INTERNATIONAL JOURNAL ENGINEERING RESEARCH

International Journal for Innovative Engineering and Management Research

PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL

www.ijiemr.org

IJIEMR Transactions, online available on 31st Mar 2023. Link :http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 03

10.48047/IJIEMR/V12/ISSUE 03/68

Title **MALWARE DETECTION USING MACHINE LEARNING**

Volume 12, ISSUE 03, Pages: 488-497

Paper Authors

Ms.T.Hemalatha, Obula Reddy Kasireddy, Sri Krishna Var Aripi, Likhitha Doma , Yamini Borra

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Malware Detection Using Machine Learning

Ms.T.Hemalatha,Assistant Professor, Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh.
Obula Reddy Kasireddy, IV B. Tech Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh.
Sri Krishna Var Aripi, IV B. Tech Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh.
Likhitha Doma , IV B. Tech Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh.
Yamini Borra , IV B. Tech Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh.

.

Abstract

There is still new Android spyware being developed despite the widespread adoption of Android applications. Since Android devices account for 72.2% of all smartphone purchases, it stands to reason that they would be a primary target for malware activities. Hackers use a wide variety of techniques to compromise mobile devices, including stealing credentials, wiretapping, and displaying harmful advertisements. Android virus identification has been the subject of extensive study, with numerous experts presenting their hypotheses and methods. Because they can generate a classification from a collection of training examples, ML-based methods have proven useful in detecting these assaults, as they do away with the need for specific signature characterization in malware detection. As part of this effort, we looked closely at how different methods of detecting Android adware use machine learning. Recent studies suggest that machine learning is effective for detecting Android adware, making this an attractive and practical option.

Keywords: machine learning, K-Mean

INTRODUCTION

As a shorthand for "malicious software," or "Spyware," "Malware" is commonly used to refer to all of these threats collectively. In addition to taking, encoding, or erasing data, these programmes can also alter or commandeer fundamental computer functions and track user activity. Do not hide the user's approval.

Malware commonly uses one of three primary entry methods to compromise Android devices: – Repackaging – Updating – Downloading

Repackaging: Hackers entice users by recreating popular apps with malicious intent, often by promising a host of desirable benefits that aren't actually present in the original. These applications are copies of legitimate ones that have been remade and resold under different names and are infected with malware.

Updating: The update mechanism built into many apps today can be used to secretly acquire malware even while the app is running.

Downloading: In order to spread malware, cybercriminals create deceptively appealing mobile applications that promise users a wide range of benefits they cannot deliver.

Content signatures, which match an app's identity to a library of known malware signature descriptions, form the backbone of many malware detection methods. This defence only works against previously identified adware. Studies have shown that signature-based methods can't keep up with the rapid pace at which new software is created. Therefore, there is an immediate requirement for investigation and development of methods to address the problem of adware going undetected on Android devices. Solutions based on characteristics or behaviours (whether set or dynamic) are two good examples of what can work. Analysis of a malware's asked authorization list and resource usages (such as Location Services, Contact Information, WiFi, etc.) is one of the most widely used passive behavioral-based techniques of malware discovery. More than a thousand user-defined rights were mined for information for our method. The dynamic

behavioral-based method is superior because it captures the application's actions in real time by monitoring its behaviour as it runs, for instance by monitoring API calls made by the application. Such a method, however, is difficult and time-consuming to implement. In light of this, we propose a novel paradigm for Android app categorization that makes use of machine learning (ML) strategies on big and varied datasets. More than a thousand distinct rights are aggregated by our system from app requests. To help teach ML models to identify fraudulent apps, permissions are recorded. Testing with a variety of high-quality and high-volume machine learning input datasets.

What spyware can be used:

To secretly obtain private details. In general, these kinds of programmes seek highly sensitive data, such as user names, passwords, bank account numbers, and the like. Further, they can keep tabs on the user's online actions, record their movements as they navigate the web, and report this information to an external website.

Expose people to advertising that they don't want to see. Malware has the potential to flood the screen with distracting pop-ups. Adware bugs are more likely to engage in such behaviour.

forcing visitors to click through to potentially dangerous webpages. Some malware risks can even alter your browser's options, such as your default search engine and main page.

Develop a network of backlinks in the victim's search results to lead them where you want them to go (third party spyware sites, websites and other associated fields).

Initiate critical configuration modifications to the system. These alterations can cause efficiency problems and weaken security in general.

Establishing a link to a machine that has been corrupted in some way. The vast majority of malware dangers can secretly grant remote entry to the system to hackers. degradation of system efficiency and unreliability.

RELATED WORK

After the initial release of the Android smartphone in September 2008, devices running the new open-source Android OS quickly became ubiquitous. Android is the most popular mobile OS worldwide, accounting for 84% of the global smartphone market in 2021 after the introduction of nearly 12 new, improved variants. [1].

Because of Android's open-source character and its widespread use, security assaults are ramping up and pose a significant danger to the reliability of the platform's apps. Over fifty million malicious and PUA apps have been discovered for Android, according to statistics. [2].

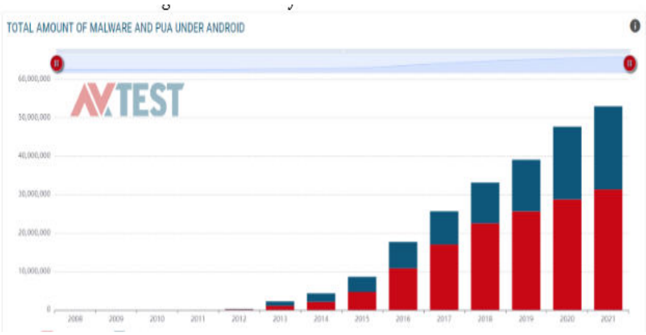


Figure 1: Number of Android Malwares Per Year

For many years, researchers have examined adware apps, eventually classifying them into distinct groups based on their shared characteristics. [3]

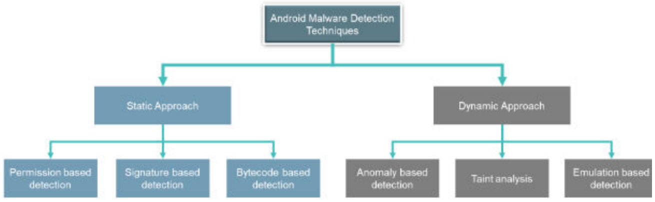
Trojans: Despite their seemingly innocuous exteriors, these applications are designed to take sensitive data from unsuspecting users.

Backdoors: These take advantage of the elevated rights that being "root" grants in order to take over the device and conduct any action they please without the user's awareness.
Worms: This software replicates itself and then spreads to other mobile devices through the networks they use.
Spyware: These apps pose as innocuous alternatives, but in reality they are secretly monitoring the user's communications, contacts, position, financial data, etc.
Botnets: In this case, the Android devices were hacked and joined into a malware that was managed by a central computer.
Ransomware: Phones infected with this software keep users out of their data until a fee is made.

Riskwares: These are legal features that can be exploited by cybercriminals to slow down a device or compromise its data.
There are two primary kinds of commonly used methods for identifying malicious software– Static and Dynamic.

Static Approach:
This technique circumvents the need to actually run an application in order to determine its functionality and potential for harm through an examination of its source code.

Dynamic Approach:
In this approach, the application is examined during execution and can help identify undetected malware by static analysis techniques due to code obfuscation and encryption of the malware.
Depending on the strategy taken to spotting outliers, these strategies take on additional classifications. The following map illustrates some of these subcategories–
Figure 2: Android Malware Detection Techniques


Android Malware Detection Techniques
Static Approach
Dynamic Approach
Permission based detection
Signature based detection
Bytecode based detection
Anomaly based detection
Taint analysis
Emulation based detection

Permission-based malware detection:
When it comes to Android devices, the permits an app is allowed play a crucial role in determining what kind of data and functionality it has access to. For instance, a user can give an app authorization to transmit data over the internet or access contacts at the moment of download. It is believed that the programme requires these rights in order to carry out its intended features. However, frequently, apps will ask for rights that are unnecessary to the app's operation.

It is possible to determine whether or not an app is malicious by comparing the rights it requests to those of known malicious apps. Machine learning categorization models can be trained with examples of both malicious and safe software to accomplish this.

Signature-based malware detection:
In fact, this is a standard procedure for many paid antimalware programmes. Signatures for the different API requests the programme will make are created in this procedure. It is possible to determine whether an application is malicious or not by finding patterns of such fingerprints and matching them with the signatures of known malware families.

PROBLEM STATEMENT
Using a combination of the Windows API code and machine learning, we can identify malicious software. Identifying previously unseen harmful code by using categorization methods on contrasting patterns Scareware detection through instruction sequence mining with a dynamic instruction duration. Adware identification with high accuracy using anti-sequence extraction. Searching for malware by analysing programme files. Malware detection using neural networks. In order to identify previously unidentified harmful code, we will apply categorization methods to a contrasting pattern. Numerous trials were evaluated, and it was determined that a 2-gram, 300-feature, Df-measured, TF configuration outperforms the absence of ML-specific methods. Finding Scareware Through Exploring the code of directions that can be any duration. For the goal of scareware identification, this project presents a data-mining-based static analysis method that employs a variable-length instructions sequence mining strategy. However, the

method is vulnerable to flaws in the absence of metrics-specific and unstructured methods.

PROPSOED SYSTEM

Features extracted from the programme source code will be the focus of the study. In this experiment, we tried out three different approaches to selecting relevant features. The chosen parameters of each classifier will then be put to the test, and the categorization will be made using the parameters that have been accepted after careful consideration of the number of characteristics. The most typical characteristics of viruses are presented next. For each of the 5 tried classifiers, we will collect the best findings in terms of the number of properly categorised cases, as well as the running time of the algorithm, the time spent in preparation, and the time spent extracting features. Malware in data and

databases is easy pickings for machine learning. K-nearest neighbours, decision trees, random forests, support vector machines, and naive bayes are just a few of the machine learning methods used. Static analysis is a type of analysis that allows one to analyse a programme or virus even without running it. Analyzing data statically Given that the parts of the malware that aren't included in normal execution can be analysed independently, it's fascinating to think about the possibilities this analysis opens up for understanding how malware or a programme would efficiently behave in conditions that aren't usual or normal in behaviour.

dynamic analysis' benefits One of the greatest features is how quickly and effectively it works. quicker than static or any other, and with unmatched precision.

IMPLEMENTATION

DATASET COLLECTION

Data collection enables the recording of prior occurrences so that trends can be identified through statistical analysis. Predictive models are constructed from these patterns using machine learning techniques to identify trends and foresee potential changes. Since the quality of a predictive model depends on the quality of the data used to construct it, adopting effective data gathering methods is essential for creating accurate forecasts. To avoid producing useless results, the input data must be accurate (garbage in, trash out). A debt failure model, for instance, could profit from petroleum price trends but not from the number of the tiger community. The data for this unit comes from the kaggle dataset files. Filed here are records of divorces that have occurred in past years.

DATA CLEANIN

It is essential to clear the data before beginning any machine learning endeavour. In this step, we sanitise the data so it's ready to be analysed by getting rid of or fixing anything that's wrong with it, like typos, missing information, duplicates, or bad formatting. You can use a wide variety of statistical and data display methods to investigate your tabulated data and discover potential data cleansing processes.

Feature Extraction

By doing so, we can speed up training and boost precision by reducing the total number of characteristics in the collection. Feature extraction is a process used in machine learning, pattern recognition, and image processing that takes raw measurement data and creates derived values (features) that are meant to be informative and non-redundant, thereby easing the learning and generalisation processes and, in some cases, improving human interpretations. Dimensionality reduction is linked to feature separation. When an algorithm's input data is too large to process in its current form and is thought to be redundant (such as the same measurement expressed in both feet and metres or the repetitiveness of images presented in pixels), it can be transformed into a reduced set of features (also named a feature vector)

Model training

Selecting relevant characteristics from a large pool of candidates is known as feature selection. The goal is to execute the desired job using this limited depiction rather than the full original data, and the chosen features are anticipated to hold the pertinent information from the incoming data. One way to get an ML system up and running is to use a training model, which is just another name for the information being used. Input data sets that are relevant to the outcome samples are also included. The raw data is fed into the algorithm and correlated with the example output using the training model. The model is refined in light of the correlation's findings. Model fitting refers to this repeated procedure. It is essential for the model's efficiency that the training dataset or the validation dataset is accurate. Training a machine learning model entails providing an ML programme with data in order to discover and learn suitable values for all relevant characteristics. Machine learning models can be divided up into a few different categories, the most prevalent of which are controlled and unstructured. Following dimensionality reduction, this section employs supervised categorization methods, such as linear regression, to train the model.

Model Testing

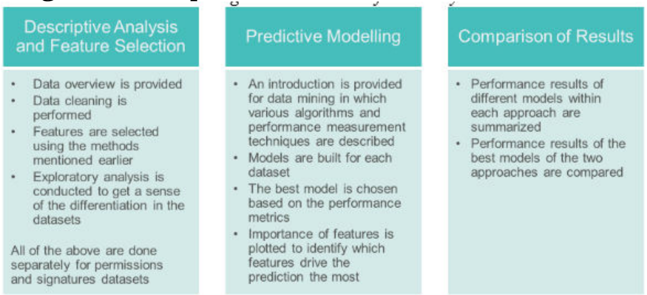
Here we put the machine learning model through its paces on the test dataset to see how well it performs. To guarantee the software system meets all of the specifications, quality testing must be performed. When it comes to the agreed-upon characteristics, were they all incorporated? Does the software act in a predictable manner? The technical design paper should list all the inputs and outputs used in testing the software. In addition, software testing can reveal any problems with the code as it is being written. It's not good for business if customers discover problems with the product after it's been published. It is possible to detect runtime errors with the help of various testing methods.

Prediction

The term "prediction" is used to describe the results of an algorithm after it has been taught on a previous dataset and applied to new data in order to predict the probability of a specific event, such as whether or not a client will quit in 30 days. For each entry in the new data, the programme will produce likely values for an uncertain variable, enabling the model creator to determine what that value will most likely be. When used carelessly, the term "forecast" can be deceiving. Using machine learning to determine the next best action in a marketing effort is an example of a situation in which a prediction of a future result is actually being made. However, in other cases, the "forecast" concerns an already completed event, such as the possibility of deception in a previously completed transaction. The deal has already occurred, but you can take corrective action based on your best estimate of its legitimacy. In regards to both test and training data, we have a virus defintion.

Follows is a description of the in-depth study. The code, along with the outputs and findings, is displayed, and a methodical description is given for how to use it.

There are three major parts to this analysis: the description, the modelling, and the comparison. Here's a quick breakdown of the diagram's components.



Descriptive Analysis and Feature Selection

- Data overview is provided
- Data cleaning is performed
- Features are selected using the methods mentioned earlier
- Exploratory analysis is conducted to get a sense of the differentiation in the datasets

All of the above are done separately for permissions and signatures datasets

Predictive Modelling

- An introduction is provided for data mining in which various algorithms and performance measurement techniques are described
- Models are built for each dataset
- The best model is chosen based on the performance metrics
- Importance of features is plotted to identify which features drive the prediction the most

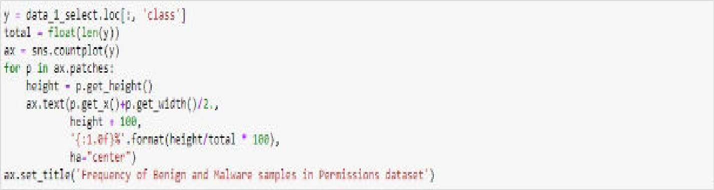
Comparison of Results

- Performance results of different models within each approach are summarized
- Performance results of the best models of the two approaches are compared

Figure 3: Structure of the Analysis

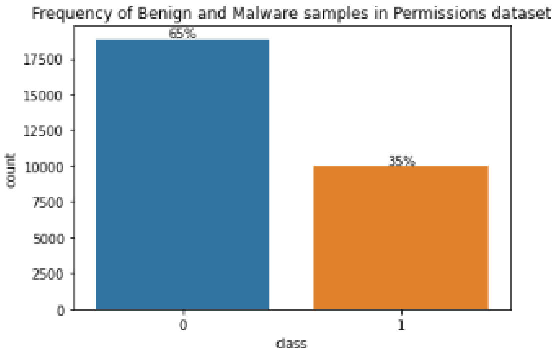
OUTPUT RESULTS

Here, I start by verifying how the data is separated between malicious and safe examples. For data mining methods to function properly, a large enough sample size of the relevant data is required. The next steps in checking this are.



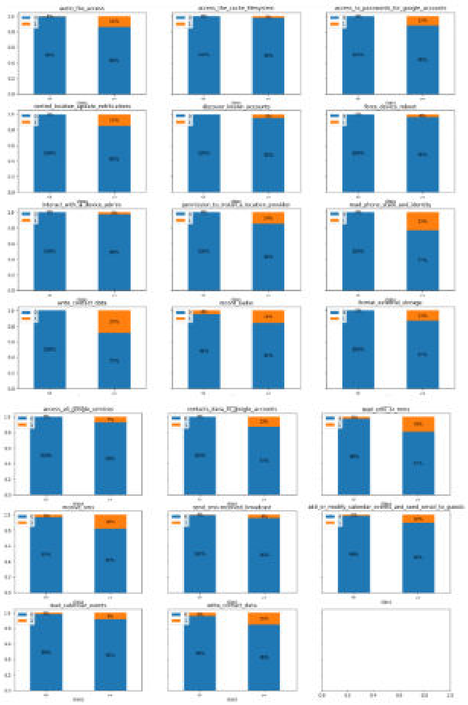
```
y = data_1_select.loc[:, 'class']
total = float(len(y))
ax = sns.countplot(y)
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 100,
            '{:1.0f}%'.format(height/total * 100),
            ha="center")
ax.set_title('Frequency of Benign and Malware samples in Permissions dataset')
```

Figure 8: Class Distribution of Permissions Data



Therefore, 65% of the records belong to the "Benign" category, while 35% belong to the "Malware" category. Statistical research can continue because there are enough data across all categories.

Now I examine the proportion of benign to malicious samples that voted yes or no to each feature. The key visible differences between malicious and safe software will be easier to spot.
Below, you'll find the final findings.
Figure 9: Permissions Data Exploratory Analysis Results



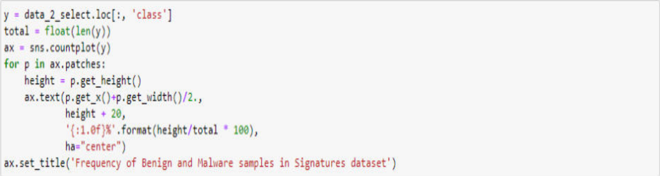
Based on the aforementioned graphs, it is clear that the features'read phone state and identity' and 'write contact data' provide the greatest numerical difference in separating malicious from innocuous examples.

As a result, 23% of malicious apps ask to access your phone's location services and identification data, while no good apps make use of these features at all. If a programme requires this access, it is almost certainly malicious.

We can anticipate that the following parts' forecast modelling will rely heavily on these highly dissimilar characteristics.

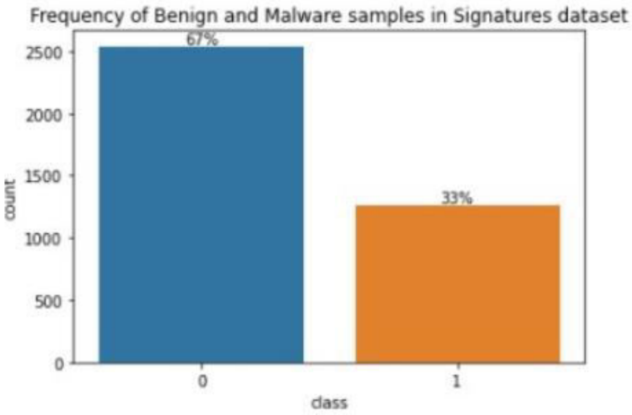
- **Exploratory Analysis**

Here, we do some preliminary digging into the rights information.
First, we need to see how the data is divided between malicious and safe examples. If you want to use data mining techniques, you'll need a good amount of examples of each kind.



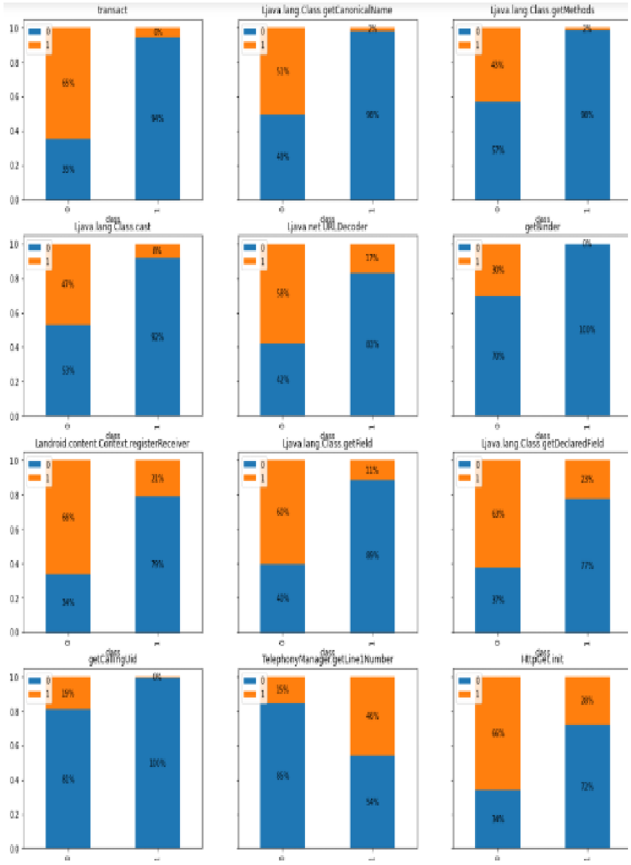
```
y = data_2_select.loc[:, 'class']
total = float(len(y))
ax = sns.countplot(y)
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 20,
            '{:1.0f}%'.format(height/total * 100),
            ha="center")
ax.set_title('Frequency of Benign and Malware samples in Signatures dataset')
```

Figure 11: Class Distribution of Signatures Data



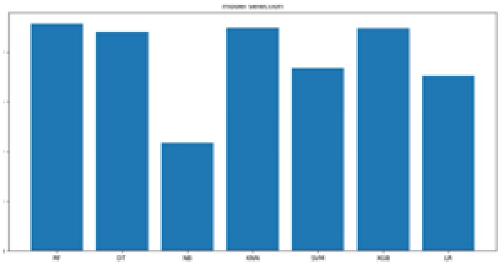
So, according to the statistics, 33% of the entries are malicious while 67% are classified as benign. An adequate amount of data points exist in each category to enable further analysis.

Separated into categories of benign and malicious samples, we can now examine the proportion of samples that have and do not have each characteristic in question. The following graphs illustrate this point.
Figure 12: Signatures Data Exploratory Analysis Results

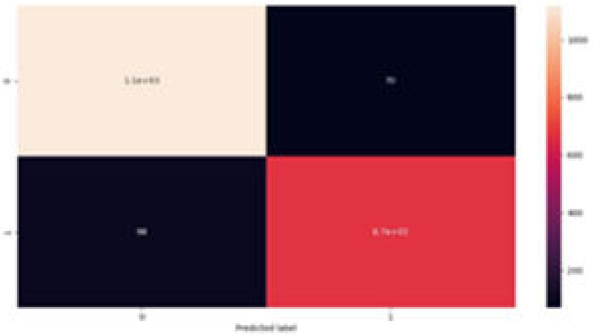


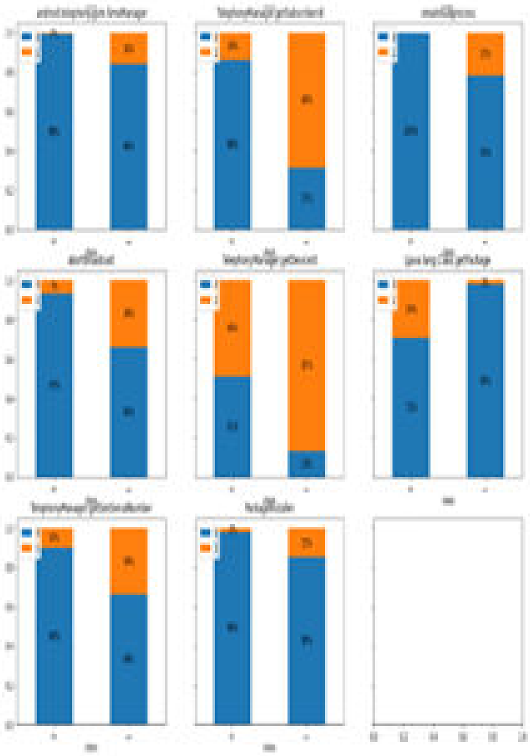
From an ocular inspection, we can tell that 'transact,' 'getCanonicalName,' 'getSubscriberId,' etc. are the characteristics that appear to distinguish Spyware examples from Innocuous ones the most. These factors might be crucial indicators in the subsequent stage.

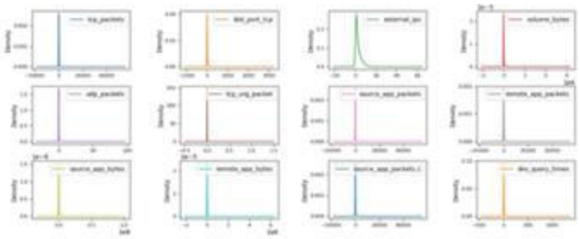
The above studies end the preliminary analysis and feature selection process. Next, we'll construct categorization models to evaluate the efficiency of each virus detection technique in relation to the other.



Confusion matrix of RF







CONCLUSION

Our primary objective was to design a machine learning system that can identify as many malware samples as possible in a generic fashion while adhering to the strict requirement of producing no false positives. We came close, but our erroneous positive rate is still above zero. This framework needs several predictable exception methods added in order for it to be included in a business product that can compete favourably with others in its field. Our prediction is that anti-virus companies will add machine learning-based malware identification to their existing techniques rather than supplant them. There are performance and capacity constraints on any business anti-virus software. The most trustworthy methods can help circumvent these problems. This paper addresses the problem of static analysis-based virus identification for the Android mobile operating system, which is presently the most common mobile platform. An introduction to Android malware analysis is provided in this thesis, and a special collection of characteristics is selected for use in the subsequent research on malware categorization. With the goal of selecting the most efficient malware detection algorithms, we analysed five classification algorithms (Random Forest, SVM, K-NN, Nave Bayes, and Logistic Regression) and three attribute selection algorithms. Based on the data we gathered, we were able to determine what features fraudulent software typically has. Features derived from Java class code were the focus of this study. Classification quality was evaluated across different feature sources to determine which one is superior.

References

[1].Almin, S. B., & Chatterjee, M. (2015). A novel approach to detect Android malware. *Procedia Computer Science*, *45*, 407-417.

[2].An Odusami et al. (2018, November). Android malware detection: A survey. In *International conference on applied informatics* (pp. 255-266). Springer, Cham.

[3].Arshad et al. (2016). Android malware detection & protection: a survey. *International Journal of Advanced Computer Science and Applications*, *7*(2), 463-475.

[4].Assisi, A., Abhijith, A., Babu, A., & Nair, A. M. Significant permission identification for machine learning based android malware detection: a review.

[5].Atlas. (n.d.). Malware statistics. Retrieved Oct, 2021 from https://portal.av-atlas.org/malware/statistics

[6].Christiana, A., Gyunka, B., & Noah, A. (2020). Android Malware Detection through Machine Learning Techniques: A Review.

[7].Daoudi et al. (2021, August). Dexray: A simple, yet effective deep learning approach to android malware detection based on image representation of bytecode. In *International Workshop on Deployable Machine Learning for Security Defense*, 81-106.

[8].Fallah, S., & Bidgoly, A. J. (2019). Benchmarking machine learning algorithms for android malware detection. *Jordanian Journal of Computers and Information Technology (JJCIT)*, *5*(03).

[9].Jiang et al. (2020). Android malware detection using fine-grained features. *Scientific Programming.*

[10].Kumar, R., Xiaosong, Z., Khan, R. U., Kumar, J., & Ahad, I. (2018, March). Effective and explainable detection of android malware based on machine learning algorithms. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence* (pp. 35-40).

[11].Kyaw, M. T., & Kham, N. S. M. (2019). *Machine Learning Based Android Malware Detection using Significant Permission Identification* (Doctoral dissertation, MERAL Portal).

[12].Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, *14*(7), 3216-3225.

[13].Liu et al. (2020). A review of android malware detection approaches based on

machine learning. *IEEE Access*, *8*, 124579-124607.

[14].Needham, M. (2021). Smartphone market share: Supply chain constraints finally catch up to the global smartphone market, contributing to a 6.7% decline in third quarter shipments, according to IDC. Retrieved Oct, 2021 from https://www.idc.com/promo/smartphone - market-share/os

[15].Rana, M. S., Gudla, C., & Sung, A. H. (2018, December). Evaluating machine learning models for Android malware detection: A comparison study. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing* (pp. 17-21).

[16].Rathore et al. (2020, December). Detection of malicious android applications: Classical machine learning vs. deep neural network integrated with clustering. In *International Conference on Broadband Communications, Networks and Systems*, 109-128.

[17].Roy et al. (2020). Android Malware Detection based on Vulnerable Feature Aggregation. *Procedia Computer Science*, *173*, 345-353.

[18].Shao, K., Xiong, Q., & Cai, Z. (2021). FB2Droid: A Novel Malware Family-Based Bagging Algorithm for Android Malware Detection. *Security and Communication Networks*.

[19].Sharma, S., Krishna, C. R., & Kumar, R. (2020, November). Android Ransomware Detection using Machine Learning Techniques: A Comparative Analysis on GPU and CPU. In *2020 21st International Arab Conference on Information Technology (ACIT)* (pp. 1-6). IEEE.

[20].Singh, D., Karpa, S., & Chawla, I. (2022). "Emerging trends in computational intelligence to solve real-world problems" Android Malware Detection Using Machine Learning. In the International *Conference on Innovative Computing and Communications* (329-341). Springer, Singapore.

[21].Syrris, V., & Geneiatakis, D. (2021). On machine learning effectiveness for malware detection in Android OS using static analysis data. *Journal of Information Security and Applications*, *59*, 102794.

[22].Wen, L., & Yu, H. (2017, August). An Android malware detection system based on machine learning. In *AIP Conference Proceedings* (Vol. 1864, No. 1, p. 020136). AIP Publishing LLC.

[23].Yerima, S. Y., & Sezer, S. (2018). Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE Transactions on Cybernetics*, *49*(2), 453-466.