MALWARE DETECTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

ANANTHAPADMANABHAN S

REG NO: TKM20MCA2007

In partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS JULY 2022

DECLARATION

I ANANTHAPADMANABHAN S hereby declare that the project report**" MALWARE DETECTION USING MACHINE LEARNING"**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Prof. ALSHAINA S. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University

Place: Kollam ANANTHAPADMANABHAN S

Date: 18.07.2022

Thangal Kunju Musaliar College of Engineering
Dept. of Computer Applications

C E R T I F I C A T E

This is to certify that, this report titled ***MALWARE DETECTION USING MACHINE LEARNING*** is a bonafide record of the **Project Work** presented by **ANANTHAPADMANABHAN S (TKM20MCA2007)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **MASTER OF COMPUTER APPLICATIONS** in **APJ Abdul Kalam Technological University** .

Internal Supervisor | Head of the Department | External Examiner

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a memorable one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T A SHAHUL HAMEED**, Principal of T.K.M College of Engineering for giving me an opportunity to present my project. I would like to thank **Dr. Fousia M Shamsudeen**, Assistant Professor and Head of the Department, MCA, TKMCE, for her constant support and encouragement throughout the project work.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my guide **Prof. ALSHAINA S**, Assistant Professor, MCA, TKMCE, for her expert guidance, cooperation, and immense encouragement. I also extend my thanks to the entire faculty and staff of the Department of Computer Applications, TKMCE, who has encouraged me throughout this work.

I also express my thanks to my loving parents, brother, and friends, for their support and encouragement in the successful completion of this project work.

ANANTHAPADMANABHAN S

ABSTRACT

Malware is any programme that is intended to cause harm to other software or hardware. The growth of malware poses a grave threat to cyber security. Malware can result in sluggish computation, frequent system crashes, memory consumption, etc. Protection should increase at the same rate as technological development. In the field of cyber security, machine learning has played a significant role in the detection of malware. In this proposed system, machine learning algorithms such as random forest and gradient boosting are utilised to combat the rising prevalence of malware. Random forest creates a collection of decision trees, each of which is a fundamental forecast, and combines the results into a single output. In gradient boosting, successive decision trees are constructed to address the deficiencies of the preceding trees. Gradient boosting accumulates results along a route. Using datasets of malware and non-malware samples, a malware classifier capable of detecting the presence or absence of malware is constructed, and the accuracy of both algorithms is evaluated to determine which method is more efficient.

Contents

List of Figures

Chapter 1

Introduction

Malware is essentially a programme designed to interfere with operations and steal data, which can result in loss of the privacy, unauthorised access, and so on. Malware includes deceptive adware, malicious software, computer viruses, worms, and Trojan horses, among others. Computer systems are attacked by hackers eager to display their skills. As a result of recent advancements in computer software and technology, the frequency of attacks on computer systems has increased dramatically. There are numerous new Internet services available today. According to multiple sources, the impact of malware is growing at an alarming rate. Today, ransomware and malware are utilised to wreak havoc on commercial and government institutions. Variable-sized and shaped grave threats and obstacles. Anti-malware software is the most important cyber security practise for organisations and individual users. This is due to the fact that a single attack can put your data at risk and cause significant damage. Malware is rapidly evolving, making it more difficult for security researchers and scholars to fortify cyber defences. Malware developers utilise register renaming, code reduction, code sequence, and code extension as conversion techniques. Signature-based and behavior-based techniques are two prevalent malware detection methods. A signature is essentially a collection of distinguishing characteristics, such as fingerprints or other characteristics that distinguish executable files. Typical antivirus applications that rely on signature-based techniques have numerous flaws. Typically, signature-based methods cannot identify unknown or novel malware file types. Instead of relying on signatures, security research offers a behavior-based approach to address these issues.

Behavior-based detection examines the attributes and behaviour of a file to determine whether or not it is malicious software. Due to the fact that machine learning is well-suited for processing vast amounts of data, a number of researchers have used it to overcome the limitations of existing antivirus engines and to construct defences against new attack types. The proposed research investigates the use of machine learning algorithms in cybersecurity and how they can be employed to detect malware. After training, the system constructs a classifier that identifies whether a file has been compromised by malware based on patterns found in malware and non-malware files. For the purpose of resolving the issue, the system implements various machine learning algorithms, such as random forests and gradient boosting. The precision of both algorithms is identified and compared[1].

1.1 Evolution of Malware

One of the most essential things for users to do in terms of cybersecurity is to protect themselves from malware. This is the situation because a single assault has the potential to compromise data and cause considerable loss all by itself. The proposed research investigates the use of machine learning algorithms in cybersecurity and how they can be employed to detect malware. For the purpose of resolving the issue, the system implements various machine learning algorithms, such as random forests and gradient boosting. The accuracy of each algorithm is analysed and evaluated against one another.[3]. Malware encrypts and modifies portions of the source code, which makes it difficult for researchers to examine the software. The crypter contains code fragments for encrypting and decrypting malicious code. Each time it is released, metamorphosing malware rewrites its code. Numerous conversion techniques, such as register renaming, code sequence, code expansion, code reduction, and trash code insertion, may be utilised by malware developers. As a result of the combination of the aforementioned techniques, the volume of malware continues to increase, and forensic analysis of malware instances becomes more time-consuming, expensive, and complex. Signature-based, heuristic, or behavior-based antivirus programmes are flawed. Identifiable characteristics or groups of attributes that distinguish an executable file, similar to a fingerprint. However, the signature-based method cannot identify

unidentified malware strains. Security researchers propose behavior-based detection as a solution to these problems. It analyses the file's properties and behaviour to determine whether or not it contains malware, a lengthy process[1].

1.2 Malware Detection

Malware is disseminated by hackers with the intent of convincing people to install it. The user is unaware of the nature of the programme because it appears authentic. It is often perceived as secure and installed, although posing a serious threat. Once it appears on your screen, it can spread and hide in a large number of files, making it incredibly difficult to identify. To obtain access to and collect your sensitive and useful data, you can connect directly to the operating system and begin encryption.

1.3 Need for Machine Learning in Malware Detection

Machine learning has revolutionised a slew of industries in the last decade, not the least of which is cybersecurity. The latest malware threats may be detected more quickly and accurately with AI-powered anti-malware technology, according to cybersecurity experts. Many investigations on malware detection systems that exploit machines support this hypothesis. Recently reported education 7720 research papers were published in 2018, an increase of 95 percent from 2017 and 76 percent from 2016. According to Google Scholar, a rise is anticipated in 2010. This growth in the number of surveys is due to a combination of causes, including an increase in publicly labelled malware feeds, a rise in computing power as prices fall, and the development of machine learning applications. Success in a range of endeavours, including visual computations and speech recognition. In many instances, the capacity to learn from raw data outperforms neural networks. The influence of neural networks on malware has recently been demonstrated in machine learning for cybersecurity[2].

1.4 Current Antivirus Software

Antivirus software is utilised to prevent, detect, and eliminate malicious software such as computer viruses, computer worms, trojans, spyware, and adware. Typically, antivirus engines employ a range of tactics. Signature-based discovery includes examining executable code for known data patterns. However, new viruses that are not signed can infiltrate your machine. By executing the file in a sandbox and studying the results, certain antivirus applications can also generate predictions. Antivirus software can frequently hinder the operation of your machine. At the kernel level of a trusted operating system, erroneous decisions can lead to security vulnerabilities. Antivirus software that performs heuristic detection must achieve a balance between false positives and false negatives in order to be effective. Historically, antivirus software has relied heavily on signatures to identify malicious software. However, due to the new type of malware, the signature-based method is worthless. [4]. Despite the fact that ordinary antivirus software may efficiently prevent virus outbreaks, any compromise can be devastating for a major enterprise. Manufacturers of viruses utilise oligomorphic, polymorphic, and metamorphic viruses. These viruses are encrypted or partially modified to obfuscate such that their signatures do not match those in the viral dictionary. Anti-virus software was much less successful in 2007, particularly against known and unknown assaults, according to a 2007 study. The percentage of detection rose from 0 to 50 percent in 2006 to 20 to 30 percent in 2007. Changes in the virus vendor's objectives worsen this situation.

1.5 Problem Statement

Machine learning algorithms such as random forest, support vector machine, and gradient boosting are used in the suggested system to identify malicious code as a threat. In addition to analysing the efficiency of the algorithms, the most efficient and precise algorithm is identified.

1.6 Objective

Malware poses a grave risk to a user's computer system since it can steal important data and disable security. Machine learning algorithms, such as random forest and gradient boosting, are directly applied to malware datasets in the proposed strategy. Next, assess the role of different malware detection algorithms based on precision.

Chapter 2

Literature Survey

Literary studies are an in-depth examination and analysis of literature on a certain subject. When identifying research questions from a review of literature, attempt to answer these questions by discovering and studying the pertinent literature. In literary studies, the ability to gain fresh ideas through the study of new data is crucial. Literature reviews are a comprehensive and current body of knowledge, as found in scholarly books and journal articles, along with a description of the topic. In college, there are two forms of literary criticism that can be written. On the one hand, it is an individual writing project for the course, while on the other, it is a lengthy essay (usually a treatise or research report). The distinction between these two categories can be understood by reading a published literature review or the first chapter of a treatise in your field. Consider the structure of their discourse and how they approach the problem.

Purpose of the Literature Review

- By picking relevant, useful, significant, and valid high-quality articles and research and compiling them into a comprehensive report, readers have easy access to research on a specific issue.
- This is an excellent beginning point for new-field researchers. By requiring you to summarize, assess, and compare your own study in a specific field.
- This discourages researchers from duplicating already completed work. This may offer hints on future study directions or potential focus areas.

- This may offer hints on future study directions or potential focus areas.
- It emphasises the principal findings.
- It identifies discrepancies, gaps, and inconsistencies in the literature.
- Provides a constructive analysis of the methodologies and approaches of other scholars.

2.1 Related Works

As a part of the project, the following journals and papers were referred:

1. A survey on machine learning-based malware detection in executable files was proposed by Jagsir Singh and Jaswinder Singh (2021)[5] stating various challenges in developing a malware classifier. The method uses machine learning to teach malware classification systems.In addition, there is discussion regarding the development of an efficient malware detection system by resolving several malware detection challenges. Static, dynamic, and hybrid malware detection algorithms are all included in the proposed framework. In static analysis, features are retrieved from malware samples without executing them. After analysis, characteristics such as hash value, N-grams, opcodes, and strings are extracted. Then, malware detecting software is created. Static methods are efficient and have a low false positive rate. Nonetheless, there are limitations to managing obfuscation strategies while acquiring static qualities. In the dynamic technique, harmful files are executed and malware's run time actions are identified. File system activity, registry key modifications, process execution, and network activity are all examples of run time characteristics. However, real-time implementation is time-consuming. The hybrid method to malware detection employs both static and dynamic techniques to detect both new and known malware.

2. Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection, proposed by Luca Caviglione and Michal et al(2021)[11], provides insight into malware development. This paper examines significant advancements in the evolution of detection systems, with a focus on machine learning techniques. Existing countermeasures are challenged by the complexity and diversity of malware, demanding their ongoing improvement. This research begins with a comprehensive

examination of previous malware-related investigations and their respective detection approaches, exposing a race between the two obstacles. Examine the growth of modern dangers in communication networks, focusing specifically on approaches that employ information concealment. Following this, presents a complete assessment of important advancements in the development of detection systems, with a focus on machine learning techniques.

3. The survey proposed by Harsha Latha, R Mohanasundaram, and colleagues identifies malware and malware varieties, as well as innovative machine learning methods for categorization and detection efficacy. There are several approaches to classify new malware based on existing code.[12]. Traditional tactics are inadequate for contending with newly released malware samples. Increasingly more anti-virus software provides defence against malware, but the zero-day attack has not yet occurred. Algorithms based on machine learning are utilised to improve the mechanism and yield positive test results. Traditional signature-based approaches have failed to compete with modern malware. This article outlines malware and its classifications and suggests a novel way for measuring the effectiveness and efficiency of machine learning-based analysis. It emphasises the major challenges connected with malware detection and classification.

4. The rise of machine learning for detection and classification of malware: research developments, trends and challenges. This framework, proposed by Daniel Gibert, CarlesMateu, and Jordi Planes (2020) et al[13]., describes in detail the methodologies and characteristics of conventional machine learning workflows for malware detection and categorization. It analyses the difficulties and limitations of conventional techniques to machine learning. It examines the difficulties of conducting research and the as-yet-undiscovered difficulties of modern methods. Machine learning techniques for malware detection are the focus of this survey, with a specific emphasis on deep learning. Traditional machine learning workflows for malware detection and classification are described in detail in this paper. Research problems and unsolved issues with current approaches are also examined in this study. This study helps researchers comprehend the topic of malware detection, as well as the new developments and research directions being followed to address the problem.

5. Malware Detection Module using Machine Learning Algorithms to Assist in

Centralized Security in Enterprise Networks was proposed by Priyank Singhal(2012) et al[14]. Rather than just scanning data, this antivirus engine can also learn and identify files as potentially malicious. Machine learning approaches are used to rate the files based on the system API calls made by various legitimate and malicious executables. The level of security risk. In spite of the fact that this system requires a significant amount of processing power, it is highly successful when installed centrally to protect commercial networks that may be particularly sensitive. Anti-virus software that can not only scan data but also learn and recognise potential infections in files is suggested by this study. With the help of machine learning, it is possible to extract the system API calls made by a variety of regular and malicious applications on the scale of potential danger to security. When used centrally to protect networks that are more vulnerable to such attacks, this technology, despite its processing requirements, is quite successful.

Chapter 3

Methodology

Malware Detection using machine learning, this project mainly aims at classifying a data-file as malware or not. This approach uses machine learning model to improve the classification accuracy.

3.1 Proposed System

In this work, a malware detection system is constructed. It includes preprocessing, training, classification, and output, among other processes. The framework receives a set of data files as input. In order for the system to become acquainted with both types of files, a training data set including both malicious and useful files is provided as input. The identification of malware or malicious files from a set of data files is the most crucial component of the design. The following training dataset is used to classify both dangerous and beneficial files. Random forest and gradient boosting classification methods are employed in this work to create a model. When new data files are presented to the proposed model, it determines whether or not they include malware. The proposed framework computes and compares the accuracy of these two machine learning methods and predicts which technique is superior for detecting malware. Using confusion matrices, the performance of the models is assessed.

3.2 System Architecture

The system architectural design is used to abstract the software system's layout, relationships, constraints, and component boundaries. It provides a comprehensive view of the physical deployment of the software system, making it an essential tool.

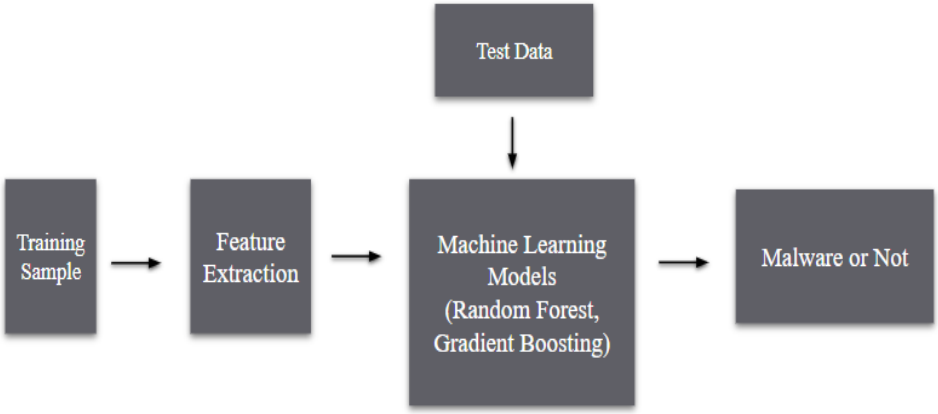


Figure 3.1: Architecture

3.3 Dataset

In the suggested system for malware detection, two datasets are used. Malware dataset (Malwaredata. Csv) is the dataset used. All non-essential null values and columns will be removed from the dataset to render it worthless. This model is inserted into maliciously valid files. Similarly, the collection contains a mixture of harmful and non-malicious files. The trained model classifies them based on the pattern analysis.

3.4 Data Preprocessing

Data preprocessing is preparation of unstructured data for a machine learning model. When constructing a machine learning model, this is the first and most critical stage.

Various situations, such as attributes with missing data or attributes with no values, have been taken into account during the preprocessing of the dataset to make it more flexible. Misleading data, such as a different datatype or a different format, has also been taken into account. The malware data set contains 56 attributes in total. This results in an enormous feature space for a relatively small number of data points. Therefore, data preprocessing is essential to the success of classifier models.

3.5 Training the model

The next step is selecting the most relevant features. Does feature selection for multiple reasons, including accelerating the training of machine learning algorithms, reducing the model's complexity to make it easier to interpret, and reducing over fitting. If the proper subset is chosen, feature selection improves the model's precision.

Fifty six logical characteristics were retrieved from the dataset, and 14 of those features were chosen for classification. For selection, employs an additional tree classifier (Extremely random tree classifier). Extra Trees Classifier is a collective learning method that generates classification results by merging the outcomes of a "forest" of multiple unconnected decision trees. The decision tree is embedded within the random forest. Each successive decision tree in the forest of trees is produced using the initial training samples. Then, at each test node, each decision tree separates the data into k features depending on a mathematical parameter (often the Gini coefficient), with a random selection of trees from the feature set. You must select the finest characteristics. A random sampling of this function produces decision trees with no correlation. In order to apply the aforementioned forest structure for feature selection, a uniform total reduction of the mathematical criterion used to calculate the feature distribution (using the composite Gini coefficient) is conducted while generating the forest for each feature. This value is known as the function's Gini value. Each feature is ranked according to its importance in descending order for feature selection, and the user selects the top k features based on this ranking.

$$Gini(p) = 1 - \sum_{i=1}^{n} (p_i)^2$$

3.6 Testing the Model

In order to ensure that the programme is free of bugs, the procedure of software testing is necessary. The objective of software testing is: detecting errors or missing requirements relative to the actual requirements. The final objective is quality assurance. The results of performed tests are compared to the expected document. A software product that has been thoroughly tested guarantees it is dependable, secure, and has high performance, results in time savings, economical, and customer satisfaction. This project phase's primary objective is to develop a churn prediction model. In an 80:20 split, the dataset is divided into training and testing parts. After that, it is put to the test with a real dataset, and the outcome confirms that the training procedure was successful.

3.7 Build Model

Algorithms used in the proposed system are:

1. DECISION TREE
2. RANDOM FOREST
3. SVM
4. XGBOOST

3.7.1 Decision Tree

When used to handle classification issues, the Supervised Learning technique of the Decision Tree can also be utilised to address regression issues. Tree-like structure, with core nodes representing the dataset's features, branches representing decision rules, and leaf nodes indicating the conclusion. The Decision Node and the Leaf Node

are both nodes in a decision tree. To make a decision, a decision node is used to find all possible solutions to a problem or decision based on the given conditions are represented graphically. Because it features a base node that branches out in a tree-like layout, it's termed a decision tree. Leaf nodes, on the other hand, record the results of these deci- Leaf nodes, on the other hand, have no more branches. The attributes of the submitted dataset are used to inform the judgments or tests. Any and all possible solutions to a problem or choice are represented graphically using the supplied criteria as a basis for their construction. Decision trees are known as such because they have a root node that branches out and produces a tree-like structure like a real tree[5].

Gini Impurity:

Gini impurity is a statistic used to evaluate how the nodes of a decision tree are divided according to the properties of a data set. Specifically, a Gini inclusion in a data set is a value between 0 and 0.5, and the chance that more random data will be misclassified, given a random class designation, depends on the data collection's class distribution[7].

3.7.2 SVM Algorithm

SVM is a supervised machine learning technique applicable to classification and regression tasks. However, they are predominantly employed for problem classification. Using the SVM algorithm, each data point in n-dimensional space is represented. The value of each function in this instance is a given coordinate value. Then, classification is achieved by locating a hyperplane that effectively divides the two categories(see snapshot below).

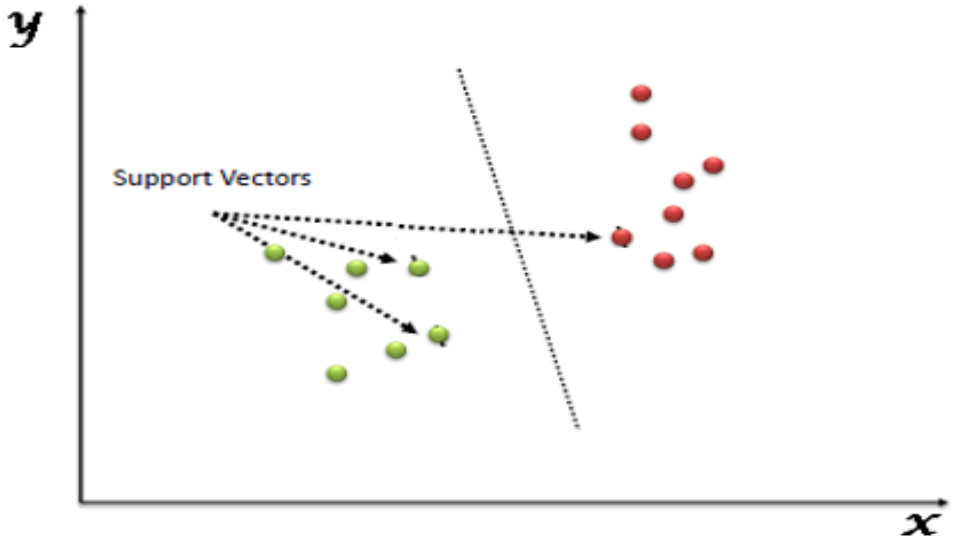


Figure 3.2: SVM Algorithm

The support vector consists of a single note's coordinates. The SVM classifier is the optimal dividing line between two classes (superlevel or line).

3.7.3 Gradient Boosting

Gradient boosting is a prominent approach for improvement. Each predictor corrects the inaccuracy of the preceding prediction by lengthening the gradient. In contrast to Adaboost, the weights of training instances are not altered. Each predictor is instead trained using the labels of its ancestors' descendants. Regression and classification issues are used in gradient boosting to create less accurate prediction models. Similar to other optimization strategies, the model is constructed and developed in phases so that each differentiable loss function may be maximised. One of the most effective methods for developing predictive models is gradient boosting. Many modes are incremental, sequential, and sequential, as is taught by gradient amplification. For the loss function's slope, the slope grows.[11].

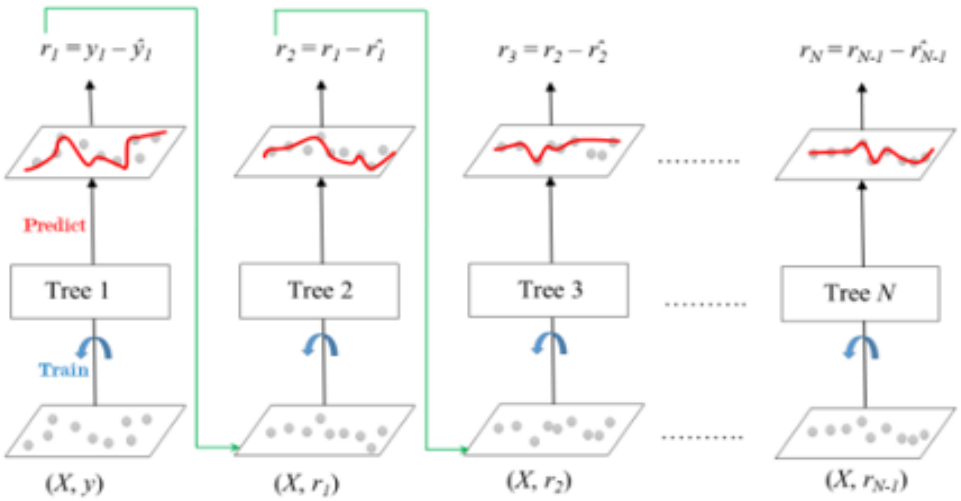


Figure 3.3: Gradient Boosting

Why ensemble learning?

Gradient boosting is a method for group learning. Depending on the findings of a single machine learning model may not be sufficient. Learning Group provides a mechanism for integrating the prediction abilities of a large number of pupils. The result is a single model that combines the results of multiple models. The group's models, also known as core instructors, might be derived from the same learning process or from several learning algorithms. Common forms of group learners include mobilisation and reinforcement. Both of these techniques are applicable to a variety of statistical models, although their principal application is in decision trees. Before delving into the idea of ascension, let's briefly examine bagging[10].

Bagging

Decision trees are among the most straightforward models to analyse, yet their behaviour varies substantially. Consider a training data set that has been randomly divided in half. Then, utilises each portion to train the decision tree, yielding two models. Comparing the two models yields distinct findings. Due to this characteristic, decision trees are referred to as height variance related. By increasing grouping or clustering, student variation can be reduced.[6].

Boosting

In boosting, trees are constructed consecutively, with each successive tree intended to reduce the mistakes of the preceding tree. Each tree gains knowledge from its predecessor and corrects any leftover faults. Consequently, the tree growing next to the sequence learns from the modified residue. Basic Boost learners are poor learners, extremely prejudiced, and somewhat superior to random guesses in their prediction skill[6].

3.7.4 Random Forest

Random Forest is a popular technique for supervised machine learning classification and regression applications. The decision tree is formed from multiple samples, and in the case of regression, the rating and mean are set primarily by popular vote. The flexibility of the Random Forest method to handle data sets containing continuous variables and categorical variables is one of its most essential characteristics. Improve performance with ranking issues[9].

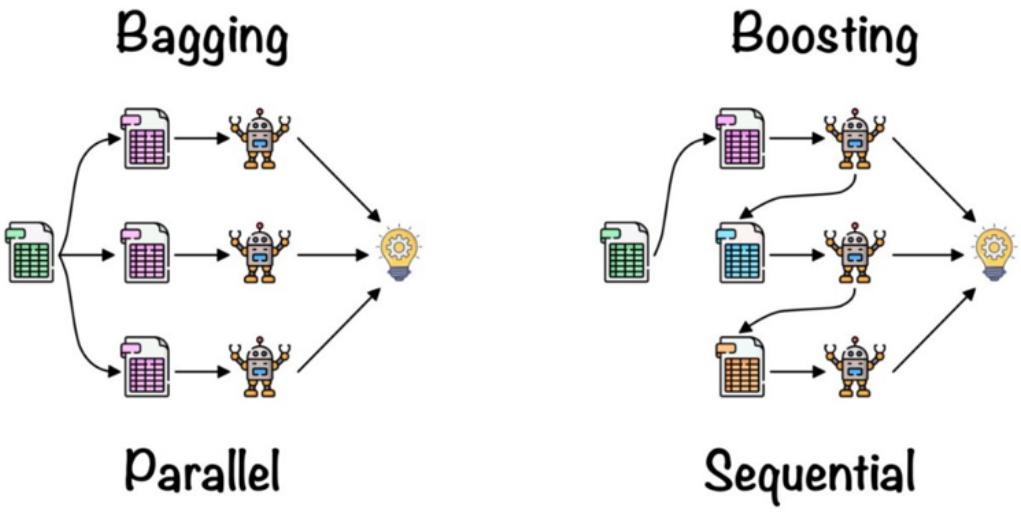


Figure 3.4: Bagging Boosting

Important Features of Random Forest

1. Not all attributes/variables/features are examined when generating a tree; each tree is unique.

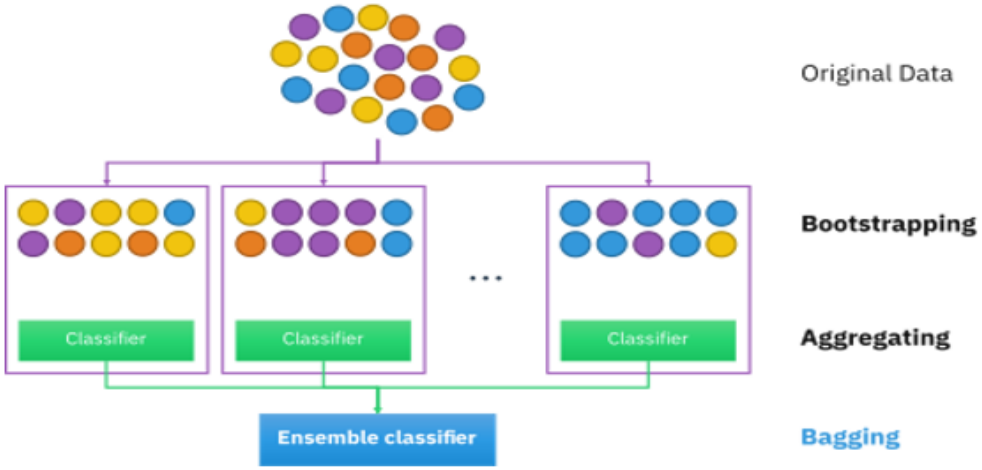


Figure 3.5: Ensemble Classifier

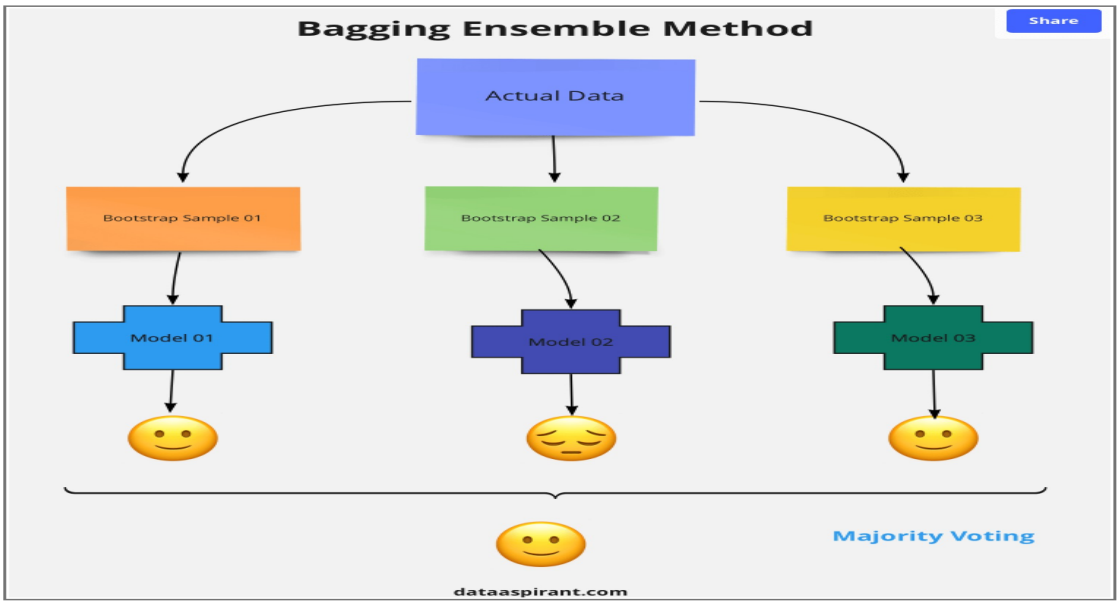


Figure 3.6: Bagging Ensemble Method

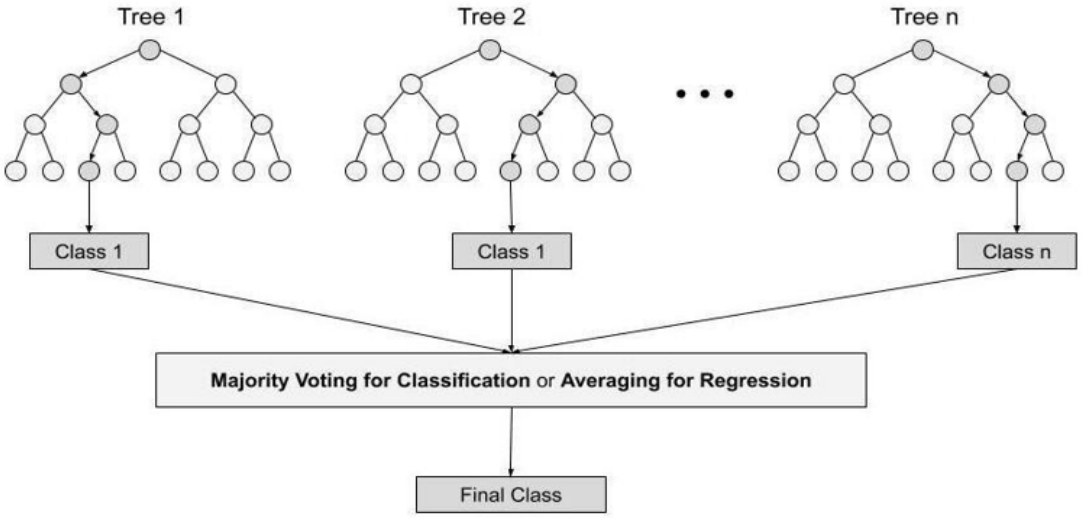


Figure 3.7: Random Forest

2. Each tree does not describe every characteristic, hence restricting the feature space.

3. Each tree is built independently of its own data and attributes. This means that you can develop a random forest using your CPU's maximal processing power.

4. In a random forest, there is always 30% of the data that does not appear in the decision tree, hence training and testing data need not be separated.

5. The results are stable because they are based on the majority or average.

Chapter 4

Result and Discussion

After training and testing both algorithms, it is discovered that their output was extremely accurate. Gradient boost classifier and random forest classifier with confidence intervals of 98.76 and 99.311 percent, respectively, were utilised to train the data. This concludes that the false positive rate is 0.5193 and the false negative rate is 0.8136. This method's confusion matrix shows that there are 0.768 false positives and 2.3099 false negatives, respectively.

```
RandomForest : 0.994386091996
GradientBoosting : 0.988373777617
GNB : 0.702897500905
DecisionTree : 0.990981528432
LinearRegression : 0.54036008649
Adaboost : 0.986381745744
```

Figure 4.1: Accuracy

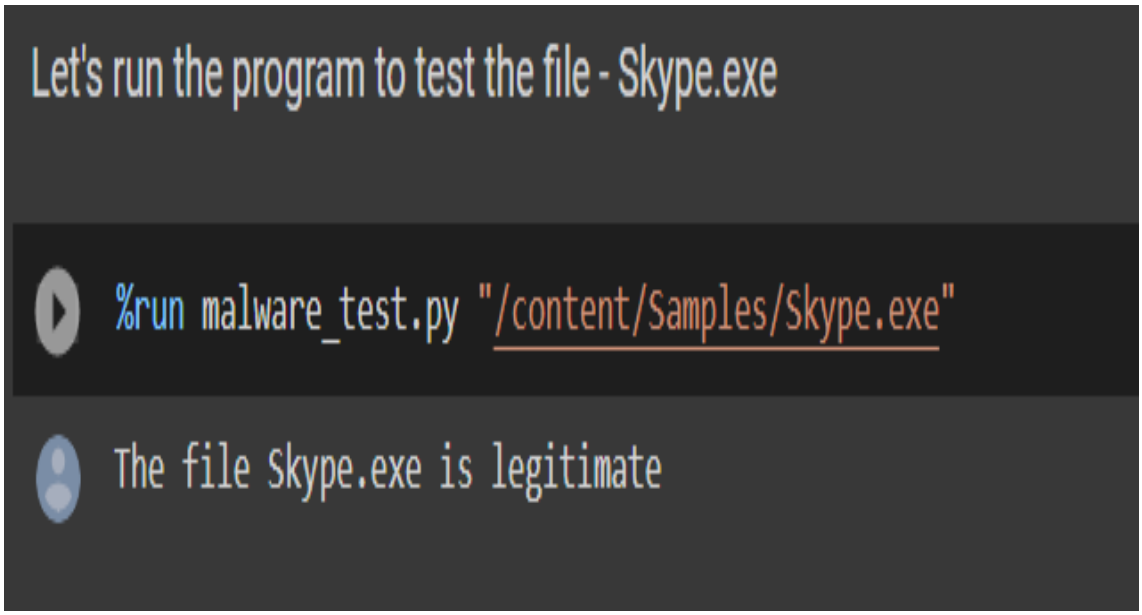


Figure 4.2: Legitimate file identified

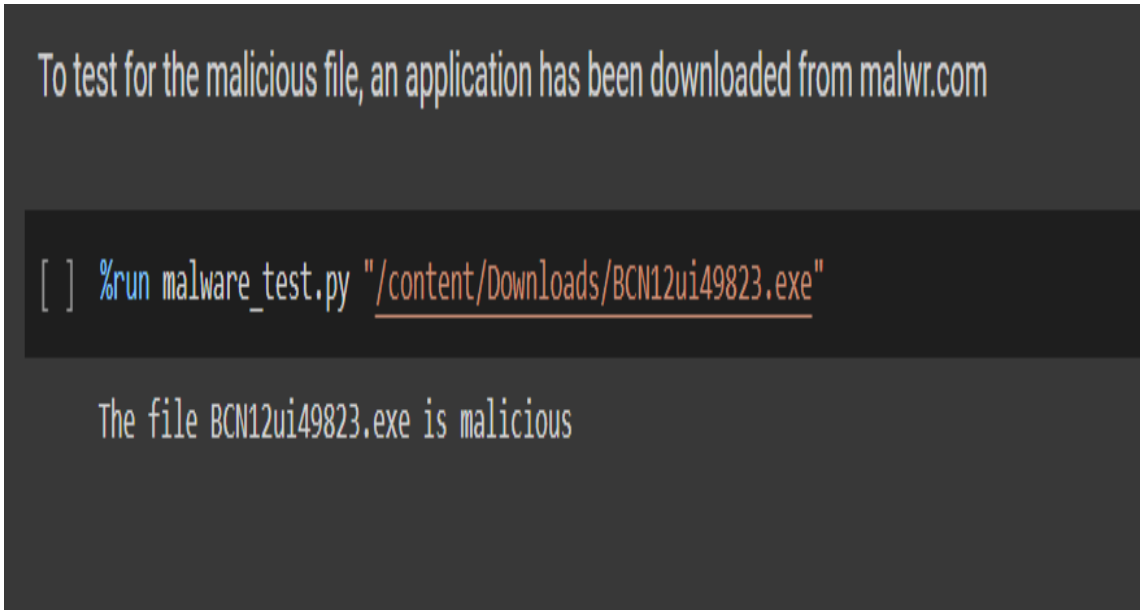


Figure 4.3: Malware file detected

Chapter 5

Conclusion

Designing a machine learning system to detect as many malware samples as feasible while guaranteeing no false positives occurs is often the primary objective. It is quite close to achieving the objective, but false alarm rate is still not zero. Various deterministic exception mechanisms must be added to this system before it can be incorporated into a highly competitive product. Machine learning-based malware detection, in opinion, enhances rather than replaces the existing detection methods of antivirus firms. Due to the speed and memory constraints of over-the-counter antivirus software, the most precise algorithm should be employed.

5.1 Limitations & Future Enhancements

- This type requires a very powerful processor, making it difficult for home users to operate and necessitating a higher-end system.
- This model does not detect all dangerous files automatically. The user must manually determine whether the file contains malware.
- Future upgrades may feature an auto-detection system that determines automatically whether newly added files are malicious software.

References

[1] Christodorescu,M., Jha, S., 2003. Static analysis of executables to detect malicious patterns. In: Proceedings of the 12th USENIX Security Symposium. Washington .pp. 105-120.

[2] Filiol, E.,2005. Computer Viruses: from Theory to Applications. New York, Springer, ISBN 10: 2- 287-23939-1.

[3] Filiol, E., Jacob, G., Liard, M.L., 2007: Evaluation methodology and theoretical model for antiviral behavioral detection strategies. J. Comput. 3, pp 27–37.

[4] Kephart, J., Arnold, W., 1994. Automatic extraction of computer virus signatures. In: Proceedings of 4th Virus Bulletin International Conference, pp. 178–184.

[5] A survey on machine learning-based malware detection in executable files Jagsir Singh , Jaswinder Singh

[6] Ahmadi etal., 2016. M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto - Novel feature extraction, selection and fusion for effective malware family classification

[7] AL-Hawawreh etal., 2018 M. AL-Hawawreh, N. Moustafa, E. Sitnikova - Identification of malicious activities in industrial internet of things based on deep learning models(2021)

[8] A Survey on Malware Classification Using Machine Learning and Deep Learning Manish Goyal(2021)

[9] Athiwaratkun etal., 2017 B. Athiwaratkun, J.W. Stokes – Malware classification with lstm and gru language models and a character- level cnn

[10] D. Bekerman, B. Shapira, L. Rokach, A. Bar - Unknown malware detection using network traffic classification 09 2015B. Biggio, F. Roli - Wild patterns: ten years after the rise of adversarial machine learning

[11] Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection, proposed by Luca Caviglione and Michal et al. (2021)

[12] The survey proposed by Harsha Latha, R Mohanasundaram, and colleagues identifies malware and malware types(2020)

[13] The rise of machine learning for detection and classification of malware: research developments, trends and challenges. This framework, proposed by Daniel Gibert, CarlesMateu, and Jordi Planes (2020)

[14] Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks was proposed by Priyank Singhal (2012)

APPENDIX

Screenshots

```
import os

!pip install sklearn

import joblib

!pip install --upgrade scikit-learn==0.20.3

from six import StringIO

import os
import pandas
import numpy

import sklearn.ensemble as ek
from sklearn import tree, linear_model
from sklearn.feature_selection import SelectFromModel
from sklearn.externals import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from sklearn import svm
from sklearn.linear_model import LinearRegression

dataset = pandas.read_csv('Malware_Detection_data.csv',sep='|', low_memory=False)

dataset.head()

dataset.tail()

dataset.describe()

dataset.groupby(dataset['legitimate']).size()
```

Figure A.1: Libraries Import

```
X = dataset.drop(['Name','md5','legitimate'],axis=1).values
y = dataset['legitimate'].values

"""#Part 1"""

import pandas as pd
import numpy as np

df = pd.read_csv('MalwareData.csv', sep='|')

malware_csv = pd.read_csv('MalwareData.csv', sep='|')
legit = malware_csv[0:41323].drop(['legitimate'],axis=1)
malware = malware_csv[41323::].drop(['legitimate'],axis=1)

malware_csv

malware_csv.head(40)

malware_csv.tail()

malware_csv.describe()

malware_csv.tail(50)

malware_csv.info()

import matplotlib.pyplot as plt
import seaborn as sns

malware_csv.plot()

malware_csv.hist()

print("The no of samples are %s and no of features are %s for legitimate part "%(legit.shape[0],legit.shape[1]))
print("The no of samples are %s and no of features are %s for malware part "%(malware.shape[0],malware.shape[1]))
```

Figure A.2: Validation and Feature Extraction

```
pd.set_option("display.max_columns",None)
malware

from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate

data_input = malware_csv.drop(['Name','md5','legitimate'],axis = 1).values
labels = malware_csv['legitimate'].values
extratrees = ExtraTreesClassifier().fit(data_input, labels)
select = SelectFromModel(extratrees, prefit = True)
data_input_new = select.transform(data_input)

import numpy as np
features = data_input_new.shape[1]
importances = extratrees.feature_importances_
indices = np.argsort(importances)[::-1]
for i in range (features):
    print("%d"%(i+1),malware_csv.columns[2+indices[i]],importances[indices[i]])

from sklearn.ensemble import RandomForestClassifier
legit_train,legit_test,mal_train,mal_test = train_test_split(data_input_new,labels,test_size=0.2)

classifier = RandomForestClassifier(n_estimators=50)
classifier.fit(legit_train,mal_train)

print("The score of algorithm is " + str(classifier.score(legit_test,mal_test)*100))

"""Confusion Matrix"""

from sklearn.metrics import confusion_matrix
result = classifier.predict(legit_test)
conf_matrix = confusion_matrix(mal_test,result)
```

Figure A.3: Feature Selection

```
"""Gradiant Boost"""

print("False Positives:",conf_matrix[0][1]*100/sum(conf_matrix[0]))
print("False Negatives:",conf_matrix[1][0]*100/sum(conf_matrix[1]))

from sklearn.ensemble import GradientBoostingClassifier
grad_boost = GradientBoostingClassifier(n_estimators=50)
grad_boost.fit(legit_train,mal_train)

print("Score:", grad_boost.score(legit_test,mal_test)*100)

"""#part2"""

import os
import pandas
import numpy


import sklearn.ensemble as ek
from sklearn import tree, linear_model
from sklearn.feature_selection import SelectFromModel
from sklearn.externals import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from sklearn import svm
from sklearn.linear_model import LinearRegression

model = { "DecisionTree":tree.DecisionTreeClassifier(max_depth=10),
          "RandomForest":ek.RandomForestClassifier(n_estimators=50),
          "Adaboost":ek.AdaBoostClassifier(n_estimators=50),
          "LinearRegression":LinearRegression()
}
```

Figure A.4: Applying machine learning algorithms

```
results = {}
for algo in model:
    clf = model[algo]
    clf.fit(legit_train,mal_train)
    score = clf.score(legit_test,mal_test)
    print ("%s : %s " %(algo, score))
    results[algo] = score
```

Figure A.5: Accuracy Evaluation

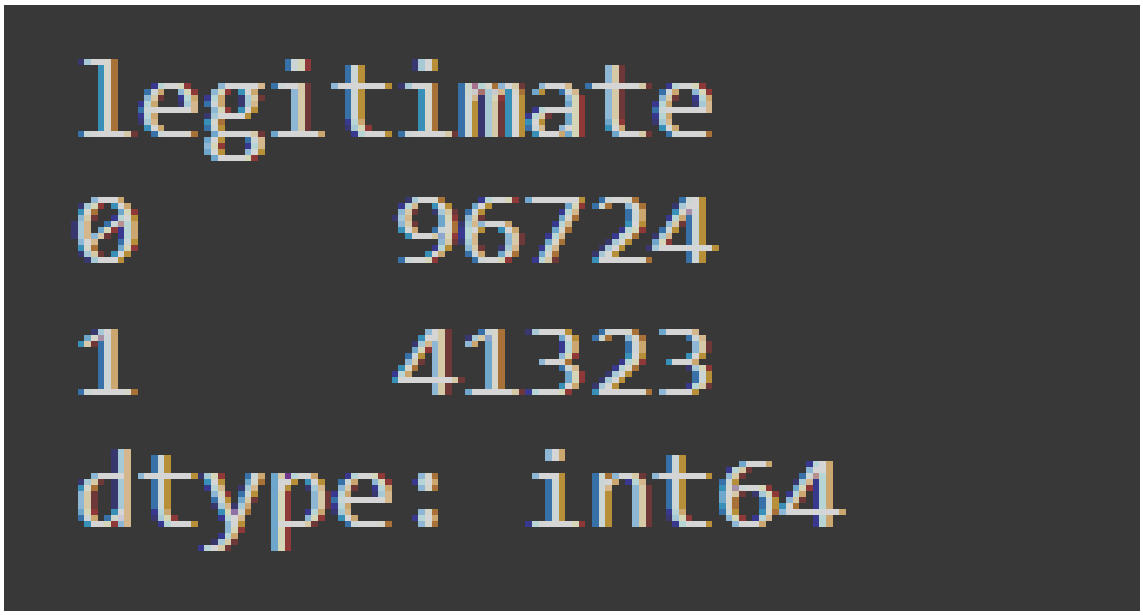


Figure A.6: Data files

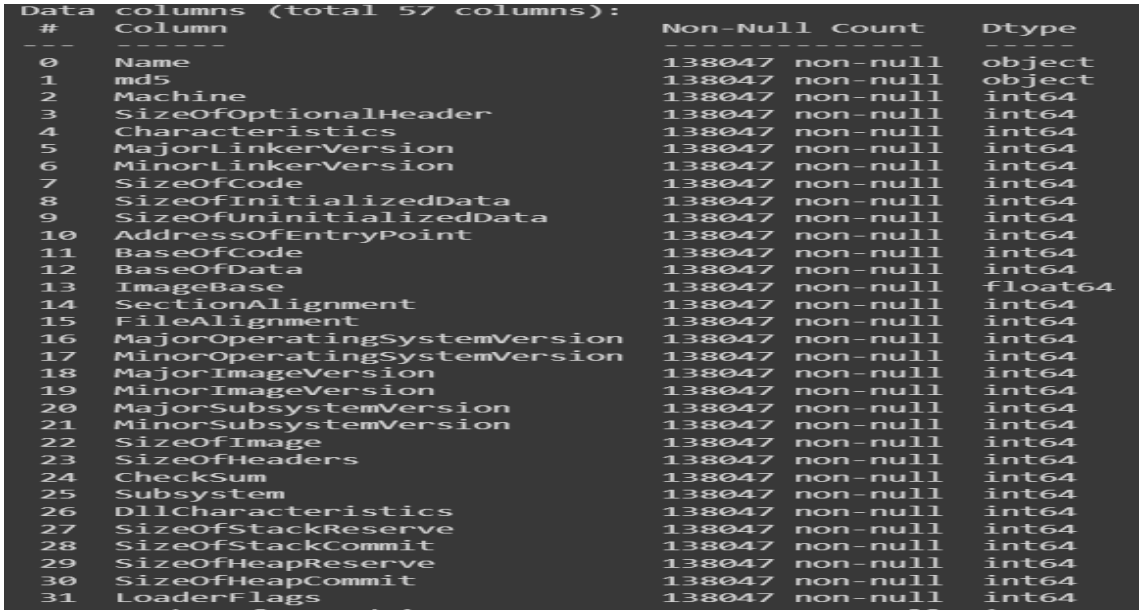
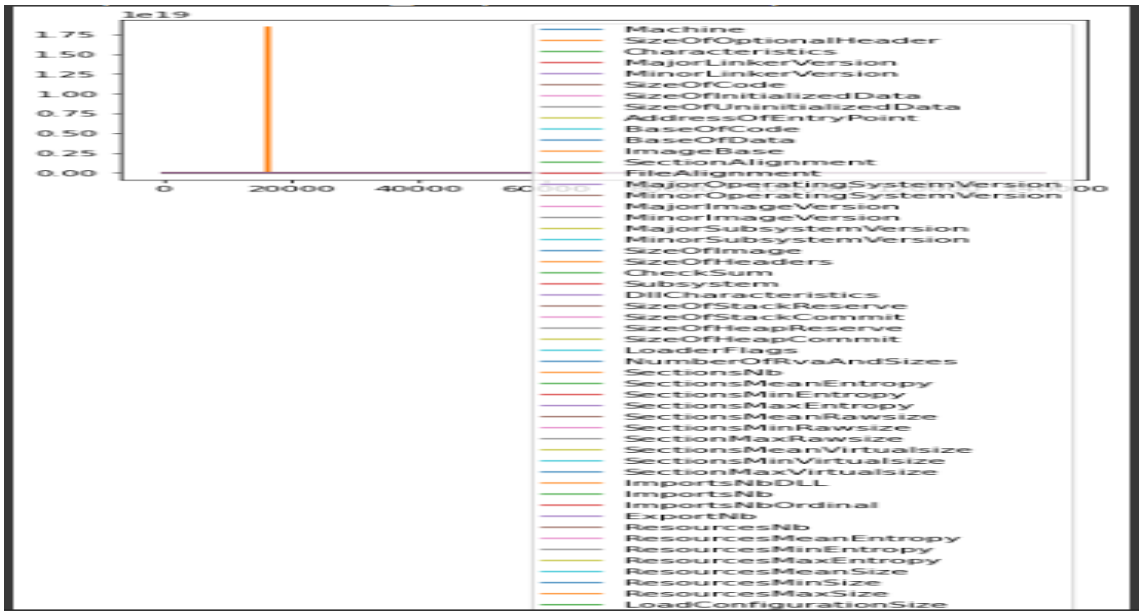


Figure A.7: Features Extracted




```
1 Characteristics 0.14900308058841 45
2 MajorSubsystemVersion 0.11155090779291392
3 DllCharacteristics 0.09846616284121343
4 SectionsMaxEntropy 0.08766022424898307
5 Subsystem 0.08137882499122222
6 SizeOfOptionalHeader 0.06541608846263167
7 VersionInformationSize 0.05708554667534184
8 Machine 0.047378469389344 01
9 MajorOperatingSystemVersion 0.040729248883841 83
10 ResourcesMinEntropy 0.033980485899966 27
11 ImageBase 0.029908847472996 48
12 ResourcesMaxEntropy 0.023398066920262316
```

Figure A.8: Selected Features

```
conf_matrix

array([[19153,   100],
       [   68,  8289]])
```

Figure A.9: Confusion Matrix

```
False Positives: 0.5193995740923493

False Negatives: 0.8136891228909896
```

Figure A.10: False positive and negative