



ENSEMBLE APPROACH WITH LASSO REGULARIZATION ALGORITHM FOR IMPROVED DETECTION OF ANDROID MALWARE

*Anyaora, P.C¹, Adebayo, O.S², Ojeniyi, J.A³, Adeleke N.D⁴, Gungbias, H.I⁵, Nwodo R.C⁶, Okechukwu, C⁷

¹Cyber Security Science Department, Federal University of Technology, PMB 65 Minna
Niger State, Nigeria

*Corresponding author email: P.anyaora@futminna.edu.ng +2348139319538

Abstract

Android malware is a trending topic in computing and cyber security science, due to over 60% of people usage of android phones. The volume and sophistication of android malware have increased due to this enormous use and creation of different applications which pose serious risks to the security of mobile devices and the services they support. Therefore, there is a growing interest in using machine learning to enhance android malware detection for industry adoption to ensure application security on android phones. This research proposes an efficient ensemble learning approach of the Multilayer Perceptron Neural Network-Random Forest-Light Gradient Boost android malware detection model with Lasso regularisation algorithm for feature selection. Experiment was conducted on two based datasets, CicandMalInvest2019 of a static analysis and TUANDROMD of a dynamic analysis. The results show the data pre-processed with hyperparameter tuning, got an accuracy of 99.601% and 96.92%, false positive rate of 0.0026 and 0.01124, F1-score of 0.91764 and 0.99441, on the static and dynamic analysis datasets respectively. The proposed model outperformed the benchmarked machine learning based models.

Keywords: Ensemble learning, Android, Malware, Static analysis, Dynamic analysis

1. Introduction

In terms of market share, android smartphones currently make up more than 80% of all smart phones, and by 2020, analysts estimated that they will reach 85%. A record-breaking increase in the number of new android applications has been brought on by the increasing popularity of android smartphones, and these apps have also attracted the attention of hackers. By the end of 2018, there were 856.52 million different types of android malware in existence. There are around 137.5 million new malicious applications(apps) found in 2018 (or 350,000 new viruses every day) (Zhang *et al.*, 2021).

There are several operating systems available for mobile devices, for smartphones and tablets, the most popular operating system is Android (Linux based operating system), which is free and open-source. Google claims that 1.3 million android smartphones are activated every day (Arshad *et al.*, 2021). The growth of Android technology has attracted a surge of malware developers. Driven by financial gain, these malicious actors are constantly refining their techniques. These programs can directly compromise the security of the Android operating system, putting users' financial credentials and personal information at risk. Android device malware attacks are becoming increasingly sophisticated. As smartphone usage continues to rise, it's expected that malware threats will become even more



prevalent.

The use of android application features to combat android malware is a proactive approach to enhance security and protect users from potential threats (Guerra-manzanares et al., 2021). Android devices receive regular security updates and patches from manufacturers and Google. These updates address known vulnerabilities and help protect against malware that exploits them. Android runtime permission model prompts users to grant permissions when an app actually needs them, rather than during installation which reduces the chances of apps abusing permissions. To further reduce the chances of apps abusing permissions, several security measures are in place. The Google Play Store enforces strict guidelines and verification processes for app submissions. Apps that violate these policies are removed, minimising the risk of downloading malicious apps from the official store. Additionally, some Android phones attempt to verify app features, scan apps downloaded from outside the Play Store, and warn users about potential security risks (Guerra-manzanares et al., 2021). This discourages users from sideloading apps from untrusted sources. Using Android's security features, developers can perform security audits and code reviews to identify vulnerabilities and potential entry points for malware (Mahindru, 2020).

This research introduces two key contributions to the field of Android malware detection:

1. Optimization of Lasso Regularization algorithm for feature selection.
2. Development of an ensemble approach with Multilayer Perceptron Neural Network-Light Gradient Boost-Random Forest android malware detection model.

The structure of the remainder of this paper is as follows: Section 2 provides an overview of related work, Section 3 discusses the methodology, Section 4. result and analysis, Section 5. conclusion and acknowledgement.

2. Literature Review

Lasso Regularization Algorithm when used for selecting pertinent and optimal features from a set of input variables by promoting sparsity in the model coefficients. Before a model is being trained, Lasso Regularisation adds a penalty term to the loss function to encourage the model to minimise the total of the absolute values of the coefficients (Zhang et al., 2020). Lasso Regularisation can be used to select optimal features from a dataset, enhancing the performance of an ensemble learning classification model during training and testing. This technique also helps eliminate

redundant features (Zhang et al., 2020). Lasso adds a penalty term to the loss function, which is a combination of the error term and the sum of the absolute values of the coefficients multiplied by a regularisation parameter (λ). The regularisation parameter determines the strength of the penalty and controls the amount of sparsity in the model. Involving feature selection algorithm which aid in building the performance of the ensemble algorithms (Ansori et al., 2024) iterated that choosing the right feature selection model is one of the issues for producing a reliable detection model, Using the Canadian Institute for Cybersecurity Malware of Android 2020 (CICMalDroid2020) dataset for experimentation demonstrates that the suggested ensemble learning technique can enhance detection performance. Gain ratio feature selection increased the detection accuracy by 2.59% in Naïve Bayes, 0.90% in k -Nearest Neighbour, and 2.29% in Support Vector Machine, three machine learning classification techniques. With an accuracy of 94.57% and a precision score of 94.71%, the Random Forest, Extra Tree, and k -Nearest Neighbors ensembled machine learning models performed the best. (S. Sharma et al., 2021) Also proposed an ensemble learning algorithm based on OC-SVM, LOF, and modified isolation forest (M-iForest) classifiers for classification using datasets DREBIN and CICAndMal2017, The accuracy for the DREBINdataset was 98.7%, and 95.67% F-Score, while the accuracy for CICAndMal2017 was 98.99% and F-Score of 96.82%.

Roy et al., (2023) also proposed a novel hybrid classification model called MalHyStack, which detects such obfuscated malware within the network. This proposed working model is built incorporating a stacked ensemble learning scheme, wherein traditional machine learning algorithms, such as Random Forest, Extremely Randomised trees classifier (ExtraTrees), and Extreme Gradient Boosting (XgBoost) Classifier, are used in the first layer, followed by a deep learning layer in the second stage. An ideal subset of features was chosen using Pearson correlation analysis, which increased the model accuracy by more than 2% for multiclass classification and reduced time complexity by roughly two and three times for bidirectional classification the CIC-MalMem-2022 balanced dataset, which was just released. The suggested model overall experimental results using this dataset show improved performance accuracy 85.04% for multiclass and 99.98% for binary classification.

Additionally Zelinka et al. (2024) present an innovative and non-traditional method of visualising malware using fractal geometry, dynamic analysis and complicated network conversion, such as Julia sets visualisation. Subsequently, very intriguing visuals used to be categorised as goodware and malware. Deep learning networks were used for classification. The database that ESET provided, which included 4,174,203 malware samples, 827,853 possibly undesirable apps, and 6,589,997 goodware samples, served as the basis for the fractal conversion and subsequent classification trials that were presented. The goal of their

work was to demonstrate a novel approach to malware visualisation by utilising fractal geometry and its potential for analysis and categorization.

The research of Sandeep et al. (2019) also recommended using a static analysis technique in conjunction with a fully connected deep learning model to detect Android malware. One of the main features of the work is the identification of Android malware, version packages and detection of Android malware even before installation. They had an extraordinary 94.65% accuracy rating and the model also learnt every feature from every conceivable combination of features.

Experimenting with other known algorithm like Artificial Immune System (AIS) (Ndatsu and Adebayo, 2020) proposed a framework using AIS for feature selection of permission-based features only for android malware classification technique, for detection of android malicious applications, the results show that random forest model of selected permission-based feature accuracy performed better 98.7% than other algorithm compared with and models without the selection of features. Chandok et al. (2022) proposed an API's function call graph as a model, one may analyse both static and dynamic malware activity. After being transformed into lower dimensional feature vectors, the call graph embedding and permissions are trained using a Support Vector Classifier (SVC), whose hyper parameters were adjusted using the Grid Search method. Moreover, the same dataset is trained using artificial neural networks (ANNs), and comparisons are made by adjusting various hyper parameters, which then achieved an accuracy of 94%. They evaluated how effective explicit and implicit android intents are at identifying fraudulent applications. It was shown that Intents are semantically rich features capable of expressing malevolent intentions, especially when compared to other well-studied characteristics like permissions.

Android app features play a crucial role in effectively detecting malicious software (malware). By analysing these features, researchers can gain valuable insights into an app's functionality and potential risks.

For instance, Dhalaria (2021) proposed a method for detecting Android malware that leverages the power of app features. Their approach utilises both static and dynamic features, which capture information about the app's code and its behaviour at runtime, respectively. To identify the most informative features for malware detection, they employed a chi-square feature selection technique. This technique helps pinpoint features that are statistically significant in distinguishing between benign and malicious apps. Furthermore, Dhalaria (2021) implemented an ensemble classification approach combining K-Nearest Neighbors (K-NN) and

Random Forest (RF) algorithms. This ensemble leverages the strengths of both algorithms, ultimately achieving a high detection accuracy of 98.02%. This success highlights the effectiveness of using app features in conjunction with advanced machine learning techniques for robust malware detection.

Stacking is one of the ensemble's learning techniques. The identification and selection of different features from malware samples is a complicated task in malware detection and classification. (Sihag et al., 2021) introduced a malware detection framework for Android called Deep Learning based Android malware detection utilising Dynamic characteristics (De-LADY), which is based on both dynamic features analysis and deep learning. Logs that document an application's operating behaviour are essential to De-LADY. Their suggested model had a 98.84% F-measure and a 98.08% detection rate of accuracy.

A novel Apriori association rule with better malware detection model was suggested by (Adebayo & Aziz, 2019) During an unsupervised learning experiment, it had a root mean square error of 0.0355, an average accuracy of 98.17%, a detection rate of 98.25%, a false alarm rate of 0.0192, and a false alarm rate of 0.0192. Additionally, the memory and temporal complexity of the new model show increased computational effectiveness. It was investigated how the Apriori method may be improved for the extraction and selection of candidate detectors for classifier training. Particle swarm optimization was used to enhance the Apriori approach and boost its efficiency in producing candidate detectors for supervised learning.

3. Methodology

The problems identified after thorough review of the literatures in chapter two includes(Chandok et al., 2022) train selected algorithm to understand recent android malware behavioural pattern and be able to detect them, the false positive rate still high for android malware detection(Adebayo & Aziz, 2019), the research is intended to improve detection accuracy and reduce false positive rate. The successful application of ensemble learning algorithms in the detection of malware is the motivation to improve the algorithms performance. This research uses the ensemble learning technique of majority voting and Lasso Regularization Algorithm to improve the performance of the new model in relation to the existing ones.

3.1 Android Datasets Description

This is a partially pre-processed version of one of the datasets obtained from the Canadian Institute of Cyber Security's Malware Investigation (CICandMALINVEST2019) and Tezpur University's Android Malware Dataset (TUANDROMD) repositories. The datasets used in this research come from two groups: CICandMALINVEST2019, with 1,594 instances (1,187 benign and 407 malware) and 8,111 features, and

TUANDROMD, with 4,465 instances (899 benign and 3,565 malware) and 241 features. Both datasets have undergone partial pre-processing, as further preprocessing was still required for each individually. This combined dataset was used for classification in this research. The datasets serve as a benchmark for the training, testing, and evaluation of the proposed Android malware detection model.

3.2 Data Pre-Processing

Data preprocessing is a crucial step in machine learning, ensuring the data is clean, transformed, and prepared for training a model. In this case, the classification data was already normalised to a range of 1 and 0 (likely for both rows and columns). Preprocessing is essential to address potential issues that could affect the efficiency and performance of machine learning algorithms. These issues include missing values, redundant features, outliers, and the need for feature transformation, as described below.

1. During data pre-processing for the CICandMALINVEST2019 and TUANDROMD datasets, we first checked for missing values. We also identified features containing mixed data types, including strings. These strings were converted to integers to ensure consistency. Additionally, some cells contained unknown symbols, which were removed to eliminate data redundancy.
2. One crucial step in machine learning is encoding categorical variables into numerical representations that algorithms can understand. Both datasets had a categorical target feature stored as text. To address this, label encoding from scikit-learn's Python module was used to encode the target column in the TUANDROMD dataset. (Note: If label encoding was used for both datasets, you can mention it here instead of just TUANDROMD).
3. To ensure all features in the CICandMALINVEST2019 and TUANDROMD datasets have a comparable scale, normalisation or scaling was necessary. This step is crucial for models that utilise optimization techniques or distance-based calculations. Several common methods exist, including normalisation (achieved by dividing by the standard deviation or mean) and min-max scaling (which scales values to a specific range, typically 0 to 1).

3.2 Hyper-parameter tuning of ensemble

learning approach

To optimise the performance of the ensemble model (MLPNN-RF-LGBM with Lasso Regularization), we employed hyperparameter tuning for specific algorithms, namely the Multilayer Perceptron (MLPNN). Hyperparameter tuning involves iteratively adjusting these parameters while evaluating the model's performance on validation data. This process aims to find the configuration that leads to the best results. The specific parameters used for the MLPNN are shown in Table 3, while the details of Lasso Regularization are presented in Table 1.

Table 1: Multilayer Perceptron Neural Network Classifier Parameter

Parameter	value
Hidden_layer_sizes	250
Max_iter	300
Solver	sgd
Random_state	42

Lasso Regularization-Based Algorithm for Android Feature Selection

Android's (singular possessive) pertinent permissions, intent manifests, and APIs converge to elevate classification performance by offering transparency, insight, and context. These features empower classification algorithms to make informed decisions based on an app's permissions, communication patterns, and functionalities. After collecting datasets from the CICandMALINVEST2019 and TUANDROMD repositories and performing pre-processing, hyperparameter tuning of the Lasso Regularization algorithm was done to optimize performance and avoid overfitting (high variance and bias), as shown in Table 1. Feature selection was performed using the Lasso Regularization algorithm to identify the most informative features from the datasets. This process resulted in a reduced feature set of 172 features for one dataset and 4,169 features for the other (details on the feature selection process can be found in Table 2 and Figure 1).

Table 2: Lasso Regularisation Parameters

Parameter	Value
Alpha	0.00001
Random_state	0

3.4. Splitting of the dataset

The dataset was splitted in 80% for training and 20% for testing and validation, the Creation of training, validation, and test sets from the pre-processed data help in various ways to build the Android malware detection (AMD) model. The validation and test set aids in tuning the hyperparameters model selection and is utilised for the model final performance assessment, and the training set is used to train the selected algorithms.

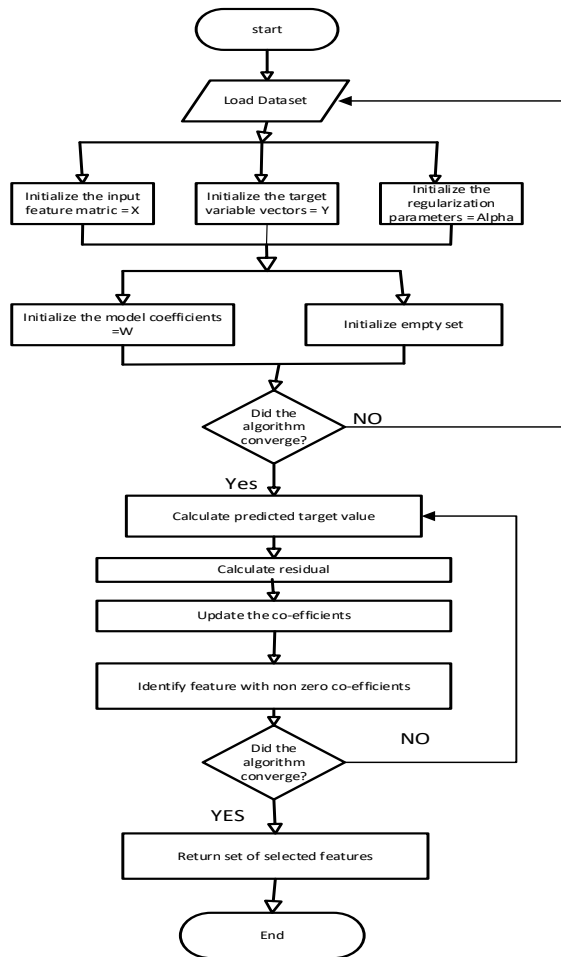


Figure 1: Design Process of Lasso Regularization AI

The flowchart of Figure 2 talks about the development of the ensemble-based approach of MLPNN-RF-LGBM from data collection, then data-preprocessing, tuning of the parameters of Lasso Regularization and Multilayer perceptron neural network to optimise their performance, the intrinsic conditions and process the Lasso Regularization feature selection and ensemble algorithm use was displayed on the flowchart.

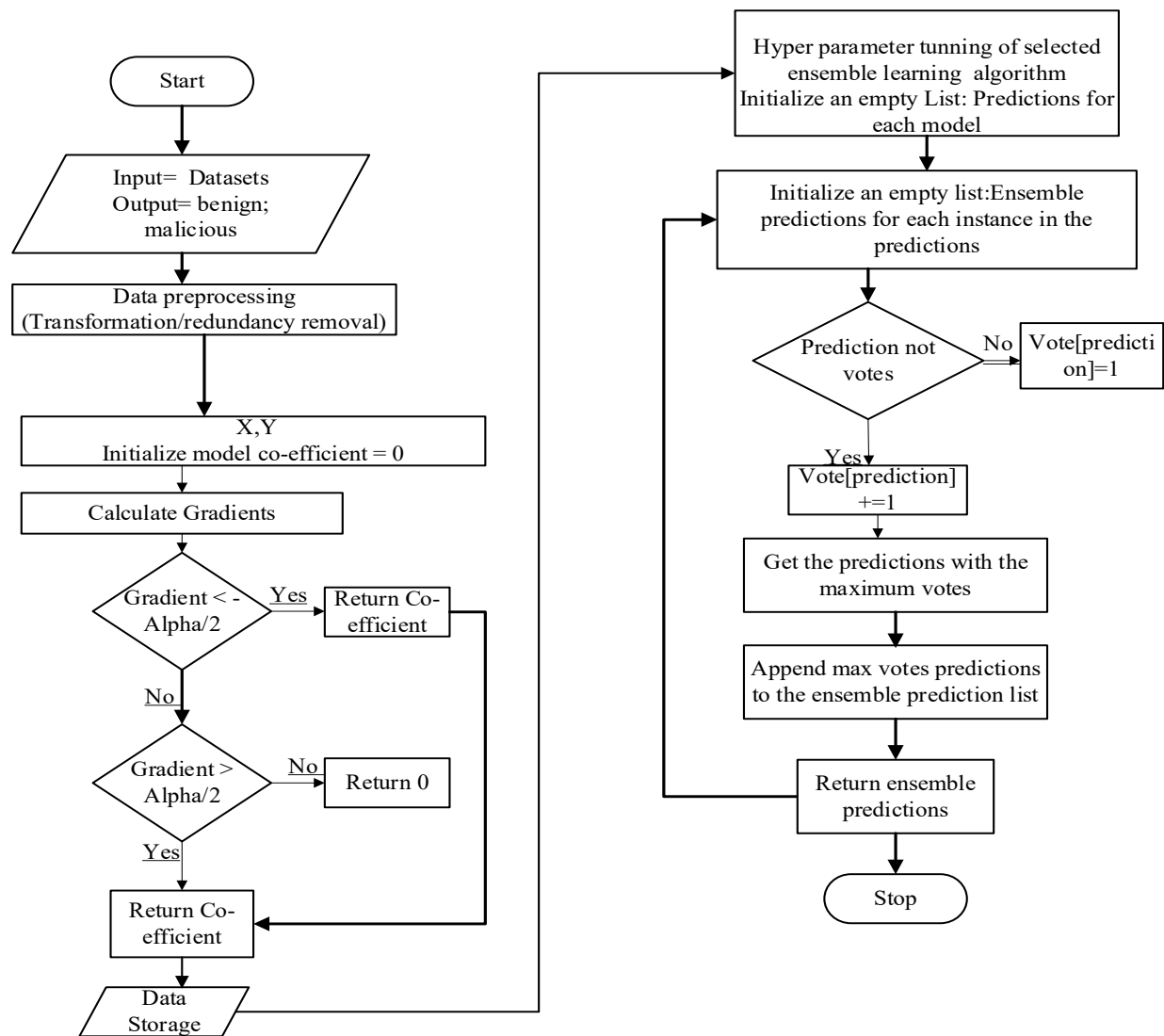


Figure 2. Flowchart of the Android Malware Detection Model

3.5. Ensemble Approach-MLPNN-RF-LGBM AMD Model

This session depicts a pictorial view architecture of the ensemble approach-MLPNN-RF-LGBM model at figure 2. In the context of Android malware detection, an ensemble approach involves combining multiple individual models to enhance the overall performance and reliability of the malware detection system. The primary goal is to overcome the limitations of single models by leveraging the strengths of multiple algorithms. By combining the strengths of these different models, it (referring to the ensemble

approach) achieves better predictive performance, handles different types of data patterns, and increases the robustness of the predictions.

Start by training a multilayer perceptron neural network (MLP) on the labelled training data. The neural network consists of multiple layers of interconnected neurons and can learn complex patterns in the data. To optimise performance, we carefully tune the network's hyperparameters. Next, train a random forest model on the same labelled training data. Random forests are made of an ensemble of decision trees, and they are known for their ability to handle high-dimensional data and reduce overfitting. Experiment with different tree depths, number of trees. Also training a light

gradient boost algorithm, light gradient boosting (LightGBM) is a gradient boosting framework that uses decision trees as base learners. Train a LightGBM model on the labelled training data. Gradient Boosting Machine combines weak models into a strong model by iteratively adding models that correct the mistakes of previous models. tree depth, and number of iterations to achieve optimal performance.

The model is combined, once you have trained the individual models, the predictions are collated and stored to make the final ensemble prediction. Such as voting, each model makes a prediction, and the final prediction is determined by majority vote. This research made use of majority voting amongst the three techniques of ensemble learning, which gave a better result.

Majority voting is a technique used in ensemble learning, where multiple machine learning models are combined to make predictions and vote the most prediction as benign or malicious. In our case, we used three different algorithms (MLPNN-RF-LGBM) and applied majority voting to combine their predictions. the pseudo-code that depicts majority voting is stated.

Table 4: Confusion Matrix

Predicted Actual		
	Negative	Positive
Benign		TN

	FP	
Malicious		FN
	TP	

Source: (Patil and Patil, 2018)

Using the above confusion matrix, the following measures were carried out to evaluate the performance of the model. A binary classifier predicts all data instances of a test dataset as either positive or negative. This classification (or prediction) produces four outcomes: true positive, true negative, false positive and false negative.

1. **True Positive (TP):** implies the total number of malicious applications software instances “correctly” labelled by the classifier as malicious attack.
2. **True Negative (TN):** represents the total number of benign application software instances “correctly” labelled by the classifier as benign.
3. **False Positive (FP):** the total number of benign application instances “incorrectly” labelled by the classifier as malicious application.

Table 3 details the step-by-step process of the ensemble-based MLPNN-RF-LGBM model, which utilises ensemble learning to combine predictions from multiple machine learning algorithms. To complement this explanation, Figure 4 provides a visual representation of the overall research process for Android malware detection.

Table 3: Ensemble-based Approach MLPNN-RF-LGBM Algorithm

Algorithm: Ensemble- based approach MLPNN-RF-LGBM

Input: Dataset of CICANDMAL2019 and TUANDROMD

Output: ‘Ensemble Predictions’

1. **Input:** The algorithm takes two input parameters:
2. **Models:** A list of trained machine learning models including Multi-Layer Perceptron neural network, Random Forest, and Light Gradient Boost.
3. **X:** The input features for which predictions are to be made.
4. **Initialize:** Initialise an empty list called “Predictions” to store the individual model predictions.
5. For each model in Models:
6. Make predictions for the input features ‘x’ using the current model and store the predictions in the variable ‘model_predictions’.
7. Append the ‘model_predictions’ to the “Predictions” list.
8. **Initialize:** Initialise an empty list called ‘Ensemble_Predictions’ to store the final ensemble predictions.
9. **For each instance in Predictions:**
10. Create an empty dictionary called ‘votes’ to keep track of the vote count for each unique prediction.
11. For each prediction in the instance:
12. Check if the prediction is already present in the ‘votes’ dictionary.
13. If the prediction is not in ‘votes’, add it as a key and set its value to 1.
14. If the prediction is already in ‘votes’, increment its value by 1.
15. Get the prediction with the maximum votes using the ‘max’ function and the ‘key’ parameter set to ‘votes.get’.

-
16. Append the maximum voted prediction to the 'Ensemble_Predictions' list.
 17. **Return Ensemble Predictions:** Once all instances have been processed, return the final ensemble predictions stored in the 'Ensemble Predictions' list.
-

4. Results and Analysis

To evaluate the performance of the proposed machine learning model for detecting malware on Android phones, we employed metrics such as accuracy, false positive rate, and F1-score. Table 4 presents the confusion matrix used in this research, which served as the basis for calculating false positives.

4.1. Results for Lasso Regularization Based Feature Selection

This research leverages two publicly available datasets for Android malware detection:

1. CICandMalInvest2019 (Chandok et al., 2022): This dataset, created in 2019, contains static analysis data for five classes of Android malware: scareware, adware, ransomware, SMS malware, and SMS premium malware.

2. TUANDROMD (Borah et al., 2020): This dataset, created in 2020, consists of dynamic analysis data for Trojan and rootkit malware. The features in TUANDROMD were extracted from host-based and network traffic generated by Android applications.

Both datasets underwent feature selection using the Lasso Regularization algorithm to identify the most informative features for malware classification. This process resulted in a reduced feature set of 172 features for TUANDROMD and 4,169 features for CICandMalInvest2019.

Some of these selected features are visualised in Figures 5 and 6, which can provide insights into the characteristics that differentiate benign and malicious applications.

```
print(selected_feature_names)

Index(['actionandroidnameandroidintentactionFILEEXPLORE',
      'actionandroidnamecomjbgkeyboardmessagecentermsgcontentactivity',
      'actionandroidnameandroidnetwifip2pTHIS_DEVICE_CHANGED',
      'actionandroidnamecomcitcweatherwidgetWIDGET_FORECAST_UPDATE',
      'actionandroidnamebrcommobicarewifimanagersACTION_WITHOUT_KNOW_NETWORKS',
      'actionandroidnamecomkreordercallbroadcastUPLOAD_RETRY',
      'actionandroidnamecomkreactionivedigischoolbacACTION_UPDATE_CATEGORY_QUIZ_PROGRESS',
      'actionandroidnamecomtraceupandroidtracesnowAddVisitCommentactionlaunch',
      'actionandroidnameWifi_SMS_DELIVERED',
      'actionandroidnamecomciamediacalleridintentSEND_PROFILE_UPDATE',
      ...,
      'comgoogleandroidproviderstalkpermissionWRITE_ONLY',
      'comjoelapennafoursquaredpermissionC2D_MESSAGE',
      'worksexamplesSyllabuspermissionC2D_MESSAGE',
      'comgoogleandroidgoogleappspermissionGOOGLE_AUTHah',
      'androidpermissionBROADCAST_PACKAGE_REPLACED',
      'caautotraderuserapppermissionC2D_MESSAGE', 'sednpermissionC2D_MESSAGE',
      'comfoursquarepermissionACCOUNTS', 'comtulaviatepermissionC2D_MESSAGE',
      'androidpermissionFORCE_STOP_PACKAGES'],
      dtype='object', length=4169)
```

Figure 4: CICANDMalinvest2019 Selected Data Features

```
print(selected_feature_names)

Index(['ACCESS_ALL_DOWNLOADS', 'ACCESS_CACHE_FILESYSTEM',
      'ACCESS_CHECKIN_PROPERTIES', 'ACCESS_COARSE_LOCATION',
      'ACCESS_COARSE_UPDATES', 'ACCESS_FINE_LOCATION',
      'ACCESS_LOCATION_EXTRA_COMMANDS', 'ACCESS MOCK_LOCATION',
      'ACCESS_NETWORK_STATE', 'ACCESS_SUPERUSER',
      ...,
      'Landroid/content/pm/PackageManager;->getInstalledPackages',
      'Landroid/telephony/TelephonyManager;->getLine1Number',
      'Landroid/telephony/TelephonyManager;->getNetworkOperator',
      'Landroid/telephony/TelephonyManager;->getNetworkOperatorName',
      'Landroid/telephony/TelephonyManager;->getNetworkCountryIso',
      'Landroid/telephony/TelephonyManager;->getSimOperator',
      'Landroid/telephony/TelephonyManager;->getSimOperatorName',
      'Landroid/telephony/TelephonyManager;->getSimCountryIso',
      'Landroid/telephony/TelephonyManager;->getSimSerialNumber',
      'Lorg/apache/http/impl/client/DefaultHttpClient;->execute'],
      dtype='object', length=172)
```

Figure 5: TUANDROMD Selected Data Features

4.2. Results Comparison of the AMD Model with Other Models

Table 5 compares the performance of the proposed ensemble learning model (MLPNN-RF-LGBM) with other approaches that have also utilised ensemble learning on the CICandMalInvest2019 (static analysis) dataset. The evaluation metrics employed are accuracy and false positive rate, as shown in Table 4.

The proposed model achieves a superior accuracy of 0.9692 and a low false positive rate of 0.0026. This performance suggests that the model is robust, efficient, and effective in malware detection. The specific algorithms chosen for the ensemble approach likely contribute to these positive results. The research work of (Sharma and Agrawal, 2022) had a combined use of analysis for obtaining their dataset which is hybrid analysis but (Sandeep, 2019) research work made use of static analysis dataset. Figure 6 and Figure 7 shows the learning curve during the training of the ensemble algorithms with the dynamic and static dataset, it also shows how it gradually learns and becomes better in its accuracy using both datasets. During the training it tends to perform higher in accuracy due the algorithms being still getting to know the data better but during validation it reduces.

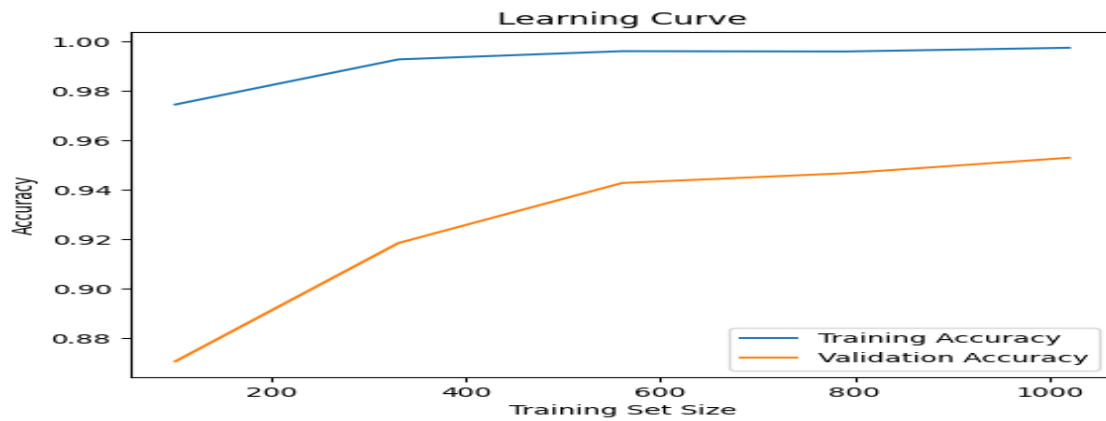


Figure 6: Learning Curve for During The Static Dataset

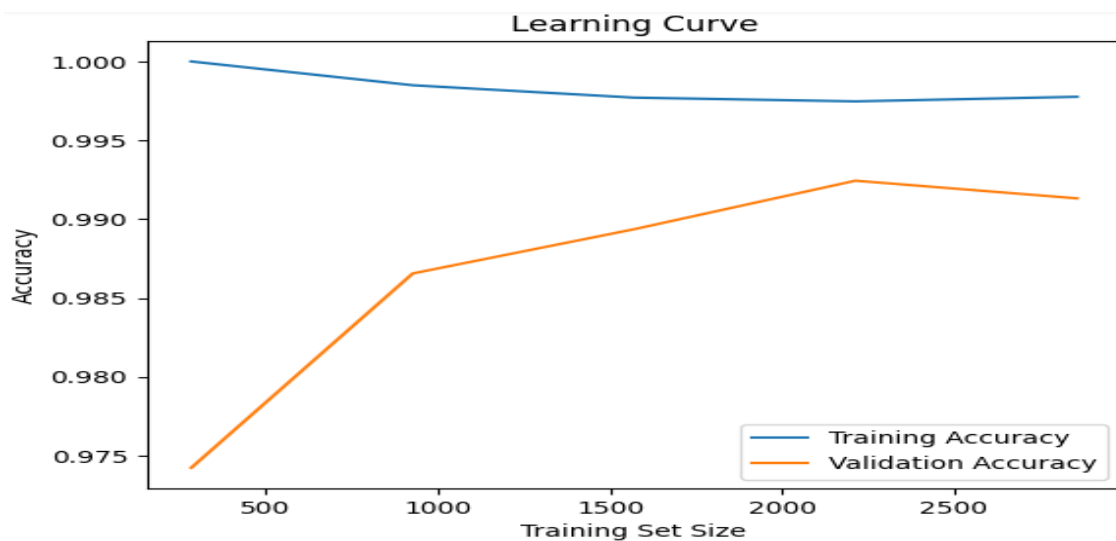


Figure 7: Learning Curve for During the Dynamic Dataset

Table 5: Result Comparison of the MLPNN-RF-LGBM Model with Same Existing Method Approaches (Static dataset)

Reference	Approach	Type of analysis data	FPR	F1-Score	Accuracy
Propose Model	LRMLPNN-RF-LGBM Ensemble learning Approach	Static	0.0026	0.91764	0.9692
(R. M. Sharma & Agrawal, 2022)	A BPSO Ensemble Learning Approach	Hybrid	0.2435	0.9893	0.9492
(Sandeep, 2019)	Ensemble Learning Approach	Static	11.4285		0.9465

Tables 5 and 6 present a comparison of the proposed ensemble learning model (MLPNN-RF-LGBM) with other approaches that have also employed ensemble learning for malware detection. It's important to note that while these approaches did not use the exact datasets used in this research, they utilised dynamic analysis data and targeted similar malware types (Trojans and rootkits). Therefore, the comparison focuses on the effectiveness of the ensemble learning methodology (combining multiple algorithms) rather

than a direct comparison of datasets.

The proposed model achieves a high level of performance on the dynamic analysis dataset, with an accuracy of 0.99601, F1-measure of 0.99441, and a low false positive rate of 0.01124. These results suggest that the ensemble learning approach, particularly the chosen algorithms within the ensemble, is effective for dynamic malware detection.

Table 6: Result Comparison of the MLPNN-RF-LGBM Model with Same Existing Method Approaches (Dynamic Dataset)

Reference	Approach	Type of analysis data	FPR	F1-Score	Accuracy
Propose model	LR-MLPNN-RF-LGBM Ensemble learning Approach	Dynamic	0.01124	0.99441	0.99601
(Sihag <i>et al.</i> , 2021)	De-LADY Ensemble learning approach	Dynamic	0.0192	0.9884	0.9808
(Feng <i>et al.</i> , 2018)	Ensemble learning approach	Dynamic	0.0167	0.9642	0.9719
(Dhalaria, 2021)	Ensemble Learning approach	Dynamic	0.020	0.980	98.02

4.2.1: Lasso Regularization algorithm was evident in improving the performance of the ensemble learning algorithms by removing irrelevant features and selecting pertinent features, hyperparameters tuning also improved the performance of the feature selection, we will see in the table 7 as it very necessary to using a feature selection algorithm in machine learning and classification data analysis.

Table 7 presents detection results of the Android Malware Detection ensemble MLPNN-RF-LGBM model without the feature selection algorithm, the result which comprises the following evaluation metric false positive rate, F1-Score and majority vote-Accuracy of the AMD model. It shows the results of the model had lower accuracy and low FPR without the feature selection algorithm being applied

Table 7: MLPNN-RF-LGBM Model without Lasso Regularization Algorithm

Models	Type of data analysis	FPR	F1-Score	Majority Voting Accuracy
MLPNN-RF- LGBM	Dynamic	0.02247	0.99510	0.99216
MLPNN-RF-LGBM	Static	0.02575	0.91764	0.9692

The Table 8 shows results obtained from the android malware detection model, using majority voting for ensemble learning (LRMLPNN-RF-LGBM) with lasso regularisation algorithm for feature selection. We have the F1-score, False positive, Accuracy of the majority voting results of both the static and dynamic data analysis separately. The performance of the model is quite recommendable. A robust, effective, and efficient model.

5.0 Conclusion

Android phones are increasingly vulnerable to malware attacks due to their reliance on a vast application ecosystem. The constant influx of new apps, often created for diverse functionalities, presents a growing challenge for cybersecurity.

This research addresses this challenge by proposing a novel Android malware detection model. The model, named MLPNN-RF-LGBM, leverages an ensemble learning approach, combining the strengths of Multilayer Perceptron Neural Networks (MLPNNs), Random Forests (RFs), and Light Gradient Boosting Machines (LGBMs).

To achieve optimal performance, the model underwent a series of experiments that involved hyperparameter tuning. This process fine-tunes the internal settings of the algorithms within the ensemble, leading to improved accuracy during the validation phase. The proposed MLPNN-RF-LGBM model demonstrates superior performance compared to other benchmarked research works. Notably, it achieves a high accuracy rate during testing, indicating its effectiveness in identifying malicious applications. Additionally, the model exhibits a low false positive rate, minimizing the chances of incorrectly classifying benign apps as malware.

5.1 Acknowledgement

The authors would like to extend sincere gratitude to the Department of Cybersecurity Science, School of Information Communication Technology, Federal University of Technology, Minna, Nigeria, for their support and resources throughout the course of this research. This work would not have been possible without their invaluable guidance and assistance

Reference

- Adebayo, O. S., & Abdulaziz, N. (2014). Android malware classification using static code analysis and Apriori algorithm improved with particle swarm optimization. *2014 4th World Congress on Information and Communication Technologies, WICT 2014*, 123–128. <https://doi.org/10.1109/WICT.2014.7077314>
- Adebayo, O. S., & Aziz, N. A. (2019). Improved Malware Detection Model with Apriori Association Rule and Particle Swarm Optimization. *Security and Communication Networks*, 2019. <https://doi.org/10.1155/2019/2850932>
- Ansori, D. B., Slamet, J., Ghuftron, M. Z., Aidiel, M., Putra, R., & Ahmad, T. (2024). *Android Malware Classification Using Gain Ratio and Ensemble Machine Learning*. 14(1), 259–266.
- Arshad, A., Rehman, A. U., Javaid, S., Ali, T. M., Sheikh, J. A., & Azeem, M. (2021). *A Systematic Literature Review on Phishing and Anti-Phishing Techniques*. <http://arxiv.org/abs/2104.01255>
- Borah, P., Bhattacharyya, D. K., & Kalita, J. K. (2020). Malware dataset generation and evaluation. *4th IEEE Conference on Information and Communication Technology, CICT 2020*. <https://doi.org/10.1109/CICT51604.2020.9312053>
- Chandok, A., Verma, A., & Gupta, R. (2022). *Dro-Mal Detector : A Novel Method of Android Malware Detection*. 1–9. 2022 3rd International Conference for Emerging Technology | 978-1-6654-9499-1/22/\$31.00 ©2022 IEEE | DOI: 10.1109/INCET54531.2022.9824877
- Dhalalaria, M. (2021). *Android Malware Detection using Chi-Square Feature Selection and Ensemble Learning Method*. 36–41. 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC) | 978-1-7281-7132-6/20/\$31.00 ©2020 IEEE | DOI: 10.1109/PDGC50313.2020.9315818 2020
- Feng, P., Ma, J., Sun, C., & Ma, Y. (2018). A Novel Dynamic Android Malware Detection System With Ensemble Learning. *IEEE Access*, 6, 30996–31011. <https://doi.org/10.1109/ACCESS.2018.2844349>
- Guerra-manzanares, A., Bahsi, H., & Nömm, S. (2021). KronoDroid : Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization. *Computers & Security*, 110, 102399. <https://doi.org/10.1016/j.cose.2021.102399>
- Islam, R., Islam, M., Saha, S., Jamal, M., & Masud, A. (2023). Internet of Things and Cyber-Physical Systems Android malware classification using optimum feature selection and ensemble machine learning. *Internet of Things and Cyber-Physical Systems*, 3(January), 100–111. <https://doi.org/10.1016/j.iotcps.2023.03.001>

Mahindru, A. (2020). MLDroid — framework for Android



- malware detection using machine learning techniques. In *Neural Computing and Applications* (Vol. 4). Springer London.
<https://doi.org/10.1007/s00521-020-05309-4>
- Ndatsu, Z. & Adebayo, O. . (2020). *Framework for the Detection of Android Malware Using Artificial Immune System*. 2017, 117–126.
- Patil, D. R., & Patil, J. B. (2018). Malicious URLs detection using decision tree classifiers and majority voting technique. *Cybernetics and Information Technologies*, 18(1), 11–29.
<https://doi.org/10.2478/cait-2018-0002>
- Roy, K. S., Ahmed, T., Udas, P. B., & Karim, E. (2023). Intelligent Systems with Applications MalHyStack : A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. *Intelligent Systems with Applications*, 20(September), 200283.
<https://doi.org/10.1016/j.iswa.2023.200283>
- Sandeep, H. R. (2019). Static analysis of android malware detection using deep learning. 2019 *International Conference on Intelligent Computing and Control Systems, ICCS 2019*, *Iciccs*, 841–845.
<https://doi.org/10.1109/ICCS45141.2019.9065765>
- Sharma, R. M., & Agrawal, C. P. (2022). A BPSO and Deep Learning Based Hybrid Approach for Android Feature Selection and Malware Detection. *Proceedings - 2022 IEEE 11th International Conference on Communication Systems and Network Technologies, CSNT 2022*, 628–634.
<https://doi.org/10.1109/CSNT54456.2022.9787671>
- Sharma, S., Challa, R. K., & Kumar, R. (2021). An ensemble-based supervised machine learning framework for android ransomware detection. *International Arab Journal of Information Technology*, 18(3 Special Issue), 422–429.
<https://doi.org/10.34028/IAJIT/18/3A/5>
- Sihag, V., Justice, C., Vardhan, M., Singh, P., & Choudhary, G. (2021). *De-LADY : Deep learning based Android malware detection using Dynamic De-LADY : Deep learning based Android malware detection using Dynamic features*. May.
<https://doi.org/10.22667/JISIS.2021.05.31.034>
- Zelinka, I., Szczypka, M., Plucar, J., & Kuznetsov, N. (2024). From malware samples to fractal images : A new paradigm for classification. *Mathematics and Computers in Simulation*, 218(October 2023), 174–203.
<https://doi.org/10.1016/j.matcom.2023.11.032>
- Zhang, H., Wang, J., Sun, Z., Zurada, J. M., & Pal, N. R. (2020). Feature Selection for Neural Networks Using Group Lasso Regularization. *IEEE Transactions on Knowledge and Data Engineering*, 32(4), 659–673.
<https://doi.org/10.1109/TKDE.2019.2893266>
- Zhang, N., Tan, Y. an, Yang, C., & Li, Y. (2021). Deep learning feature exploration for Android malware detection. *Applied Soft Computing*, 102, 107069.
<https://doi.org/10.1016/j.asoc.2020.107069>