Network Intrusion Detection of Android Smartphones Using Machine Learning and Ensemble Learning Techniques

MSc Research Project
MSc Cybersecurity

Sumit Kumar
Student ID: X20258526

School of Computing
National College of Ireland

Supervisor: Mr. Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sumit Kumar		
Student ID:	X20258526		
Program:	MSc in Cybersecurity	**Year:**	2022-2023
Module:	MSc Research Project		
Supervisor:	Mr. Jawad Salahuddin		
Submission Due Date:	01-02-2023		
Project Title:	Network Intrusion Detection of Android Smartphones using Machine Learning and Ensemble Learning Techniques		
Word Count:	7744	**Page Count**	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sumit Kumar

Date: 01-02-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Network Intrusion Detection of Android Smartphones Using Machine Learning and Ensemble Learning Techniques

Sumit Kumar
X20258526

Abstract

Mobile risks are growing at a quick rate as the number of mobile users significantly increases. Modern age's major cyberattacks have been fuelled by the advanced development of mobile devices and technologies. The popular operating system Android, which is used in smartphones, is a prime target for dubious activities carried out by various intrusions and malware. Due to its popularity and open-source platform, Android has turned into a top target for unethical intrusions. A mobile virus can result in a number of cybersecurity problems. The volume of data created and the increase in zero-day threats make the present security applications insufficient to predict and detect intrusion with variable properties. These challenges have been addressed in recent years using machine learning classification algorithms, and this study compares classic machine learning models with ensemble learning models to determine which model can yield the greatest results. So, utilizing the Random Forest, Decision Tree, KNN techniques, and the Android Mischief dataset as training data, we presented a comparison between ensemble learning and the classic ML classification model in this study to detect remote access trojan (RAT). Metrics like accuracy, precision, and F1 score are used to measure the model's performance, and its performance is contrasted with that of more established models like Decision Tree and K Nearest Neighbor (KNN).

Keywords: Android, Remote Access Trojan, RAT, Network Intrusion Detection, Machine Learning, Ensemble Learning, Random Forest, Decision Tree, KNN

1 Introduction

Since the turn of the century, high-speed mobile communication networks have developed quickly, leading to an increase in the use of portable electronics like smartphones. Due to their portability and array of capabilities, which range from simple alarms to payments to the bank, people use their phones to manage their life. Worldwide, there are more than 3.6 billion phone owners, and a Statista Research study from April 2021 indicates that more than 54% of all Internet consumption is now done on mobile devices. (*Smartphone subscriptions worldwide 2027*, no date) Google has a very popular platform called Android, which powers billions of smartphones worldwide. Anyone can independently create an android-based application because Android is an open-source platform. A malware-filled Android application is created by a variety of intruders to carry out numerous destructive actions. As a result of its acceptance, there has been a rise in the variety and number of cyberattacks. Kaspersky statistics show that about 7% of these Android mobile attacks were carried out by Remote Access Trojans (RAT). Because they make it possible to remotely manage the infected machine in a variety of ways, RATs are regarded as one of the most hazardous types of malware. (*Remote Access Trojan (RAT)*, no date) Long-term attempts to identify RATs have been made by the security community, with some measure of success. Antivirus detections of binary files outside of Android mobiles or link analysis are the finest detections currently discovered. RAT network traffic detection, however, has not proven effective. Although network traffic-based RAT detections may significantly increase the security of our mobile devices, to our knowledge, no extensive research has been done to identify Android RATs in the network. This study contributes to this field of study by comparing ensemble learning and conventional machine learning models on the most recent dataset to identify remote access trojans (RATs) for Android devices with the highest level of accuracy.

The research contribution and novelty as compared to prior work on this dataset are based on research and findings that have been done in the same area, and it is believed that nobody has used machine learning algorithms such as Decision tree and Random Forest for the Android remote access trojan (RAT) detection and provided a comparison between the ensemble learning and traditional machine learning model's accuracy result of RAT detection for Android. Also, the same dataset that is the **Android mischief dataset** is not used by any other researcher so far since this dataset is published in the year 2021 and there is no latest ongoing research where the same dataset was used with similar machine learning models for Android RAT detection.

Remote Access Trojan (RAT)
Malware called a "remote access trojan" enables an attacker to take over a target device and control it from a distance. Given that they are a component of the majority of assaults, including APTs and Ransomware, RATs are one of the most significant threats today. Particularly when it comes to Android RATs in phones, it is difficult to identify RATs in network traffic. Why? The fundamental issue is that our mobile devices lack any simple means of viewing the network traffic. The security of our phones is significantly less assured than that of our computers. (*What is a RAT (Remote Access Trojan)? | Definition from TechTarget*, no date)

1.1 Motivation for Research

Due to the daily introduction and development of new and emerging remote access trojans (RAT) by several hackers or attackers, RAT and its detection have always had a number of problems. It is vital to look into new technologies that would protect mobile devices from unidentified and ambiguous dangers even though current operating mobile computing systems combine a number of security features, such as (post) authentication mechanisms and access control. In order to protect our mobile phones, this thesis uses traditional machine learning models like Decision Tree, KNN, and Ensemble learning to address the issue of Android RAT detection in network traffic. In order to examine the processes and prevent malicious assaults and keep confidential data safe, it is the responsibility of reliable secondary means to gather information on the system and its infrastructure.

1.2 Research Questions

- How effectively, and with a low false-positive rate, can proposed machine learning models identify the intrusion in the network of Android mobile devices?

- How do feature selection strategies assist us in enhancing accuracy?

1.3 Research Objectives

- To establish the most efficient method for finding Remote Access Trojan (RAT) on Android mobile networks.

1.4 Research Hypothesis

H1: The suggested machine learning models have a high degree of accuracy in recognising the remote access trojan in Android mobile networks.
H0: The suggested machine learning models have not a high degree of accuracy in recognising the remote access trojan in Android mobile networks.

H2: By using feature selection strategies, we can increase our accuracy.
H0: By using feature selection strategies, we cannot increase our accuracy.

1.5 Structure of the Report

Chapter #	Description	Details
2	Related Work	This section will cover the literature related to network intrusion detection along with examination of the earlier research on this topic.
3	Research Methodology	This section will cover a thorough explanation of the algorithms utilised and all the specifications for creating this thesis.
4	Design Specification	This section will cover the design process of the proposed model for the thesis. This will also cover the methods utilised and the structure for each of the approach.
5	Implementation	This section will explain the implementation of the proposed machine learning models.
6	Evaluation	This section will cover the outcomes following the application of the thesis. Based on a comparison of

		the outcomes produced by various algorithms, a conversation will be conducted
7	Conclusion and Future Work	This section will cover the conclusion withdrawn following the application of this theory. This part provides the response to the research question. It will also be explored what the future will entail.

Table 1: Report Structure

2 Related Work

A discussion of the material I read for the purpose of developing my thesis will be presented in this part. There will be a brief review of the advantages and disadvantages of the literature. A discussion of the literature will then be conducted.

In 2008, the first Android phone was developed, and sales have skyrocketed since. As more people began utilising android devices for various purposes, cybercriminals began to launch assaults. Cybercriminals and threat actors have been more interested in android phones as their popularity has increased. As a result, there are a tonne of mobile malware programmes that target Android-powered smartphones and do malicious tasks like stealing money or private data, among others. These malicious software programmes were created by inventive hackers and detecting them requires processing intelligence and detection software. Most work on utilising machine learning techniques to detect intrusions on Android is based on system and API calls. Network-based functionalities for Android malware have just recently been the focus of a small number of research.(Olson, no date)

2.1 Penetration of Android Devices

The smartphone is a typical and important item today. However, the security risks have escalated in proportion to the development of Android smartphones. According to the (Ahmad, Ali Shah and Ahmad Al-Khasawneh, 2021) even if many security measures have been developed for platforms like Android, not all risks can be eliminated by only employing encryption and authentication. Due to the increase in viruses and breaches, smartphones are at serious danger for security. According to recent studies and reports, malware and viruses are becoming more common on operating systems of smartphone, particularly Android. Smartphone development is being held back by its lack of computing power and inadequate energy supply, which depend on battery usage.

As per the (Rahmat *et al.*, 2019) 90% of Android cell phones are vulnerable to at least one significant flaw. As a result, after exploiting existing vulnerabilities, attackers look for opportunities to create new attack vectors in order to compromise an organization's infrastructure as a whole. However, during the past few years, machine learning has achieved significant results with its malware classification and feature selection methods. In addition to providing a brief history of some conventional techniques, this paper will discuss key classification techniques that were employed to identify anomalies in the Android mobile network sector.

2.2 Prior Contribution and Their Limitations

Malware was discovered on Android for the first time in 2010; following that, its prevalence progressively rose. With an emphasis on the development of new and effective IDS, substantial research has already been conducted on mobile device security. With the help of signature-enabled and anomaly-based techniques, it is possible to collect intrusion features and subsequently identify the software or behaviours that make up the intrusion. (Zhou *et al.*, 2012) did a study that revealed certain categories of harmful programmes that have been disclosed during the past year. A study proposed by (Shabtai *et al.*, 2012) depending on the host, an Android malware detection framework. The proposed strategy continuously examines a wide range of smartphone characteristics and events before classifying the acquired data using methods of machine learning. However, no actual virus testing was done on the indicated system. A study conducted by (Yuan *et al.*, 2013) proposed an intrusion detection system (IDS) for detecting anomalies in Android mobile devices. The recently created framework classified the data gathered from a smartphone as either malicious or valid using the Naive Bayes algorithm. The claims regarding the detection rates, however, remain under question because the malware utilised in the Classifier's training of Naive Bayes is not made public. (Ghorbanian *et al.*, 2013) a signature-based IDS was shown to be a proposed system for Android smartphones. The authors used the detection technique which was signature-based with pre-established rules to find irregularities. This methodology solely takes into account risks that is known, leaving out unknown dangers.

Due to the high level of security concerns, it is crucial to keep in mind that novel solutions need to be researched in order to protect current mobile devices from both known and unknown threats. In the study of (Yerima *et al.*, 2013) the applications were divided into benign and suspicious categories by the authors using Bayesian classification. For training and classification, 1000 samples of each type of programme were used by them, including malware. Where in the study of (Sahs and Khan, 2012) Authors trained support vector machine (SVM) models to distinguish between good and bad applications using permission and call flow graphs. A study conducted by (Mahindru and Singh, 2017) using Random Forest, Naive Bayes, K-star, J48 Decision Tress, and Simple Logistic Technique, the author attempted to categorise malicious applications of Android. The study's findings show that a simple logistic approach has a 99.6% accuracy rate on a dynamic application dataset, but that the choice of classifier and the sample size that is smaller frequently cause false positives.

According to the study conducted by (Arp *et al.*, 2014) the author use DREBIN to locate phoney Android apps. The SVM Classifier was used to build the model, which categorises good and dangerous applications based on factors including API calls, permissions, and network addresses. It has already been highlighted that many users fail to install the DREBIN programme on their mobile device, which is required to detect fake programmes. The inability of DREBIN to detect malware that uses obfuscation or dynamic code is just one of the many limitations that it has. A model presented by (Song *et al.*, 2016) a static identification technique for detecting malware on Android, where a filtering technique has been merged with detection. The primary benefit of this work was its ability to reduce workload, which led to high efficiency. The outcome produced additionally displays a rate of 98.80%. Multiple number of studies that were conducted in the middle of the previous time period. A approach for categorising malware was suggested by (Ali Alatwi *et al.*, 2016) which focuses primarily on improving the model's performance in the domain of undeniable. The quality of the characteristics played a significant role in accuracy. In this study, classifier training has been carried out on each group separately to improve performance. A related investigation was carried out by (Bhatia and Kaushal, 2017) which increases dynamic analysis's effectiveness for spotting malware on Android by twofold. They suggested using a virtual box for the execution and employed a monkey tool that produces movements in this approach, they gathered a lot of traces, and the data they gathered was utilised to analyse different learning approaches, which gave accuracy levels of more than 80%. The study found that dynamic analysis is an effective way to find malware.

Later, an approach has been suggested by (Feizollah *et al.*, 2017) When evaluating the effectiveness of the intents as a feature for detecting malware, the study makes use of a substantial dataset that includes 7406 programmes andriodialysis which is capable of checking two unrelated intent objects termed implicit and explicit. When manipulating androids with purpose, there have been reports of an efficiency of over 90%, and for consent, it was over 80%. The indent feature is not regarded as the ultimate last resort, which is a significant limitation for the paper. The study suggested by (Feng *et al.*, 2018) where author suggested a framework for a study called EnDriod, which is successful. The proposed work seeks to implement very precise intrusion detection based on several kinds of dynamic performance metrics. EnDriod's acceptance of the feature selection method to avoid the noisy or inappropriate characteristics and extract the important behaviour feature is one of the main advantages.

Another study conducted by (Yerima and Sezer, 2019) where author proposed a ensemble detector that was multi-level classifier named Droid fusion to improve classifier accuracy by removing security threats to the Android system, the base classifiers at the lower stage are combined with the predictive ranking-based algorithm at the upper level. A study conducted by (Demontis *et al.*, 2019) It has been scientifically demonstrated that using machine learning techniques increases security. Since the designed algorithm can be very resistant to avoidance, an adversary aware methodology has been used. Another system called OmniDroid was proposed by (Martín, Lara-Cabrera and Camacho, 2019) the tool's primary goal is to assist researchers and developers while developing anti-malware programmes. It was a large and widely used dataset that contains 22000 real set of intrusion and goodware samples. The dataset was produced using AndroPyTool and a number of classifiers. Researcher (Rana and Sung, 2020) proposed a study on several machine learning strategies using ensemble learning approaches as sacking, boosting, etc. The study found that the boosting algorithm can be useful and effective since it reduces most main errors by separating strong learners from weak learners. According to the study conducted by (Aminanto and Kim, 2022) The use of deep learning in intrusion detection is open to criticism and other potential problems, which lends weight to this assertion. Last but not least, Deep Learning models can be beneficial in upcoming research on the detection of unknown dangers. Below figure depicts the existing malware detection techniques which uses either static analysis or dynamic analysis techniques.

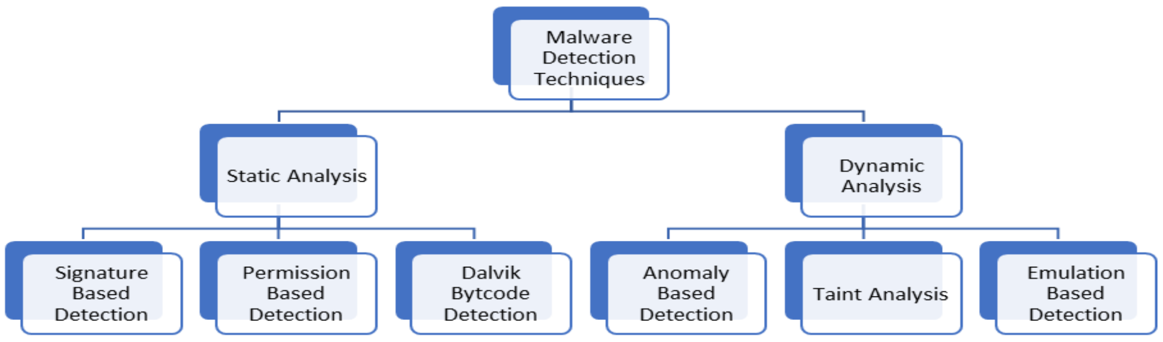

Malware Detection Techniques
Static Analysis
Dynamic Analysis
Signature Based Detection
Permission Based Detection
Dalvik Bytcode Detection
Anomaly Based Detection
Taint Analysis
Emulation Based Detection

Figure 1: Existing Malware Detection Techniques

2.3 Intrusion Detection Using Machine Learning

Intrusion that targets computers, and computer systems has been categorised using machine learning in recent years. Machine learning is widely utilised to identify dangerous applications in recent years as more Android intrusion samples have become available. In order for the Android phone to be able to detect any intrusion activities that might be injected by any undesirable sources, the notion of traditional machine learning and Ensemble Learning Algorithm was taken into consideration in this research.

Types of Machine Learning Approach

Supervised Learning:
In supervised machine learning, the dataset for the model includes both the input and the output data. Several facets are displayed as input data. The objective of a supervised machine learning algorithm is to discover any relationships between the input and output data using these specified attributes. The model is biased toward the result when using the supervised learning approach. Simply learning the actions to take will get it there. For instance, the algorithm must decide whether this email is spam or not. These supervised learning tasks involve 0 and 1 based Boolean equations. In general, supervised learning includes two categories of algorithms: classification algorithms and regression algorithms.(Edwards, 2020)

Unsupervised Learning:
Unsupervised machine learning is a type of learning when the system only receives input data and the results are unknowable. In this situation, the algorithm is allowed to choose a path of action of its own. The algorithm must look at certain previously undetectable patterns in the input dataset in order to predict future results or develop a solution. This is the trickiest instructional technique. The algorithm must go through a process of trial and error to reach a result. Unsupervised machine learning is a powerful tool for developing original solutions to

any challenge. Consequently, this technique is also known as the technique of knowledge discovery.(Edwards, 2020)

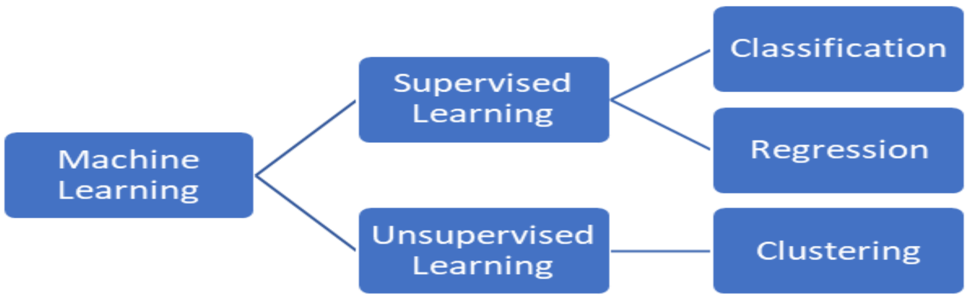


Figure 2 Machine Learning Approaches

3 Research Methodology

The fundamental framework for the proposed study is an experimental design, and secondary sources is used for the datasets. The source used was a remote access trojan intrusion dataset namely Android mischief dataset, which was very relevant because it represented genuine intrusion attacks. The outcome analysis also aims to be a comparison analysis carried out by contrasting various elements of the multiple under examination. This would help with the primary study question, which was to find a compromise between the new implementation and the accuracy of the evolving intrusion detection models for Android handsets. Therefore, by comparing the different machine learning models shown in this thesis, the developers may decide the optimal detection technique to use going forward. The focus of the research technique to be used in the proposed method of research is the significant research models discovered in the literature, as determined from the earlier pertinent works above. The proposed model will be constructed in a number of steps, as illustrated in the figure below.

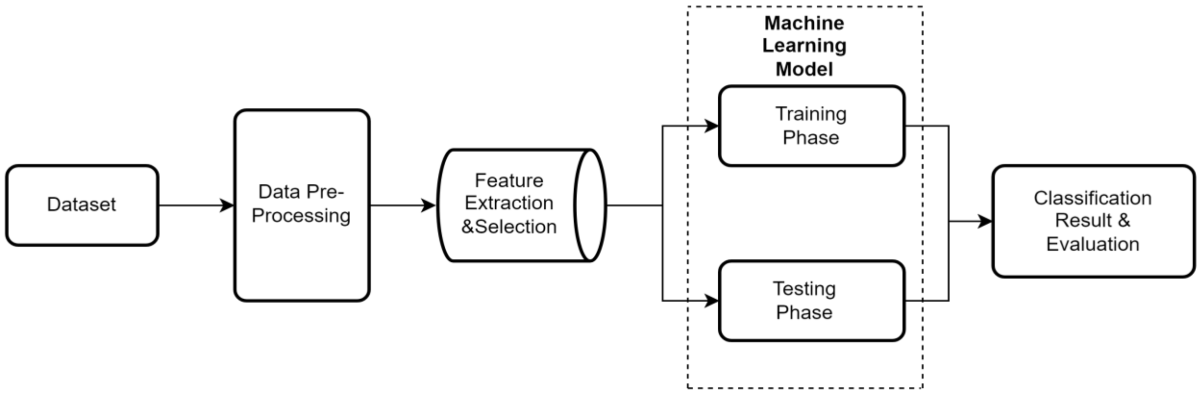


Figure 3 Process Flow of Proposed Model

We will first take the dataset we'll be working with to a program. Data preprocessing will take place next as a next step. This stage involves establishing a solid grasp of the facts. The process of data generalisation is carried out in this step, which also checks for missing values and provides no null values. After that, extraction of the feature and selection are carried out. To score the features at this stage depending on their importance level in achieving the target variable, we are employing the select from model feature selection approach. After comparing the scores, those with the lower scores are removed from the data.

The following step would be model training, Decision tree, K Nearest Neighbor from classic machine learning model and Random Forest from Ensemble learning are the models that will be used. Accuracy score, precision, and F1 score are the evaluation measures that are discussed. They are all classification models, which is what they are all. With the help of the data, classification models can predict to which category or class an observation belongs. Training the models on training data comes first, then testing them on test data, where a portion of the data set will be used to train the model in this scenario. The population as a whole is what training and testing are really just subsets of. The remaining data set is used for testing after training, and the results are assessed via Accuracy.

4 Design Specification

This section of the report is a description of the machine learning models and their framework and assessment metrics used in this proposed model.

4.1 Decision Tree Model

The first algorithm that is used in the development of this thesis is Decision Tree. Decision Tree is a supervised machine learning algorithm that can be used to address problems involving classification and regression. With the help of decision nodes and leaf nodes, it creates the model as a tree structure. Any number of branches make up a decision node. Decisions are represented by leaf nodes. The root node is a top-level decision node. It is capable of handling both continuous and categorical data.(Arora, 2020) Tests or assessments are made based on the qualities of the dataset that is presented. It presents a graphic representation of every possible response to a problem or choice, given specified criteria. It is known as a decision tree because, like a tree, it begins with the root node and grows by adding more branches to take the form of a tree.

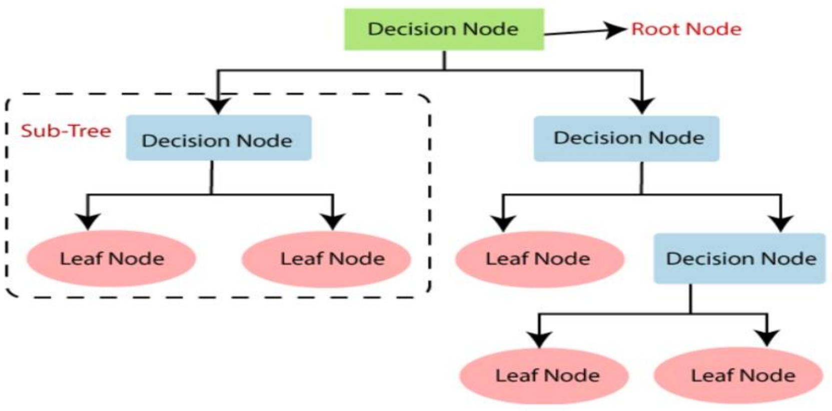


Figure 4 Decision Tree Model[1]

4.2 K Nearest Neighbor (KNN) Model

KNN is utilised in classification and regression applications. But the classification component is where it really excels. To find new data points, KNN uses a concept called "feature similarity." The distance of the newly generated data points from the training dataset will be used to calculate a measure for them. A majority vote is used to select a class label for a classification problem, which means that the classification that is most often used to refer to a particular data point is accepted.(Okfalisa *et al.*, 2017)

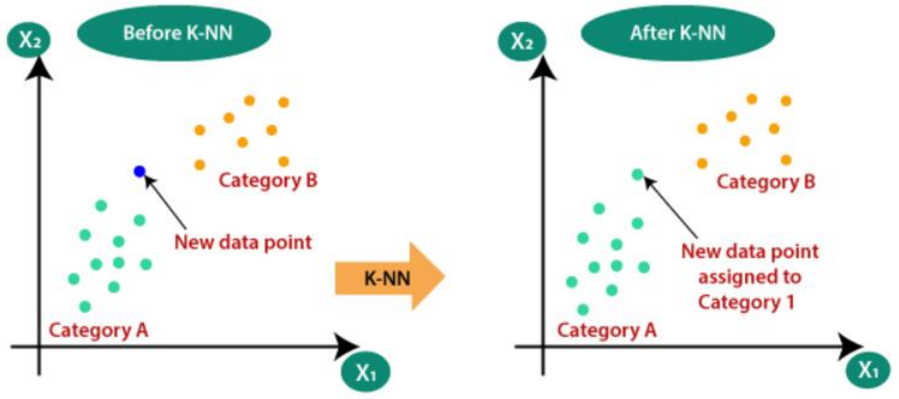


Figure 5 K-Nearest Neighbor (KNN) Model[2]

The procedures for putting the KNN algorithm into practise is as follows.

- Both the training and testing datasets are loaded, and the dataset is separated into these two categories.
- Depending on their kind, the training dataset is separated into various clusters.

[1] https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
[2] https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

- For the data point that needs to be classified, the k-value is chosen.
- The categorised clusters are all used to identify the K closest neighbour data points. These closest neighbours are located using a distance function.
- The cluster with the greatest number of neighbours is given the new data point.

4.3 Random Forest Model

Using data that was partitioned randomly, an ensemble of decision trees was created. Forest is the name given to the collection of trees. Each tree is produced using a selection of attributes, such as information gain, gain ratio, etc., and is based on an independent random sample. When solving classification issues, we select the top-voting tree as the final solution. The average of all the trees is taken into account as the final outcome for regression issues.(Arora, 2020)

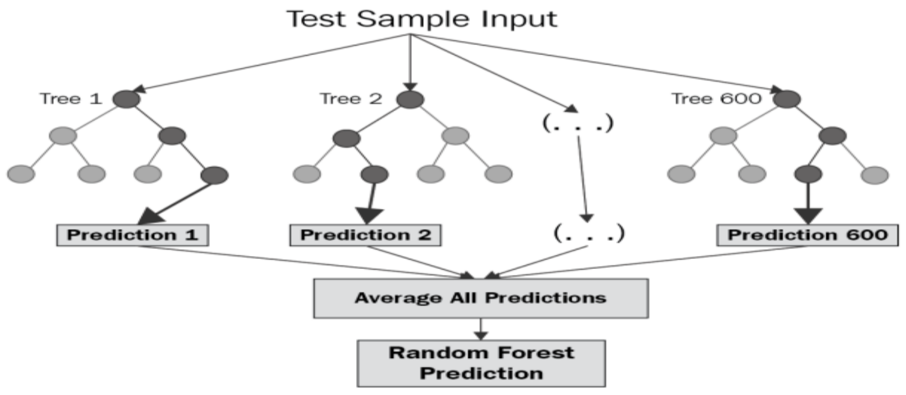


Figure 6 Random Forest Model[3]

Random Forest vs Decision Tree

Although decision trees are straightforward, they have numerous major flaws, including overfitting, variance- or bias-related errors. A Random Forest is a group of decision trees that produce an aggregated result. The random forest uses several trees to reduce the possibility of overfitting. They are also challenging to understand. A decision tree is easier to read and understand when compared to a random forest.

Despite being quick to implement, a single decision tree is not good at forecasting the outcomes. A stronger model will result from adding more trees, which also reduces overfitting. We must produce, process, and analyse each and every tree in the forest. Consequently, this is a lengthy procedure that occasionally takes hours or even entire days.

4.4 Proposed Metrics for Performance Evaluation

The definition of the evaluation parameters that will be used is now the most important component of the technique that will be used for the comparative examination of the

[3] https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

performances of the suggested models that will be used to address the primary research issue in the proposed research. The evaluation metrics are an essential component of a machine learning model. There is no use in developing a machine learning model without a clear feedback mechanism. We get a model's feedback through the evaluation metrics. We continue till we achieve the necessary accuracy score; we will improve model performance. The effectiveness of a model is guided by evaluation metrics.

The following criteria are used in the proposed approach to assess the models' performance:

4.4.1 Accuracy

Accuracy, which is calculated as the ratio of successfully identified objects to all evaluated objects, is the most sensible performance statistic. A scale of statistical evaluation for a model is called the accuracy score. The quantity of values that the model can accurately anticipate is its accuracy. In mathematics, it can be written as:

$$Accuracy = \frac{TP+FP}{TP+TN+FP+FN}$$

4.4.2 F1-Score

The F1-Score for a classification problem is the harmonic mean of the precision and recall values. The weighted average for precision and recall might be used to characterise it. An F-score is said to be at its maximum when it reaches 1, and at its minimum when it reaches 0.

$$F - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$$

Precision: Only a small portion of all positively anticipated positive data objects—referred to as precision—were really predicted with any degree of accuracy.

$$Precision = \frac{TP}{TP+FP}$$

Recall: Keep in mind that the answer is calculated by dividing the number of precise affirmative outcomes by the total number of relevant samples.

$$Recall = \frac{TP}{TP+FN}$$

4.4.3 Confusion Matrix

The most popular tool for evaluating the effectiveness of a classification algorithm is the confusion matrix. A few crucial parameters will be used to compare the performance of the models used in the proposed research. The performance of the model is revealed through an insightful confusion matrix. However, different performance metrics can be calculated utilising its component pieces to get even more information.

- **True Positive (TP):** The percentage of values that are both truly true and as predicted by the model.
- **True Negative (TN):** True negative is the percentage of values that are actually negative, and the model also an anticipated negative outcome.
- **False Positive (FP):** The number of genuinely negative values for which the model also made a prediction of truth is known as the False Positive.
- **False Negative (FN):** The proportion of genuinely negative values that the model also predicted to be true is known as false-negative.

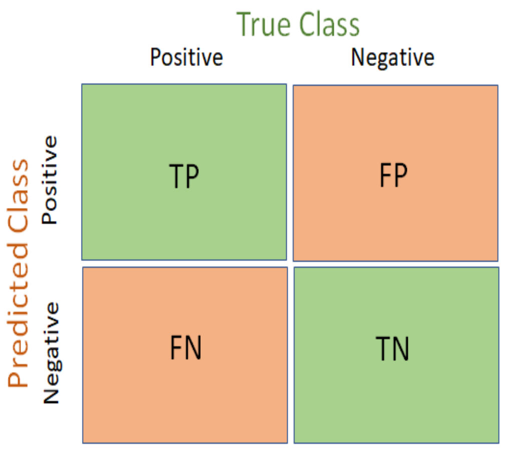


Figure 7 Confusion Matrix

5 Implementation

Here, we'll go through how machine learning and ensemble learning were used to create a powerful classification model and get the best results possible. In this section, the modelling and feature selection procedure is also explained.

5.1 Software and Programming Language Used

Python is used to programme the suggested model. The machine learning project is best suited for the Python programming language. Python is the best choice for machine learning applications due to its consistency, platform freedom, ease of use, accessibility to top-notch ML tools and frameworks, flexibility, huge community, and ease of use. On the Microsoft Windows 11 platform, code is written and executed using **Anaconda software** and **Jupyter Notebook** (IDE).

5.2 Description of Dataset

In this study, the model is trained and tested using the **Android Mischief Dataset**, which was generated in the Stratosphere Laboratory at the Czech Technical University in Prague. The dataset can be downloaded from https://mcfp.felk.cvut.cz/publicDatasets/Android-Mischief-Dataset/. The network traffic from mobile devices infected with Android RATs is collected in the Android Mischief Dataset. In order to propose new detections to safeguard our devices, it aims to provide the community with a dataset to learn about and evaluate the network behavior of RATs. 8 packet captures from 8 running Android RATs are part of the dataset's current iteration.(*Android Mischief Dataset*, no date) The following 8 RATs are included in the most recent edition of the Android Mischief Dataset: Android Tester v6.4.6, DroidJack v4.4, HawkShaw, SpyMax v2.0, AndroRAT, Saefko Attack Systems v4.9, AhMyth and command-line AndroRAT.

5.3 List of Installed Python Packages

For the purpose of conducting our investigation, the following libraries and packages are installed:

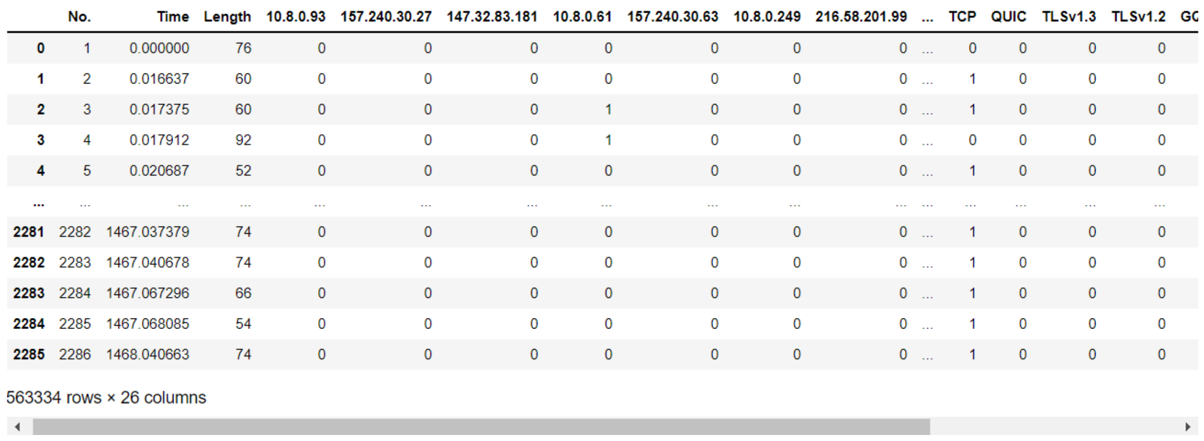
- **NumPy:** NumPy refers to numerical Python and the Python package NumPy is used to manipulate arrays. With NumPy, array objects should be up to 50 times faster than those of conventional Python lists.
- **Pandas:** Pandas is a Python module for analysis of data. It offers numerous data structures and procedures for working with time series and numerical data.
- **Matplotlib:** A comprehensive tool for creating static, animated, and interactive visualisations is offered by the Matplotlib toolbox for Python. Matplotlib makes both tough problems and simple ones doable.
- **Seaborn:** Matplotlib serves as the foundation for the Seaborn library, which plots graphs. It will be used to view random distributions.
- **Scikit-learn or sklearn:** Scikit-learn is probably the most useful Python machine learning library. The sklearn toolkit includes a number of efficient machine learning and statistical modeling methods, including as classification, regression, clustering, and dimensionality reduction.(*Libraries in Python - GeeksforGeeks,* no date)

5.4 Dataset Pre-processing

When all necessary programmes have been installed and data has been imported into the system, we start data pre-processing, which entails doing a basic statistical description of each feature, cleansing of the data collection involves checking for missing values and ensuring that they aren't there. We separate the data and labels from the data set after removing all of the null columns from the data set and filling in the null values with zeros. Since we are working with integer data, we remove the object type column from the data. We next convert the data and labels to a NumPy array because object type includes both strings and numbers.

5.5 Feature Selection

Selecting meaningful features for training our model is one of the most crucial tasks in a machine learning classification. To train our model with the right number of features and to disregard those that are not important in order to improve precision and shorten training times, this research uses feature selection. Features are evaluated using the **correlation-based feature selection (CFS) method**. A technique for choosing features in a dataset based on their connection with the target variable is called correlation-based feature selection (CFS). According to CFS, variables that have a strong correlation with the target variable are better suited for making predictions than features that have a weak correlation. CFS begins by figuring out the association between each feature and the desired outcome. After that, it employs a heuristic to order the features according to their connection with the desired variable. In order to provide a machine learning model with the most highly associated features, it finally chooses a subset of them. After separating the features of the data from the target set, the relevance of the features is assessed. Below figure shows the final selected features of the dataset on which our machine learning models has trained.



```
        No.         Time  Length  10.8.0.93  157.240.30.27  147.32.83.181  10.8.0.61  157.240.30.63  10.8.0.249  216.58.201.99  ...  TCP  QUIC  TLSv1.3  TLSv1.2  GC
   0      1     0.000000      76          0              0              0          0              0           0              0  ...    0     0        0        0
   1      2     0.016637      60          0              0              0          0              0           0              0  ...    1     0        0        0
   2      3     0.017375      60          0              0              0          1              0           0              0  ...    1     0        0        0
   3      4     0.017912      92          0              0              0          1              0           0              0  ...    0     0        0        0
   4      5     0.020687      52          0              0              0          0              0           0              0  ...    1     0        0        0
 ...    ...          ...     ...        ...            ...            ...        ...            ...         ...            ...  ...  ...   ...      ...      ...
2281   2282  1467.037379      74          0              0              0          0              0           0              0  ...    1     0        0        0
2282   2283  1467.040678      74          0              0              0          0              0           0              0  ...    1     0        0        0
2283   2284  1467.067296      66          0              0              0          0              0           0              0  ...    1     0        0        0
2284   2285  1467.068085      54          0              0              0          0              0           0              0  ...    1     0        0        0
2285   2286  1468.040663      74          0              0              0          0              0           0              0  ...    1     0        0        0

563334 rows × 26 columns
```

Figure 8 Feature Selection Output

5.6 Modelling (Training and Testing)

The train test split approach was used to divide the data and labels. The dataset will be split in half, with 70% of the data being used for training and 30% being utilised for testing. The goal of this study was to develop a model utilising ensemble learning to categorise Android intrusion detection software as malicious or benign. Our ensemble learning model is based on Random Forest, and our comparison analysis uses Decision Tree and KNN algorithms. The model has been developed, and the split data and labels have been fitted to the developed ML model. The model is then trained using the split data and labels from the training data set. The trained model was subsequently saved. Last but not least, we use the testing data set to determine the trained model's accuracy and F1 score.

6 Evaluation

The outcomes of using the proposed machine learning models will be covered in this section. In this part, the accuracy, f1-score, and confusion matrix outcomes from each model will be compared.

As the models are implemented and evaluated for effectiveness, the results of the implemented models are displayed below. The results show that Random Forest received the best score in remote access trojan (RAT) detection. Random Forest's accuracy is **96.44%** and its F1 score is **96.3%,** while Decision Tree's accuracy is **86.7%** and its F1 score is **86.12%**.

ML Model	Test Accuracy	Precision	Recall	F1-Score
Random Forest	96.44%	95.87%	97%	96.3%
Decision Tree	86.7%	93.62%	83.12%	86.12%
K Nearest Neighbor (KNN)	Discarded	Discarded	Discarded	Discarded

Table 2 Machine Learning Models Result

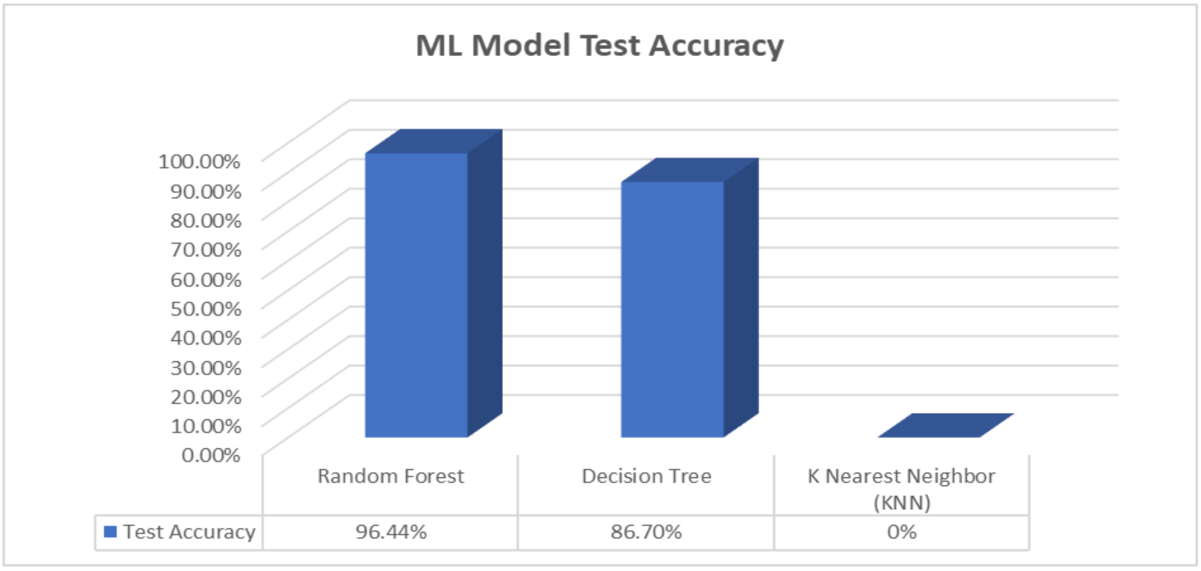


Figure 9 Bar graph of ML Model Test Accuracy

For a better understanding of the results, the various remote access trojan (RAT) types that we searched for throughout the intrusion testing are categorised.

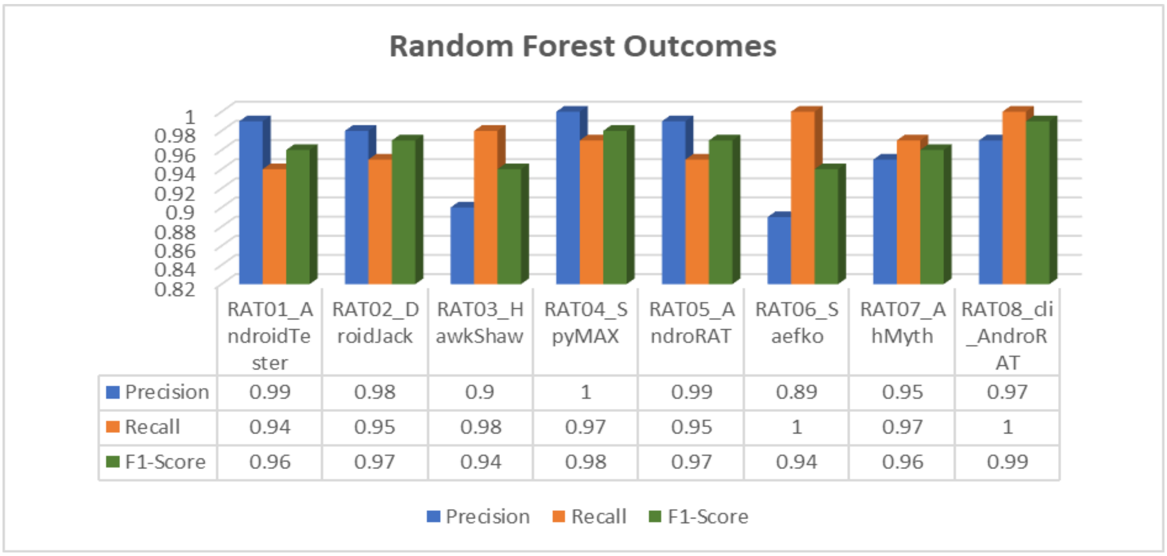
Class	Remote Access Trojan (RAT)
Class 0	RAT01_AndroidTester
Class 1	RAT02_DroidJack
Class 2	RAT03_HawkShaw
Class 3	RAT04_SpyMAX
Class 4	RAT05_AndroRAT
Class 5	RAT06_Saefko
Class 6	RAT07_AhMyth
Class 7	RAT08_cli_AndroRAT

Table 3 Class of Remote Access Trojan

6.1 Case Study 1: Random Forest Outcomes

Accuracy of Random Forest algorithm in detection of RAT = **96.44%**

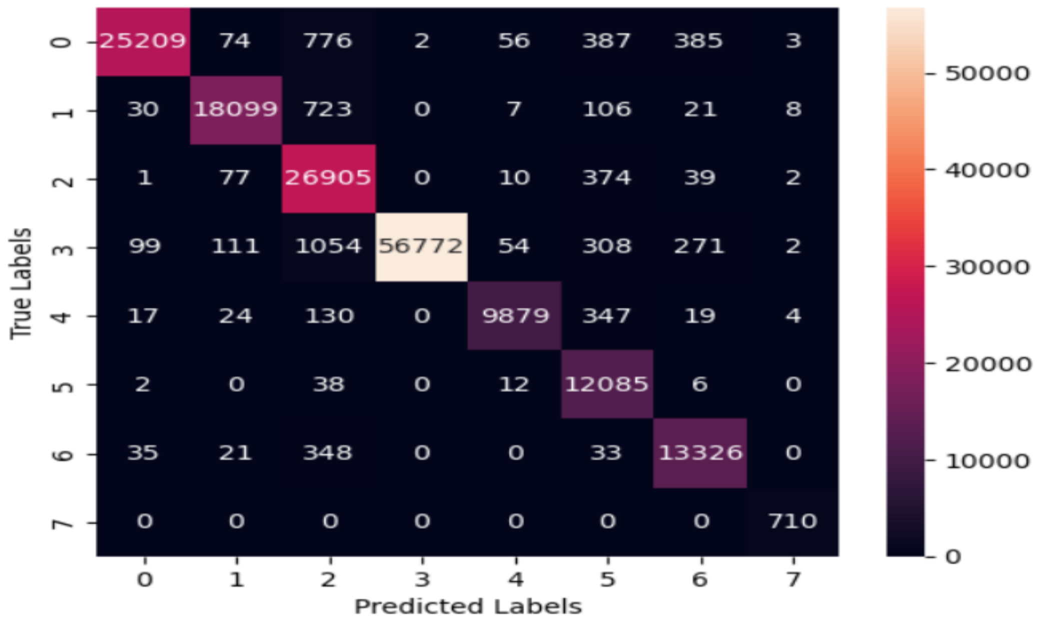
Below figure shows the individual remote access trojan (RAT) detection score of precision, recall, and F1 score by Random Forest model.



	RAT01_AndroidTester	RAT02_DroidJack	RAT03_HawkShaw	RAT04_SpyMAX	RAT05_AndroRAT	RAT06_Saefko	RAT07_AhMyth	RAT08_di_AndroRAT
🟦 Precision	0.99	0.98	0.9	1	0.99	0.89	0.95	0.97
🟧 Recall	0.94	0.95	0.98	0.97	0.95	1	0.97	1
🟩 F1-Score	0.96	0.97	0.94	0.98	0.97	0.94	0.96	0.99

Figure 10 Bar Graph of Random Forest Result

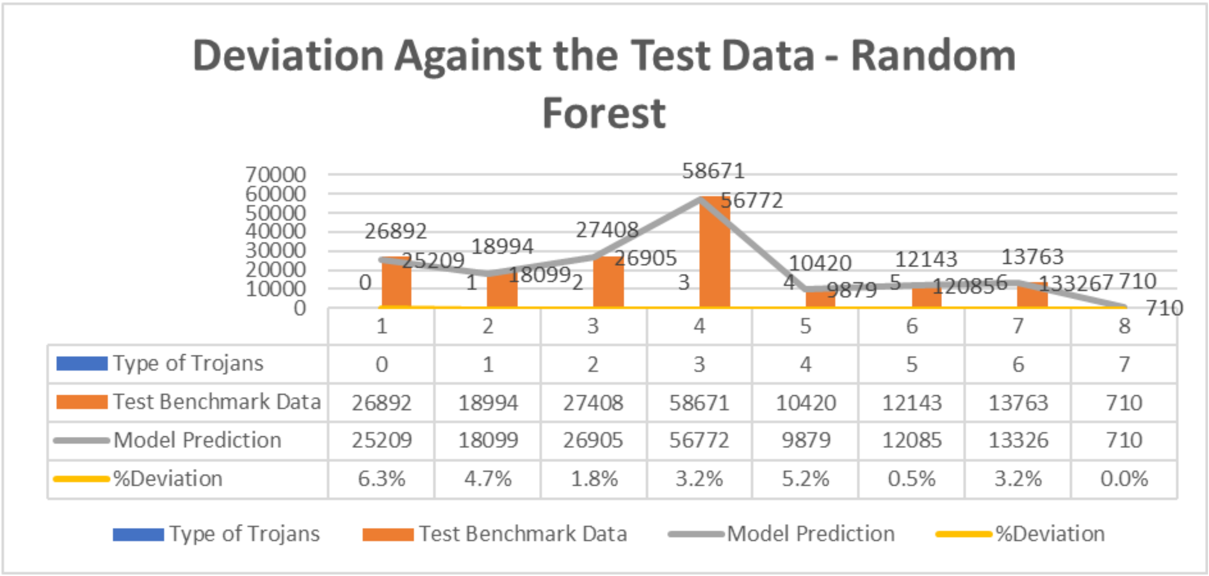
Below figure depicts the confusion matrix of the Random Forest classifier where highlighted diagonal boxes show the **true positive (TP)** value which means exact match the between predicted sample of the Random Forest trained machine learning model and the true test sample. **162,985** samples were classified correctly out of **169,001** samples and rest **6,016** were classified incorrectly.



	0	1	2	3	4	5	6	7
0	25209	74	776	2	56	387	385	3
1	30	18099	723	0	7	106	21	8
2	1	77	26905	0	10	374	39	2
3	99	111	1054	56772	54	308	271	2
4	17	24	130	0	9879	347	19	4
5	2	0	38	0	12	12085	6	0
6	35	21	348	0	0	33	13326	0
7	0	0	0	0	0	0	0	710

Figure 11 Confusion Matrix of Random Forest Model

Deviation of output from Random Forest Model against Test Data



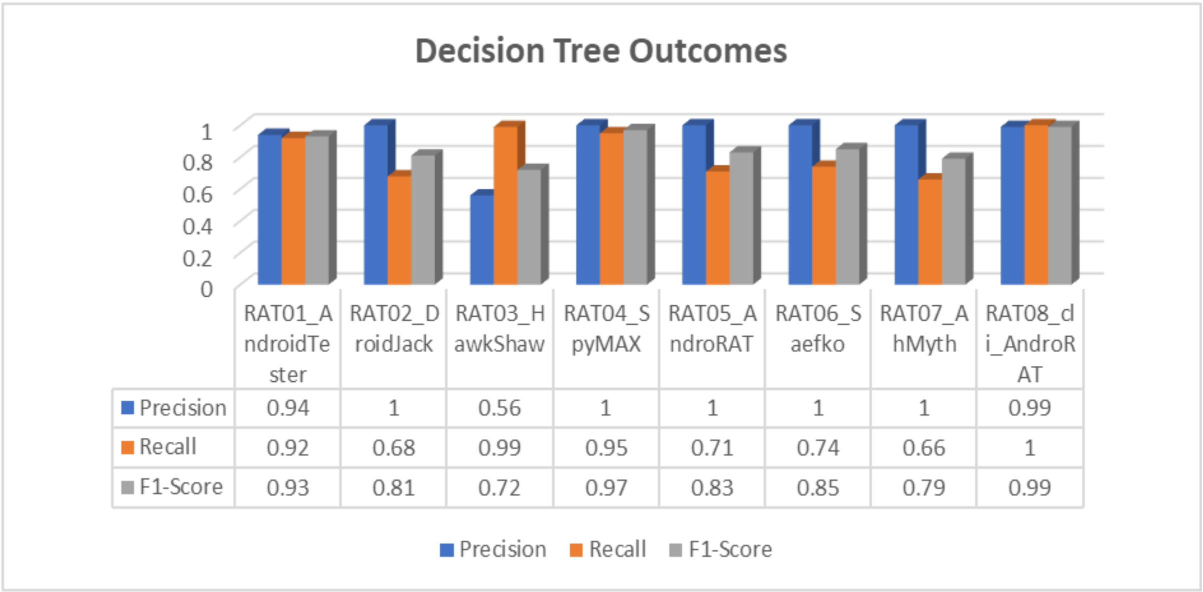
	1	2	3	4	5	6	7	8
Type of Trojans	0	1	2	3	4	5	6	7
Test Benchmark Data	26892	18994	27408	58671	10420	12143	13763	710
Model Prediction	25209	18099	26905	56772	9879	12085	13326	710
%Deviation	6.3%	4.7%	1.8%	3.2%	5.2%	0.5%	3.2%	0.0%

Figure 12 Output Deviation of Random Forest

6.2 Case Study 2: Decision Tree Outcomes

Accuracy of Decision Tree algorithm in detection of RAT = **86.7%**

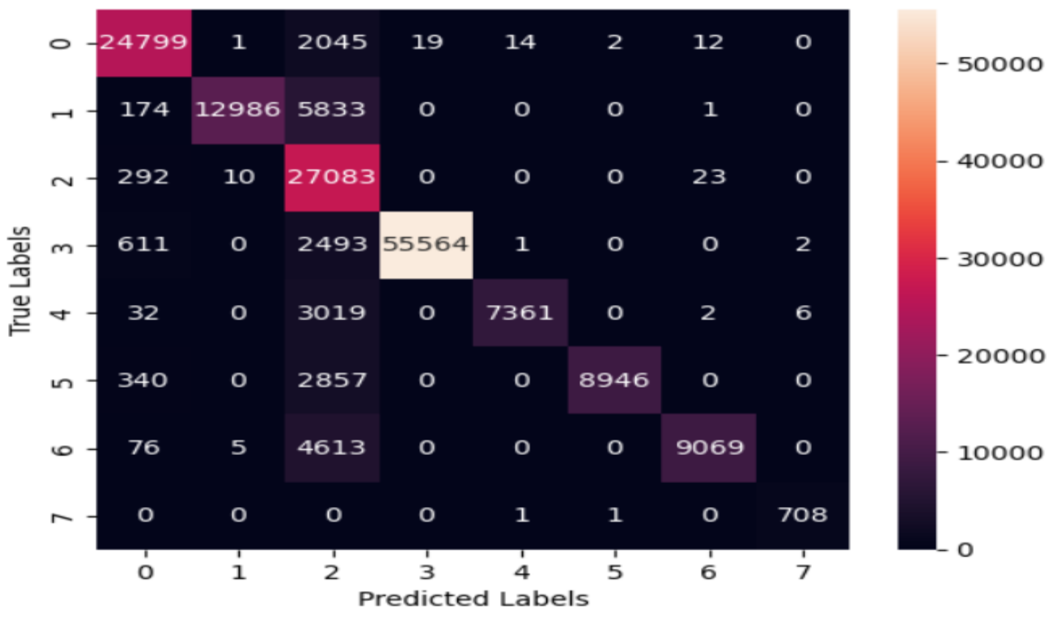
Below picture shows the individual remote access trojan (RAT) detection score of precision, recall, and F1 score by decision tree model.



	RAT01_AndroidTester	RAT02_DroidJack	RAT03_HawkShaw	RAT04_SpyMAX	RAT05_AndroRAT	RAT06_Saefko	RAT07_AhMyth	RAT08_di_AndroRAT
Precision	0.94	1	0.56	1	1	1	1	0.99
Recall	0.92	0.68	0.99	0.95	0.71	0.74	0.66	1
F1-Score	0.93	0.81	0.72	0.97	0.83	0.85	0.79	0.99

Figure 13 Bar Graph of Decision Tree Result

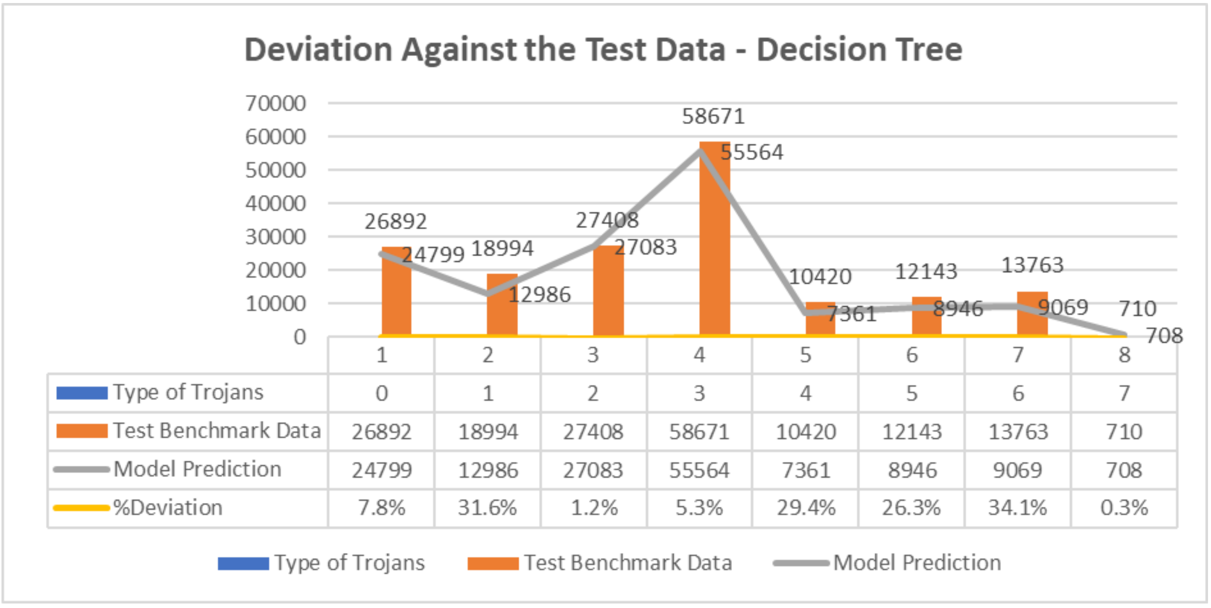
Below figure depicts the confusion matrix of the Decision Tree classifier where highlighted diagonal boxes show the **true positive (TP)** value which means exact match the between predicted sample of Decision Tree machine learning trained model and the true test sample. **146,516** samples were classified correctly out of **169,001** samples and rest **22,485** were classified incorrectly.



	0	1	2	3	4	5	6	7
0	24799	1	2045	19	14	2	12	0
1	174	12986	5833	0	0	0	1	0
2	292	10	27083	0	0	0	23	0
3	611	0	2493	55564	1	0	0	2
4	32	0	3019	0	7361	0	2	6
5	340	0	2857	0	0	8946	0	0
6	76	5	4613	0	0	0	9069	0
7	0	0	0	0	1	1	0	708

Figure 14 Confusion Matrix of Decision Tree

Deviation of output from Decision Tree Model againt Test Data



	1	2	3	4	5	6	7	8
Type of Trojans	0	1	2	3	4	5	6	7
Test Benchmark Data	26892	18994	27408	58671	10420	12143	13763	710
Model Prediction	24799	12986	27083	55564	7361	8946	9069	708
%Deviation	7.8%	31.6%	1.2%	5.3%	29.4%	26.3%	34.1%	0.3%

Figure 15 Output Deviation of Decision Tree

6.3 Discussion

When the models are put into practise and their effectiveness is assessed, it is discovered that Random Forest from Ensemble Learning surpasses the traditional machine learning Decision Tree and KNN models in terms of detecting remote access trojans (RATs) of Android phones. Since this thesis is the comparison between ensemble machine learning model and traditional machine learning model whether which model detect the Android network intrusion, we can observe that the Decision Tree model takes less time, but it has less accuracy and stability if compared with Random Forest's accuracy, stability and reliability as **Random Forest involves collection of Decision Trees with aggregate and single result** and **decreases the likelihood of overfitting.**

However, we have trained KNN model as well and we got that the KNN has the accuracy score and F1 score for the trained model, but **we have completely discarded this model as KNN is taking more than 3 hours to build the model in detecting the remote access trojan (RAT)** and we can say that the KNN is slow learner in Android intrusion detection and can not be implemented for Android device.

With reference to the figure 12 and figure 15, it is clearly observed that Random Forest model gave more accurate prediction against the test data.

7 Conclusion and Future Work

The objective of this research is to improve intrusion detection for Android devices by utilising machine learning and ensemble learning models. As per the objective, this implementation focuses on false negatives while comparing models according to their accuracy, precision, recall, and F- 1 score to conclude on the most suitable models in this process.
Each model computes and produces results with a different precision quickly. The confusion matrix is used to identify which model has the best accuracy and the less false negatives. We have put following machine learning models:

1. **Random Forest** from ensemble learning
2. **Decision Tree, KNN** from traditional machine learning

into practice for comparison and trained the models on the **Android Mischief Dataset**. This dataset contains **8 different Remote Access Trojans (RAT)**. After assessing their effectiveness, I have observed that **Random Forest** surpasses the traditional ML models in detecting remote access Trojan (RAT) with the test accuracy of **96.44%.** It is also having less false positive observed from the confusion matrix, whereas **Decision Tree** model is having accuracy limited to **86.7%** in detecting RAT. **We have trained KNN model as well and discarded it completely as KNN model was taking more than 3 hours and even more to build the model which is not at all suitable for the targeted remote access trojan for Android**.

However, Decision Tree has taken less time than Random Forest to build the model but less accurate whereas Random Forest is more accurate and best suited to the targeted remote access trojan detection. Therefore, I have concluded that **Random Forest** from ensemble learning is more efficient in detection of remote access trojan. It involves **collection of Decision Trees with aggregate and single result** concept that **minimises the possibility of overfitting,** and it is more **stable and reliable** compared to **Decision Tree** and **KNN** machine learning models.

7.1 Future Work

The scope of this thesis was to focus on detection of the intrusion in Android network and not the prevention or on how to stop the intrusion of Android devices. Preventive measures of the identified remote access trojans can also be implemented in future using ensemble machine learning along with other machine learning algorithms by using large dataset to make it more robust and efficient system for RAT detection. Additionally, a dataset with the right attributes for a particular RAT attack type is required to increase the accuracy of that attack type along with some hybrid ensemble learning model to get more accurate result. The model created in this thesis can be turned into appropriate mobile software that can operate in real time alongside the Android system to constantly look for any RAT infection and safeguard Android devices.

8 Acknowledgment

I would like to express my gratitude to Mr. Jawad Salahuddin, my supervisor, who supported me with his valuable support and guidance throughout the research. His timely guidance and consistent encouragement helped me understand the topic of research better and develop a deep understanding about the topic.
He inspired me to do better in my research. I genuinely appreciate his contribution in advancing my study and helping me drive and complete my project in time and with great details.

9 References

Ahmad, I., Ali Shah, S.A. and Ahmad Al-Khasawneh, M. (2021) 'Performance Analysis of Intrusion Detection Systems for Smartphone Security Enhancements', in *2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*. *2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pp. 19–25. Available at: https://doi.org/10.1109/ICSCEE50312.2021.9497904.

Ali Alatwi, H. *et al.* (2016) 'Android Malware Detection Using Category-Based Machine Learning Classifiers', in *Proceedings of the 17th Annual Conference on Information Technology Education*. New York, NY, USA: Association for Computing Machinery (SIGITE '16), pp. 54–59. Available at: https://doi.org/10.1145/2978192.2978218.

Aminanto, M.E. and Kim, K. (no date) ‘Deep Learning in Intrusion Detection System: An Overview’.

Android Mischief Dataset (no date) *Stratosphere IPS*. Available at: https://www.stratosphereips.org/android-mischief-dataset (Accessed: 10 December 2022).

Arora, S. (2020) ‘Decision Tree vs Random Forest in Machine Learning’, *AITUDE*, 8 February. Available at: https://www.aitude.com/decision-tree-vs-random-forest-in-machine-learning/ (Accessed: 9 December 2022).

Arp, D. *et al.* (2014) ‘Drebin: Effective and Explainable Detection of Android Malware in Your Pocket’, in *Proceedings 2014 Network and Distributed System Security Symposium. Network and Distributed System Security Symposium*, San Diego, CA: Internet Society. Available at: https://doi.org/10.14722/ndss.2014.23247.

Bhatia, T. and Kaushal, R. (2017) ‘Malware detection in android based on dynamic analysis’, in *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security). 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1–6. Available at: https://doi.org/10.1109/CyberSecPODS.2017.8074847.

Demontis, A. *et al.* (2019) ‘Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection’, *IEEE Transactions on Dependable and Secure Computing*, 16(4), pp. 711–724. Available at: https://doi.org/10.1109/TDSC.2017.2700270.

Edwards, G. (2020) *Machine Learning | An Introduction*, *Medium*. Available at: https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0 (Accessed: 30 July 2022).

Feizollah, A. *et al.* (2017) ‘AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection’, *Computers & Security*, 65, pp. 121–134. Available at: https://doi.org/10.1016/j.cose.2016.11.007.

Feng, P. *et al.* (2018) ‘A Novel Dynamic Android Malware Detection System With Ensemble Learning’, *IEEE Access*, 6, pp. 30996–31011. Available at: https://doi.org/10.1109/ACCESS.2018.2844349.

Ghorbanian, M. *et al.* (2013) ‘Signature-based hybrid Intrusion detection system (HIDS) for android devices’, in *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC). 2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pp. 827–831. Available at: https://doi.org/10.1109/BEIAC.2013.6560251.

Libraries in Python - GeeksforGeeks (no date). Available at: https://www.geeksforgeeks.org/libraries-in-python/ (Accessed: 14 December 2022).

Mahindru, A. and Singh, P. (2017) *Dynamic Permissions based Android Malware Detection using Machine Learning Techniques*, p. 210. Available at: https://doi.org/10.1145/3021460.3021485.

Martín, A., Lara-Cabrera, R. and Camacho, D. (2019) ‘Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the

OmniDroid dataset', *Information Fusion*, 52, pp. 128–142. Available at: https://doi.org/10.1016/j.inffus.2018.12.006.

Okfalisa *et al.* (2017) 'Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification', in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE). 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pp. 294–298. Available at: https://doi.org/10.1109/ICITISEE.2017.8285514.

Olson, P. (no date) *First-Known Targeted Malware Attack On Android Phones Steals Contacts And Text Messages*, *Forbes*. Available at: https://www.forbes.com/sites/parmyolson/2013/03/26/first-known-targeted-malware-attack-on-android-phones-steals-contacts-and-text-messages/ (Accessed: 13 December 2022).

Rahmat, S. *et al.* (2019) 'Network Traffic-Based Hybrid Malware Detection for Smartphone and Traditional Networked Systems', in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0322–0328. Available at: https://doi.org/10.1109/UEMCON47517.2019.8992934.

Rana, Md.S. and Sung, A.H. (2020) 'Evaluation of Advanced Ensemble Learning Techniques for Android Malware Detection', *Vietnam Journal of Computer Science*, 07(02), pp. 145–159. Available at: https://doi.org/10.1142/S2196888820500086.

Remote Access Trojan (RAT) (no date). Available at: https://encyclopedia.kaspersky.com/glossary/remote-access-trojan-rat/ (Accessed: 7 December 2022).

Sahs, J. and Khan, L. (2012) 'A Machine Learning Approach to Android Malware Detection', in *2012 European Intelligence and Security Informatics Conference*. *2012 European Intelligence and Security Informatics Conference (EISIC)*, Odense, Denmark: IEEE, pp. 141–147. Available at: https://doi.org/10.1109/EISIC.2012.34.

Shabtai, A. *et al.* (2012) '"Andromaly": a behavioral malware detection framework for android devices', *Journal of Intelligent Information Systems*, 38(1), pp. 161–190. Available at: https://doi.org/10.1007/s10844-010-0148-x.

Smartphone subscriptions worldwide 2027 (no date) *Statista*. Available at: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (Accessed: 7 December 2022).

Song, J. *et al.* (2016) 'An integrated static detection and analysis framework for android', *Pervasive and Mobile Computing*, 32, pp. 15–25. Available at: https://doi.org/10.1016/j.pmcj.2016.03.003.

What is a RAT (Remote Access Trojan)? | Definition from TechTarget (no date) *Security*. Available at: https://www.techtarget.com/searchsecurity/definition/RAT-remote-access-Trojan (Accessed: 13 December 2022).

Yerima, S.Y. *et al.* (2013) 'A New Android Malware Detection Approach Using Bayesian Classification', in *2013 IEEE 27th International Conference on Advanced Information*

Networking and Applications (AINA). *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, Barcelona: IEEE, pp. 121–128. Available at: https://doi.org/10.1109/AINA.2013.88.

Yerima, S.Y. and Sezer, S. (2019) 'DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection', *IEEE Transactions on Cybernetics*, 49(2), pp. 453–466. Available at: https://doi.org/10.1109/TCYB.2017.2777960.

Yuan, F. *et al.* (2013) 'Research of Intrusion Detection System on Android', in *2013 IEEE Ninth World Congress on Services*. *2013 IEEE Ninth World Congress on Services*, pp. 312–316. Available at: https://doi.org/10.1109/SERVICES.2013.77.

Zhou, W. *et al.* (2012) 'Detecting repackaged smartphone applications in third-party android marketplaces', in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. New York, NY, USA: Association for Computing Machinery (CODASPY '12), pp. 317–326. Available at: https://doi.org/10.1145/2133601.2133640.