International Journal of Engineering Sciences & Emerging Technologies, Dec. 2024
ISSN: 22316604 Volume 11, Issue 4, pp: 710-721 ©IJESET

ANDROID MALWARE DETECTION USING MACHINE LEARNING

Amay Bhatnagar[1], Mahendra Singh Sagar[2], Priyanka Goel[3],
Yamini Yadav[4], Sherine K. Davasia[5]

Department of Computer Science Engineering, M.I.T Moradabad, Moradabad, India
Amaybhatnagar13@gmail.com
mahendra.singh12jan@gmail.com
priyanka070goel@gmail.com
Yaminiyadav583@gmail.com
davsherine@gmail.com

ABSTRACT

Android operating systems have become a prime target for malware due to their widespread usage. Traditional antivirus software struggles to keep up with the rapid evolution of malware, necessitating more dynamic and adaptive solutions. This paper presents a novel approach to Android malware detection using machine learning techniques. We propose a system that extracts meaningful features from Android applications and employs various machine learning algorithms to classify these applications as benign or malicious. Our system leverages the strengths of both static and dynamic analysis for feature extraction. The machine learning models are trained and tested using a large dataset of both benign and malicious application. Various classification techniques such as Logistic Regression, Decision Tree, and Naïve Bayes applied to test the efficiency of classification machine learning algorithms. Experimental results demonstrate that our system can effectively detect Android malware with high accuracy, precision, and recall. This research contributes to the ongoing efforts in cyber security, providing a robust and scalable solution to Android malware detection. Quantitative metrics including accuracy, precision, and computational efficiency are rigorously measured and analysed. Results demonstrate good accuracy rates across the respective dataset. In conclusion, Android Malware detection using Machine learning approach that compares various classification detection techniques that are logistic regression, decision tree, Naïve Bayes to provide comprehensive protection against evolving threats and tell us about the performance of those algorithms respectively.

KEYWORDS— *Classification, Logistic Regression, Dtree, Naïve Bayes, Machine Learning, Malware*

I. INTRODUCTION

The threat of malware attacks has grown significantly as the use of mobile devices, notably Android smartphones, has increased.

Apps may contain hidden malicious software that might damage the device or steal critical user data. Conventional

Security measures and anti-virus software have limitations when it comes to identifying and stopping these threats, which are getting more complex. Machine learning has emerged as a viable strategy for Android malware prediction in response to this difficulty. Machine learning algorithms can examine a variety of app properties and traits to spot possibly dangerous activity. This can assist in finding malware that more conventional methods would not have been able to find.

Despite the growing threat of malware, there is still no reliable and robust method for detecting malicious applications. However, with the increasing use of machine learning in various fields, we believe that this issue can be addressed through the application of machine learning techniques. Our project aims to conduct a thorough and systematic investigation into the use of machine learning for malware detection, with the ultimate goal of developing an efficient ML model capable of accurately classifying apps as either benign (0) or malware (1) based on their requested permissions. This study Proposes:Conducting

an in-depth examination and evaluation of Android metadata and permissions as predictors of malware and Introducing a machine learning-based malware detection strategy that utilizes publicly available metadata information.

By training models on diverse sets of features extracted from Android applications, such as permissions, API calls, and code structures, ML can learn to distinguish between benign and malicious software effectively. This introduction sets the stage for exploring how ML can revolutionize Android malware detection, offering proactive defenses against evolving threats in the mobile ecosystem.

The benefits of employing ML for Android malware detection are manifold. Firstly, it offers unparalleled adaptability, constantly learning and evolving alongside the ever-shifting threat landscape. This agility ensures that even zero-day attacks, previously unknown threats, can be effectively identified and neutralized. Secondly, ML empowers automated detection, significantly streamlining the process compared to manual analysis, thus improving efficiency and responsiveness. Finally, ML algorithms can be fine-tuned for optimal performance on resource-constrained mobile devices, ensuring smooth operation without hindering user experience.

However, implementing ML for Android malware detection is not without its challenges. One primary hurdle is the scarcity of high-quality training data. Labelling vast datasets of apps as benign or malicious requires significant time and resources. Additionally, model interpretability can be an issue, as understanding why a particular ML model classifies an app as malicious can be challenging. Finally, there is a delicate balance to strike between detection accuracy and performance, ensuring robust protection without compromising user experience on mobile devices.

Despite these challenges, the potential of ML for Android malware detection is undeniable. Driven by continuous research and development, these challenges are being actively addressed, paving the way for a future where ML forms the cornerstone of our mobile security defences. As the mobile landscape continues to evolve, so too will our fight against malware. Machine learning, with its adaptability and resilience, stands as a powerful ally in this ongoing battle, safeguarding our devices and the data they hold in the face of ever-evolving threats.

ML algorithms, trained on vast datasets of both benign and malicious applications, have the remarkable ability to identify subtle patterns and relationships that are invisible to traditional methods. By analyzing app permissions, code structure, network activity, and even user reviews, these algorithms can learn to distinguish legitimate apps from their malicious counterparts with remarkable accuracy, even when faced with previously unseen malware variants. This adaptability and resilience are crucial in the ever-evolving battle against cyber threats.

II. LITERATURE REVIEW

The proliferation of Android devices has unfortunately witnessed a concomitant rise in malicious applications (malware). To address this escalating threat, researchers have increasingly focused on leveraging the capabilities of Machine Learning (ML) for robust and efficient detection methodologies. Extensive literature analysis reveals the effectiveness of diverse ML algorithms, including decision trees and neural networks, in identifying intricate patterns within features extracted from Android applications. Notably, while feature extraction often utilizes static code analysis of permissions and API calls, dynamic behavior

While each analysis approach has its merits and limitations, researchers often employ a combination of techniques for a more comprehensive and robust detection strategy. Static analysis, examining the app's code and manifest file without execution is efficient and scalable but might miss malware that relies on dynamic behavior. Dynamic analysis, running the app in a controlled environment, can uncover complex malware but can be computationally expensive and time-consuming. Hybrid analysis combines both approaches, offering a more balanced and effective solution.

Despite the promising results achieved by ML-based approaches, several challenges remain. The ever-evolving landscape of Android malware, with new variants constantly emerging and employing novel evasion techniques, necessitates adaptable and generalizable models capable of identifying unseen threats. Additionally, datasets used for training ML models often suffer from imbalanced class distribution, where the number of benign apps significantly outnumbers malicious ones. This imbalance can bias the model towards the majority class, leading to poor detection of rare malware samples. Addressing this issue through data augmentation techniques or employing appropriate cost-sensitive

learning algorithms is crucial. Finally, ensuring the explainability and interpretability of ML models is essential for building trust and improving transparency, particularly in security-critical applications like malware detection.

The evolving nature of malware, resource constraints on mobile devices, and the increasing need for interpretable models necessitate continuous research and development. Although Deep Learning presents promising avenues with its advanced feature learning capabilities, it requires further investigation and necessitates substantial computational resources. In conclusion, ML has established itself as a powerful tool for Android malware detection. However, continuous advancement remains paramount to remain vigilant in the face of evolving threats.

III. HARDWARE AND SOFTWARE EMPLOYED

A. Hardware Platform

The Chatbot has been completely made on the Windows OS based machine.The Specifications are as follows:

- Processor: i5 11th gen Hexa-core @2.7ghz
- Memory: 16gb Ram @3200mhz
- GPU: NVidia GeForce RTX 3050 4gb

B. Sofware Platform

The Following software, libraries & techniques are used in building this chatbot.

- Programmed Using: Python
- LibrariesImported: Sklearn,matplotlib,seaborn,numpy,pandas
- Dataset: Kaggle
- IDE: Jupyter Notebook

IV. ABOUT THE LIBRARIES & TECHNIQUES

This Project uses various types of libraries and various data cleaning and mining techniques that help us in optimal threat detection.

- SKLEARN: The scikit-learn (sklearn) library is a powerful tool in the field of machine learning, offering a comprehensive suite of algorithms and utilities for data pre-processing, modelling, and evaluation. Developed in Python and built on NumPy, SciPy, and Matplotlib, sklearn provides a user-friendly interface for implementing various machine learning techniques, making it accessible to both beginners and experts alike.
- MATPLOTLIB is a versatile Python library widely used for creating static, interactive, and animated visualizations. It offers a straightforward interface for generating various types of plots, including line plots, scatter plots, histograms, bar charts, and more.. With Matplotlib, users can control elements such as colours, labels, fonts, annotations, and styles, enabling the creation of publication-quality figures.
- NUMPY: short for Numerical Python, is a fundamental Python library for numerical calculations and works on the principle of matrix and arrays formation. NumPy's primary data structure is the nparray, which enables fast and memory-efficient manipulation of numerical data. With NumPy, users can perform various array operations, including arithmetic, logical, statistical, and linear algebraic operations.
- PANDAS is a powerful Python library widely used for data manipulation and analysis. It provides high-level data structures, primarily the Data Frame, which allows users to work with labelled and structured data easily. Similar to a spreadsheet or SQL table, a Data Frame consists of rows and columns with customizable index and column labels. Pandas offers a rich set of functions and methods for reading, writing, cleaning, transforming, and analysing data stored in various

formats, including CSV, Excel, SQL databases, and JSON. Users can perform operations such as filtering, sorting, grouping, merging, and joining data efficiently using Pandas.

- SEABORN is a Python data visualization library based on that provides us different tools for creating well-mannered visual graphs and graphics. It is a subset of matplotlib and integrates well with data structures and pandas.

V. SYSTEM OVERVIEW

An Android malware detection system using machine learning (ML) typically consists of several key components designed to detect and mitigate the risks posed by malicious software on Android devices. Here's an overview of the system:

1). **Data Collection & Pre-processing**:

- Raw data collection involves gathering various attributes and features from Android applications, such as permissions, API calls, code snippets, and metadata.
- Pre-processing steps involve cleaning and transforming the collected data into a format suitable for ML model training. This may include feature extraction, normalization, and handling missing values.

2). **Feature Engineering:**

- Feature engineering plays a crucial role in determining the effectiveness of the ML models. Relevant features extracted from the collected data are selected or engineered to capture meaningful patterns and behaviours indicative of malware.
- Features may include static features extracted from the application's binary files (e.g., permissions, API calls), dynamic features obtained during runtime analysis (e.g., system calls, network traffic), and metadata features (e.g., app category, installation source).

3). **Machine Learning Models**:

- Various ML algorithms are trained on the extracted features to learn patterns distinguishing between benign and malicious applications.
- Commonly used ML algorithms include decision trees, random forests, logistic regression, naïve bayes,kmeans.
- Model selection and optimization are performed to improve detection accuracy, generalization, and efficiency.

4). **Training and Evaluation**:

- The ML models are trained on labeled datasets consisting of both benign and malicious applications.
- Training involves splitting the dataset into training and validation sets, tuning hyperparameters, and optimizing the models to achieve the desired performance metrics.
- Evaluation is performed using separate testing datasets to assess the models' performance in terms of metrics such as accuracy, precision, recall, F1-score, and ROC.

5). **Deployment and Integration:**

- Once trained and evaluated, the ML models are deployed within an Android malware detection system, either on-device or on a remote server. Integration with existing security solutions or mobile device management (MDM) systems may be necessary for real-time monitoring and enforcement.

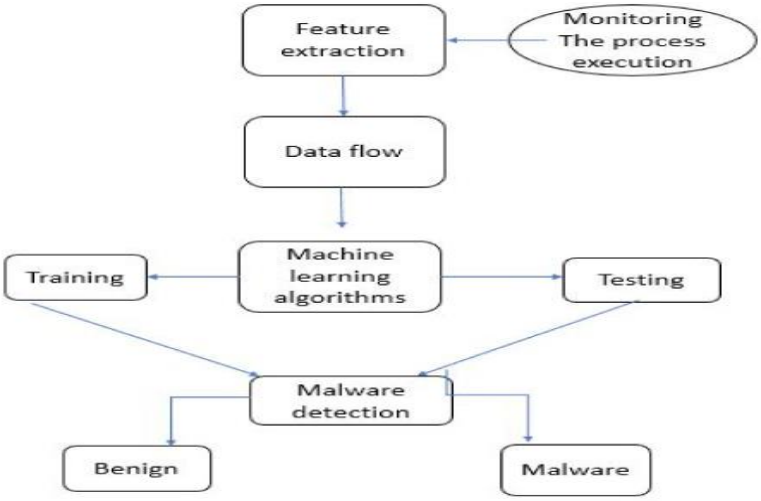

Fig:-1.0 System Overview

VI. PROPOSED METHODOLOGY

1). **Data Acquisition and Preprocessing:**

- **Source:** Collect data from diverse sources like app stores, user submissions, and malware repositories.

2).**Pre-processing:**

- **Decompile:** Convert apps to readable code format for feature extraction.
- **Feature Engineering:** Extract relevant features like:
  - **Staticfeatures:** App permissions, API calls, strings, code structure, etc.
  - **Dynamic features:** Network activity, resource usage, system calls, etc.
  - **Metadata:** App size, developer info, user reviews, etc.
- **Normalization:** Scale features to a common range for better model performance.

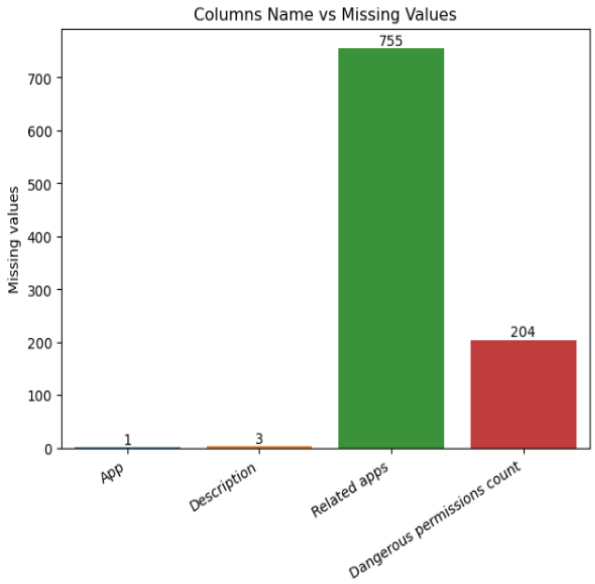

Fig:- 2.0 Check Missing Values

- Will be replacing missing values with mean of that respective column.

```
def replace_mean(x):
    if x.dtype!='object':
        return x.fillna(x.mean())
    return x
```

Fig:-3.0 Replacing missing values

- Checking the app rating distribution using histplot will help us know user review of the apps in given dataset as we know more less review means more malware infected apps are.

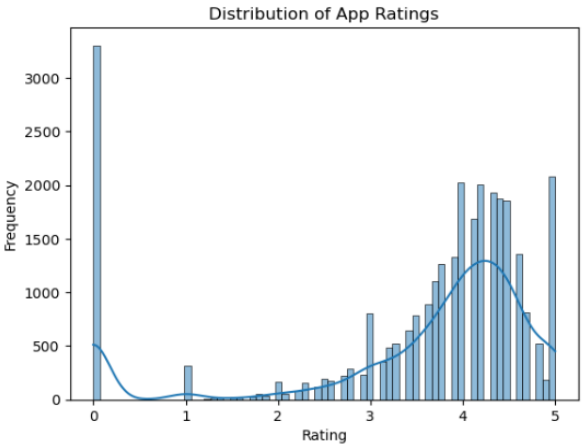

Fig:- 4.0 App Ratings Distribution

- For the features selected we will be evaluating them using various visualization methods. Like below we have checked various features correlation using various graphs and plots.

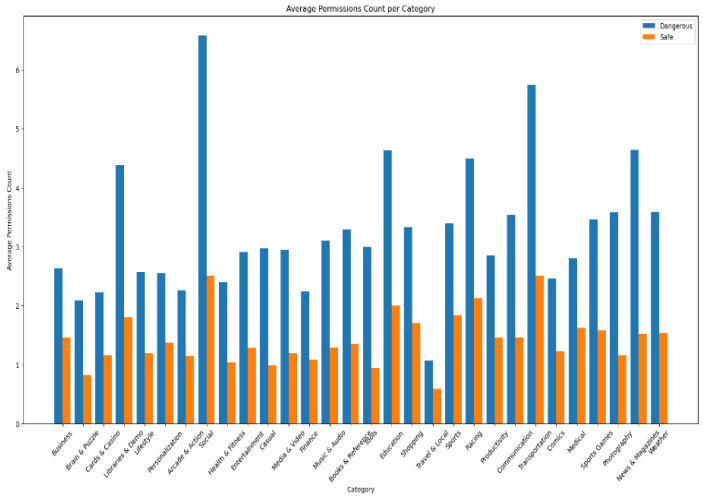

Fig 5.0 Permissions w.r.t app category

3).Machine Learning Pipeline:

3.1Feature Selection: Use techniques like correlation analysis to identify the most informative features. Using correlation heatmap to find the most appropriate features.

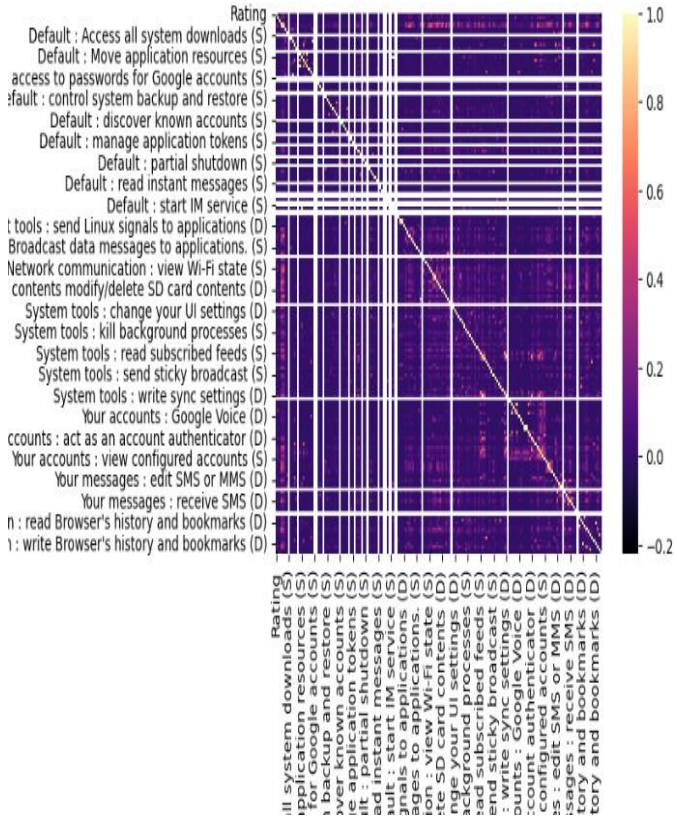

Fig:-6.0 Correlation heatmap

3.2. Model Training:

- **Decision Tree (DT):** Train a DT model to learn decision rules for classifying apps. This model is interpretable and good for understanding feature importance.
- **Logistic Regression (LR):** Train an LR model to estimate the probability of an app being malicious. This model is efficient and works well with binary classification.
- **Naive Bayes (NB):** Train an NB model based on conditional independence assumptions between features. This model is simple and efficient but may not capture complex relationships.

4).Model Selection and Ensemble:

- Evaluate each model on separate validation data based on metrics like accuracy, precision, recall, and F1-score.
- Consider ensemble methods like voting or stacking to combine the strengths of different models and potentially improve performance.
-

5).Feedback Loop:

1. Continuously collect user reports and expert analysis on flagged apps.
2. Use this feedback to update the training data and retrain the models for improved accuracy and adaptability to new threats.

Challenges:

- **Data quality and quantity:** Requires large, diverse, and well-labelled datasets for effective training.

- **Model interpretability:** Balancing complex models with the need to understand their decision-making.
- **Performance optimization:** Ensuring efficient resource usage on mobile devices.

This is just one possible approach, and the specific methodology may need to be adapted based on your specific requirements and data availability.

VII. ALGORITHMS ANALYSIS TECHNIQUES

Result Analysis for Android Malware Detection System will done using various performance metrics.

1) Training Accuracy:

- Training accuracy help us to measure the training data set accuracy with respect to total features. It is basically no of correct predictions by total number of instances.
- Training instance help us to know if our model is optimal for dataset or not

2). Testing Accuracy:

- Training accuracy measures the proportion of correctly classified instances in the training dataset by the model. It is basically total correct prediction by total number of data features
- While training accuracy gives insights on how well the model fits the training data, but not the model's performance on unseen data.

3) ROC:

- The Receiver Operating Characteristic (ROC) curve is a visual representation between the true positive rate and the false positive rate for different threshold values.
- The ROC score quantifies the model's discriminative power across all possible threshold values. It is calculated as the area under the ROC curve. A higher ROC score indicates better discriminative performance of the model

4) Recall Score:

- A recall score of 1.0 indicates that the model correctly identifies all relevant instances of the positive class (i.e., no false negatives), while a score of 0.0 indicates that the model fails to identify any relevant instances of the positive class.
- Higher recall values are desirable, especially in applications where it is crucial to minimize false negatives, such as medical diagnoses or fraud detection.

5) Confusion Matrix:-

- A confusion matrix is a tabular representation that is commonly used in the evaluation of machine learning models, particularly in binary classification tasks. It provides us details of the classification model by displaying the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

VIII. RESULT ANALYSIS

Analyzing the classification algorithms in Android malware detection involves comparing and evaluating the performance of machine learning algorithms in distinguishing between benign and malicious applications. This analysis aims to identify the most effective algorithm or combination of algorithms for accurately detecting malware on Android devices.

1) Classification Algorithms Analysis:-

1.1 Logistic Regression

```
logistic(X_train, X_test, y_train, y_test)
```

```
Training Accuracy 0.58
Test Accuracy 0.50
Recall Score 1.00
ROC Score 0.50
```

```
LogisticRegression
LogisticRegression(max_iter=200, n_jobs=-1)
```

Fig:- 7.0 Logistic Regression

1.2 Naïve Bayes

```
Naive(X_train, X_test, y_train, y_test)
```

```
Training Accuracy 0.58
Test Accuracy 0.50
Recall Score 1.00
ROC Score 0.50
```

```
GaussianNB
GaussianNB()
```

Fig:-8.0 Naïve Bayes

1.3 Decision Tree

```
reg = dtree(X_train, X_test, y_train, y_test)
```

```
Training Accuracy 0.93
Test Accuracy 0.75
Recall Score 1.00
ROC Score 0.83
```

Fig 9.0:- Dtree

2) Confusion Matrix:-

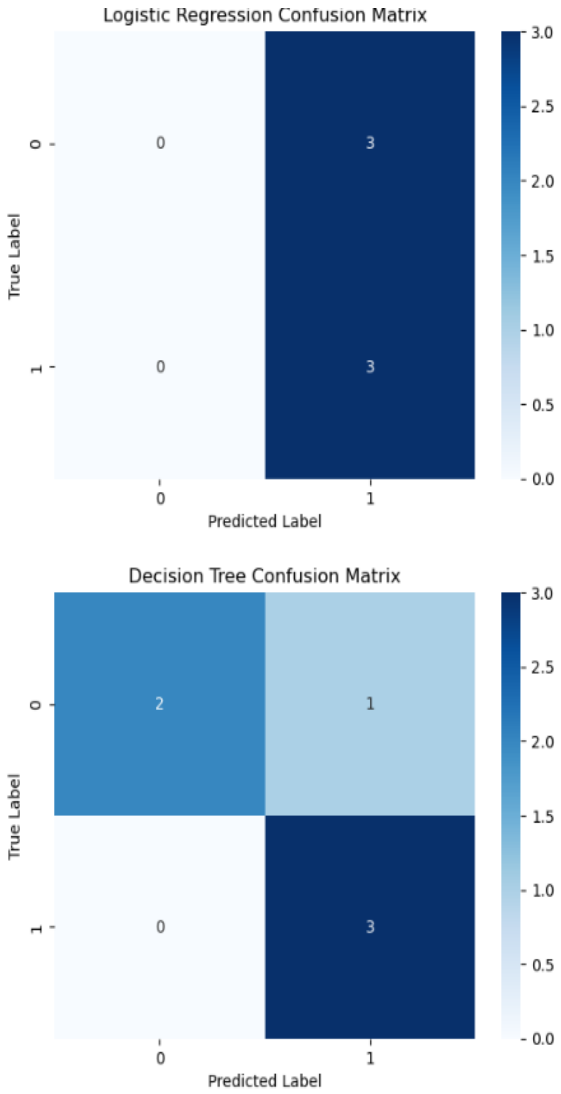

Fig 11:- Confusion Matrix

IX. CONCLUSION

From Implementing the following machine learning classification techniques we can say that **Decision Tree > Logistic Regression > Naïve Bayes.**

Within the domain of Android malware detection, decision tree algorithms have demonstrated advantages over logistic regression and naive Bayes approaches. This superiority stems from several key factors. Firstly, malware frequently exhibits intricate patterns within its permissions, code structure, and network behavior. Decision trees excel at capturing these non-linear relationships, allowing for a more nuanced understanding of malicious applications. Conversely, logistic regression is restricted to linear models, potentially overlooking crucial complexities. Similarly, naive Bayes suffers when features are interdependent, as it presupposes independence between them.

This assumption can be particularly detrimental in malware detection, where various indicators often work in concert. Decision trees offer the distinct benefit of highlighting the features most critical for malware identification.

In conclusion, leveraging machine learning for Android malware detection represents a promising approach in the ongoing battle against evolving security threats in the mobile ecosystem. Though this project contain only different classification comparison, various algorithms performance such as bagging techniques, svm, deep learning will be evaluated in further implementation respectively.

REFERENCES

[1]. https://www.mdpi.com/2073-8994/14/11/2304

[2]. https://www.ijraset.com/research-paper/malware-detection-using-ml-and-deep-learning.

[3]. https://www.sciencedirect.com/science/article/pii/S1319157821003049

[4]. https://repository.rit.edu/cgi/viewcontent.cgi?arti-cle=12311&context=theses

[5]. https://ieeexplore.ieee.org/document/9077693

[6]. https://ieeexplore.ieee.org/document/8376705

[7]. https://paperswithcode.com/paper/unraveling-the-key-of-machine-learning

[8]. https://eprints.whiterose.ac.uk/128366/1/MalwareAnalysis.pdf

AUTHORS

Amay Bhatnagar is a B.Tech 4th Year Student in the Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. His research interests include Artificial Intelligence, Machine Learning and Natural language Processing.

Mahendra Singh Sagar is working as an assistant professor in Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. He had done B. Tech, M.Tech .His research area & expertise involves NLP, Machine learning, Deep Learning, Data Science.

Priyanka Goel is working as an Assistant Professor in Department of Computer Science and Engineering Moradabad Institute of Technology (affiliated to Dr. A.P.J. Abdul Kalam Technical University), Moradabad. She is having 13 years of teaching experience. Her research interests include Network Security using Machine Learning/Deep Learning techniques and Swarm Intelligence Algorithms. She has also served as reviewer in the International Conferences held in recent years.

Yamini Yadav is a B.Tech 4th Year Student in the Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. Her research interests include Cyber security,Machine Learning & Artifical Intelligence.

Sherine K Davasia is a B.Tech 4th Year Student in the Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. Her research interests include Cyber security,Machine Learning & Artifical Intelligence.