning algorithm. The architecture of this model, illustrating the sequential flow of data processing and analysis, is depicted in Figure 1.

3.1. Information Gain

Information Gain (IG) [20] is a fundamental concept in machine learning and data mining, particularly in the context of feature selection. It serves as a criterion for measuring the relevance of a feature in predicting the target variable of a dataset. The IG metric quantifies the reduction in entropy achieved by splitting the data based on a specific feature, thereby indicating the informativeness of that feature. By selecting features with higher Information Gain, machine learning models can focus on the most relevant aspects of the data, leading to improved model performance and interpretability. The detailed explanation of the Information Gain feature selection technique, includes its calculation process and significance in enhancing the efficiency and effectiveness of machine learning algorithms.

Entropy Calculation: Entropy is a measure of uncertainty or disorder in a set of data. It is calculated for the target variable (or class labels) before and after splitting the data based on a feature. The formula to calculate entropy is

$$Entropy\ (S) = -\sum_{i=1}^{c} p_i\ log_2\ p_i \quad (1)$$

Where, S is the set data, c is the number of classes in target variable, p_i is the proportion of the number instance in class i to the total number of instances.

Information Gain Calculation: Information Gain (IG) serves as a measure of the decrease in entropy, or uncertainty, achieved by partitioning a dataset based on a specific feature. In essence, IG quantifies the informativeness of a feature in discerning between different classes within the dataset. The formula for calculating IG is as follows:

$$IG\ (S, A) = Entropy(S) - \Sigma_{v\ \in\ values\ (A)} \frac{|S_v|}{|S|} \times Entropy(S_v) \quad (2)$$

Where, S is the original dataset, A is the feature being considered for splitting, $values\ (A)$ are the possible values of feature A. S_v is the subset of S for which feature A has value v. $|S|$ and $|S_v|$ denote the number of instances in sets S and S_v respectively.

Selecting the Best Feature: Calculate the IG for each feature in the dataset and select the feature with the highest IG. This feature is considered the most informative and is chosen for splitting the dataset.

Repeat: This recursive partitioning process is then repeated for each resulting subset, further splitting the data based on the selected features. This iterative procedure continues until a predefined stopping criterion is met. Such criteria may include reaching a maximum tree depth, achieving a minimum number of instances within a node, or attaining a desired level of purity within the subsets.

3.2. Chi-Square Test

Feature selection is a crucial step in machine learning model development, aimed at identifying the most relevant features that contribute to the prediction task. Among the various techniques available, the Chi-Square (χ^2) [21] test stands out as a valuable method for feature selection, particularly in scenarios involving categorical data. In the context of machine learning, the Chi-Square test is applied to assess the relationship between each feature and the target variable, aiding in the selection of informative features for model training. The detailed explanation of the step-by-step process involved in the Chi-Square feature selection technique, includes the calculation of the Chi-Square statistic, determination of degrees of freedom, and assessment of significance.

Understanding the Chi-Square Test: The Chi-Square test is based on the principle of comparing the observed frequencies of categorical variables with the expected frequencies under the assumption of independence.

Calculating the Contingency Table: To perform the Chi-square test, a contingency table, also known as a cross-tabulation table, is constructed. This table displays the frequency distribution for every possible combination of categories between the two variables under consideration. In the context of feature selection, one of these variables typically represents the target variable (e.g., class labels), while the other represents the feature variable being evaluated (e.g., feature values).

Calculating Expected Frequencies: Subsequently, the expected frequencies for each cell within the contingency table are calculated. These expected frequencies represent the values that would be observed if there were no association between the target variable and the feature variable, i.e., under the assumption of independence. The formula for calculating the expected frequency of a cell is as follows:

$$Expected\ Frequency = \frac{Row\ Total\ x\ Column\ Total}{Grand\ Total} \quad (3)$$

Calculating the Chi-Square Statistic: The Chi-Square statistic is computed by first determining the squared differences between the observed frequencies (obtained from the data) and the expected frequencies (calculated based on the assumption of independence). Each squared difference is then divided by the respective expected frequency. These values are subsequently summed across all cells within the contingency table. This process is mathematically expressed by the following formula:

$$\chi^2 = \frac{(Observed - Expected)^2}{Expected} \quad (4)$$

Calculating Degrees of Freedom: Degrees of freedom (df) is calculated as

$$(number\ of\ rows - 1)\ X\ (number\ of\ columns - 1) \quad (5)$$

Determining Significance: Finally, the significance of the Chi-Square statistic is assessed using a critical value from the Chi-Square distribution table. If the calculated Chi-Square value is greater than the critical value for a given significance level (e.g., 0.05), then the null hypothesis of independence is rejected, indicating that the two variables are dependent and the feature is considered important for classification.

Feature Selection: Features with high Chi-Square values and low p-values (significance level) are considered important and can be selected for further analysis or model building.

3.3. Fisher's Score

Fisher's Score [22], also known as Fisher's Discriminant Ratio, is a widely used feature selection technique that evaluates the discriminatory power of individual features in classification tasks. By measuring the ratio of between-class variance to within-class variance, Fisher's Score identifies features that are most effective in distinguishing between different classes. The detailed explanation of the step-by-step process involved in Fisher's

Score feature selection, includes the calculation of mean vectors for each class, the computation of between-class and within-class scatter matrices, and the derivation of Fisher's Score for each feature.

Calculate the Mean Vectors: Calculate the mean vector for each class. If you have C classes and n features, you will have C mean vectors of size n. Mean vector for class c:

$$\mu_c = \frac{1}{N}\sum_{i=1}^{N_c} x_{c_i} \quad (6)$$

Where, N_c is the number of samples in class c and x_{c_i} is the i^{th}sample is class c.

Calculate the Between-Class Scatter Matrix (SB): This matrix, known as the between-class scatter matrix, quantifies the variance between different classes within the dataset. Its calculation involves summing the outer products of the differences between each individual class mean vector and the overall mean vector of the entire dataset. This process effectively captures the degree of separation between the classes based on their feature values.

$$SB = \sum_{c=1}^{C} N_c(\mu_c - \mu)(\mu_c - \mu)^T \quad (7)$$

Where, μ is the overall mean vector calculated as the mean of all samples.

Calculate the Within-Class Scatter Matrix (SW): This matrix measures the variance within classes and is calculated as the sum of the scatter matrices for each class.

$$SW = \sum_{c=1}^{C}\sum_{i=1}^{N_c}(x_{ci} - \mu_c)\left(x_{cj} - \mu_c\right)^T \quad (8)$$

Calculate the Fisher's Score (F): Fisher's Score for each feature is calculated as the ratio of the trace of SB to SW.

$$F = \frac{trace\ (SB)}{trace\ (SW)} \quad (9)$$

Where, $trace\ (A)$ denotes the sum of the diagonal elements of matrix A.

Select Features: Features with higher Fisher's scores are selected as they are deemed more discriminatory for classification tasks.

4 Experimentation Setup

The experimental setup for this study was conducted on a Windows 10 (64-bit) operating system running on an Intel® Core™ i5 CPU @1.80 GHz processor, equipped with 8 GB of RAM and a 1 TB HDD. The system employed the Anaconda distribution of Python, integrating essential machine learning libraries.

The CICInvesAndMal2019 dataset, selected for its high dimensionality, served as the evaluation dataset. This dataset comprises 1594 instances, including 1187 benign and 407 malware samples, described by 4115 Android permissions. The malware samples are classified into 42 distinct families, which are further grouped into broader categories such as Adware, SMS, Ransomware, and Scareware. A detailed breakdown of the dataset's composition is presented in Table 2.

Table 2. Dataset description

Dataset	Features	No. of Samples		Size
CICInvesAndMal2019	4115	1594		12.7 MB
		Malware	Benign	
		1187	407	

5 Performance Analysis and Experimental Results

The proposed Android malware detection system evaluates its classification accuracy using several standard metrics, including Mean Squared Error (MSE), Root Mean Square Error (RMSE), Precision, Recall, F1-Score, and Accuracy. These metrics are defined as follows:

$$\text{Precision} = \frac{True\ Positive}{(False\ Positive + True\ Positive)} \quad (10)$$

$$\text{Recall} = \frac{True\ Positive}{(False\ Negative + True\ Positive)} \quad (11)$$

$$\text{F1-Score} = \frac{2 * Precision * Recall}{(Precision + Recall)} \quad (12)$$

$$\text{Accuracy} = \frac{True\ Positive + True\ Negative}{(True\ Positive + False\ Positive + False\ Negative + True\ Negative)} \quad (13)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad (14)$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \quad (15)$$

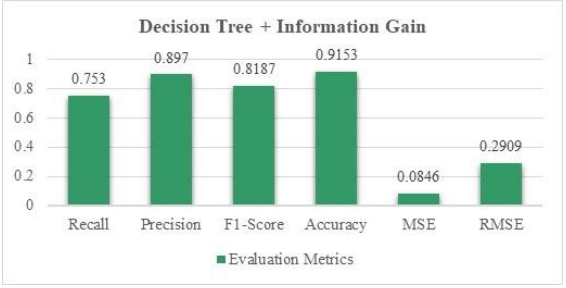
In evaluating the performance of the classification models, standard metrics were employed, including $True\ Positive$ (TP), $True\ Negative$ (TN), $False\ Positive$ (FP), and $False\ Negative$ (FN). TP represents the number of benign samples correctly classified, while TN represents the number of malware samples correctly classified. Conversely, FP indicates the number of benign samples incorrectly classified as malware, and FN represents the number of malware samples incorrectly classified as benign. These metrics are formally defined as follows: TP represents instances where the predicted output $\hat{Y}_i$ correctly identifies a benign sample, TN represents instances where $\hat{Y}_i$ correctly identifies a malware sample, FP occurs when $\hat{Y}_i$ incorrectly classifies a benign sample as malware, and FN occurs when Y^_i incorrectly classifies a malware sample as benign. Here, $\hat{Y}_i$ denotes the actual output or true class label, and n represents the total number of samples in the dataset.

To assess the efficacy of the Information Gain, Chi-Square test, and Fisher's Score feature selection methods, five widely-used classification algorithms were employed: logistic regression, k-nearest neighbors, decision tree, support vector machine, and random forest. Each feature selection method was evaluated in conjunction with each of these classifiers, resulting in a comprehensive set of experiments. The performance metrics for each combination of feature selection method and classifier are presented in Table 3. This comparative analysis provides insights into the effectiveness of each feature selection technique in enhancing the performance of different classification algorithms.

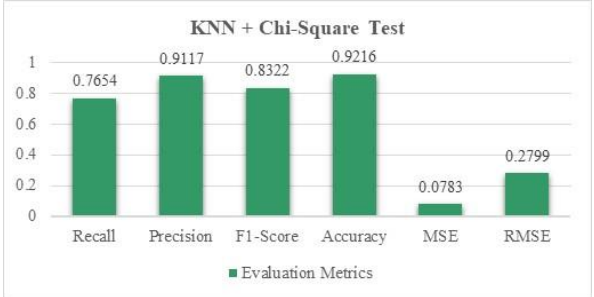
Table 3. Accuracy comparison of experimented filter-based feature selection methods

Feature Selection Method	Classifier	Accuracy Before Feature Selection	Accuracy After Feature Selection	% Change in Accuracy	No. of Features Selected	% Decrease in Features	Time taken for Feature Reduction in Sec.
Information Gain	SVM	0.915360502	0.89968652	-1.71%	49	98.81%	35.7
	DT	**0.909090909**	**0.915360502**	**0.69%**	**49**	**98.81%**	
	KNN	0.912225705	0.87460815	-4.12%	49	98.81%	
	LR	0.909090909	0.887147335	-2.41%	49	98.81%	
	RF	0.940438871	0.909090909	-3.33%	49	98.81%	
Chi Square Test	SVM	0.915360502	0.902821317	-1.37%	50	98.78%	0.204558849
	DT	0.909090909	0.912225705	0.34%	50	98.78%	
	KNN	**0.912225705**	**0.921630094**	**1.03%**	**50**	**98.78%**	
	LR	0.909090909	0.887147335	-2.41%	50	98.78%	
	RF	0.940438871	0.915360502	-2.67%	50	98.78%	
Fishers Score	SVM	0.915360502	0.896551724	-2.05%	49	98.81%	4.167310238
	DT	**0.909090909**	**0.912225705**	**0.34%**	**49**	**98.81%**	
	KNN	0.912225705	0.887147335	-2.75%	49	98.81%	
	LR	0.909090909	0.884012539	-2.76%	49	98.81%	
	RF	0.940438871	0.927899687	-1.33%	49	98.81%	

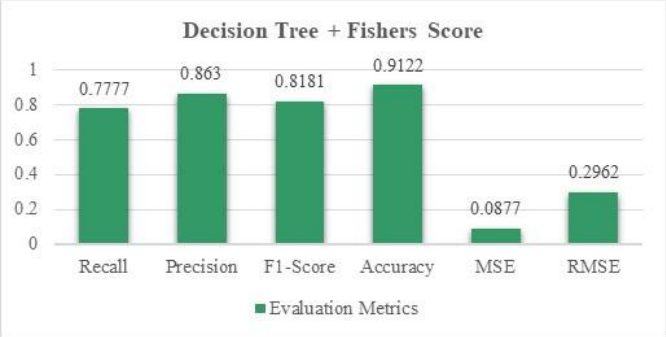
The performance analysis of prominent classification algorithms integrated with feature selection methods, namely Information Gain, Chi-Square test, and Fisher's Score, are illustrated from Graphs 1 to 3.



Graph 1. Performance Analysis of Decision Tree with Information Gain



Graph 2. Performance Analysis of KNN with Chi-Square Test



Graph 3. Performance Analysis of Decision Tree with Fishers Score

Figures 2 to 4 depict the Area Under the Curve (AUC) of the ROC curves corresponding to the aforementioned graphs. The results indicate that the Decision Tree algorithm, when employed with Information Gain and Fisher's Score, exhibits a larger area under the curve, suggesting better performance in terms of feature space optimization.

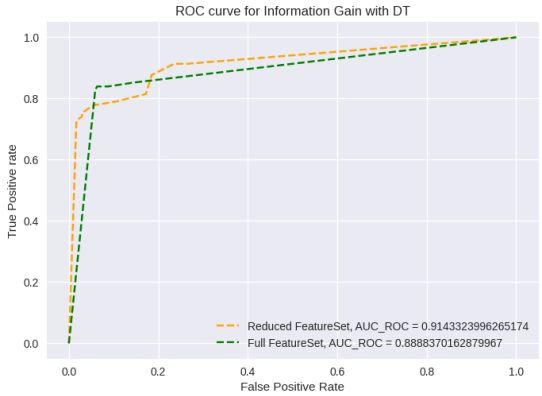


Fig 2. ROC curve for DT – IG

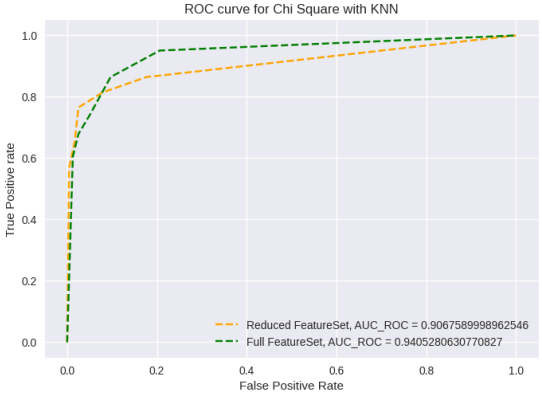


Fig 3. ROC curve for KNN – Chi Square

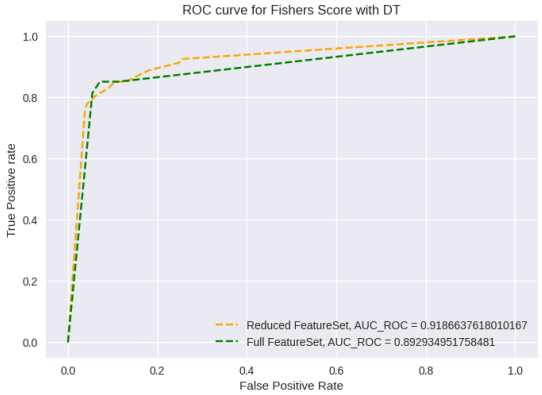


Fig 4. ROC curve for DT – Fishers Score

Among the various experiments conducted, the decision tree classifier using Information Gain for feature selection exhibited the best performance, achieving a 0.69% improvement in accuracy with only 49 features selected out of 4115. Similarly, the decision tree classifier using Fisher's Score as the feature selection criterion outperformed other comparison algorithms, achieving a 0.34% increase in accuracy with 50 features. Lastly, the KNN classifier using the Chi-Square test for feature selection demonstrated superior performance compared to other feature selection algorithms, achieving an overall increase in accuracy of 1.03% with only 49 selected features, indicating a substantial 98.81% reduction in the feature space.

6 Conclusion and Future Work

The increasing reliance on smartphones necessitates robust security measures to protect user data from Android malware attacks. This study addresses this challenge by exploring the efficacy of three filter-based feature selection algorithms – Information Gain, Chi-Square Test, and Fisher's Score – in conjunction with five machine learning classifiers: KNN, Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM), and Random Forest (RF). Utilizing the high-dimensional CICInvesAndMal2019 dataset, which encompasses 4115 features, the performance of each feature selection method paired with each classifier was rigorously evaluated. Notably, the Decision Tree classifier, when combined with either Information Gain or Fisher's Score, achieved the

highest accuracy of 95.09%, utilizing a significantly reduced subset of only 49 features. While the Chi-Square Test, in conjunction with the KNN classifier, also demonstrated respectable accuracy (92.16% with 50 features), its performance in terms of AUC ROC in the reduced feature spaces was lower compared to using the full feature set. These findings underscore the effectiveness of Information Gain and Fisher's Score in reducing dimensionality and improving the performance of machine learning models for Android malware detection in high-dimensional datasets.

This research lays the groundwork for several promising avenues of future exploration. Subsequent studies will focus on developing and evaluating wrapper-based feature selection techniques, aiming to further refine the selection process and potentially uncover even more performant feature subsets. Additionally, incorporating dynamic analysis techniques, particularly behavior-based analysis for real-time data monitoring, will be a crucial area of investigation to enhance the detection and classification of evolving Android malware threats.

References

[1] P.R.K.Varma, K.V.S.Raju and K.P.Raj, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," in International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC), SCAD Institute of Technology, Palladam,India, 2017.

[2] N.Anusha and B.Rajalakshmi, "Sensor based application for malware detection in android OS (Operating System) devices," in 2017 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2017.

[3] https://www.techrepublic.com/article/new-android-malware-found-every-10-seconds-report-says/. [Online].

[4] S.H.Moghaddam and M.Abbaspour, "Sensitivity analysis of static features for Android malware detection," in 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2014.

[5] W.Qing-Fei and F.Xiang, "Android Malware Detection Based on Machine Learning," in 2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, 2018.

[6] Taheri, Laya, Kadir, Abdul, Lashkari and A.Habibi, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," in 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019.

[7] P.Xiong, X.Wang, W.Niu, T.Zhu and G.Li, "Android malware detection with contrasting permission patterns," China Communications, vol. 11, no. 8, pp. 1-14, Aug 2014.

[8] F.M.Darus, N.A.A.Salleh and A.F.Ariffin, "Android Malware Detection Using Machine Learning on Image Patterns," in 2018 Cyber Resilience Conference (CRC), Putrajaya, Malaysia, 2018.

[9] S.Arshad, M.A.Shah, A.Wahid, A.Mehmood, H. Song and H.Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," vol. 6, pp. 4321-4339, 2018.

[10] C L P M Hein and K M Myo, "Permission-based Feature Selection for Android Malware Detection and Analysis," International Journal of Computer Applications, vol. 181 , no. 19, pp. 29-39, 2018.

[11] B Sanz et al., "PUMA: Permission Usage to Detect Malware in Android," in International Joint Conference CISIS'12-ICEUTE´12-SOCO´12, Ostrava, Czech Republic., 2013, pp. 289-298.

[12] Z Aung and W Zaw, "Permission-Based Android Malware Detection," International Journal Of Scientific & Technology Research, vol. 2, no. 3, pp. 228-234, 2013.

[13] A K Singh, C D Jaidhar, and M A A Kumara, "Experimental analysis of Android malware detection based on combinations of permissions and API-calls," Journal of Computer Virology and Hacking Techniques, vol. 15, no. 3, pp. 209-218, 2019.

[14] S S Y Yerima, S Sezer, G McWilliams, and I Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," in IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 2013, pp. 121-128.

[15] L Zhang, V L Thing, and Y Cheng, "A Scalable and extensible framework for android malware detection and family attribution," Computer & Security, vol. 80, pp. 120-133, 2019.

[16] O Yildiz and I A Doğru, "Permission-based Android Malware Detection System Using Feature Selection with Genetic Algorithm," International Journal of Software Engineering and Knowledge Engineering, vol. 29, no. 2, pp. 245-262, 2019.

[17] K.Allix, Jerome, T.F.Bissyandé, Klein, State and Y.L.Traon, "A Forensic Analysis of Android Malware -- How is Malware Written and How it Could Be Detected?," in International computer software and applications conference, Vasteras, Sweden, 2014.

[18] M.Zaman, T.Siddiqui, M.R.Amin and M.S.Hossain, "Malware detection in Android by network traffic analysis," in 2015 International Conference on Networking Systems and Security (NSysS), Dhaka, Bangladesh, 2015.

[19] Z.D.Patel, "Malware Detection in Android Operating System," in 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018.

[20] Prasetiyowati, M.I., Maulidevi, N.U. & Surendro, K. Determining threshold value on information gain feature selection to increase speed and prediction accuracy of random forest. J Big Data 8, 84 (2021). https://doi.org/10.1186/s40537-021-00472-4

[21] Zhai, Y.; Song, W.; Liu, X.; Liu, L.; Zhao, X. A chi-square statistics based feature selection method in text classification. InProceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China,23–25 November 2018; pp. 160–163

[22] Q. Q. Gu, Z. H. Li, J. W. Han. Generalized fisher score for feature selection. https://arxiv.org/abs/1202.3725, 2012.

[23] Rehman, Khan, Muhammad, Lee, Lv, Baik, P.A.Shah, K.Awan and Mehmood, "Machine learning-assisted signature and heuristic-based detection of malwares in Android devices," Computers and Electrical Engineering, vol. 69, pp. 821-841, July 2018.
[24] M.Kedziora, P.Gawin, I.Jozwiak and M.Szczepanik, "Android Malware Detection Using Machine Learning and Reverse Engineering," in CS & IT Conference Proceedings, Wroclaw University of Science and Technology, Poland, 2018.
[25] B.Wu, W.Wen and J.Li, "Android Malware Detection Method Based on frequent Pattern and Weighted Naive Bayes," in Cyber Security, Beijing, China, Springer, 2019, pp. 36-51.
[26] V. Grampurohit, "Android App Malware Detection," IIIT, Hyderabad, India, 2016.
[27] P.Girdhar and D.Virmani, "An Analysis of Feature Selection Method in Mobile Malware Detection," International Journal of Engineering Applied Sciences and Technology, vol. 3, no. 3, pp. 56-61, 2018.