J. Integr. Sci. Technol. 2024, 12(1), 702

Article

Journal of Integrated
SCIENCE& TECHNOLOGY

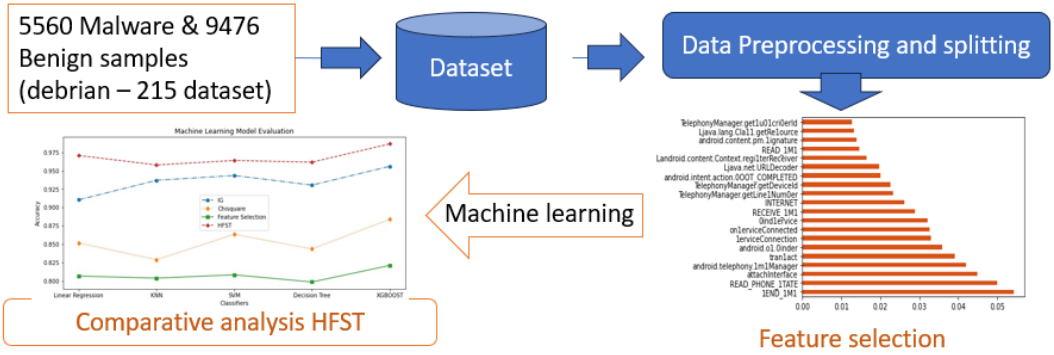
Hybrid Feature Selection Technique to classify Malicious Applications using Machine Learning approach

Umesh V. Nikam,* Vaishali M. Deshmukh

Computer Science and Engineering Department, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, Maharashtra 444701. India

Received on: 26-Apr-2023, Accepted and Published on: 03-Aug-2023

ABSTRACT



The malware identification is crucial factor in secure uses of mobile phones. A method for malware classification utilizing hybrid features is described in this study. This method selects features using: information gain, chi-square, and feature importance techniques. This approach aims to check how feature selection affects the performance of a classifier. First, top 20 significant features are chosen using each technique, and performance of classifiers is assessed. At the end, all 60 features chosen using three techniques are combined, and features that are common to at least two techniques are selected as hybrid features. 11 out of 60 features were discovered to be hybrid features. Using these 11 hybrid features, the performance of classifiers like linear regression, SVM, KNN, decision trees, and XGBoost is re-examined. The outcome reveals that hybrid features improve the performance of all classifiers when compared to performance using individual features. Out of the five classifiers examined, XGBoost has achieved the highest accuracy (0.9866), precision (0.9900), recall (0.9833), and f measure (0.9866). Hybrid feature selection technique ultimately proved to be successful because the classifiers could distinguish between malicious and benign apps with a higher level of classification accuracy than they could with an individual feature selection strategy.

Keywords: Hybrid Features, Malware, Benign, Machine Learning, Performance Evaluation.

INTRODUCTION

Nowadays, smartphones are the most widely available and handy platform for running any kind of application. The number of applications downloaded per day per user is increasing very rapidly. Due to this rapid expansion in the smartphone industry, many consumers now use smart phones to access the internet and a variety of services. By enabling constant contact everywhere and offering a variety of functionalities, Android apps significantly improve the quality of our lives. The growth of Android applications is essential for the development of the mobile Internet and the emerging economy.

Smartphones maintain every activity performed by the user, and by that, they store images, messages, and private or personal information about it. Due to this reason, smart phones are an easy target for attackers.[1] Almost all smart phones use Android as their operating system. Android allows downloading and installing any application from open source as well as from any third-party market. Because of this choice, it is simple for attackers to distribute apps and viruses, which they then employ to fool people by launching malicious code. Attackers use Android applications to spread malware and steal the users' private information. As per the McAfee report, 31 million Android malwares were found in 2018, and studies have shown that 1.9 million new malware samples are

*Umesh V. Nikam, Prof. Ram Meghe Institute of Technology & Research, Badnera, Amravati, Maharashtra 444701. India
Email: umeshnikam3@gmail.com

Cite as: *J. Integr. Sci. Technol., 2024, 12(1), 702.*
URN:NBN:sciencein.jist.2024.v12.702

 ISSN: 2321-4635
http://pubs.thesciencein.org/jist

added per year.[2] Thus, detecting a large amount of malware has become a challenging task. Both static and dynamic malware detection techniques are available but have their own limitations. A signature-based malware detection technique cannot detect unknown malware, and a dynamic malware detection technique has time and resource constraints.[3] Every application requires certain permissions at the time of installation. However, the permissions required by malicious and benign apps are different. If we applied some technique and carefully analysed these permissions, it could be used to distinguish malicious from benign applications.[4] Machine learning offers various methods for picking crucial features from an app's whole feature set.

This approach uses three techniques for the selection of features: information gain, chi-square, and the feature importance technique. This approach investigates the impact of feature selection on a classifier's performance. Initially, the top 20 important features out of 215 features with the highest gain value are selected using information gain, chi-square, and feature selection techniques, and then 11 features out of 60 that are common to at least two techniques are finalised. Using these 11 hybrid features, the performance of machine learning classifiers like linear regression, KNN, SVM decision trees, and XGBoost is examined again, and the outcome demonstrates that, when compared to each technique's individual features, hybrid features improve the performance of all five machine learning classifiers.

This study describes a hybrid feature selection technique named as HFST for malware classification utilising and combining common features generated by information gain, chi-square, and feature importance techniques. This approach aims to check how feature selection affects the performance of a classifier. First, performance of classifiers is assessed with top 20 significant features of information gain, chi-square, and feature importance techniques. After that, all 60 features from these three techniques are combined, and features that are common to at least two techniques are selected as hybrid features. 11 out of 60 features were discovered to be hybrid features. Using these 11 hybrid features, the performance of classifiers like linear regression, SVM, KNN, decision trees, and XGBoost is re-assessed. The outcome reveals that hybrid features improve the performance of all classifiers when compared to performance using individual features.The Hybrid Feature Selection Technique (HFST) is an innovative approach that brings together the top features extracted from three distinct feature selection techniques: information gain, chi-square, and feature importance. By combining the strengths of these techniques, HFST aims to achieve a more comprehensive and robust feature subset, which can lead to superior results in machine learning models. Indeed, experiments using HFST have demonstrated significant improvements in the performance of classifiers, exhibiting a remarkable 20% enhancement in accuracy, precision, F-measure, and recall when compared to models built using individual features. The ability of HFST to improve accuracy and other classification parameters showcases its potential as a novel tool in various machine learning applications, enabling practitioners to leverage the power of diverse feature selection methods for superior predictive modeling.

Research Contribution:

The following are the main contributions from this work:

1. The best sets of features are selected using three feature selection techniques.
2. New category of features as hybrid features is identified and used to investigates the impact on a classifier's performance.
3. Performance of five machine learning classifiers belonging to various categories is examined on both normal as well as hybrid features.
4. It has been found that a performance of a classifiers improves with hybrid features.

RELATED WORK

For this survey work, the author has given a comprehensive review of the evolution and current trends in analyzing malware and its detection methodologies.[5] This study is particularly interested in the perspectives that prior surveys frequently overlooked or just partially examined. Also, in order to emphasise the distinction between the phases of data collection and data extraction, instead of using analysis procedures, this study connected feature extraction techniques with real extraction processes. This survey has introduced a fresh taxonomy for feature representation techniques. In order to identify the unresolved problems and make recommendations for future research directions, a root causes of the shortcomings that each strategy or method suffers from have been explored.

By extracting some static and dynamic data, the authors Dhalaria et. al.[6] have suggested a hybrid technique for malware detection. They selected features using the information gain technique & ran multiple algorithms of machine learning on the dataset. In the experiment, they discovered that using only static or dynamic features reduces a classifier's performance, whereas using a mixed method improves malware detection accuracy.

Model by Abdul Basit A. Darem et al.[7] combining sequential deep learning and idea drift detection. By running malware in a sandpit, they were able to gather dynamic features for their strategy. Malware samples from the past are used to train a base classifier. Then, the old and new malware samples are combined, and the learning model is fed in an incremental batch-size way. A thorough analysis reveals that the suggested model outperforms the static model in terms of a detection rate and an efficiency.

In a report, Gao et.al.[8] proposed a GGN-based approach for detecting and classifying malware. They developed a heterogeneous graph using Android applications and APIs. The GDroid prototype system developed by them is found to be effective as compared to an existing system. They have also worked on the API usage pattern for malware classification, and their results utilising graph neural networks have been encouraging.

In their study, Abijah Roseline et al.[9] suggested a broad deep forest model for classifying and detecting malware. In their strategy, they have attempted to enhance a model's performance by focusing on three areas. They started by turning PE binary files into 2D grayscale graphics. Then they processed photos in a second phase using sliding window scanning and cascade layering. To decide the layering procedure, they employed cross-validation in

the third step. Results obtained demonstrate the effectiveness of the aforementioned technique in identifying and classifying malware.

On a benchmark dataset, several machine learning techniques have been employed by Shhadat et.al[10] In their experiment, the decision tree algorithm for a binary classification and a random forest algorithm for multiclass classification have shown the highest accuracy. However, nave bees have shown a significant improvement in accuracy with the change in feature set.

Rkhouya et. al.[11] used a PE header technique for malware classification. In this, they trained four machine learning algorithms using more than 130000 files. The performance of an algorithm is evaluated using accuracy, AUC, and execution time. The random forest algorithm is shown to have a best performance.

The methodology put forward in a work by Rey et.al.[12] employs FL for supervised and unsupervised model training and testing without revealing any personal data. In order to offload the processing from an IoT device, this project is intended to be implemented over a network nodes enabling access to a IoT-connected devices for WiFi, B5G, or 5G networks. With the use of N-BaIoT, they evaluated the performance of three different setups: the use of a federated approach, in which every device owner creates their own model and occasionally aggregates it on a server; (ii) a non-privacy-preserving configuration in which the server develops and centralises the full dataset; and (iii) a local configuration in which each device owner creates a solitary, isolated model. Above comparison has demonstrated that using more varied and substantial amount of data, as done by the federated & centralised techniques, significantly improves the model's performance in both supervised and unsupervised domains.

Maryam Al-Janabi et al.[13] conducted a survey in their paper to identify the best feature extraction and classification techniques that produce the most accurate malware detection. Numerous representational studies were examined and divided into three categories depending on the type of analysis they used: static, dynamic, or hybrid. They have presented a review of some classification techniques in which the J48 algorithm has outperformed others.

The study by Rabadi et.al.[14] examined a new approach to extracting API-based dynamic features by examining API calls and their lists of supporting arguments. The authors have created two ways for identifying Windows malware samples and group them according to their types. The first technique treats each API call's whole list of parameters as a single feature, whereas the second treats each argument individually as a single feature. They demonstrated that, in terms of accuracy, constraints, and necessary API knowledge, the proposed approach surpasses other current API arguments-based malware detection approaches.

Prerna Agrawal & Bhushan Trivedi[15] have suggested a method for detecting malware using machine learning. Generated dataset's performance has been evaluated by the authors using a supervised classifiers, features reduction approaches, and ensembling techniques. A number of evaluation metrics, such as accuracy, TPR, FPR, precision, , AUC and Cohen kappa score, were also used for evaluating a model's performance. Moreover, a bar graphs, ROC curve, & the Cohen Kappa Score were used to analyse the performance. With a ROC score of 0.91 indicating the model's overall performance in correctly predicting malware, a Cohen kappa score of 81.56 % indicating agreement amongst true classes, and an accuracy rating of 93.15% for predicting the overall count of correct predictions, a Cat Boost classifier performs best.

In the method by Goyal et.al.[16], more than one experiment was run using a built-in dataset for balanced and unbalanced data to compare the differences in complete accuracy and the effect of an imbalanced dataset. For balanced data, 1079 malware samples and the same number of benign samples were used, whereas 42797 malware and 1079 benign samples were used for unbalanced data. Then, different machine learning classifiers with training & testing ratios of 60:40, 70:30, and 80:20 are applied. Random forest has shown promising results in both studies. According to their observations, when a dataset is unbalanced, the resultant accuracy is excessive and appears to be manufactured. In contrast, when a data set is balanced, the accuracy is trustworthy. So, to obtain more reliable results, a balanced dataset should be used when performing machine learning.

The study by Soni et.al.[17] uses a random forest and tree-based classifier to identify fraudulent Android USI applications. This method increases the likelihood that rogue programmes will be detected. The software has the ability to recognise new malicious programmes. This technique has 93.6% accuracy in identifying malicious software. Only 22 of the 135 permissions were used for this work and has improved the runtime performance to 85.6 %. It can also be combined with dynamic scanning techniques to boost effectiveness even further.

The work by Agarkar et.al.[18] discusses behavior based detection approach and how various machine learning algorithms are applied to produce malware detections and categorization techniques based on behavior. In this approach, a machine learning algorithm is applied to static features. In situations with high load, it will be efficient to employ this strategy before signature based solutions to reduce the workload placed on the dynamic analysis of executables. Results obtained show that the Light Gradient Boosting algorithm is effective in both accuracy and model training time, and in comparison to Light GBM, Random Forest is significantly slower.

In order to reduce the need for specialised knowledge, memory dump files of malware are transformed into grayscale in the study by Shah et.al.[19] This work introduces two key methods: non-local denoising to remove noise and the discrete wavelet transform for reducing the dimension of an image. Finally, machine-learning-based classifiers are fed with these images. The best classifier is SVM with a RBF kernel, which also has the highest precision, accuracy, f1 score and recall. Focusing on malware classes with a detection rate < 90% will be necessary in the future.

METHODOLOGY

Below is depicted the suggested architecture for malware detection and classification. This methodology consists of a number of steps, which are explained below:

1) **Dataset Collection:** The first and most important step for malware classification is a dataset.[20] This methodology uses the debrian-215 dataset available on the Kaggle website. This dataset consists of 5560 malware and 9476 benign application samples.

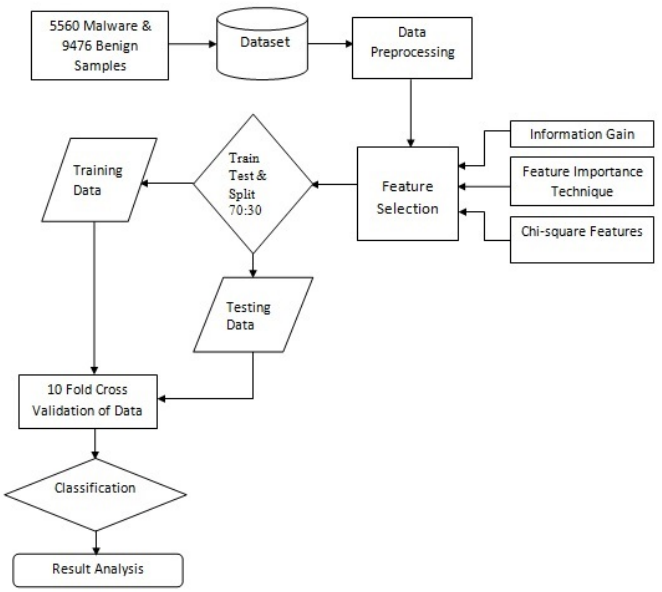


Figure 1. Methodology

2) **Data Preprocessing:** Once a dataset is selected, it needs to be cleaned before it is used for training a machine learning model. Because it may contain inappropriate data[21] like null values, categorical data, etc. Due to this inappropriate data, a model may not be trained properly, leading to inaccurate result generation. In this step, a dataset is preprocessed using the average value technique of machine learning from the Scikit-Learn library.[22]
3) **Feature Selection:** The choice of appropriate features is a crucial stage in virus detection. A model's accuracy can be low as a result of improperly chosen relevant characteristics, but it can also be high when those features are appropriately chosen.[23] As a result, this strategy employs three feature selection techniques: information gain, feature importance technique employing an additional tree classifier, and chi-square technique.[24] The best 20 attributes from each approach are first chosen. After combining all 60 features, the features that are common to at least two techniques are chosen as the final feature and are considered hybrid features. Following is a description of feature selection techniques:

A. **Information Gain:** The gain of each feature from a dataset is provided by IG. The most pertinent attribute is that which has the biggest gain value. The top 20 features in a debrian dataset with the highest information gain are chosen from the dataset's 215 features. In Figure 2, it can be seen that RECEIVE_lMl is a feature with an IG value of 0.077 and translate has the highest IG value of 0.195. Figure 2 shows a list of features chosen using the IG technique.

B. **Chi-Square Technique:** The second technique used is the chi-square technique. It looks for a stronger association in the dependent and independent features.[25] Using a chi-square feature named LND, which was found to be highly important, and another feature named Landroid.content.Context.regilter, Receiver has the lowest importance feature amongst 20 features. The link

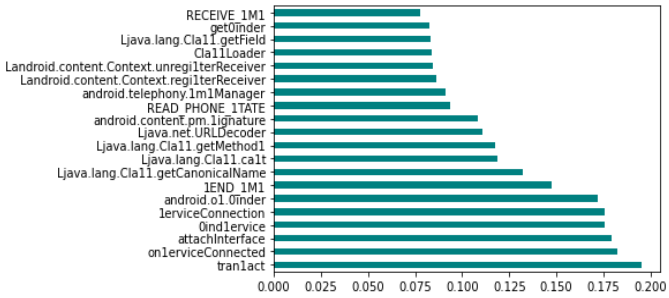


Figure 2. Top 20 features selected with Information Gain technique.

between dependent and independent properties is revealed using the Chi-Square approach. The stronger the association, the more significant the trait, the more valuable it is. By taking into account the top 20 values of the association, this method selects 20 features out of 215 features in a debrian dataset. The list of the top 20 features chosen using the chi-square method is displayed in figure 3 below.

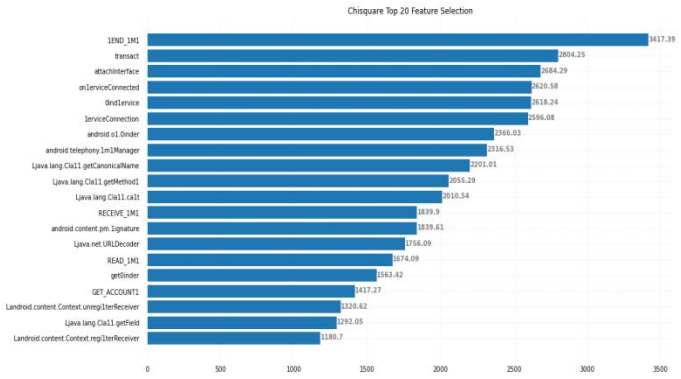


Figure 3. Top 20 features selected with Chi-square technique.

C. **Feature Importance Technique:** Using the feature importance technique, one may determine the weight of every feature in a dataset. Higher the weight more significant is a feature.[26] This method selects the top 20 features from a dataset by taking a feature's weight into account. Figure 4 below shows the features that have been chosen as the top 20 features.

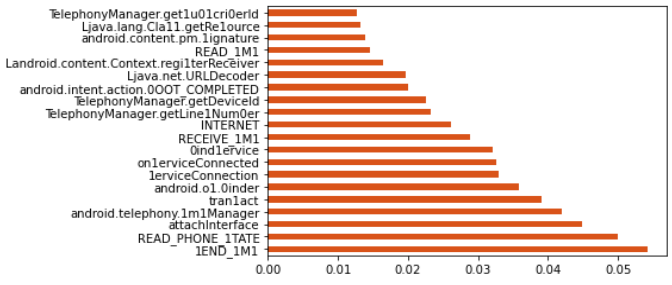


Figure 4. Top 20 features selected with Feature importance technique.

An information gain technique selects features using the I. G. value, the chi-square technique uses associations between dependent and independent features, and the feature importance technique works on the weight of a feature.[27] From figures 2, 3, and 4, it can be seen that each technique has identified different top 20 features. Combining all these features and selecting only those features that are common in at least two techniques can be the strongest set of features for malware identification. Considering this idea, all the 60 features from three techniques are combined together, and features that are common to at least two techniques are selected. They are called hybrid features. As hybrid features, 11 out of the 60 are found, and a list of such hybrid features is shown in Table 1 below:

Table 1. Features selected using hybrid feature selection technique. (hfst)

S.N	Name of Common Features
1	RECEIVE_1M1
2	Landroid.content.Context.regi1terReceiver
3	android.telephony.1m1Manager
4	android.content.pm.1ignature
5	Ljava.net.URLDecoder
6	1END_1M1
7	android.o1.0inder
8	1erviceConnection
9	0indlervice
10	OnlerviceConnected
11	tranlact

The results obtained reveal that the performance of classifiers is boosted with these hybrid features as compared to performance using individual features.

4) **Training & Testing Data:** Using data from a dataset, machine learning models are trained and tested [28]. Here, a dataset is split into 70:30 ratios. 70% of the data is used for the purpose of training a model, while the remaining 30% is used for the purpose of testing a trained model.
5) **Cross Validation:** Machine learning models' efficiency can be assessed using the cross validation technique [29]. This approach uses a 10-fold cross-validation technique. It divides the dataset into 10 subsets of equal size. Each time, nine subsets are used to train a model and one subset is used to test a machine learning model. Every time different subsets are used, which helps in enhancing the performance of a model with more accurate results.[30]
6) **Classification:** To classify the sample as benign or malicious, this approach uses machine learning classifiers like KNN, SVM, linear regression, decision trees, and XGBoost.[31] Below is some brief information about the machine learning algorithms used and their evaluation parameters.

A. Machine Learning algorithms

- Linear Regression: Linear regression, a straightforward and unassuming statistical technique, is employed in predictive analysis to highlight the connection between continuous variables.[32] A statistical technique that shows a correlation between input and output variables is known as linear regression. The objective of Linear Regression is to discover a linear association between the independent (x) and dependent (y) variable. The relationship is represented as a straight line equation:

$$y = \beta_0 + \beta_1 * x \qquad \text{Eq. 1}$$

- Where: y is a predicted value for a dependent variable. x is the input value of the independent variable. B_0 is the y-intercept (the value of y when x = 0). B_1 is the slope of the line, representing the change in y for a unit change in x. The coefficients (β_0 and β_1) are estimated during the training process for minimizing the error between the predicted & actual target values in a training data This is done using some methods such as Ordinary Least Squares (OLS) or Gradient Descent. For a multiple linear regression an equation extends to include additional coefficients (β_2 to β_i) representing different independent variables. The algorithm finds the optimal coefficients to create the best-fitting linear model for the given data.

- KNN: KNN is straightforward supervised machine learning algorithm applicable for the classification, regression, and missing value imputation tasks. KNN operates based on the principle of selecting the closest observations in a dataset for predicting the class or value of a new data points. The user can control a number of nearest neighbors considered (K value) to influence the algorithm's performance. Larger K values tend to be more robust and generate stable decision boundaries compared to very small K values, which may produce less desirable results. [33] K-Nearest Neighbors (KNN) algorithm predicts a class label for a new data point by selecting a K nearest neighbor based on distance and assigning the majority class among them as the prediction. Mathematically,

$$\hat{y} = \text{mode}(\{y_i\}) \qquad \text{Eq. 2}$$

for i in K-nearest-neighbors of x.

- SVM: SVM is a well-known supervised learning algorithm used for classification and regression purposes, with a primary focus on classification in machine learning. SVM aims to construct an ideal decision boundary, termed a hyperplane, which effectively separates classes in a multi-dimensional space. The algorithm identifies the extreme points, termed support vectors, which contribute to constructing the hyperplane, leading to its name "Support Vector Machine".[34] It is a binary classification algorithm which aims to find the optimal hyperplane ($w \cdot x + b = 0$) that separates two classes in the feature space. It maximizes a margin between two classes minimizing the classifications error. SVM identifies support vectors, a closest data points in the hyperplane, to achieve this objective. A decision function $f(x) = \text{sign}(w \cdot x + b)$ predicts a class labels for new data points.

- Decision Tree: The decision tree method looks for a solution by modeling the issue as a tree [35]. While each inner node in the tree is related to a feature, every leaf

node is associated with a label of class. The mathematical representation of the decision tree algorithm is as follows: Given a training dataset with labeled samples: Training data : (x 1, y 1), (x 2, y 2), …., (x_i, y_i), …., (xn, yn) where, x_i is a input feature vector & y_i is a corresponding class label. Decision Tree algorithm recursively builds a tree-like model to make predictions for new data points. At each node of the tree, a feature and a corresponding threshold value are selected for splitting the data into subsets. Up until a stopping requirement is satisfied, as like reaching a max depth or a minimum amount of samples per leaf, the algorithm divides the subgroups into more child nodes.

- XGBoost: A XGBoost method sequentially builds decision trees. Weights are considered as crucial parameter while using XGBoost. Prior to entering the decision tree that predicts the results, each independent variable is given a weight. [36] Mathematically, the prediction of the XGBoost model (F(x)) is given by: $F(x) = \Sigma_i f_i(x)$ Eq. 3

- Where, F(x) is a final prediction of the XGBoost model for a input feature vector x and $f_i(x)$ is a prediction of the i-th weak learner (tree) for x.

- The XGBoost algorithm also incorporates regularization terms to prevent overfitting and control the complexity of the model. In summary, XGBoost combines weak learners (trees) iteratively and optimizes the model to minimize the loss function, producing a robust and accurate predictive model.

B. Evaluation Parameters: The following factors are used for evaluation of the performance of various machine learning algorithms:

- Accuracy: It's a ratio of a correct number of predictions divided by a total count of predictions.[37] ACC = TP + TN / TP + TN + FP + FN Eq. 4

- Precision: It counts number of positive samples that actually belong to the positive class.[38] Precision = TP/TP + FP Eq. 6

- Recall: Recall counts the number of successful class predictions which were made using all of the dataset's successful cases.[39] Recall = TP/TP + FN Eq. 7

- F-Measure: The value of F-measure balances precision & recall in a single integer.[40] F Measure = (2*Precision * Recall)/(Precision + Recall) Eq.8

ALGORITHM: HYBRID FEATURE SELECTION TECHNIQUE (HFST) FOR MALWARE DETECTION

Input: Dataset of Malware and benign Applications
Output: Performance of machine learning algorithms using HFST.

1. Step 1: Preprocess a dataset.
2. Find and fill missing values.
3. Find and remove noisy data using clustering
4. Find and remove the outliers.
5. Step 2: Find top 20 features each by using Information Gain, Chi square & Feature Importance technique.
6. Step 3: Select top features common in at least 2 techniques.(Hybrid Features)
7. $(F_{(HFST)}=F_{(IG)} \cap F_{(FIT)} \cap F_{(CT)})$
8. Step 4: Split dataset into 70:30(Training & Testing part)
9. Step 5: Apply 10 fold cross validation to data.
10. Step 6: Measure performance of machine learning algorithms for I.G, Chi-square, Feature Selection and HFST.
11. Step 7: Compare performance of machine learning algorithms with and without HFST.
12. Step 8: Find best performing technique.
13. End

RESULTS

An experiment is carried out to determine which of the features, the hybrid features shown in Table 1 or the top 20 features shown in figure 2 to 4 are more effective to improve the performance of a machine learning classifier. As shown in Figure. 1 Methodology, It uses a ten fold cross validation technique, which has been shown to be statistically effective in classifier performance evaluation. 70% of a total dataset was used for a training of model, while 30% was used for testing purpose. Accuracy, recall, precision, and f measure are used to assess the performance of the classifiers'. Comparative performance of various classifiers using information gain, chi-square, and feature importance techniques is shown in Table II, and performance using hybrid features is shown in Table III. It can be seen from a Table II that the performance of all the classifiers is higher using the information gain technique. Of all the feature selection techniques, the XG Boost classifier has demonstrated the highest accuracy, precision, recall and F-measure.

After evaluating the performance of classifiers with the top 20 features, how hybrid features affect the performance has to be checked. An experiment is conducted, and the performance of the same classifiers is evaluated this time using hybrid features shown in Table I. Above Table II, shows the result obtained. Comparing the result obtained in Table II with Table III, it can easily be seen that there's a significant improvement in the performance of all the classifiers with hybrid features.

Table IV, below, shows the comparative performance of classifiers using top 20 individual features and hybrid features. Following are the observations shown in columns 2 and 3 of Table IV below-

- When the classifiers' performance was assessed, it was found that using the top 20 features, the decision tree algorithm had the lowest accuracy of 0.7986, but its accuracy was improved to 0.9617 using the hybrid feature.
- Related to the precision, using the top 20 features of logistic regression has shown a precision of 0.8049; on the

Table 2. Performance of Algorithms with top 20 feature using ig, chi-square & feature importance technique.

	Performance Using IG				Performance Using Chi-Square				Performance Using Feature Importance			
Name of Algorithm	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Logistic Regression	0.9104 79452	0.9135 441	0.91354 406	0.91354 406	0.85177 918	0.8576 74	0.85767 3952	0.85767 3952	0.80673 3697	0.8049 649	0.80496 4897	0.80596 4897
KNN	0.9370 22153	0.9655 936	0.90897 7352	0.93643 05	0.84893 0689	0.8541 93	0.85335 7107	0.85377 4867	0.80388 9693	0.8203 209	0.80090 0941	0.81049 463
SVM	0.9435 92728	0.9844 389	0.90391 8005	0.94246 1706	0.85358 1926	0.8658 388	0.84865 6271	0.85716 1429	0.80838 3895	0.8311 069	0.79658 1559	0.81347 8089
Decision Tree	0.9305 20045	0.9532 654	0.90837 5832	0.93027 9391	0.84355 716	0.8457 855	0.85273 5844	0.84924 6467	0.79862 1234	0.8122 995	0.80024 6618	0.80622 7997
XG BOOST	0.9560 76497	0.9889 84	0.92393 2977	0.95535 2397	0.86640 6505	0.8724 151	0.86757 1433	0.86998 6514	0.82118 5138	0.8384 44	0.81438 8772	0.82624 1342

Table 3. Performance of algorithms using 11 hybrid features.

	Performance Using 11 hybrid features			
Name of Algorithm	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Logistic Regression	0.97078	0.9724418	0.9724418	0.973441839
KNN	0.9679758	0.9687607	0.9676142	0.96818711
SVM	0.9740134	0.9859137	0.9622173	0.973921383
Decision Tree	0.9617636	0.9574964	0.9670081	0.962228776
XG BOOST	0.9866864	0.9900195	0.9833849	0.986691068

Table 4 Comparison of performance using top 20 features and Hybrid features

Evaluation Parameter	**Minimum Performance obtained Using top 20 features**	**Performance obtained Using Hybrid features**	**Findings**
Accuracy	0.7986 by Decision Tree Algorithm	0.9617 by Decision Tree Algorithm	Performance of classifiers is improved by 20% using HFST.
Precision	0.8049 by Logistic Regression Algorithm	0.9724 by Logistic Regression Algorithm	
Recall	0.7965 by SVM Algorithm	0.9622 by SVM Algorithm	
F-measure	0.8059 by Logistic Regression Algorithm	0.9734 by Logistic Regression Algorithm	

-

other hand, it is improved to 0.9724 with the use of hybrid features.

- For the recall, the SVM algorithm using the top 20 features has a recall value of 7965, which is increased to 0.9622 using hybrid features.
- F-measure for logistic regression with hybrid features is improved to 0.9734 from 0.8059 using the top 20 features.

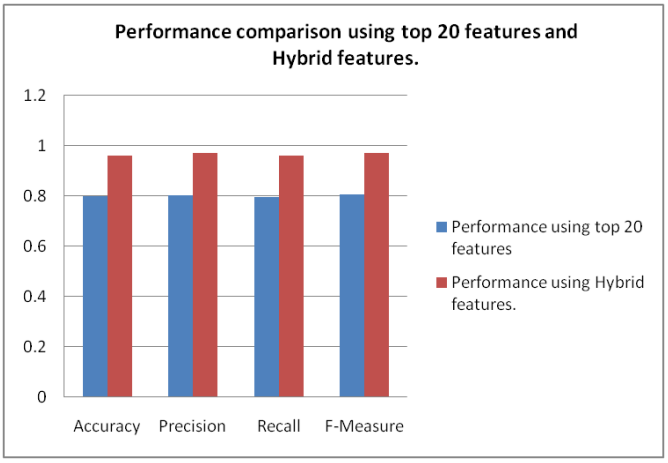


Figure 5. Performance comparison using top 20 features and Hybrid features.

DISCUSSION

1. Accuracy:

Below, Figure 6 shows graph plots of the accuracy of each classifier using information gain, chi-square, feature importance, and hybrid feature techniques. The graph clearly shows that the hybrid feature selection technique outperforms all other techniques in terms of classifier performance for accuracy. Each classifier's

accuracy is calculated based on the top 20 features found for each technique. XG Boost has the highest accuracy, with a score of 98.66, and logistic regression has shown the second-highest accuracy. Similar to how well XG boost performed with the hybrid feature selection technique, it also performed better with the other three techniques.

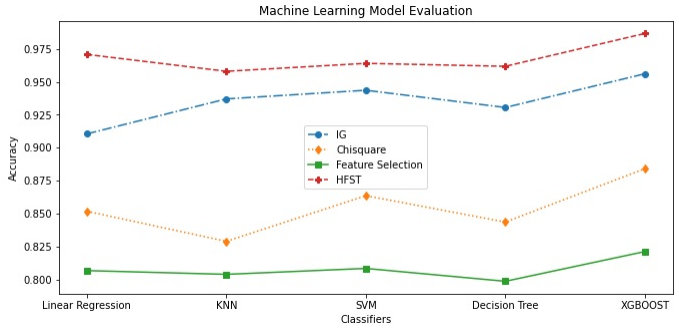


Figure 6. Performance comparison of classifiers in terms of accuracy.

2. Precision: Figure 7 below displays the precision performance of all classifiers. Each classifier has demonstrated superior performance for information gain and hybrid feature selection techniques, as seen in the graph. By using the information gain technique, logistic regression performs relatively poorly compared to when using the hybrid feature selection strategy, as can be seen in the graph in Figure 7. Furthermore, when compared to all other performances of classifiers, every classifier has demonstrated improved performance when adopting the hybrid feature selection technique, with SVM and XG Boost in particular outperforming other classifiers.

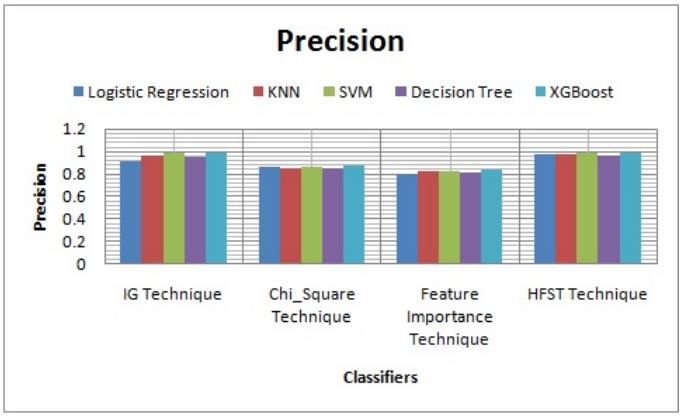


Figure 7. Performance comparison of classifiers in terms of precision.

3. Recall: Across all the positive samples in the data, recall tallies of how many positive predictions the classifier made. Figure 8 below shows how well classifiers performed in terms of recall. As can be seen from the diagram, the performance of all classifiers is decreased when using the feature importance technique; however, it is improved when using the hybrid feature selection technique. When applying HFST, all classifiers have demonstrated improved performance; XG Boost's performance stands out with the highest value, 0.98, as shown in table III.

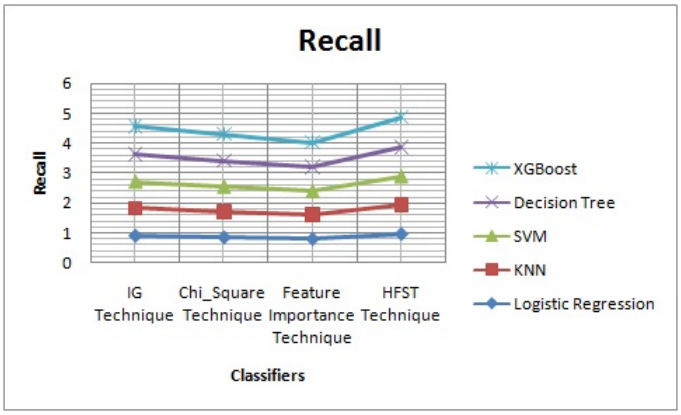


Figure 8. Performance comparison of classifiers in terms of recall.

4. F Measure: As shown in figure 9. Straightaway, all the classifiers have shown higher performance for the hybrid feature selection technique as compared to all other techniques. Performance using the information gain technique is the second largest, whereas it is the lowest in the case of the feature importance technique.

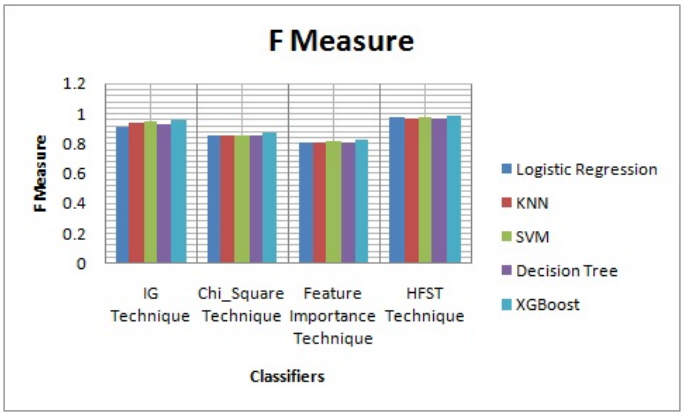


Figure 9. Performance comparison of classifiers in terms of F Measure.

Considering the performance of classifiers in terms of accuracy, precision, f measure & recall as shown in figures 6, 7, 8 and 9. After the evaluation and as per the results shown in tables II, III, and IV, the hybrid feature selection method is discovered to be the most successful. Additionally, the impact on classification accuracy, recall, precision, and f-measure were evaluated, and with the use of the hybrid feature selection technique, a significant improvement in classification accuracy and other parameters was made. The classification procedure benefits from the hybrid feature selection technique.

CONCLUSION AND FUTURE SCOPE

Android users are constantly at risk from mobile viruses. It is critical to make sure these devices are safe and secure as they become more crucial to our daily lives. Therefore, it is essential to give top priority to the creation and testing of fresh, potent, and effective malware detection approaches. In this approach, the

hybrid features derived from three feature selection strategies were investigated and compared with the characteristics of the individual features. The hybrid feature selection technique was found to be most effective. Additionally, the impact on classification accuracy, precision, recall, and f-measure were evaluated, and with the use of this technique, a significant improvement in classification accuracy and other parameters was made. The classification procedure benefits from the hybrid feature selection technique. In the end, utilising a hybrid feature selection strategy, the XGBoost classifier was able to discriminate between malicious and benign apps with a classification accuracy of 98.66%, a precision of 99.00, a recall of 98.33, and an f-measure of 98.66.

The future scope for this work involves exploring deep learning architectures, adaptive learning algorithms, and multi-modal analysis for improved malware detection. Additionally, investigating federated learning, explainability, and large-scale deployment challenges can enhance the system's effectiveness and user privacy. The research can also focus on detecting zero-day malware, real-time monitoring, and conducting Android app market analysis to stay ahead of evolving threats and provide proactive security solutions.

ACKNOWLEDGMENTS

Authors thanks department of Comp. Science & Engineering, P.R.M.I.T & R, Badnera for providing necessary facilities for study.

REFERENCES

1. D.H. Gillani. A perspective study on Malware detection and protection, A review. *Authorea Preprints*, **2022**.
2. S. Priyadharshini, S. Shanthi. A Survey on Detecting Android Malware Using Machine Learning Technique. *2021 7th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2021* **2021**, 1621–1627.
3. A. Altaher. Classification of Android Malware Applications using Feature Selection and Classification Algorithms. *VAWKUM Trans. Comput. Sci.* **2016**, 10 (1), 1.
4. K.P. Raj, K.V.S. Raju. Using Machine Learning Algorithms To. *Int. Conf. I-SMAC (IoT Soc. Mobile, Anal. Cloud) (I-SMAC* **2017**, 1 (1), 2007.
5. F.A. Aboaoja, A. Zainal, F.A. Ghaleb, et al. Malware Detection Issues, Challenges, and Future Directions: A Survey. *Appl. Sci.* **2022**, 12 (17).
6. M. Dhalaria, E. Gandotra. A hybrid approach for android malware detection and family classification. *Int. J. Interact. Multimed. Artif. Intell.* **2021**, 6 (6), 174–188.
7. A.A. Darem, F.A. Ghaleb, A.A. Al-Hashmi, et al. An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning. *IEEE Access* **2021**, 9, 97180–97196.
8. H. Gao, S. Cheng, W. Zhang. GDroid: Android malware detection and classification with graph convolutional network. *Comput. Secur.* **2021**, 106.
9. S.A. Roseline, S. Geetha, S. Kadry, Y. Nam. Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm. *IEEE Access* **2020**, 8, 206303–206324.
10. I. Shhadat, B. Bataineh, A. Hayajneh, Z.A. Al-Sharif. The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware. In *Procedia Computer Science*; Elsevier B.V., **2020**; Vol. 170, pp 917–922.
11. S. Rkhouya, K. Chougdali. Malware Detection Using a Machine-Learning Based Approach. *Int. J. Inf. Technol. Appl. Sci.* **2021**, 3 (4), 167–171.
12. V. Rey, P.M. Sánchez Sánchez, A. Huertas Celdrán, G. Bovet. Federated learning for malware detection in IoT devices. *Comput. Networks* **2022**, 204 (April 2021), 108693.
13. M. Al-Janabi, A.M. Altamimi. A comparative analysis of machine learning techniques for classification and detection of malware. *Proc. - 2020 21st Int. Arab Conf. Inf. Technol. ACIT 2020* **2020**.
14. D. Rabadi, S.G. Teo. Advanced Windows Methods on Malware Detection and Classification. *ACM Int. Conf. Proceeding Ser.* **2020**, 54–68.
15. P. Agrawal, B. Trivedi. Evaluating Machine Learning Classifiers to detect Android Malware. *2020 IEEE Int. Conf. Innov. Technol. INOCON 2020* **2020**, 1–6.
16. M. Goyal, R. Kumar. Machine Learning for Malware Detection on Balanced and Imbalanced Datasets. *2020 Int. Conf. Decis. Aid Sci. Appl. DASA 2020* **2020**, 867–871.
17. H. Soni, P. Arora, D. Rajeswari. Malicious Application Detection in Android using Machine Learning. *Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020* **2020**, 846–848.
18. S. Agarkar, S. Ghosh. Malware detection & classification using machine learning. *Proc. - 2020 IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. iSSSC 2020* **2020**.
19. S.S.H. Shah, N. Jamil, A. ur R. Khan. Memory Visualization-Based Malware Detection Technique. *Sensors* **2022**, 22 (19).
20. T.C. Miranda, P.F. Gimenez, J.F. Lalande, V.V.T. Tong, P. Wilke. Debiasing Android Malware Datasets: How Can I Trust Your Results If Your Dataset Is Biased? *IEEE Trans. Inf. Forensics Secur.* **2022**, 17, 2182–2197.
21. P. Agrawal, B. Trivedi. Unstructured Data Collection from APK files for Malware Detection. *Int. J. Comput. Appl.* **2020**, 176 (28), 42–45.
22. J. Zhu, X. Wang, L. Hu, et al. abess: A Fast Best Subset Selection Library in Python and R. **2021**, 23, 1–7.
23. X. Zhang, J. Wang, J. Xu, C. Gu. Detection of Android Malware Based on Deep Forest and Feature Enhancement. *IEEE Access* **2023**, PP, 1.
24. E. Odat, Q.M. Yaseen. A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features. *IEEE Access* **2023**, 11 (December 2022), 15471–15484.
25. X. Xing, X. Jin, H. Elahi, H. Jiang, G. Wang. A Malware Detection Approach Using Autoencoder in Deep Learning. *IEEE Access* **2022**, 10, 25696–25706.
26. N.O.R.Z. Gorment, A.L.I. Selamat, L.I.M. Kok. Machine Learning Algorithm for Malware Detection : Taxonomy , Current Challenges and Future Directions. *IEEE Access* **2023**, PP, 1.
27. M.S. Akhtar, T. Feng. Malware Analysis and Detection Using Machine Learning Algorithms. *Symmetry (Basel).* **2022**, 14 (11), 2304.
28. J. Kim, Y. Ban, E. Ko, H. Cho, J.H. Yi. MAPAS: a practical deep learning-based android malware detection system. *Int. J. Inf. Secur.* **2022**, 21 (4), 725–738.
29. X.L. Zhang, M. Xu. AUC optimization for deep learning-based voice activity detection. *Eurasip J. Audio, Speech, Music Process.* **2022**, 2022 (1).
30. M. Mimura, R. Ito. Applying NLP techniques to malware detection in a practical environment. *Int. J. Inf. Secur.* **2022**, 21 (2), 279–291.
31. S. Kurnaz, M.E. Khudhur. Comparative and Analysis Study for Malicious Executable by Using Various Classification Algorithms. **2018**, 08 (7), 18–26.
32. U.-H. Tayyab, F.B. Khan, M.H. Durad, A. Khan, Y.S. Lee. A Survey of the

Recent Trends in Deep Learning Based Malware Detection. *J. Cybersecurity Priv.* **2022**, 2 (4), 800–829.

33. E.J. Alqahtani, R. Zagrouba, A. Almuhaideb. A survey on android malware detection techniques using machine learning Algorithms. *2019 6th Int. Conf. Softw. Defin. Syst. SDS 2019* **2019**, 110–117.
34. R. Chaganti, V. Ravi, T.D. Pham. Deep learning based cross architecture internet of things malware detection and classification. *Comput. Secur.* **2022**, 120 (May).
35. S.R.T. Mat, M.F.A. Razak, M.N.M. Kahar, J.M. Arif, A. Firdaus. A Bayesian probability model for Android malware detection. *ICT Express* **2022**, 8 (3), 424–431.
36. R. Ali, A. Ali, F. Iqbal, M. Hussain, F. Ullah. Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review. *Secur. Commun. Networks* **2022**, 2022.
37. H. Manthena, J. Kimmell, M. Abdelsalam, M. Gupta. Analyzing and Explaining Black-Box Models for Online Malware Detection. *IEEE Access* **2023**, No. February, 25237–25252.
38. S. Yan, J. Ren, W. Wang, et al. A Survey of Adversarial Attack and Defense Methods for Malware Classification in Cyber Security. *IEEE Commun. Surv. Tutorials* **2022**, 25 (1), 467–496.
39. M. Asam, S.H. Khan, A. Akbar, et al. IoT malware detection architecture using a novel channel boosted and squeezed CNN. *Sci. Rep.* **2022**, 12 (1), 1–12.
40. H. Lee, S. Kim, D. Baek, D. Kim, D. Hwang. Robust IoT Malware Detection and Classification using Opcode Category Features on Machine Learning. *IEEE Access* **2023**, 11 (February), 18855–18867.