A MINI PROJECT REPORT ON

MALWARE ANALYSIS AND PREDICTION

Submitted to JNTUH in the partial fulfillment of the Academic Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

AMBATI BHARGAV	**18QM1A0406**
BRUNDAVAN SAHOO	**18QM1A0417**
A.S.SANDHYA	**18QM1A0407**
C. SAI NITIN	**18QM1A0421**

ELECTRONICS AND COMMUNICATION ENGINEERING

BY UNDER THE GUIDANCE OF

Dr. D .Chandraprakash

KG REDDY
College of Engineering
& Technology
Engineering India's Changemakers

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

(Accredited by NAAC, Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)

Chilkur (Village), Moinabad (Mandal), R. R Dist, TS-501504

2021-22

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Accredited by NAAC, Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
Chilkur (Village), Moinabad (Mandal), R. R Dist, TS-501504

KG REDDY
College of Engineering
& Technology
Engineering India's Changemakers

CERTIFICATE

This is to certify that the Project report on "**MALWARE ANALYSIS AND PREDICTION**" is a bonafide record work carried out by **AMBATI BHARGAV (18QM1A0406), BRUNDAVAN SAHOO (18QM1A0417), SAI NITIN (18QM1A0421), A.S.SANDHYA (18QM1A0407),** in partial fulfillment for the requirement for the award of degree of **BACHELOR OF TECHNOLOGY** in "**ELECTRONICS AND COMMUNICATION ENGINEERING**", **JNTUH,** Hyderabad during the year 2020-2021.

Internal Guide	**Head of the department**
DR. D. Chandraprakash	Mr. M.N. Narsaiah (PHD)
ASSISTANT PROFESSOR	

External Examiner

ACKNOWLEDGEMENT

We wish to thank the Management, Sri. **Ln. K. Krishna Reddy, CHAIRMAN** of KG Reddy College of Engineering and Technology for providing the facilities to complete our project work.

We would also especially like to express our extreme gratitude and sincere thanks to our beloved **Director**, **Dr. Rohit Kandakatla**, KG Reddy College of Engineering and Technology, who encouraged and gave support to complete the project successfully.

At the very outset, we wish to place on record our sincere thanks and gratitude to our beloved head of the Institution, **Dr. R.S. Jahagirdar, Principal**, KG Reddy College of Engineering and Technology, who allowed us to complete this endeavor successfully.

We would like to express our sincere deep sense of gratitude and indebtedness to **Mr. M N Narsaiah, HOD**, Department of Electronics and Communication Engineering, for his significant suggestions and help in every respect to accomplish the project work. His persisting encouragement, everlasting patience and keen interest in discussions have benefited us to the extent that cannot be spanned by words. He has been the continuous source of inspiration for us throughout the work.

We would like to sincerely thank to our Major **Project coordinator, Mrs. Dr B Vandana, Associate Professor** who stimulated many thoughts for this project and Staff-Members of Department of Electronics and Communication Engineering Department for their goodwill gestures towards us.

We would especially like to express our extreme gratitude and sincere thanks to our project guide **Mr. Angotu Saida, Assistant Professor,** Department of Electronics and Communication Engineering for his enthusiastic and innovative guidance and support.

Last but not least we would also like to thank teaching and non-teaching staff and also all my family, friends without whose support this project work would remain unfulfilled.

AMBATI BHARGAV	**18QM1A0406**
BRUNDAVAN SHAOO	**18QM1A0417**
A.S.SANDHYA	**18QM1A0407**
C.SAI NITIN	**18QM1A0421**

ABSTRACT

A malicious URL is a typical and genuine threat to cybersecurity. A Malicious URL has an assortment of spontaneous content in the form of phishing, spam in order to launch attacks. Innocent users who visit such websites move toward becoming casualties of various sorts of scams, including monetary loss, theft of private information (identity, credit cards, etc.). It is essential to identify and follow up on such dangers in an opportune way. We had studied different techniques for detecting malicious URLs and discussed each and every technique and its merits and demerits.

Machine learning algorithms offer promising techniques to detect malicious websites performing unethical anonymous activities on the Internet. Attackers have been found to continuously evolve with updated techniques to attack web users using malicious Uniform Resource Locators (URLs). The main objective of such attacks is to gain financial benefits through acquiring personal information. Machine learning (ML)-based approach is proposed to identify malicious users from URL data.

Keywords: *Cyber Forensics, Malicious, Cybersecurity, Tor Browser, Machine learning , URLs,*

Contents

CHAPTER -1

1.INTRODUCTION

Malware analysis is the study or practice of extracting as much information as possible from a specific software vulnerability in order to determine its capability, origin, and potential impact. The data obtained aids in the identification of malware's functioning and extent, as well as how the system got infected and how to guard against future attacks.

Given malware's constant growth in complexity and volume, dealing with it is becoming extremely difficult. Using machine learning techniques to automatically acquire models and patterns beneath such complexity, and also developing technology to keep up with malware progress, is one of the most popular approaches in the literature. This survey seeks to give an overview of how machine learning has been applied to malware analysis in Windows settings, particularly for the analysis of Portable Executables. We categorize surveyed studies' goals (i.e., the desired conclusion), the malware information they utilize (i.e., the characteristics), and the machine learning techniques they apply (i.e., what algorithm is used to process the input and produce the output). We also identify the key current topical trends and how to possibly advance them, as well as a number of concerns and obstacles, including those related to the datasets utilized. Offer the innovative notion of malware analysis economics, which is concerned with the examination of existing trade-offs among important variables such as analysis accuracy and economic costs.

Today's antivirus vendors are dealing with a variety of potentially harmful samples. It's not typical to receive thousands of fresh samples every day. Because most signatures used to recognize confirmed malicious threats are still produced by hand, it's critical to distinguish between samples that offer a fresh unknown danger and those that are just versions of existing malware. This survey article provides an overview of strategies for analysing possibly harmful samples that are based on the simulation process. It also includes applications that use these to do analysis. It also includes analytical tools that use these approaches to aid human analysts in making timely and suitable decisions.

Now that we understand what malware is and what malware analysts perform, let's look at some malware analysis methodologies. The strategies provided here are only the tip of the iceberg when it comes to malware. For additional in-depth learning, I've given a list of resources below.

- **Static Analysis**

To examine malware, you don't need to run it. You can gain some useful information just by looking at the static information associated with a file by conducting what is known as Static Analysis. Here are a few instances of useful data that static analysis may provide.

- **Debugging**

Any malware researcher should be familiar with how to utilise a debugger. It allows you to follow the program's execution flow and includes handy features that offer you more control over the program's execution. You can, for example, put breakpoints on certain instructions to interrupt execution. You may also inspect the contents of registers and memory locations while the programme is executing, and even change their values. These are just a few of the options available to you using a debugger.

Malware detection is an important feature of the Page describes Security Cloud services, which further includes a secure access service edge (SASE) architecture. Malware is malicious software that is developed with both the intention of harming and exploiting devices or computer networks. Viruses, worms, Trojan horses, ransomware, and spyware, among other types of malware, continue to be a severe concern for companies and government institutions. Anti-virus signatures, heuristics, and patterns of behavior in sandboxes have been used in traditional malware detection systems, which involve a large number of human analysis by security analysts and researchers. It's difficult for firms to keep up with malware attacks as new attacks and variants emerge every day. Artificial intelligence (AI) and machine learning (ML), on the other hand, have the ability to detect unknown and zero-day malware by learning virus patterns automatically from enormous amounts of historical data. Because of this one-of-a-kind capabilities, AI/ML is now an essential component of modern malware detection solutions, complementing heuristic and signature-based approaches.

CHAPTER-2

2.Literature survey

This is a essential issue with distinct results for virtual protection, going from in addition evolved enemy of infection programming to creating and conveying designated updates to all of the greater efficiently ascertaining the economic price of breaks.

We saw how to expect the scope of real malware contaminations in a nation dependent on authentic measurements. Since we can most straightforward distinguish a little percent of malware pervasions, that is a troublesome mission to clear up. There are a few capacity benefits to definitively anticipating malware amount, which incorporate further developed enemy of infection programming program and centered fixing.

There have already been well documented incidents of android malware such as droid dream which detected over 50apps we can compare programmes to known malware based on code check for signatures or use other metamorphic malware can use techniques to reprogram itself the information flow of the program under review over period of time the system is then tested against a set of rules to verify if it is correct harmful behaviour has been observed.the issue with static methods is that static binary analysis is difficult to perform .

They presented the consequences of the first try at growing a chance prediction version based totally on each social-demographic and behavioral factors. Using statistics amassed from 50 users over a 4-month duration, we decided on 12 capabilities to construct a predictive version using neural networks .More broadly, their findings show that user demographics and high-level activity data are sufficient for basic risk modelling of users. However, further research is needed to expand the breadth and quality of this model before it can be used in a real-world setting.

CHAPTER-3

3.Methodology

Static analysis (or code analysis) and dynamic analysis are the two most prevalent malware analysis approaches employed by malware analysts (or behaviour analysis). These two methods enable analysts to swiftly and thoroughly comprehend the threats and goals of a specific sample malware.

Static analysis necessitates a solid understanding of programming and the x86 assembly language paradigm. You don't have to run the malware during the static analysis. Malware samples' source code is rarely published. You must first disassemble and decompile, and then reverse engineer the low-level assembly code. Because static analysis is safer than dynamic analysis, most malware analysts undertake it first. The intricacy of modern malware makes static analysis difficult, and some malware uses anti-debugging techniques to prevent malware analysts from examining the code.

In malware analysis, dynamic analysis (behaviour analysis) is a technique that executes the malware and monitors its activity. It also keeps track of any changes that occur during the malware's execution. Infecting a computer with malware that has been downloaded from the internet can be extremely dangerous. Malware infestation on your system might result in file deletion, registry changes, file alteration, data theft, and other issues. You'll need a secure environment to conduct malware analysis, and the network shouldn't be connected to any production networks.

Machine learning algorithms employ computational approaches to "learn" information directly from data rather than depending on a model. As the number of samples available for learning grows, the algorithms adaptively enhance their performance.

Detecting Malicious URLs using Machine Learning

The malicious URLs can be detected using the lexical features along with tokenization of the URL strings. I aim to build a basic binary classifier that would help classify the URLs as malicious or benign. Steps followed in building the machine learning classifier:

- Data Preprocessing
- Data Visualization
- Algorithm and Predication

Data Preprocessing

In Machine Learning, data preprocessing refers to the process of cleaning and organising raw data in order to make it appropriate for creating and training Machine Learning models.Preprocessing data is necessary to ensure high-quality results. Data cleansing, data integration, data reduction, and data transformation are the four stages of data preprocessing, which make the process easier.

Data Visualization

Data visualisation in machine learning is essential for understanding how data is used in a machine learning model and for assessing it. Facets is a free, open-source Python tool that may be used to efficiently visualise and analyse data.

Algorithm and Predication

In machine learning, an "algorithm" is a technique that is conducted on data to build a "model." Pattern recognition is performed by machine learning algorithms. Algorithms can either "learn" from data or "fit" to a dataset. There are numerous machine learning algorithms to choose from.

- Logistic Regression
- Decision Trees
- Random Forest

❖ Logistic Regression

A statistical analysis approach for predicting a data value based on past observations of a data collection is known as logistic regression. By examining the connection between one or more existing independent variables, a logistic regression model may predict a dependent data variable. It's a type of statistical software that estimates probabilities using a logistic regression equation to analyse the relationship between a dependent variable and one or more independent variables. This form of analysis can assist you in predicting the probability of an event or a decision occurring.

Logistic regression uses an equation as the representation, very much like linear regression. Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

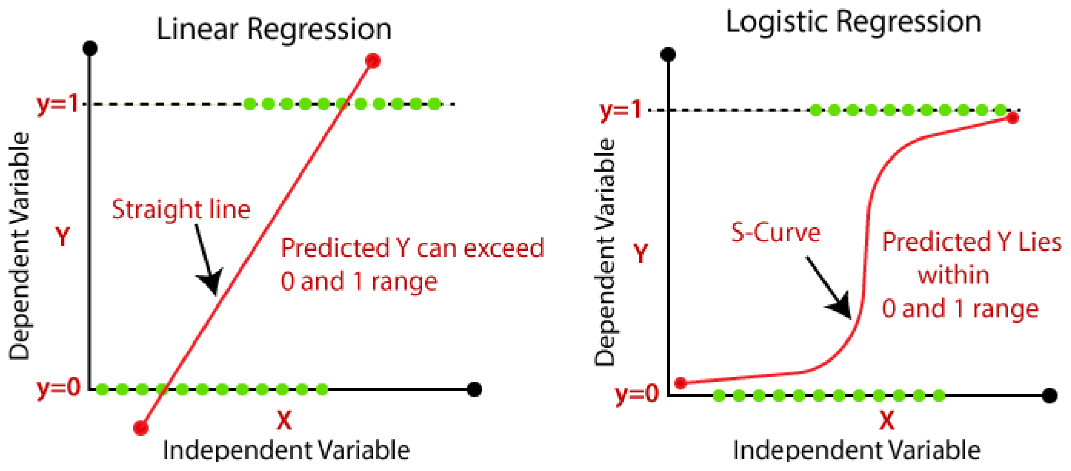

Fig: Linear Regression Variable Graph

Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

❖ **Decision Trees**

The Decision Tree is a supervised learning technique that may be used to solve both classification and regression issues, however it is most commonly employed to solve classification problems. Internal nodes represent dataset properties, branches represent decision rules, and each leaf node reflects the conclusion. The Decision Node and the Leaf Node are the two nodes of a Decision Tree. Decision nodes are used to make any decision and have several branches, whereas Leaf nodes are the results of such decisions and have no additional branches. The decisions or tests are based on the characteristics of the given dataset.

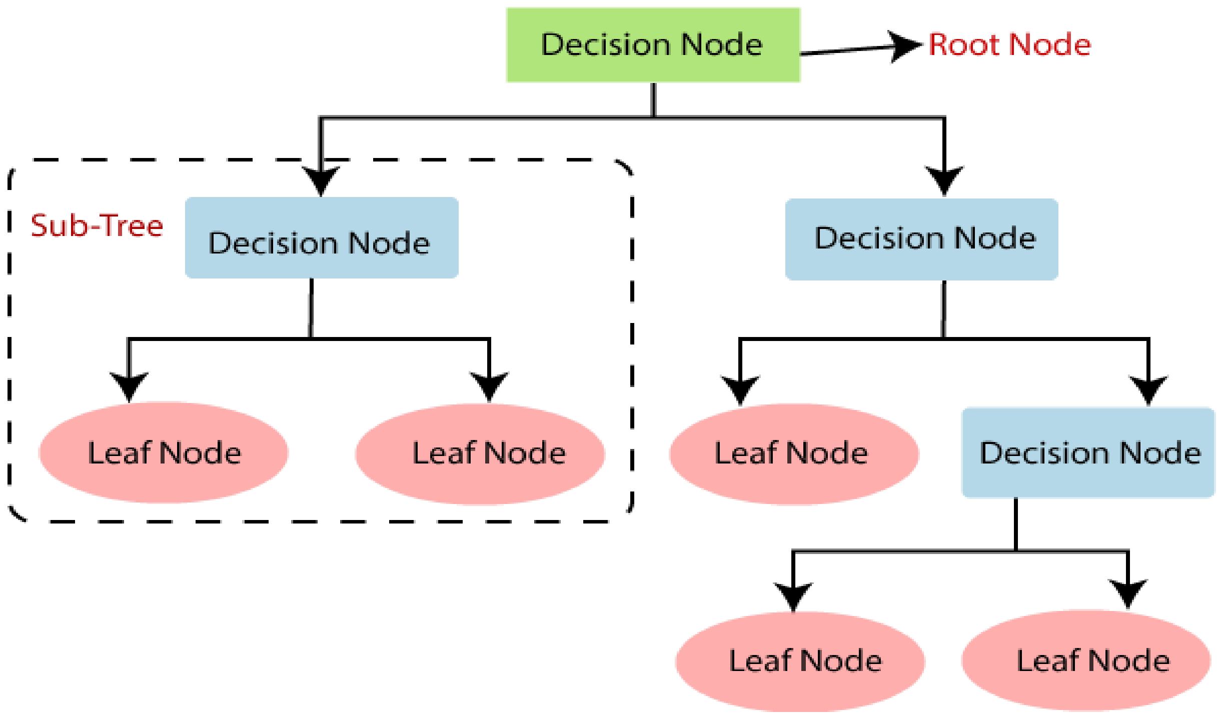

Fig: Decision Tree Flow Chat

The procedure for determining the class of a given dataset in a decision tree starts at the root node of the tree. This algorithm checks the values of the root attribute with the values of the record (actual dataset) attribute and then follows the branch and jumps to the next node based on the comparison.

❖ **Random Forest**

Random Forest is a well-known supervised machine learning algorithm. In machine learning, it can be used for both classification and regression. It is based on ensemble learning, which is the process of integrating numerous classifiers to solve a complex problem and improve the model's performance. "Random Forest is a classifier that contains a number of decision trees on various subsets of a given dataset and takes the average to enhance the predictive accuracy of that dataset," as the name suggests. Rather than depending on a single decision tree, the random forest collects predictions from each tree and ranks them according to the majority of votes.

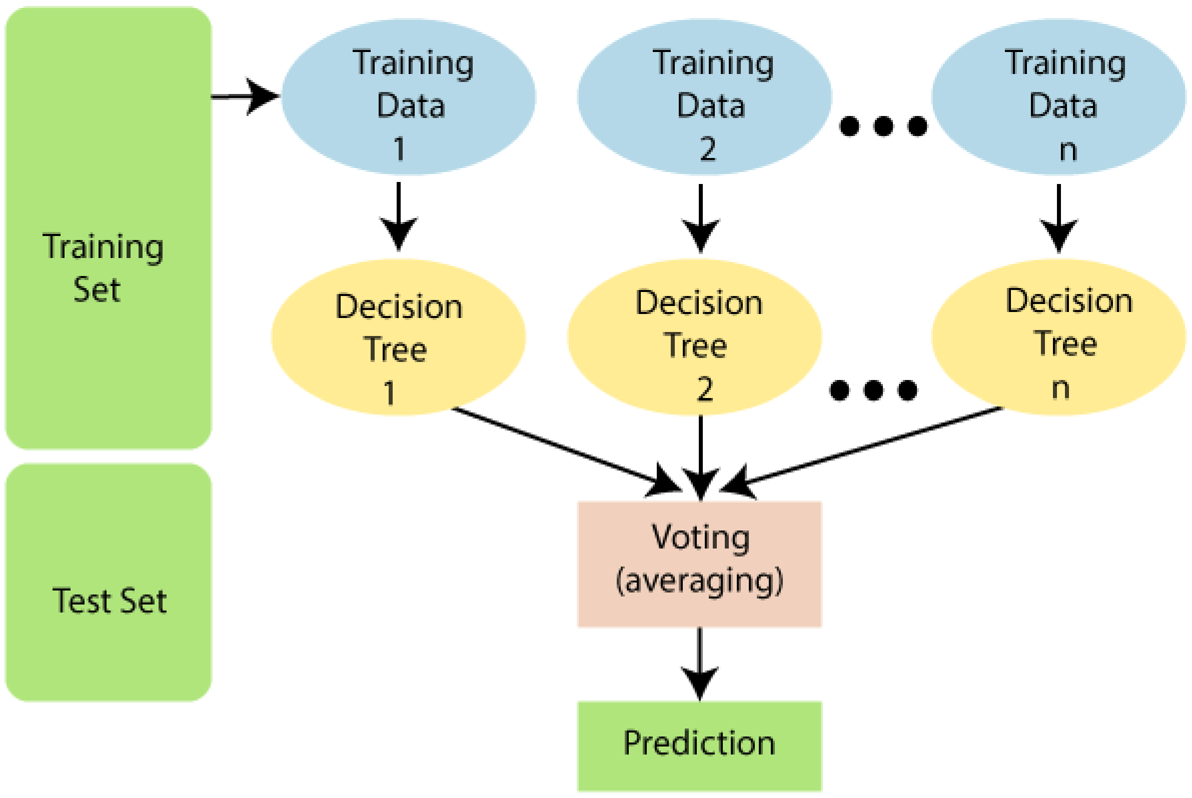

Fig: Random Forest Flowchart

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

CHAPTER-4

4. Design Coding

```
# Detection of Maclicious URLs using ML<br>
# By Team 10 ECE Final year
The malicious urls can be detected using the lexical features along with tokenization of the url
strings. I aim to build a basic binary classifier which would help classify the URLs as malicious
or benign.
Steps followed in building the machine learning classifier<br>
1. Data Preprocessing / Feature Engineering
2. Data Visualization
3. Building Machine Learning Models using Lexical Features.
Importing The Dependencies

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import os
from google.colab import files
from scipy.stats import ttest_ind
import io

uploaded = files.upload()
urldata = pd.read_csv(io.BytesIO(uploaded['urldata.csv']))
urldata.head()
#Removing the unnamed columns as it is not necesary.
urldata = urldata.drop('Unnamed: 0',axis=1)
```

```
urldata.head()
urldata.shape
urldata.info()
Checking Missing Values
urldata.isnull().sum()
No missing values in any column.




## 1. DATA PREPROCESSING
The following features will be extracted from the URL for classification. <br>
<ol>
    <li>Length Features
    <ul>
        <li>Length Of Url</li>
        <li>Length of Hostname</li>
        <li>Length Of Path</li>
        <li>Length Of First Directory</li>
        <li>Length Of Top Level Domain</li>
    </ul>
    </li>
    <br>
   <li>Count Features
    <ul>
    <li>Count Of  '-'</li>
    <li>Count Of '@'</li>
    <li>Count Of '?'</li>
    <li>Count Of '%'</li>
    <li>Count Of '.'</li>
    <li>Count Of '='</li>
    <li>Count Of 'http'</li>
    <li>Count Of 'www'</li>
    <li>Count Of Digits</li>
    <li>Count Of Letters</li>
    <li>Count Of Number Of Directories</li>
    </ul>
    </li>
    <br>
    <li>Binary Features
    <ul>
        <li>Use of IP or not</li>
        <li>Use of Shortening URL or not</li>
    </ul>
    </li>

</ol>
```

Apart from the lexical features, we will use TFID - Term Frequency Inverse Document as well.

### 1.1 Length Features

```
!pip install tld
#Importing dependencies
from urllib.parse import urlparse
from tld import get_tld
import os.path
#Length of URL
urldata['url_length'] = urldata['url'].apply(lambda i: len(str(i)))
#Hostname Length
urldata['hostname_length'] = urldata['url'].apply(lambda i: len(urlparse(i).netloc))
#Path Length
urldata['path_length'] = urldata['url'].apply(lambda i: len(urlparse(i).path))
#First Directory Length
```

```
def fd_length(url):
    urlpath= urlparse(url).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0

urldata['fd_length'] = urldata['url'].apply(lambda i: fd_length(i))
#Length of Top Level Domain
urldata['tld'] = urldata['url'].apply(lambda i: get_tld(i,fail_silently=True))
def tld_length(tld):
    try:
        return len(tld)
    except:
        return -1

urldata['tld_length'] = urldata['tld'].apply(lambda i: tld_length(i))
urldata.head()
urldata = urldata.drop("tld",1)
Dataset after extracting length features
urldata.head()
### 1.2 Count Features
urldata['count-'] = urldata['url'].apply(lambda i: i.count('-'))
urldata['count@'] = urldata['url'].apply(lambda i: i.count('@'))
urldata['count?'] = urldata['url'].apply(lambda i: i.count('?'))
urldata['count%'] = urldata['url'].apply(lambda i: i.count('%'))
urldata['count.'] = urldata['url'].apply(lambda i: i.count('.'))
urldata['count='] = urldata['url'].apply(lambda i: i.count('='))
urldata['count-http'] = urldata['url'].apply(lambda i : i.count('http'))
urldata['count-https'] = urldata['url'].apply(lambda i : i.count('https'))
urldata['count-www'] = urldata['url'].apply(lambda i: i.count('www'))
```

```
def digit_count(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
urldata['count-digits']= urldata['url'].apply(lambda i: digit_count(i))
def letter_count(url):
    letters = 0
    for i in url:
        if i.isalpha():
            letters = letters + 1
    return letters
urldata['count-letters']= urldata['url'].apply(lambda i: letter_count(i))
def no_of_dir(url):
    urldir = urlparse(url).path
    return urldir.count('/')
urldata['count_dir'] = urldata['url'].apply(lambda i: no_of_dir(i))
Data after extracting Count Features
urldata.head()
import re
#Use of IP or not in domain
def having_ip_address(url):
    match = re.search(

'(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])
\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4

'((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'
# IPv4 in hexadecimal
```

```
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # Ipv6
    if match:
        # print match.group()
        return -1
    else:
        # print 'No matching pattern found'
        return 1
urldata['use_of_ip'] = urldata['url'].apply(lambda i: having_ip_address(i))
def shortening_service(url):
    match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                      'tr\.im|link\.zip\.net',
                      url)
    if match:
        return -1
    else:
        return 1
urldata['short_url'] = urldata['url'].apply(lambda i: shortening_service(i))
#Heatmap
corrmat = urldata.corr()
f, ax = plt.subplots(figsize=(25,19))
sns.heatmap(corrmat, square=True, annot = True, annot_kws={'size':10})
```

```
plt.figure(figsize=(15,5))
sns.countplot(x='label',data=urldata)
plt.title("Count Of URLs",fontsize=20)
plt.xlabel("Type Of URLs",fontsize=18)
plt.ylabel("Number Of URLs",fontsize=18)
plt.figure(figsize=(15,5))
sns.countplot(x='label',data=urldata)
plt.title("Count Of URLs",fontsize=20)
plt.xlabel("Type Of URLs",fontsize=18)
plt.ylabel("Number Of URLs",fontsize=18)
plt.figure(figsize=(20,5))
plt.hist(urldata['url_length'],bins=50,color='LightBlue')
plt.title("URL-Length",fontsize=20)
plt.xlabel("Url-Length",fontsize=18)
plt.ylabel("Number Of Urls",fontsize=18)
plt.ylim(0,1000)

plt.figure(figsize=(20,5))
plt.hist(urldata['hostname_length'],bins=50,color='Lightgreen')
plt.title("Hostname-Length",fontsize=20)
plt.xlabel("Length Of Hostname",fontsize=18)
plt.ylabel("Number Of Urls",fontsize=18)
plt.ylim(0,1000)
plt.figure(figsize=(20,5))
plt.hist(urldata['tld_length'],bins=50,color='Lightgreen')
plt.title("TLD-Length",fontsize=20)
plt.xlabel("Length Of TLD",fontsize=18)
plt.ylabel("Number Of Urls",fontsize=18)
plt.ylim(0,1000)
plt.figure(figsize=(15,5))
plt.title("Number Of Directories In Url",fontsize=20)
```

```
sns.countplot(x='count_dir',data=urldata)
plt.xlabel("Number Of Directories",fontsize=18)
plt.ylabel("Number Of URLs",fontsize=18)
plt.figure(figsize=(15,5))
plt.title("Number Of Directories In Url",fontsize=20)
sns.countplot(x='count_dir',data=urldata,hue='label')
plt.xlabel("Number Of Directories",fontsize=18)
plt.ylabel("Number Of URLs",fontsize=18)
plt.figure(figsize=(15,5))
plt.title("Use Of IP In Url",fontsize=20)
plt.xlabel("Use Of IP",fontsize=18)

sns.countplot(urldata['use_of_ip'])
plt.ylabel("Number of URLs",fontsize=18)
plt.figure(figsize=(15,5))
plt.title("Use Of IP In Url",fontsize=20)
plt.xlabel("Use Of IP",fontsize=18)
plt.ylabel("Number of URLs",fontsize=18)
sns.countplot(urldata['use_of_ip'],hue='label',data=urldata)
plt.ylabel("Number of URLs",fontsize=18)
## 3. Building Models Using Lexical Features Only
I will be using three models for my classification.
<br>1. Logistic Regression
<br>2. Decision Trees
<br>3. Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression


#Predictor Variables
x = urldata[['hostname_length',
       'path_length', 'fd_length', 'tld_length', 'count-', 'count@', 'count?',
       'count%', 'count.', 'count=', 'count-http','count-https', 'count-www', 'count-digits',
       'count-letters', 'count_dir', 'use_of_ip']]

#Target Variable
y = urldata['result']
#Splitting the data into Training and Testing
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.3, random_state=42)
#Decision Tree
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train,y_train)

dt_predictions = dt_model.predict(x_test)
accuracy_score(y_test,dt_predictions)

print(confusion_matrix(y_test,dt_predictions))
#Random Forest
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)

rfc_predictions = rfc.predict(x_test)
accuracy_score(y_test, rfc_predictions)*100
print(confusion_matrix(y_test,rfc_predictions))
#Logistic Regression
log_model = LogisticRegression()
```

```
log_model.fit(x_train,y_train)


log_predictions = log_model.predict(x_test)
accuracy_score(y_test,log_predictions)*100
print(confusion_matrix(y_test,log_predictions))
```

CHAPTER-5

5. Results and discussions

Using a Machine Learning classification technique, we devised an efficient and reliable malware detection scheme. We investigated parameter modifications and their impact on malware classification accuracy. The efficiency and resilience of our proposed strategy have been confirmed in experiments. We plan to use more variant features in conjunction with one another in the future to improve detection accuracy and eliminate false positives. Overall, models produced excellent results, with a high level of accuracy and a low error rate.

CHAPTER-6

6. Conclusions

Using big data and advanced analytics is now possible thanks to new technology. Machine learning algorithms can be used in cybersecurity to detect exterior incursions, such as recognizing patterns in the behaviour of attackers doing reconnaissance, as well as internal threats. The goal of the study is to produce a visual representation of human interaction so that it may be used to infer concepts. A more thorough overview of a normal scenario is imagined by merging data from system log files, historical data on IP addresses, honeypots, system and user actions, and so on. To detect inappropriate conduct, the wit analyses many sources and patterns. In addition, machine learning is utilised to detect and attribute attacks. In addition, machine learning is used in a variety of penetration testing scenarios. The research presented here demonstrates that machine learning can be used to detect malware using a variety of methodologies. A number of algorithms have been developed, trained, and put to the test.

References

[1] . V.S Subrahmanian and Edoardo serra .Ensemble models for data-driven prediction of malware infections. ACM , 08 feb 2016.

[2] . Anil Somayaji and Jose M. Fernandez . Risk prediction of malware victimization based on user behavior . IEEE , (2014).

[3] . Vinod.p . survey on malware detection . IEEE , (2009).

[4] . Justin sahs and Latifer khan .Machine learning approach to android malware detection. Springer , (2012).

[5] . Andreas moser and Engin kirdo. Limits static analysis for malware detection. IEEE , (2007).

[6] . Paul miller and NMC laughlin ,Suleiman . Deep android malware detection . IEEE , (2012).

[7] . Nir Nissim and Lior Rokach . Dynamic malware analysis in the modern Era . Association for computing machinery . 13th Sep (2019).

[8] . E.filiol. Malware pattern scanning schemes secure against black box analysis.
Journal of computer virol , (2006)

[9] . Hossein Fereidooni and Mauro conti . Android detection using static analysis of applications. IEEE ,(2016).

[10] . Abhijit yewale and Mahinder Singh . Malware detection based on opcode frequency.
IEEE Xplore , jan (2017)

[11]. Arvind Mahindru and Lov Kumar . Android Malware prediction using extreme learning machine with different kernel functions. Association for Computing Machinery , 7 August (2019).