International Journal of Engineering Research and Science & Technology

IJERST

ISSN : 2319-5991

www.ijerst.com

Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

ANDROID MALWARE DETECTION THROUGH INTELLIGENT PATTERN RECOGNITION USING DEEP LEARNING AND EQUILIBRIUM OPTIMIZATION

[1] *M Nithish Kumar, MCA Student, Department of MCA*

[2] *K.Samson Paul,M.Tech,(Ph.D), Assistant Professor,Department of MCA*

[12]*Dr KV Subba Reddy Institute of Technology, Dupadu, Kurnool*

ABSTRACT

OUR PROPOSED PROJECT ABSTRACT:

By precisely identifying malicious software using cutting-edge machine learning techniques, the project "Malware Analysis and Detection Using Machine Learning Algorithm" seeks to improve cyber-security measures. The project, which was created in Python, uses HTML, CSS, and JavaScript to create a responsive and interactive frontend interface and the Flask web framework for backend functions.

The Extra Tree Classifier and Logistic Regression are two machine learning models that are essential to this project. With 97.42% training accuracy and 97.23% testing accuracy, the Extra Tree Classifier model performs quite well. with contrast, 94.84% training accuracy and 93.67% testing accuracy are attained with the Logistic Regression model. The TUNADROMD dataset, which includes 242 attributes and 4465 instances, is used to train and evaluate both models. The target classification attribute is used to differentiate between malware and goodware.

Based on their significance and effect on the categorisation task, a subset of 23 attributes was chosen for the analysis. The goal of this calculated choice is to minimise computational complexity and maximise model performance. According to the project's findings, the Extra Tree Classifier provides a dependable solution for malware detection in practical applications by effectively differentiating between harmful and benign software.

All things considered, this study shows how effective machine learning algorithms are in cyber-security, offering a reliable malware detection solution that can be included into many digital security infrastructures.

I. INTRODUCTION

Smartphones, especially those with the Android operating system, are being used at an unprecedented rate in today's digitally connected world due to the proliferation of mobile devices. Android, the most popular mobile operating system, has emerged as a top target for hackers looking to take advantage of security flaws for nefarious ends. The amount and complexity of Android malware threats have significantly increased as a result of the rise in mobile device usage and the open nature of the Android platform.

Malicious software intended to damage devices, steal confidential data, or interfere with services is referred to as Android malware. These malicious apps pose major dangers to users' security and privacy since they can proliferate through several distribution methods, penetrate app stores, and pose as genuine software. Android malware identification and prevention are becoming essential parts of mobile security because of the sensitive and private nature of the data kept on mobile devices, including contacts, messages, banking information, and personal media.

The rapid growth of malware variants is too much for traditional malware detection approaches, which mostly rely on signature-based methods. The inability of these traditional methods to identify novel and unidentified

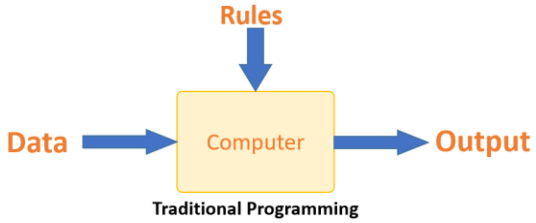
International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

dangers frequently calls for the creation of more sophisticated and flexible detection systems. Machine learning presents a viable answer to this problem because of its capacity to evaluate enormous datasets and spot intricate patterns. Even in the face of new and advanced threats, malware may be detected and classified more accurately and efficiently by utilising machine learning techniques.

The goal of this project is to use cutting-edge machine learning techniques to create a reliable system for Android virus detection. The research aims to identify dangerous software with high accuracy by using the TUNADROMD dataset and concentrating on the implementation of machine learning models like the Extra Tree Classifier and Logistic Regression. The solution is guaranteed to be efficient and user-friendly by combining these models with a useful web-based interface that was created with Python, Flask, and contemporary frontend technologies.
In conclusion, this effort is motivated by the urgent need for improved Android malware detection techniques. This project aims to make a substantial contribution to the field of mobile cybersecurity by utilising machine learning and taking a thorough approach to feature selection and model evaluation, giving users dependable tools to protect their devices and data against changing malware threats.

What is Machine Learning?

Without explicit programming, machine learning is a set of computer algorithms that can learn from examples through self-improvement. A subfield of artificial intelligence called machine learning uses data and statistical methods to forecast results that can be utilised to provide insights that can be put into practice.
The idea that a machine can learn from data alone (for instance) and generate precise outcomes is the breakthrough. Bayesian predictive modelling and data mining are strongly related to machine learning. After receiving data as input, the computer creates answers using an algorithm.
Making recommendations is a common machine learning problem. All movie or series recommendations for Netflix subscribers are based on their past usage history. Unsupervised learning is being used by tech companies to enhance user experience through personalised recommendations.
Other applications of machine learning include fraud detection, portfolio optimisation, predictive maintenance, task automation, and more.
Traditional Programming vs. Machine Learning
Machine learning is very different from traditional programming. With conventional programming, a programmer works with an industry expert to develop the rules before writing any code. Every rule has a logical basis, and the machine will carry out an output after the logical statement. More rules must be created as the system becomes more intricate. It may soon become impossible to sustain.
Machine learning is very different from traditional programming. With conventional programming, a programmer works with an industry expert to develop the rules before writing any code. Every rule has a logical basis, and the machine will carry out an output after the logical statement. More rules must be created as the system becomes more intricate. It may soon become impossible to sustain.



Traditional Programming

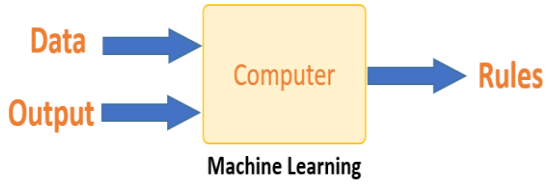
International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

This problem is meant to be solved by machine learning. The machine creates a rule after learning the correlation between the input and output data. Every time there is new data, the programmers do not have to create new rules. Over time, the algorithms' efficacy is enhanced by adapting to new information and experiences.



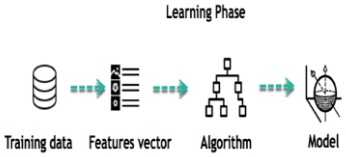
Machine Learning

How does Machine Learning Work?

All learning occurs in the brain of machine learning. Humans and machines both learn in comparable ways. Experience is how humans learn. Predicting becomes easier the more we know. By analogy, our chances of success are lower in unknown situations than in recognised ones. The same training is given to machines. The machine sees an example in order to produce a precise prediction. The machine can determine the result when we offer it a comparable example. But just like a human, the machine struggles to forecast if it is fed an example that hasn't been seen before.

Learning and inference are the main goals of machine learning. Initially, the machine learns by identifying patterns. The data is responsible for this discovery. Selecting which data to provide the computer is an important task for the data scientist. A feature vector is a collection of characteristics that are utilised to address an issue. A feature vector can be thought of as a subset of data that is utilised to address an issue.

The machine turns this information into a model by simplifying reality using some sophisticated algorithms. As a result, the data is described and condensed into a model during the learning step.

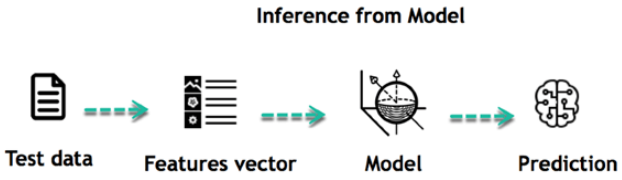


For example, the machine is attempting to determine the correlation between an individual's income and the probability of dining at a posh restaurant. As it happens, the algorithm discovers a favourable correlation between paying a wage and dining at a fancy restaurant: The model is this.

Inferring

Once the model is constructed, its strength can be evaluated on previously unseen data. After being converted into a features vector, the new data is run through the model and a forecast is made. All of this is what makes machine learning so lovely. It is not necessary to train the model afresh or update the rules. The previously trained model can be used to new data to draw conclusions.

.



Machine learning programs lead simple lives that can be summed up in the following ways:

1. State a query
2. Gather information
3. Display data 4. Develop the algorithm 5. Evaluate the algorithm
6. Get input 7. Improve the algorithm
8. Repeat steps 4–7 until you are satisfied with the outcome.
9. Create a prediction using the model.

The algorithm applies what it has learnt to new data sets after it becomes proficient at making the correct deductions.

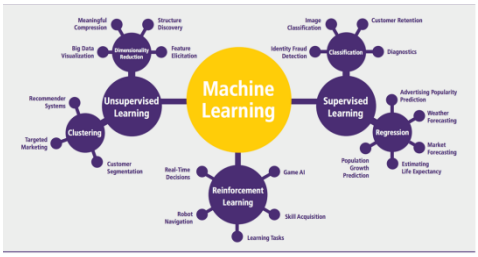
International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

Machine Learning Algorithms and Where they are Used?



Machine learning Algorithms

The two main categories of machine learning tasks are supervised and unsupervised. Numerous alternative algorithms exist.

Supervised learning

An algorithm learns the link between inputs and outputs by using training data and human feedback. For example, a practitioner can forecast can sales using weather forecasts and marketing expenditures as input data.

When the output data is known, supervised learning can be used. New data will be predicted by the program.

Two types of supervised learning exist:

1. Classification

2. Regression

Classification

Let's say you want to forecast a customer's gender for a commercial. You will begin collecting information from your customer database like height, weight, occupation, pay, shopping basket, etc. You are aware that every one of your customers is either male or female. Assigning a chance of being male or female (the label) based on the data (i.e., attributes you have gathered) will be the classifier's goal. New data can be used to make predictions once the algorithm has learnt to identify males and females. For example, you want to know if a new customer you just obtained from an unknown source is male or female. In the event that the classifier predicts male = 70%, the algorithm is certain that this customer is male at 70% and female at 30%.

Two or more classes may be represented by the label. A classifier that must predict objects contains dozens of classes (e.g., glass, table, shoes, etc. each object represents a class), whereas the machine learning example above just has two classes.

Regression

The task is a regression when the output is a continuous value. For example, a financial analyst could have to predict a stock's worth based on a variety of factors, such as equity, past stock performance, and the macroeconomic index. After training, the system will be able to predict stock prices with the least amount of error.

II. LITERATURE SURVEY

1) Deep learning-based attack detection and classification in Android devices

AUTHORS: A. Gómez and A. Muñoz

Attackers have turned their attention to Android-based devices because to their growing popularity, as they presently hold a stunning 72% global market share. As a result, identifying Android malware has become a crucial field of study. Android malware detection and classification is still a problem, despite the fact that both industry and academia have investigated a number of strategies to create reliable and effective solutions. In this work, we provide a supervised learning method that shows promise in detecting malware on Android devices. The foundation of our methodology is the development of an extensive labelled dataset that includes more than 18,000 samples categorised into five different groups: banking, riskware, adware, benign programs, and SMS. Our suggested model's efficacy is confirmed on reputable datasets including CICMalDroid2020, CICMalDroid2017, and CICAndMal2017. Our methodology beats existing semi-supervised approaches in certain dimensions when compared to state-of-the-art techniques in terms

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

of precision, recall, efficiency, and other pertinent characteristics. On the other hand, we admit that our model does not significantly differ from other methods in some aspects. All things considered, our study supports continued attempts to create sophisticated methods for identifying and categorising Android malware. We think that our research will stimulate more research, resulting in improved security protocols and defences for Android devices against changing threats.

2) On the impact of sample duplication in machine-learning-based Android malware detection

AUTHORS: Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy

In the Android space, machine learning techniques are frequently used for large-scale malware detection. When evaluated against popular datasets, state-of-the-art methods like DREBIN and MaMaDroid are said to produce high detection rates. Sadly, a significant percentage of duplicate samples may be present in these databases, which could skew the documented experimental findings and insights. In this work, we conduct comprehensive tests to quantify the performance difference that arises from de-duplicating datasets. According to our experimental findings, supervised malware classification models are only marginally affected by duplication in published datasets. Allamanis's findings on the broader issue of machine learning bias for huge code are in contradiction to this observation. However, our tests demonstrate that sample duplication has a more significant impact on unsupervised learning models (such malware family grouping). However, regardless of the potential impact, we contend that sample duplication should always be taken into account by our fellow academics and practitioners when doing machine-learning-based (either supervised or unsupervised learning) Android malware detections.

3) You are what the permissions told me! Android malware detection based on hybrid tactics

AUTHORS: H. Wang, W. Zhang, and H. He

The use of Android smartphones in many facets of our lives has significantly increased in recent years. Nevertheless, customers have the option to acquire Android apps through unaffiliated channels, which gives malware a lot of options. Attackers obtain sensitive user private information by using unsolicited permissions. There is an urgent need for effective and flexible antiviral solutions, particularly in novel versions, as signature-based solutions are no longer practicable. We suggest a hybrid Android malware detection method that blends static and dynamic strategies as a solution. First, using a machine-learning-based approach, we employ static analysis to infer distinct permission consumption patterns between malicious and benign apps. We create a dynamic feature base by extracting the object reference associations from the RAM heap in order to further identify the suspicious apps. Next, we introduce an enhanced DAMBA-based state-based algorithm. Our method surpasses the popular detector with 97.5% F1-measure, according on experimental results on a real-world dataset of 21,708 apps. Additionally, it has been shown that our system is resistant to obfuscation tactics and permission abuse behaviours.

4) Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses

AUTHORS: H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak

With billions of users now, Android smartphones have become a lucrative target for malware developers. It is therefore more of a

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

need than a desire for the anti-malware community and malware developers to stay one step ahead in this zero-sum game of malware detection. In order to create adversarially better Android malware detection models, our work focusses on a proactive adversary-aware framework. First, we examine the adversarial robustness of 36 different malware detection models built with 18 classification techniques and two static features (intent and authorisation). To take advantage of flaws in the aforementioned malware detection models, we created two reinforcement learning-based Targeted Type-II Evasion Attacks (TRPO-MalEAttack and PPO-MalEAttack). In order to trick the malware detection models, the attackers try to cause as little disruption as possible to each malicious application. With an average fooling rate of 95.75% (with 2.02 mean perturbations), the TRPO-MalEAttack lowers the average accuracy of 36 malware detection models from 86.01% to 49.11%. In contrast, the PPO-MalEAttack reduces the average accuracy from 86.01% to 48.65% in the same 36 detection models by achieving a greater average fooling rate of 96.87% (with 2.08 mean perturbations). Additionally, we create a list of the TEN most susceptible Android permissions and intents that an attacker could exploit to create more malicious apps. In order to combat the adversarial attacks on malware detection models, we subsequently suggest a defence strategy (MalVPatch). Higher detection accuracy and a significant increase in the adversarial robustness of malware detection models are both attained by the MalVPatch defence. Lastly, we suggest that in order to attain adversarial supremacy in Android malware detection, it is crucial to examine the adversarial robustness of models prior to their real-world deployment.

5) De-LADY: Deep learning-based Android malware detection using Dynamic features

AUTHORS: V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son

Malicious apps that target the Android operating system have significantly increased in popularity and market share. Since modern malware uses cutting-edge obfuscation techniques to conceal their intentions from scanning engines, traditional malware detection techniques are no longer effective. In this research, we offer an obfuscation robust method called De-LADY (Deep Learning based Android malware detection using DYnamic characteristics). It makes use of the behavioural traits found in dynamic analysis of a program running in a simulated environment. The suggested method is assessed using 13533 applications from various industries, including utilities, gaming, and banking. De-LADY has a 98.84% F-measure and a 98.08% detection rate. Additionally, it performed better than current machine learning techniques.

III. SYSTEM ANALYSIS AND DESIGN

EXISTING SYSTEM:

Equilibrium Optimiser with Deep Learning (IPR-EODL) was used to create the current Android malware detection system. This cutting-edge technology improves Android malware detection and categorisation by combining deep learning methods with the Equilibrium Optimiser algorithm.

An optimisation method inspired by nature, the Equilibrium Optimiser simulates the dynamic and equilibrium states found in real-world systems. The system successfully improves the training procedure by combining this optimiser with deep learning models, guaranteeing more precise and effective malware detection.

The deep learning part of the IPR-EODL technique usually uses sophisticated neural networks, which are good at analysing big data sets and spotting intricate patterns linked to malware activity. A highly dependable malware

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

recognition system is produced by the Equilibrium Optimiser and deep learning models working together to provide robust feature selection and classification.

This current method achieves great performance in identifying malicious software by utilising the Equilibrium Optimizer's optimisation efficiency and deep learning's large data processing capabilities. It is a major advancement in cybersecurity, especially when it comes to shielding Android devices from different types of malware.

DISADVANTAGES OF EXISTING SYSTEM:

- Although the Equilibrium Optimiser with Deep Learning (IPR-EODL) method for identifying Android malware offers notable improvements, there are a number of drawbacks:
- High Computational Complexity: The Equilibrium Optimiser requires a significant amount of processing power to integrate with deep learning models. Combining the optimisation technique with deep neural network training can be a time-consuming and computationally demanding procedure that calls for sophisticated hardware and lengthy training periods.
- Scalability Issues: It can be difficult to scale the system to handle larger datasets or more complicated models because of the high computing needs. This restriction may make it more difficult to use in settings with a variety of virus varieties or quickly expanding data.
- Energy Consumption: A considerable amount of energy is used by the optimisation and deep learning models. For mobile and embedded systems, like Android smartphones, which have constrained computing power and battery life, this is especially problematic.
- Overfitting Risk: When working with inadequate or unbalanced training data, the intricacy of deep learning models raises the possibility of overfitting. Reduced generalisation ability from overfitting can make the model less useful for identifying novel, unseen malware variants.
- Updates and Maintenance: It can take a lot of work to update and maintain the system. The models must be regularly retrained and updated to stay effective when new malware appears. This continuing upkeep necessitates constant resources and skill.
- Interpretability: Deep learning models frequently behave as "black boxes," making it challenging to understand how they make decisions, particularly when paired with optimisation techniques. Security specialists who must comprehend and have faith in the detection procedures may find this lack of transparency to be a disadvantage.
- Latency Issues: The intricate calculations required in the IPR-EODL approach may cause slowness in real-time virus detection. This delay may be harmful in situations where prompt notice and action are essential.
- Ultimately, these drawbacks underscore the need for more effective, scalable, and interpretable cybersecurity solutions, even though the IPR-EODL technique represents a substantial technological leap in malware detection.

PROPOSED SYSTEM:

Machine learning methods are used in the suggested malware analysis and detection system to improve the precision and effectiveness of detecting dangerous software.

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

Python serves as the main scripting language for this system, while the Flask web framework handles backend functions. HTML, CSS, and JavaScript are used in the frontend to provide an intuitive and dynamic user experience.

The Extra Tree Classifier and Logistic Regression are the two main machine learning models used in the suggested system. These models were chosen because they have a track record of success in classification tasks, especially when it comes to differentiating between malicious and safe software.

The TUNADROMD dataset is used in the suggested method to train and evaluate the models. This dataset has 242 attributes and 4465 instances. Its classification goal is to classify the data as either goodware or malware. To maximise the classification process, 23 criteria have been carefully chosen for this project.

The performance of the Extra Tree Classifier and Logistic Regression models is assessed through training and evaluation in the suggested system. Training accuracy for the Extra Tree Classifier is 97.42%, while testing accuracy is 97.23%. 94.84% training accuracy and 93.67% testing accuracy are achieved with the Logistic Regression model.

Out of the 242 attributes in the dataset, a subset of 23 attributes is selected for the proposed system. Reducing the dimensionality of the data, enhancing model performance, and concentrating on the most pertinent attributes for malware detection are the goals of this selection procedure. Model inference and data processing are handled by the Flask-developed backend. With the help of HTML, CSS, and JavaScript, the frontend offers a smooth and user-friendly interface through which users can view results, control the detection procedures, and interact with the system.

All things considered, the suggested system concentrates on using machine learning techniques for malware detection, guaranteeing high accuracy and effective operation through feature engineering and careful model selection.

ADVANTAGES OF PROPOSED SYSTEM:

- High Accuracy: With the Extra Tree Classifier achieving a training accuracy of 97.42% and a testing accuracy of 97.23%, the suggested system demonstrated great malware detection accuracy. Because of its high accuracy, dangerous software may be reliably detected, lowering the number of false positives and false negatives.
- Efficiency in Detection: By choosing just 23 pertinent characteristics from the TUNADROMD dataset, the suggested solution lowers processing time and computational complexity, resulting in quicker malware identification and classification.
- User-Friendly Interface: The user interface is responsive and easy to use thanks to the frontend integration of HTML, CSS, and JavaScript. Users no longer need in-depth technical expertise to engage with the system, monitor results, and control malware detection procedures thanks to this architecture.
- Scalability: The Flask web framework and Python were used in the development of this extremely scalable system. It can be easily adapted and expanded to accommodate larger datasets, more features, or additional machine learning models, making it suitable for evolving cybersecurity needs.
- Flexibility in Model Use: The system offers versatility in approach by utilising both the Logistic Regression and Extra Tree Classifier models. Whether they prefer linear models or tree-based approaches, users can select the model

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

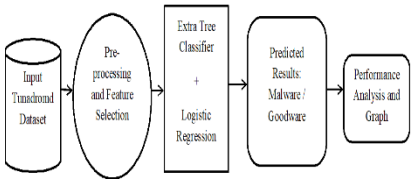
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

that best suits their unique needs or situation.

- Reduced Overfitting: The models' ability to generalise is improved by the careful selection of 23 attributes, which reduces the possibility of overfitting. This guarantees that the system operates effectively on both fresh, unseen data and the training data.
- Integration Capabilities: The suggested system can be integrated into larger cybersecurity infrastructures or merged with other security tools thanks to the Flask framework's simple integration capabilities.
- Maintenance and Update Efficiency: Maintenance and upgrades are made easier by the system's modular architecture, which clearly distinguishes between frontend presentation and backend processing. Improvements to the user interface or machine learning models can be made with little impact on the system as a whole.
- Transparency and Interpretability: In particular, the application of logistic regression provides a certain amount of interpretability, enabling users to comprehend the model's decision-making process. Gaining trust and enabling well-informed judgements in cybersecurity operations depend heavily on this transparency.
- Resource Optimization: The suggested approach maximises resource utilisation through the use of computationally efficient machine learning methods. This enables deployment across a range of platforms, including those with low processing capability, like mobile devices.
- All things considered, the suggested system offers a strong and adaptable malware detection solution that combines excellent accuracy, effectiveness, and user-friendliness with the capacity to scale and adjust to meet new cybersecurity threats.

SYSTEM ARCHITECTURE



IV. IMPLEMENTATION

MODULES:

- Data Collection
- Dataset
- Data Preparation
- Feature Extraction
- Splitting the dataset
- Model Selection
- Analyze and Prediction
- Accuracy on test set
- Saving the Trained Model
- Prediction Module
- Model Evaluation Module

MODULES DESCSRIPTION:

Data Collection:

- In the Malware Analysis and Detection Using Machine Learning Algorithm's first module, we create the data gathering procedure. Gathering information is the first actual step in creating a machine learning model. The more and better data we obtain, the better our model will perform, therefore this is a crucial phase that will cascade into how good the model will be.
- Data can be gathered using a variety of methods, including manual interventions and web scraping. The model folder contains the dataset. The dataset is sourced from Kaggle, a well-known

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

dataset resource. The dataset can be found at the following website:

- Kaggle Dataset Link: https://www.kaggle.com/datasets/jayaprakashpondy/android-malware

Dataset:

- The dataset, which serves as the system's main data source, is used in this module. This dataset includes 242 attributes and 4465 occurrences, with malware vs. goodware as the goal attribute for classification.
- User Input: Real-time malware detection based on user-uploaded files or input data is made possible by data supplied by users via the web interface.

Data Preparation:

- The TUNADROMD dataset must be prepared for analysis by this module. It entails activities including feature selection, normalisation, and data cleansing. To optimise the machine learning models, 23 pertinent features are specifically chosen from the initial 242 attributes.
- Sort data and get it ready for training. Remove duplicates, fix mistakes, deal with missing numbers, normalise, convert data types, and clean up everything that could need it.
- Data should be randomised to eliminate the impact of the specific sequence in which they were gathered and/or otherwise prepared.
- Use data visualisation to find pertinent correlations between variables, identify class imbalances (beware of bias), or carry out additional exploratory analysis.
- 'Begnin' is changed to 'Benign' when the labels are cleaned to fix spelling mistakes.
- The dataset's NaN values are removed..
- The RandomOverSampler from imblearn is used to balance the data in order to address the class imbalance between "Malware" and "Benign" apps.

Feature Extraction:

- Extract characteristics that the machine learning models can use if the dataset includes raw binary data or other non-numeric data. This could entail dynamic analysis (such as tracking the binary's behaviour while it is running) or static analysis (such as examining the binary's structure).
- For model training, a subset of features (permissions) is chosen in order to minimise dimensionality and concentrate on pertinent attributes.

Splitting the dataset:

- For the model to be trained and assessed, data splitting and validation are essential. The dataset is separated into training, validation, and testing sets by this module. It guarantees that appropriate validation methods, such as cross-validation, are used to appropriately evaluate the model's performance. Divide the dataset into test and train sets. 20% is test data and 80% is train data..

Model Selection:

- This module uses the preprocessed data to train the machine learning models. It uses the methods for Logistic Regression and Extra Tree Classifier.

Extra Tree Classifier:

- Extremely Randomised Trees, another name for the Extra Tree Classifier, is an ensemble learning technique that is mostly applied to classification and regression problems. It is implemented in the ExtraTreesClassifier module of the sklearn package and belongs to the

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448
Vol. 21, Issue 2, 2025

larger family of decision tree approaches.

- A forest of randomised decision trees is created by the Extra Tree Classifier. Extra Trees chooses the split points at random, in contrast to typical decision trees that choose the optimum feature threshold combination to minimise a criterion such as information gain or Gini impurity. There are two primary ways that this unpredictability is introduced:
- Split points are produced at random for every feature..
- For the split, a random selection of features is chosen.
- By averaging several trees, this produces a significant variance reduction that improves the model's resilience and guards against overfitting.

Logistic Regression:

- A basic statistical method for binary classification problems is logistic regression. It is a classification algorithm, not a regression algorithm, as its name suggests. The likelihood that a given input belongs to a specific class is estimated by logistic regression. It is used in the LogisticRegression module of the sklearn package.
- The logistic function, sometimes referred to as the sigmoid function, is used in logistic regression to represent the relationship between the dependent variable (the target) and one or more independent variables (the features). Any real integer can be mapped into the range (0, 1) by the sigmoid function, which can then be understood as a probability.
- For binary classification applications, logistic regression is a flexible and effective technique that strikes a balance between ease of use and efficiency. Its simplicity of interpretation and effective handling of big datasets make it popular. It can be modified to prevent overfitting and function well in a variety of situations by implementing regularisation approaches.

Analyze and Prediction:

- To find the most important characteristics for malware categorisation, this module selects features. It guarantees that the chosen features make a substantial contribution to the model's functionality.

Accuracy on test set:

- Following training, the model's performance must be assessed. In this module, the dataset is divided into training and testing subsets, and the model's accuracy, precision, recall, and F1-score are evaluated.
- The evaluation metrics shed light on how well the system anticipates and identifies Android malware. Training accuracy for the Extra Tree Classifier is 97.42%, while testing accuracy is 97.23%. Training accuracy for the Logistic Regression model is 94.84%, and testing accuracy is 93.67%.

Saving the Trained Model:

- The first step is to store your trained and tested model into an.h5 or.pkl file using a library like Pickle after you're comfortable enough to move it into a production-ready environment.
- Ensure that pickles are installed in your surroundings..
- Next, let’s import the module and dump the model into .pkl file.

Prediction Module:

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

- This module handles real-time predictions using the trained models. Users can input new data through the frontend, and the module processes this data to classify it as malware or benign.

Model Evaluation Module

- This module uses the testing dataset to assess how well the trained models perform. To evaluate the efficacy of the model, it computes accuracy metrics and other performance indicators.
- Analyse the model's F1-score, recall, accuracy, and precision.
- For each models, create confusion matrices.
- Examine the differences in performance between the Logistic Regression and Extra Tree Classifier models.
- Model performance is assessed using F1-score, recall, accuracy, and precision.
- Seaborn Heatmap is used to visualise the confusion matrix in order to comprehend the categorisation results.

ALGORITHMS

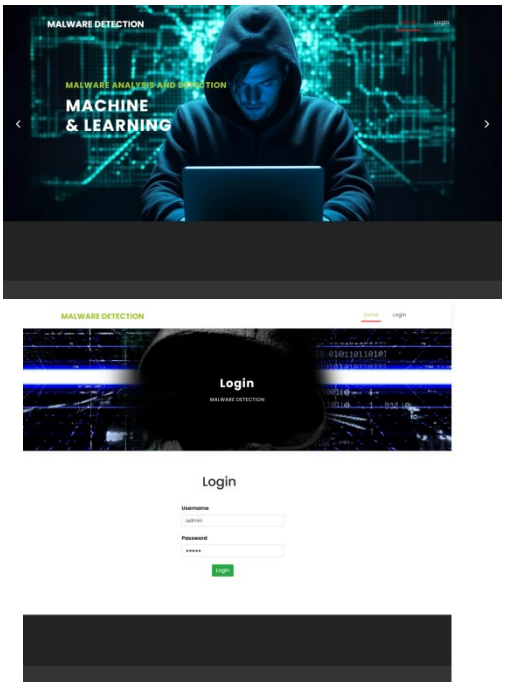
Logistic regression Classifiers

The relationship between a set of independent (explanatory) variables and a categorical dependent variable is examined using logistic regression analysis. When the dependent variable simply has two values, like 0 and 1 or Yes and No, the term logistic regression is applied. When the dependent variable has three or more distinct values, such as married, single, divorced, or widowed, the technique is sometimes referred to as multinomial logistic regression. While the dependent variable's data type differs from multiple regression's, the procedure's practical application is comparable.

When it comes to categorical-response variable analysis, logistic regression and discriminant analysis are competitors. Compared to discriminant analysis, many statisticians believe that logistic regression is more flexible and appropriate for modelling the majority of scenarios. This is due to the fact that, unlike discriminant analysis, logistic regression does not presume that the independent variables are regularly distributed.

Both binary and multinomial logistic regression are calculated by this program for both category and numerical independent variables. Along with the regression equation, it provides information on likelihood, deviance, odds ratios, confidence limits, and goodness of fit. It does a thorough residual analysis that includes diagnostic residual plots and reports. In order to find the optimal regression model with the fewest independent variables, it might conduct an independent variable subset selection search. It offers ROC curves and confidence intervals on expected values to assist in identifying the optimal classification cutoff point. By automatically identifying rows that are not used throughout the study, it enables you to validate your findings.

V. SCREEN SHOTS

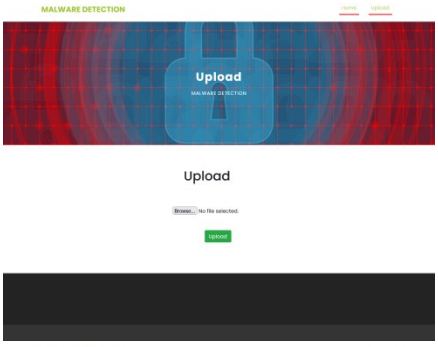


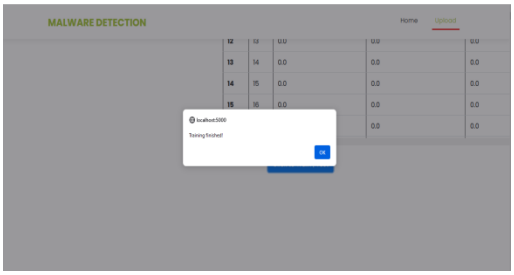
International Journal of Engineering Research and Science & Technology

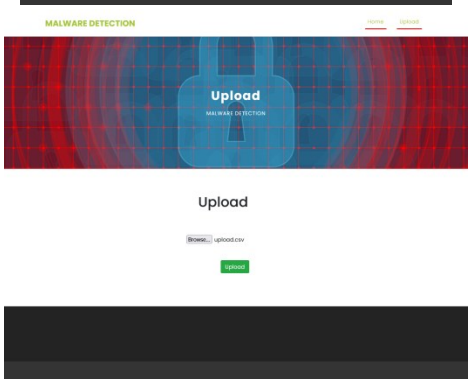
ISSN 2319-5991 www.ijerst.com

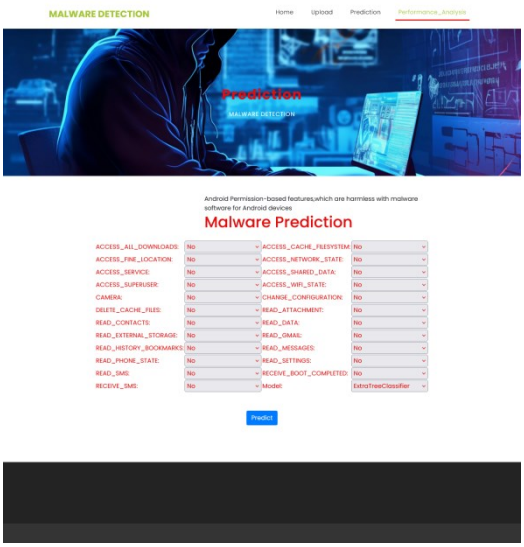
https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

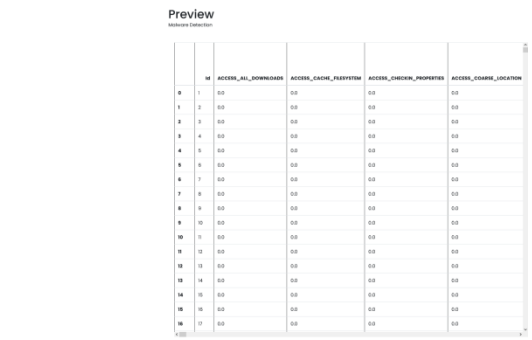
Vol. 21, Issue 2, 2025

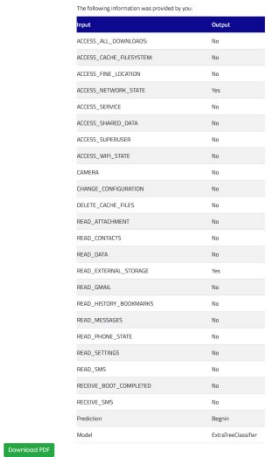












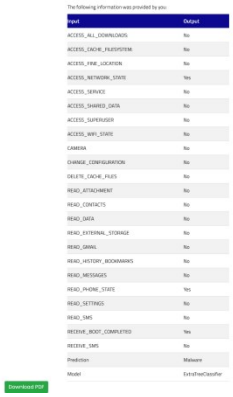
International Journal of Engineering Research and Science & Technology

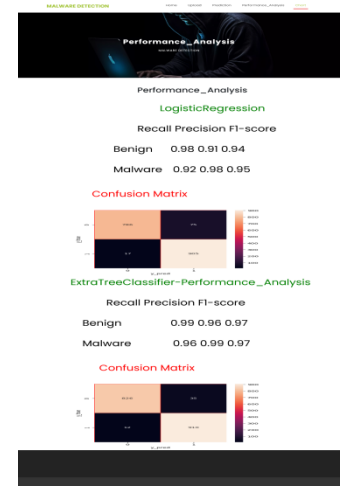
ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

Result





VI. CONCLUSION

The project "Malware Analysis and Detection Using Machine Learning Algorithm" shows how well sophisticated machine learning methods can be applied to the vital task of malware identification and classification. The system uses Flask, Python, and a combination of Logistic Regression and Extra Tree Classifier models to obtain high accuracy rates of 93.67% and 97.42% on test data, respectively.

The approach successfully lowers computational complexity while preserving strong detection skills by utilising the TUNADROMD dataset and carefully choosing 23 pertinent features. A thorough and effective approach to malware detection is ensured by the system's modular architecture, which includes data pretreatment, model training, evaluation, backend integration, and an intuitive frontend.

The promise of ensemble learning techniques in cybersecurity applications is highlighted by the Extra Tree Classifier's exceptional performance. In the meanwhile, the Logistic Regression model offers a useful baseline for comparison, highlighting the advantages and disadvantages of various machine learning techniques.

All things considered, the project effectively combines machine learning with useful application frameworks to produce a potent malware detection solution. In order to create computing environments that are more safe and resilient, our work emphasises the significance of fusing technology developments with thoughtful feature selection and system design.

FUTURE WORK:

- Building on the existing system's successful deployment and encouraging outcomes, there are a number of directions for future research that could improve malware analysis and detection even more.:
- Adding More Machine Learning Models: Increasing the variety of machine learning algorithms, such as sophisticated methods

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

like Support Vector Machines, Neural Networks, and Gradient Boosting, may increase the precision and resilience of detection.

- The system can continuously scan and identify malware in real-time scenarios by integrating real-time monitoring and detection features. This gives the system immediate protection against new threats.
- The implementation of adaptive learning algorithms guarantees that the system stays current and effective against the most recent malware variants by enabling the models to automatically update and improve depending on new data and detected threats.
- Enhanced Feature Engineering: Investigating cutting-edge feature engineering methods, like deep feature learning and automated feature extraction, to find hidden patterns and connections in the data that could improve the models' precision and effectiveness.
- The system's capacity to identify and react to a wider range of malicious activity is enhanced by the integration of external threat intelligence feeds, which supplement the dataset with real-world malware samples and threat indicators.
- User Behaviour Analysis: By using behavioural insights to identify deviations from normal user activity patterns, user behaviour analysis can be used to identify abnormalities and possible threats, hence enhancing security.
- Scalability and Deployment: To guarantee scalability and smooth integration with current cybersecurity infrastructures, the system is optimised for deployment in expansive and varied contexts, such as corporate networks and cloud-based platforms.
- Comprehensive Reporting and Analytics: Creating sophisticated reporting and analytics tools that support proactive threat management and decision-making for cybersecurity professionals by offering in-depth insights, trend analysis, and predictive analytics.
- Improved User Interface: Constantly enhancing the user interface to guarantee accessibility, usability, and improved visualisation capabilities will help people engage with and comprehend the system's outputs more successfully.
- The ability to work across platforms: expanding the system's compatibility to include more platforms, such as mobile devices and multiple operating systems, in order to provide thorough security in a variety of user scenarios.
- Future research in these areas can make the malware analysis and detection system even more resilient, flexible, and equipped to handle the changing demands of the cybersecurity industry..

REFERENCES

1. A. Gómez and A. Muñoz, "Deep learning-based attack detection and classification in Android devices", Electronics, vol. 12, no. 15, pp. 3253, Jul. 2023.
2. Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, et al., "On the impact of sample duplication in machine-learning-based Android malware detection", ACM Trans. Softw. Eng. Methodol., vol. 30, no. 3, pp. 1-38, Jul. 2021.
3. H. Wang, W. Zhang and H. He, "You are what the permissions told me! Android malware detection based on hybrid tactics", J. Inf. Secur. Appl., vol. 66, May 2022.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

4. H. Rathore, A. Nandanwar, S. K. Sahay and M. Sewak, "Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses", Forensic Sci. Int. Digit. Invest., vol. 44, Mar. 2023.
5. V. Sihag, M. Vardhan, P. Singh, G. Choudhary and S. Son, "De-LADY: Deep learning-based Android malware detection using Dynamic features", J. Internet Serv. Inf. Secur., vol. 11, no. 2, pp. 34-45, 2021.
6. M. Ibrahim, B. Issa and M. B. Jasser, "A method for automatic Android malware detection based on static analysis and deep learning", IEEE Access, vol. 10, pp. 117334-117352, 2022.
7. A. Albakri, F. Alhayan, N. Alturki, S. Ahamed and S. Shamsudheen, "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification", Appl. Sci., vol. 13, no. 4, pp. 2172, Feb. 2023.
8. A. Batouche and H. Jahankhani, "A comprehensive approach to Android malware detection using machine learning", Information Security Technologies for Controlling Pandemics, pp. 171-212, 2021.
9. P. Bhat and K. Dutta, "A multi-tiered feature selection model for Android malware detection based on feature discrimination and information gain", J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 10, pp. 9464-9477, Nov. 2022.
10. L. Hammood, I. A. Dogru and K. Kiliç, "Machine learning-based adaptive genetic algorithm for Android malware detection in auto-driving vehicles", Appl. Sci., vol. 13, no. 9, pp. 5403, Apr. 2023.
11. R. Raphael and P. Mathiyalagan, "Intelligent hyperparameter-tuned deep learning-based Android malware detection and classification model", J. Circuits Syst. Comput., vol. 32, no. 11, Jul. 2023.
12. M. N. AlJarrah, Q. M. Yaseen and A. M. Mustafa, "A context-aware Android malware detection approach using machine learning", Information, vol. 13, no. 12, pp. 563, Nov. 2022.
13. O. N. Elayan and A. M. Mustafa, "Android malware detection using deep learning", Proc. Comput. Sci., vol. 184, pp. 847-852, Jan. 2021.
14. P. Yadav, N. Menon, V. Ravi, S. Vishvanathan and T. D. Pham, "A two-stage deep learning framework for image-based Android malware detection and variant classification", Comput. Intell., vol. 38, no. 5, pp. 1748-1771, Oct. 2022.
15. P. Yadav, N. Menon, V. Ravi, S. Vishvanathan and T. D. Pham, "EfficientNet convolutional neural networks-based Android malware detection", Comput. Secur., vol. 115, Apr. 2022.
16. K. Shaukat, S. Luo and V. Varadharajan, "A novel deep learning-based approach for malware detection", Eng. Appl. Artif. Intell., vol. 122, Jun. 2023.
17. A. Desnos and G. Gueguen, "Androguard documentation", 2018.
18. C. Chen, J. Wei and Z. Li, "Remaining useful life prediction for lithium-ion batteries based on a hybrid deep learning model", Processes, vol. 11, no. 8, pp. 2333, Aug. 2023.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp432-448

Vol. 21, Issue 2, 2025

19. C. Zhong, G. Li, Z. Meng, H. Li and W. He, "A self-adaptive quantum equilibrium optimizer with artificial bee colony for feature selection", Comput. Biol. Med., vol. 153, Feb. 2023.
20. Andro-AutoPsy, Jul. 2023, [online] Available: https://ocslab.hksecurity.net/andro-autopsy.