NFSU National Forensic Sciences University

Knowledge | Wisdom | Fulfilment

An Institution of National Importance
(Ministry of Home Affairs, Government of India)

विद्या अमृतं अश्नुते

PROJECT REPORT
ON
"Malware Detection Using Machine Learning"

Submitted To

School of Cyber Security & Digital Forensics,

National Forensic Sciences University

For partial fulfilment for the award of degree

MASTER OF SCIENCE

In

DIGITAL FORENSICS AND INFORMATION SECURITY

Submitted By

SUMIT KUMAR DUBEY

(101CTMSDS2021007)

Under the Supervision of

MR. PRAKASH KHASOR

(ASSISTANT PROFFESOR CYBER SECURITY)

National Forensic Sciences University,

Gandhinagar Campus, Gandhinagar – 382009, Gujarat, India.

January, 2021

DECLARATION

I certify that

a. The work contained in the dissertation is original and has been done by myself under the supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

e. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

f. From the plagiarism test, it is found that the similarity index of whole dissertation within 25% and single paper is less than 10 % as per the university guidelines.

Date: 21/07/2022
Place: NFSU(Gandhinagar)

Sumit Kumar Dubey
Enroll. No.: 101CTMSDS2021007

CERTIFICATE

This is to certify that the work contained in the dissertation entitled **"Malware Detection Using Machine Learning"**, submitted by **SUMIT KUMAR DUBEY (Enroll. No.: 101CTMSDS2021007)** for the award of the degree of **Master of Science in Digital Forensics and Information Security** to the **National Forensic Sciences University**, **Gandhinagar Campus**, is a record of Bonafide research works carried out by him under my direct supervision and guidance.

Date:21/07/2022
Place: NFSU(Gandhinagar)

Mr. Prakash Khasor
Assistant Professor,
School of Cyber Security & Digital Forensics,
National Forensic Sciences University,
Gandhinagar Campus, Gandhinagar, Gujarat, India.

ACKNOWLEDGEMENTS

I would like to convey my heartfelt sincere thanks to my internal guide **Mr. Prakash Khasor, Assistant Professor, IFS, NFSU** for his valuable suggestion and constant encouragement and guidance provided at every stage of my project work. Gratitude is owed to the staff of department of IFS, Gujarat Forensics Sciences University for the guidance and co-operation provided.

With Sincere Regards,

Sumit Kumar Dubey

Master of Science in Digital Forensics and Information Security

ABSTRACT

In today's Digital World all the are relying on information that they carry. And this information is so crucial that if it gets into wrong hand it can lead into a huge loss or a major destruction. The main cause of all this unwanted action is malware. Early model of antivirus scanning by the help of signature based detection is not that much helpful as now the malware itself is updated and it can evade those tools very easily. So here we are going to design an automated tool for malware detection in a file. By this one can easily analyze the difference between a normal or a malicious file.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. Introduction

1.1 Introduction

Every day we hear news about data breaches and cyber-attacks as attackers are enhancing their development skills and building malware that are able to bypass company safeguards.

And in this Digital World all the things are relying on information that they carry. And this information is so crucial that if it gets into wrong hand it can lead into a huge loss or a major destruction. The main cause of all this unwanted action is malware.

Malware stands for malicious software. It intends toward intrusive behavior over an end device.

There are many types of malwares are available in world with different goals some are for information collection and some are to destroy the operations and many more. No one can assume that what it tends to do and what its nature.

Early model of antivirus scanning by the help of signature-based detection is not that much helpful as now the malware itself is updated and it can evade those tools very easily.

Only thing that an end entity can do is they can try to remove it with help of an antivirus. These methods were useful in older times but now a days even the malware can predict the end machines behaviors and nature. After that it can decide what action to take to achieve end goal and evade security and exploit system.

Here we are just going analyses such malware with the help of some machine learning.

1.2 Aim and objective

The main goal is to design an automated tool for malware detection in a file. By this one can easily analyse the difference between a normal or a malicious file.

1.3 What is malware

Malware or malicious software are piece of software that are designed to infiltrate and damage information systems without user’s consent.

Every day we hear news about data breaches and cyber-attacks as attackers are enhancing their development

1.4 Types of malwares

There is different type of malware available in the world but they are generally classified into the below mentioned category.

- **Viruses: -** A virus can be termed as a type of computer program that replicates itself by modifying other code when it gets into in any system.
- **Ransomware: -** Ransomware is type of malware when executed it will encrypt the whole data and deny the end entity from using unless the demand or moto is satisfied**.**
- **Worms: -**A worm is a type of malware that replicate itself an spreads onto computer.
- **Trojan: -** Trojan is a type of malware that leads to malicious software presenting it as legitimate software.
- **Spyware:** -Spyware is a type of malware that enters into end computer and steal information and transfer it to interested unintended users.
- **Backdoor:** - Backdoor are the hidden method of bypassing normal authentication or encryption in a computer and creating secrete channel for remote execution without getting detected.
- **Spyware:** - Spyware are the malicious software which tends to gather information about a person organization and send it to another entity.
- **Keyloggers:** - Keyloggers are the program that logs each and every keystroke for the purpose of collecting the hidden or personal data.
- **Adware:** -Adware's are the program which is used to pop-up unwanted advertisements on your device.

- **Bots:** -Bots are computer program that operates as an agent for a user or other program to pretend as a human activity.

And many other forms are available.

1.5 Malware Analysis

Malware analysis is the process of discovering exacatly what happened to system, and make sure that the machines damaged by malicious software are isolated from the network.

To perform malware analysis, we need to follow specific operations and approaches.

There is total three malware analysis methods are present:

- Static malware analysis
- Dynamic malware analysis
- Memory malware analysis

1.6 Types of malware analysis

- **Static malware analysis: -** It is used to collect all information about the malware using different techniques and utilities, but without actually executing the malware.
  Under static analysis there are two methods.
  - Basic static analysis: - in this we analyse the malware without actually looking into the instructions but without executing it.
  - Advanced static analysis: -This also known as code analysis in this we analyse code and dissect binary file to study each operation it performs.
- **Dynamic malware Analysis: -**After Collecting the information we run the malware in an isolated and secure environment which are called as sandboxes to study the nature of malware while it is executing.

Under dynamic malware analysis are two other sub sections.

- Basic Dynamic analysis: -in this we will analyse the malware behaviours while it is running.
- Advanced Dynamic analysis: -Here we run the malware but we use a debugger to determine its functionality and behaviour which is not possible basic dynamic malware analysis.

- **Memory Malware Analysis: -**In this we use the collected memory dump for analysing the malware behaviour and predicting the outcome of malware.

2. Literature Survey

2.1 A survey on machine learning-based malware detection in executable files

The first research h paper is **A survey on machine learning-based malware detection in executable files** purpose of this research is to know about different tools and techniques for malware analysis

2.2 Automatic malware classification and new malware detection using machine learning

The second paper is **Automatic malware classification and new malware detection using machine learning** purpose of this research is to know data processing, decision making, and clustering.

2.3 Malware Detection Using Machine Learning and Deep Learning

The third paper is **Malware Detection Using Machine Learning and Deep Learning** purpose of this paper is to feature extraction and feature reduction.

2.4 Malware Detection with Deep Neural Network Using Process Behavior

The fourth paper is on **Malware Detection with Deep Neural Network Using Process Behavior** purpose of this paper is to detection of malware designed for Windows operating systems.

2.5 Static and Dynamic Malware Analysis Using Machine Learning

The fifth research paper is on **Static and Dynamic Malware Analysis Using Machine Learning** the purpose of this report is Recurrent Neural Network (RNN) to extract features of process behavior. Convolutional Neural Network (CNN) to classify feature images which are generated by the extracted features from the trained RNN.

3. Malware Analysis Tool

Here we are going to use different machine learning methods to develop a tool that is used for malware detection.

This tool will perform static analysis on the basis of features extracted by a ML model that will dump it into a pickle file and predict the file category.

Weather the file is malware or a legitimate file.

3.1 Tool flow steps

- **Load Dataset: -** Here we are going to select appropriate dataset.
- **Model training: -** Then we are going to train the model with appropriate data set.
- **Generate Pickle's: -** Here we are generating the pickle files from the model and saving it for comparison.
- **Extract features from file: -** Then we are going to select the file for analysis and extract the features from it.
- **Predict File type: -** From the extracted features we are going to compare the file an predict the type whether the file is malicious or legitimate.

3.2 **Dataset**

A dataset is a collection of related discrete items of related data that may be accessed individually or in combination or managed as a whole entity.

Here we are going to use different types of datasets to select the best dataset for training model.

In the dataset we will provide a file which consists of certain features like: -

• SizeOfOptionalHeader

• Characteristics

• SizeOfCode

• VersionInformationSize

• legitimate

And a lot of other features depends on the data set you provide.

3.3 **PE File**

Portable executable files are file formats for executables, DLLs, and object codes used in 32-bit and 64-bit versions of window. They Contain many useful pieces of information for malware analysts, including imports, exports, time-date, stamps, subsystems, sections, and resources.

This following is the basic structure of PE File:

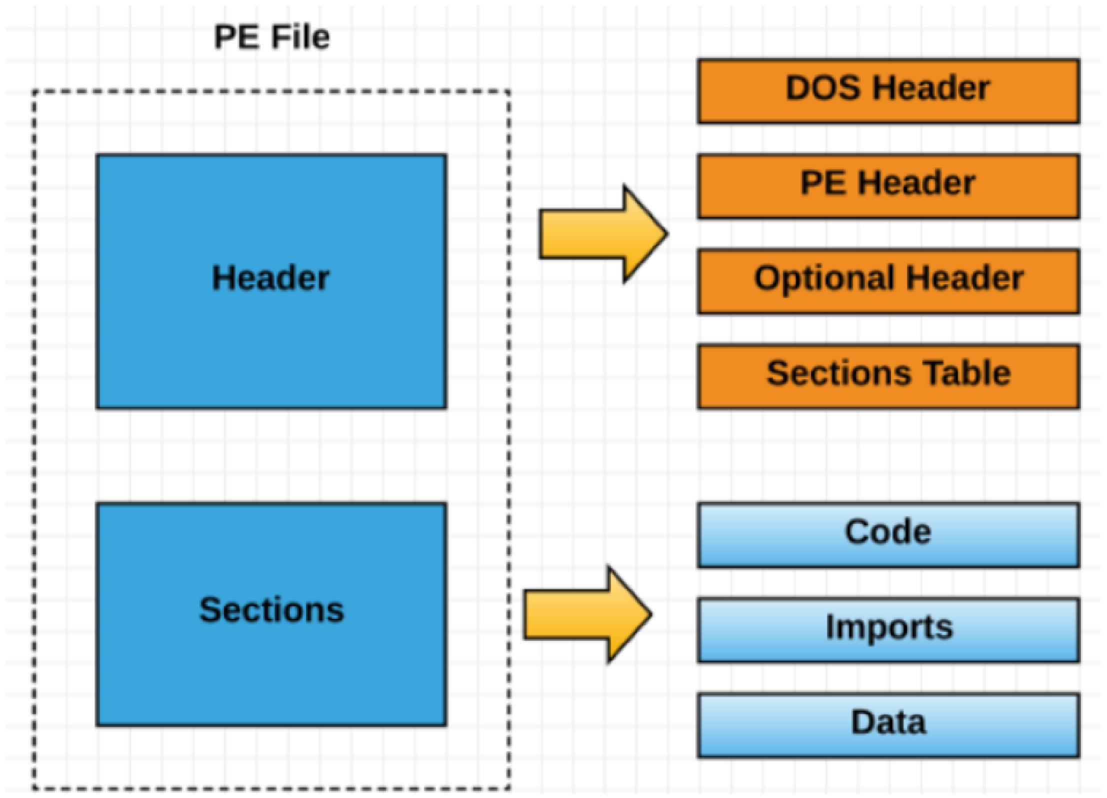


Figure 1:PE File

- **DOS Headers: -**This starts with the first 64 bytes of every PE file, so DOS can validate the executable and can run it in DOS stub mode.
- **PE Header: -**This contains information, including the location and size of the code.
- **PE Sections: -**They contains the main contents of the file.

The structures defined in the Windows header files will be accessible as attributes in the PE instance. The naming of fields/attributes will try to adhere to the naming scheme in those headers. Only shortcuts added for convenience will depart from that convention. pefile requires some basic understanding of the layout of a PE file — with it, it's possible to explore nearly every single feature of the PE file format.

3.4 Model Training

Model Training is the process of selecting the best features from all available features for that wea re using different type of models.

- **Decision tree: -** It is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- **Logistic Regression: -** Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary event occurring.

- **Random Forest:** - Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

- **Gradient Boosting:** - Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

- **ADA boost: -** AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps.

- **GuassianNB: -** Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution. An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions.

3.5 Feature Extraction

Here we will use PE File module in python to extract features from the .exe file. Then we will use this feature and compare it with our trained model. For that we will use the Pickle files generated from the training. After Comparing the we will then predict whether a file is malicious or legitimate.

4. Source Code

4.1 Main.py

```
from tkinter import *
from tkinter import filedialog

from tkinter import ttk

import learning
import threading
import analyse

Loc=None
Loc1=None


root = Tk()
root.title('Malware Detection')
root.minsize(600, 400)
root.geometry("1200x800")
root.resizable(False,False)

background_image = PhotoImage(file="img.png")
background_label = Label(root, image=background_image)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
background_label.image = background_image


def getFileLoc():
   global Loc
   Loc = filedialog.askopenfilename(title='select a .csv file',
filetypes=(('csv files','*.csv'),('all files', '*.*')))
   return str(Loc)

def getFileLoc1():
   global Loc1
   Loc1 = filedialog.askopenfilename(title='select a .exe file',
filetypes=(('exe files','*.exe'),('all files', '*.*')))
   return str(Loc1)

def trainpath():
   global trainpathloc
   trainpathloc = learning.learning1(Loc, processVar)
   return str(trainpathloc)
def mthread():
   threading.Thread(target=trainpath).start()

def analysepath():
   global trainpathloc
   trainpathloc = analyse.filepath(Loc1, processVar1)
   return str(trainpathloc)
def athread():
   threading.Thread(target=analysepath).start()

def close():
   root.destroy()
   root.quit()


titleLabel = Label(root, text="Malware Detection Using
machine Learning", anchor=CENTER,
bg="#4b76b4",fg="#c7d2e6",font=("Helvetica", 26))
titleLabel.grid(row=1, column=2,
columnspan=4,sticky="w",padx=20,ipadx=200,pady=20,ipady=
40)
```

```
datasetLb=Label(root,width=50,height=1,bg="#c7d2e6",fg="#0
50505",text=" Select Your dataset",font=("Calibri", 12))
datasetLb.grid(row=3,column=1,columnspan=2)

datasetBt = Button(root,
text='Dataset',fg="black",bg="#a6776d",width=20,height=2,
font=("Calibri", 13),command= getFileLoc)
datasetBt.grid(row=3, column=2,columnspan=2, padx=400,
pady=20)


trainLb=Label(root,width=52,bg="#c7d2e6",fg="#050505",text
="Train the Model with Dataset",height=1,font=("Calibri", 12))
trainLb.grid(row=4, column=1, columnspan=2)

trainBt = Button(root, text = 'Train Model',
fg="black",bg="#a6776d",width=20,height=2,font=("Calibri",
13), command=mthread)
trainBt.grid(row=4, column=2,columnspan=2, pady=20,
padx=20)

processVar = StringVar(value="")
processsLabel = Label(root, textvar=processVar, fg="black",
bg="white")
processsLabel.grid(row=5, column=2, columnspan=2, pady=20,
padx=20)



selectfileLb=Label(root,bg="#c7d2e6",fg="#050505",text="Sele
ct a File To Analyse", width=52,height=1,font=("Calibri", 12))
selectfileLb.grid(row=6, column=1, columnspan=2)
```

```
selectfileBt = Button(root, text = 'Select
File',fg="black",bg="#a6776d",width=20,height=2,font=("Calib
ri", 13),command=getFileLoc1)
selectfileBt.grid(row=6, column=2,columnspan=2, pady=20,
padx=20)


analysefileLb=Label(root,bg="#c7d2e6",fg="#050505",text="A
nalyse the File", width=52,height=1,font=("Calibri", 12))
analysefileLb.grid(row=7, column=1, columnspan=2)

analysefileBt = Button(root, text = 'Analyse
File',fg="black",bg="#a6776d",width=20,height=2,font=("Calib
ri", 13), command=athread)
analysefileBt.grid(row=7, column=2,columnspan=2, pady=20,
padx=20)

processVar1 = StringVar(value="")
processsLabel = Label(root, textvar=processVar1, fg="black",
bg="white")
processsLabel.grid(row=8, column=2, columnspan=2, pady=20,
padx=20)

closeBt = Button(root, text =
'Exit',fg="black",bg="#ce3531",width=10,height=1,font=("Calib
ri", 13), command=close)
closeBt.grid(row=9, column=2,columnspan=2, pady=20,
padx=20)


root.mainloop()
```

4.2 Learning.py

```
import os
import pandas as pd
import numpy as np
import pickle
import sklearn.ensemble as ske
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
qwe=None
def learning1(path, procesvar1):
        process = procesvar1
        global qwe
        qwe=path

        base_dir = os.path.dirname(os.path.abspath(__file__))
        pickel_dir = os.path.join(base_dir, "classifier")
        datset_dir = os.path.join(base_dir, "dataset")

        data = pd.read_csv(os.path.join(datset_dir,path), sep='|')
        data=data.fillna(0)
        X = [[np.nan, 2], [6, np.nan], [7, 6]]
        X = data.drop(['Name','md5', 'legitimate'], axis=1).values
        y = data['legitimate'].values

        process.set('Researching important feature based on %i
total features\n' % X.shape[1])

        # Feature selection using Trees Classifier
        fsel = ske.ExtraTreesClassifier().fit(X, y)
        model = SelectFromModel(fsel, prefit=True)
        X_new = model.transform(X)
        nb_features = X_new.shape[1]

        X_train, X_test, y_train, y_test = train_test_split(X_new, y
,test_size=0.3)

        features = []

        process.set('%i features identified as important:' %
nb_features)

        indices = np.argsort(fsel.feature_importances_)[::-
1][:nb_features]
        for f in range(nb_features):
            process.set("%d. feature %s (%f)" % (f + 1,
data.columns[2+indices[f]],
fsel.feature_importances_[indices[f]]))

        # XXX : take care of the feature order
        for f in sorted(np.argsort(fsel.feature_importances_)[::-
1][:nb_features]):
            features.append(data.columns[2+f])

        #Algorithm comparison
        algorithms = {
                "DecisionTree":
tree.DecisionTreeClassifier(max_depth=10),
                "RandomForest":
ske.RandomForestClassifier(n_estimators=50),
                "GradientBoosting":
ske.GradientBoostingClassifier(n_estimators=50),
                "AdaBoost":
ske.AdaBoostClassifier(n_estimators=100),
                "GNB": GaussianNB()
        }

        results = {}
        process.set("\nNow testing algorithms")
        for algo in algorithms:
           clf = algorithms[algo]
           clf.fit(X_train, y_train)
           score = clf.score(X_test, y_test)
           process.set("%s : %f %%" % (algo, score*100))
           results[algo] = score

        winner = max(results, key=results.get)
        process.set('\nWinner algorithm is %s with a %f %%
success' % (winner, results[winner]*100))

        # Save the algorithm and the feature list for later
predictions
        process.set('Saving algorithm and feature list in classifier
directory...')
        joblib.dump(algorithms[winner],
os.path.join(pickel_dir,'classifier.pkl'))
        open(os.path.join(pickel_dir,'features.pkl'),
'wb').write(pickle.dumps(features))
        process.set('Saved')

        # Identify false and true positive rates
        clf = algorithms[winner]
        res = clf.predict(X_test)
        mt = confusion_matrix(y_test, res)
        process.set("False positive rate : %f %%" % ((mt[0][1] /
float(sum(mt[0])))*100))
        process.set('False negative rate : %f %%' % ( (mt[1][0] /
float(sum(mt[1]))*100)))
        process.set('Training Completed Sucessfully')
```

4.3 **Analyze.py**

```
import pefile
import os
import array
import time
import math
import pickle
import joblib
import sys


qwe2=None
base_dir = os.path.dirname(os.path.abspath(__file__))
pickel_dir=os.path.join(base_dir,"classifier")
datset_dir=os.path.join(base_dir,"dataset")
```

Screenshot 1-1:Drozer

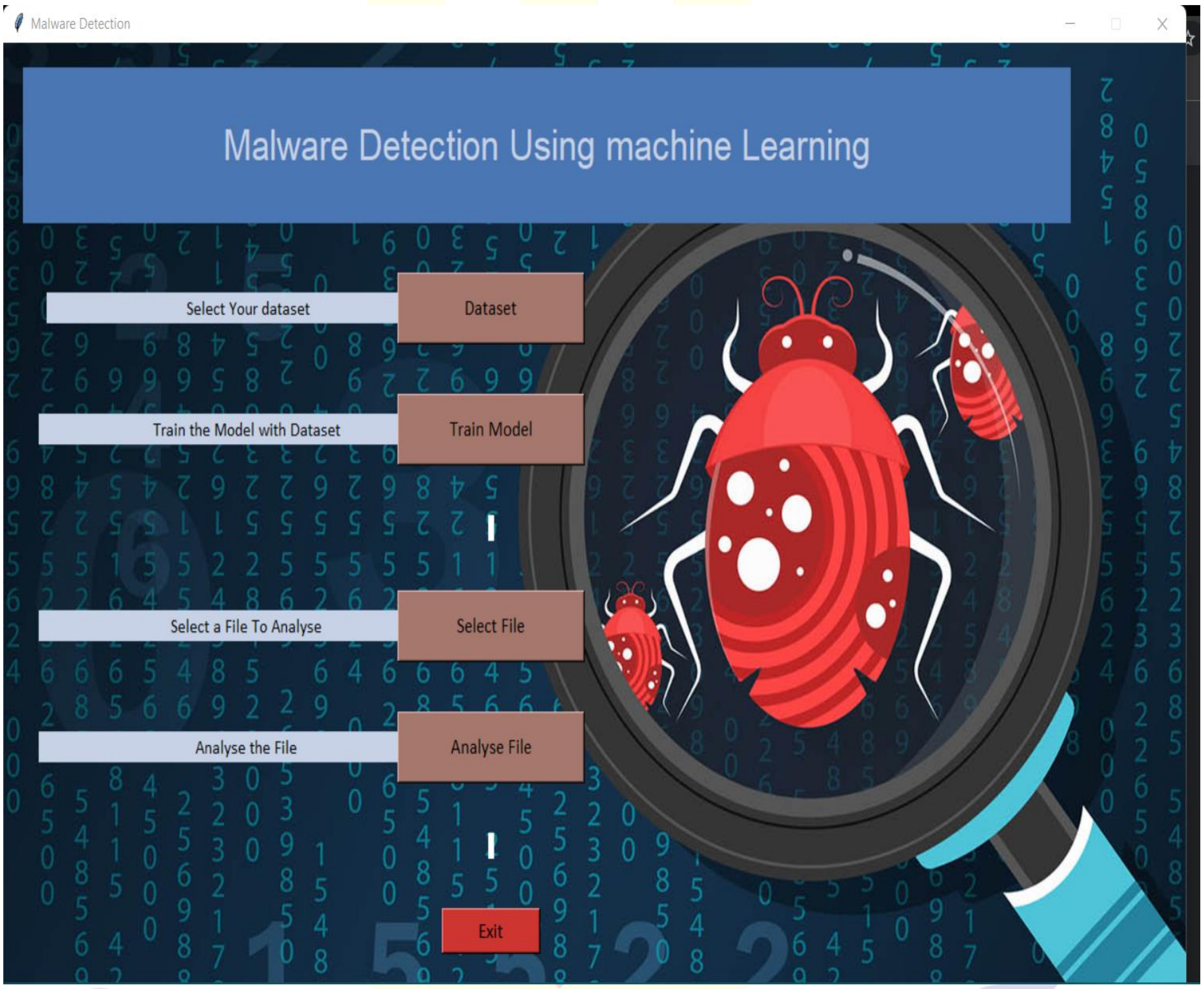
```
def get_entropy(data):


    if len(data) == 0:
        return 0.0
    occurences = array.array('L', [0]*256)
    for x in data:
        occurences[x if isinstance(x, int) else ord(x)] += 1

    entropy = 0
    for x in occurences:
        if x:
            p_x = float(x) / len(data)
            entropy -= p_x*math.log(p_x, 2)

    return entropy
```

```
def get_resources(pe):

    resources = []
    if hasattr(pe, 'DIRECTORY_ENTRY_RESOURCE'):
        try:
            for resource_type in
pe.DIRECTORY_ENTRY_RESOURCE.entries:
                if hasattr(resource_type, 'directory'):
                    for resource_id in resource_type.directory.entries:
                        if hasattr(resource_id, 'directory'):
                            for resource_lang in
resource_id.directory.entries:
                                data =
pe.get_data(resource_lang.data.struct.OffsetToData,
resource_lang.data.struct.Size)
                                size = resource_lang.data.struct.Size
                                entropy = get_entropy(data)

                                resources.append([entropy, size])
        except Exception as e:
            return resources
    return resources

def get_version_info(pe):

    res = {}
    for fileinfo in pe.FileInfo:
        if fileinfo.Key == 'StringFileInfo':
            for st in fileinfo.StringTable:
                for entry in st.entries.items():
                    res[entry[0]] = entry[1]
        if fileinfo.Key == 'VarFileInfo':
            for var in fileinfo.Var:
                res[var.entry.items()[0][0]] = var.entry.items()[0][1]
    if hasattr(pe, 'VS_FIXEDFILEINFO'):
        res['flags'] = pe.VS_FIXEDFILEINFO.FileFlags
        res['os'] = pe.VS_FIXEDFILEINFO.FileOS
        res['type'] = pe.VS_FIXEDFILEINFO.FileType
        res['file_version'] =
pe.VS_FIXEDFILEINFO.FileVersionLS
        res['product_version'] =
pe.VS_FIXEDFILEINFO.ProductVersionLS
        res['signature'] = pe.VS_FIXEDFILEINFO.Signature
        res['struct_version'] =
pe.VS_FIXEDFILEINFO.StrucVersion
    return res

def extract_infos(fpath):

    res = {}
    pe = pefile.PE(fpath)
    res['Machine'] = pe.FILE_HEADER.Machine
    res['SizeOfOptionalHeader'] =
pe.FILE_HEADER.SizeOfOptionalHeader
    res['Characteristics'] = pe.FILE_HEADER.Characteristics
    res['MajorLinkerVersion'] =
pe.OPTIONAL_HEADER.MajorLinkerVersion
    res['MinorLinkerVersion'] =
pe.OPTIONAL_HEADER.MinorLinkerVersion
    res['SizeOfCode'] = pe.OPTIONAL_HEADER.SizeOfCode
    res['SizeOfInitializedData'] =
pe.OPTIONAL_HEADER.SizeOfInitializedData
    res['SizeOfUninitializedData'] =
pe.OPTIONAL_HEADER.SizeOfUninitializedData
    res['AddressOfEntryPoint'] =
pe.OPTIONAL_HEADER.AddressOfEntryPoint
    res['BaseOfCode'] = pe.OPTIONAL_HEADER.BaseOfCode
    try:
        res['BaseOfData'] =
pe.OPTIONAL_HEADER.BaseOfData
    except AttributeError:
        res['BaseOfData'] = 0
    res['ImageBase'] = pe.OPTIONAL_HEADER.ImageBase
    res['SectionAlignment'] =
pe.OPTIONAL_HEADER.SectionAlignment
    res['FileAlignment'] =
pe.OPTIONAL_HEADER.FileAlignment
    res['MajorOperatingSystemVersion'] =
pe.OPTIONAL_HEADER.MajorOperatingSystemVersion
    res['MinorOperatingSystemVersion'] =
pe.OPTIONAL_HEADER.MinorOperatingSystemVersion
    res['MajorImageVersion'] =
pe.OPTIONAL_HEADER.MajorImageVersion
    res['MinorImageVersion'] =
pe.OPTIONAL_HEADER.MinorImageVersion
    res['MajorSubsystemVersion'] =
pe.OPTIONAL_HEADER.MajorSubsystemVersion
    res['MinorSubsystemVersion'] =
pe.OPTIONAL_HEADER.MinorSubsystemVersion
    res['SizeOfImage'] = pe.OPTIONAL_HEADER.SizeOfImage
    res['SizeOfHeaders'] =
pe.OPTIONAL_HEADER.SizeOfHeaders
    res['CheckSum'] = pe.OPTIONAL_HEADER.CheckSum
    res['Subsystem'] = pe.OPTIONAL_HEADER.Subsystem
    res['DllCharacteristics'] =
pe.OPTIONAL_HEADER.DllCharacteristics
    res['SizeOfStackReserve'] =
pe.OPTIONAL_HEADER.SizeOfStackReserve
    res['SizeOfStackCommit'] =
pe.OPTIONAL_HEADER.SizeOfStackCommit
    res['SizeOfHeapReserve'] =
pe.OPTIONAL_HEADER.SizeOfHeapReserve
```

```
    res['SizeOfHeapCommit'] =
pe.OPTIONAL_HEADER.SizeOfHeapCommit
    res['LoaderFlags'] = pe.OPTIONAL_HEADER.LoaderFlags
    res['NumberOfRvaAndSizes'] =
pe.OPTIONAL_HEADER.NumberOfRvaAndSizes

    # Sections
    res['SectionsNb'] = len(pe.sections)
    entropy = list(map(lambda x:x.get_entropy(), pe.sections))
    res['SectionsMeanEntropy'] =
sum(entropy)/float(len(entropy))
    res['SectionsMinEntropy'] = min(entropy)
    res['SectionsMaxEntropy'] = max(entropy)


    raw_sizes = list(map(lambda x:x.SizeOfRawData,
pe.sections))
    res['SectionsMeanRawsize'] =
sum(raw_sizes)/float(len(raw_sizes))
    res['SectionsMinRawsize'] = min(raw_sizes)
    res['SectionsMaxRawsize'] = max(raw_sizes)
    virtual_sizes = list(map(lambda x:x.Misc_VirtualSize,
pe.sections))
    res['SectionsMeanVirtualsize'] =
sum(virtual_sizes)/float(len(virtual_sizes))
    res['SectionsMinVirtualsize'] = min(virtual_sizes)
    res['SectionMaxVirtualsize'] = max(virtual_sizes)

    #Imports
    try:
        res['ImportsNbDLL'] =
len(pe.DIRECTORY_ENTRY_IMPORT)
        imports = list(sum([x.imports for x in
pe.DIRECTORY_ENTRY_IMPORT], []))
        res['ImportsNb'] = len(imports)
        res['ImportsNbOrdinal'] = len(list(filter(lambda x:x.name is 
None, imports)))
    except AttributeError:
        res['ImportsNbDLL'] = 0
        res['ImportsNb'] = 0
        res['ImportsNbOrdinal'] = 0

    #Exports
    try:
        res['ExportNb'] = 
len(pe.DIRECTORY_ENTRY_EXPORT.symbols)
    except AttributeError:
        # No export
        res['ExportNb'] = 0
    #Resources
    resources= get_resources(pe)
    res['ResourcesNb'] = len(resources)
    if len(resources)> 0:
        entropy = list(map(lambda x:x[0], resources))
        res['ResourcesMeanEntropy'] = 
sum(entropy)/float(len(entropy))
        res['ResourcesMinEntropy'] = min(entropy)
        res['ResourcesMaxEntropy'] = max(entropy)
        sizes = list(map(lambda x:x[1], resources))
        res['ResourcesMeanSize'] = sum(sizes)/float(len(sizes))
        res['ResourcesMinSize'] = min(sizes)
        res['ResourcesMaxSize'] = max(sizes)
    else:
        res['ResourcesNb'] = 0
        res['ResourcesMeanEntropy'] = 0
        res['ResourcesMinEntropy'] = 0
        res['ResourcesMaxEntropy'] = 0
        res['ResourcesMeanSize'] = 0
        res['ResourcesMinSize'] = 0
        res['ResourcesMaxSize'] = 0
    # Load configuration size
    try:
        res['LoadConfigurationSize'] =
pe.DIRECTORY_ENTRY_LOAD_CONFIG.struct.Size
    except AttributeError:
        res['LoadConfigurationSize'] = 0


    # Version configuration size
    try:
        version_infos = get_version_info(pe)
        res['VersionInformationSize'] = len(version_infos.keys())
    except AttributeError:
        res['VersionInformationSize'] = 0
    return res

def filepath(path2, processvar2):
    process =processvar2
    global qwe2
    qwe2 = path2
    clf = joblib.load(os.path.join(pickel_dir,'classifier.pkl'))
    features =
pickle.loads(open(os.path.join(pickel_dir,'features.pkl'),'rb').read
())
    data = extract_infos(path2)
    pe_features = list(map(lambda x: data[x], features))

    res = clf.predict([pe_features])[0]
    process.set('The file %s is %s' % (os.path.basename(path2),
['malicious', 'legitimate'][res]))
```

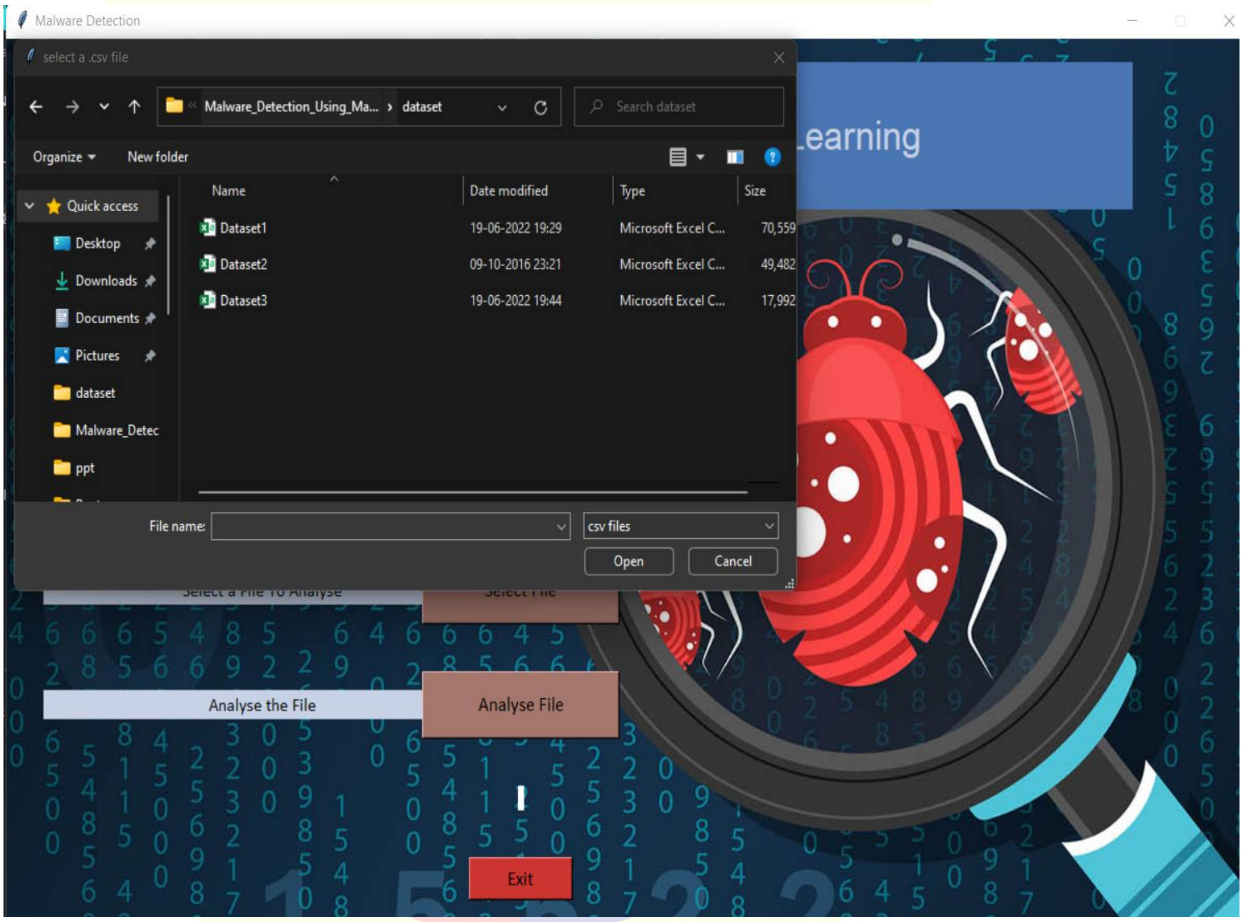
5. Implementation

5.1 Main Screen



screenshot 1:Main Screen

5.2 Select Dataset

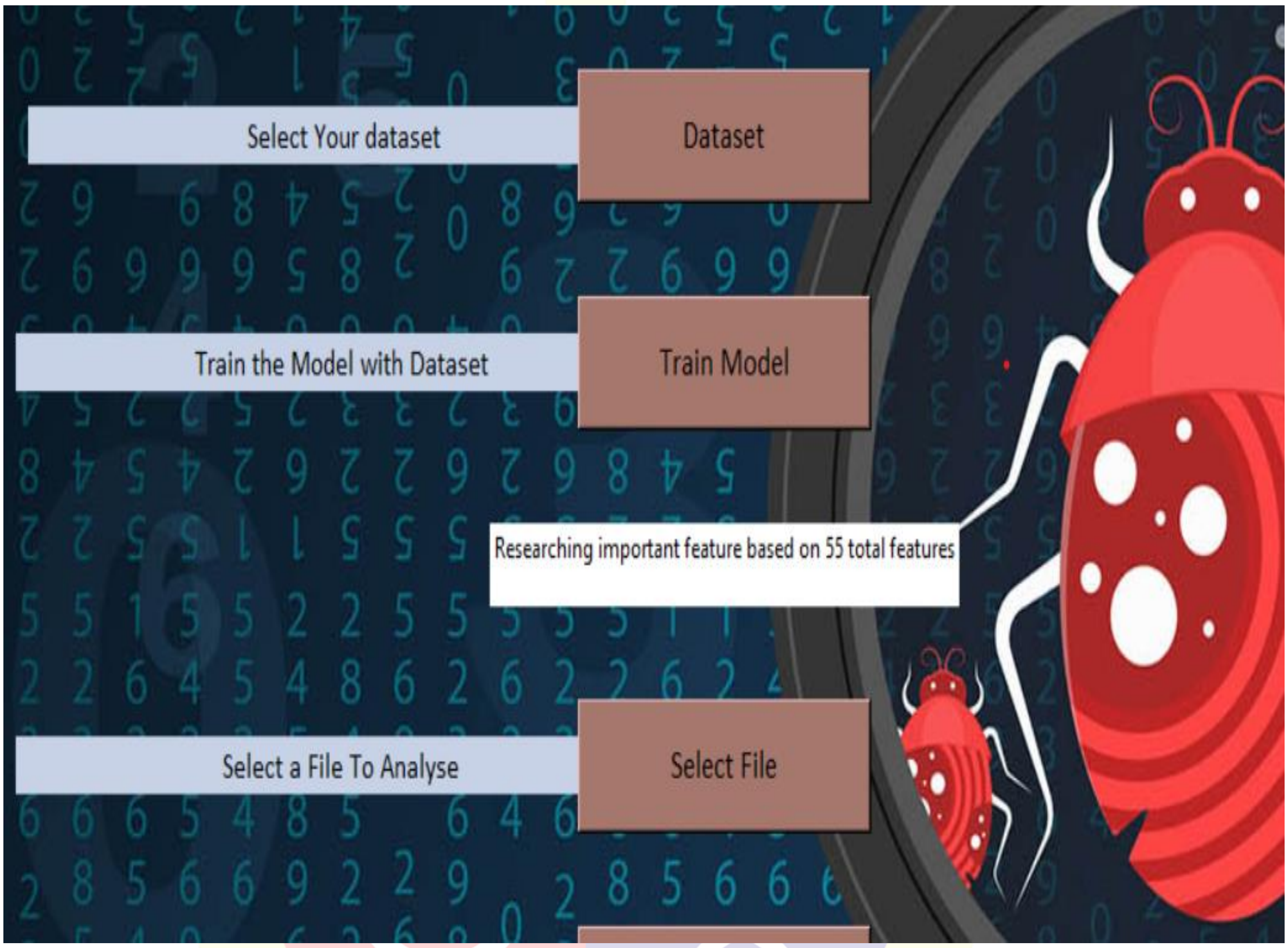


Name	Date modified	Type	Size
Dataset1	19-06-2022 19:29	Microsoft Excel C...	70,559
Dataset2	09-10-2016 23:21	Microsoft Excel C...	49,482
Dataset3	19-06-2022 19:44	Microsoft Excel C...	17,992

File name:
csv files
Open
Cancel

screenshot 2: Dataset

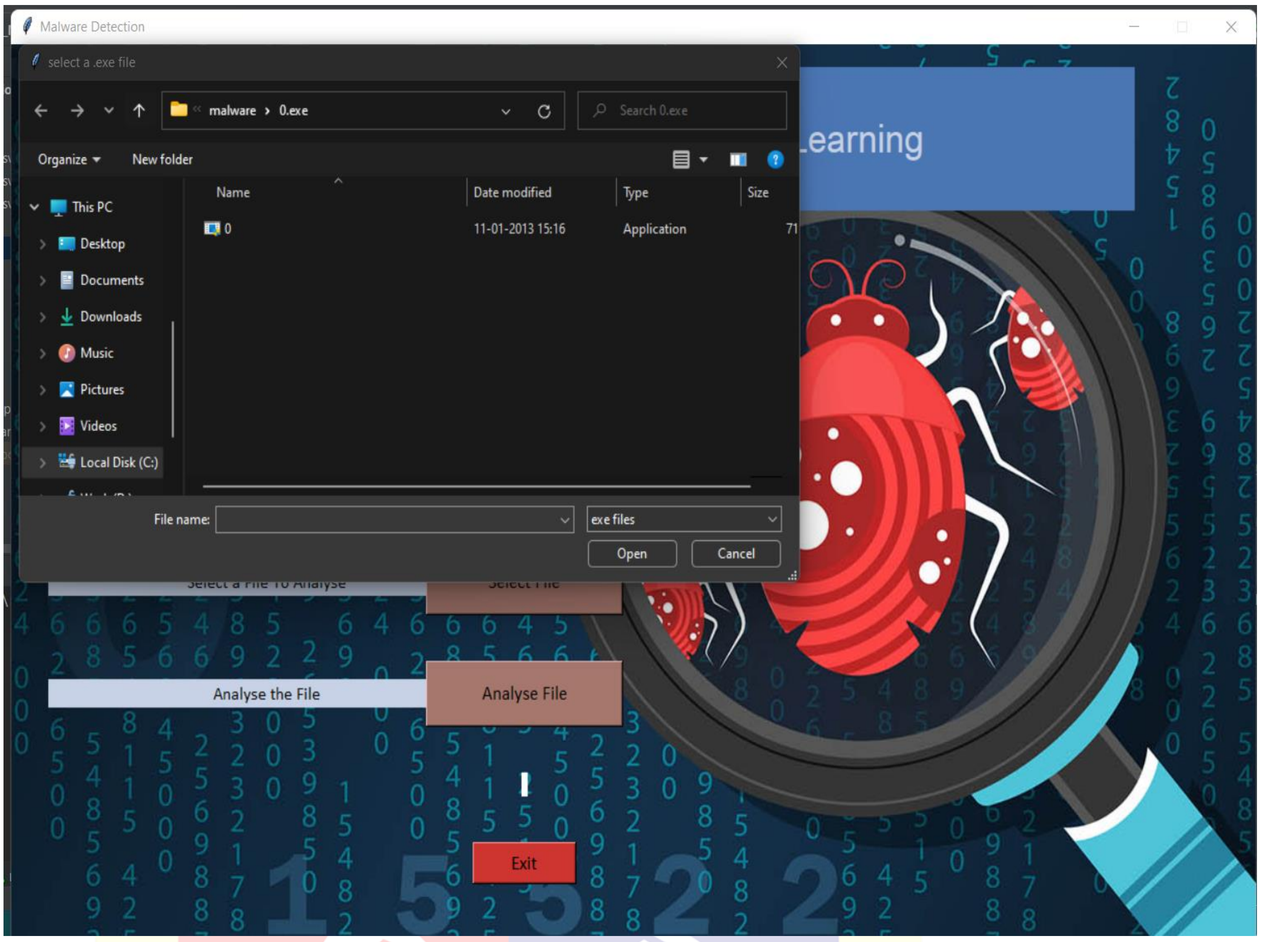
5.3 Train Model



Researching important feature based on 55 total features

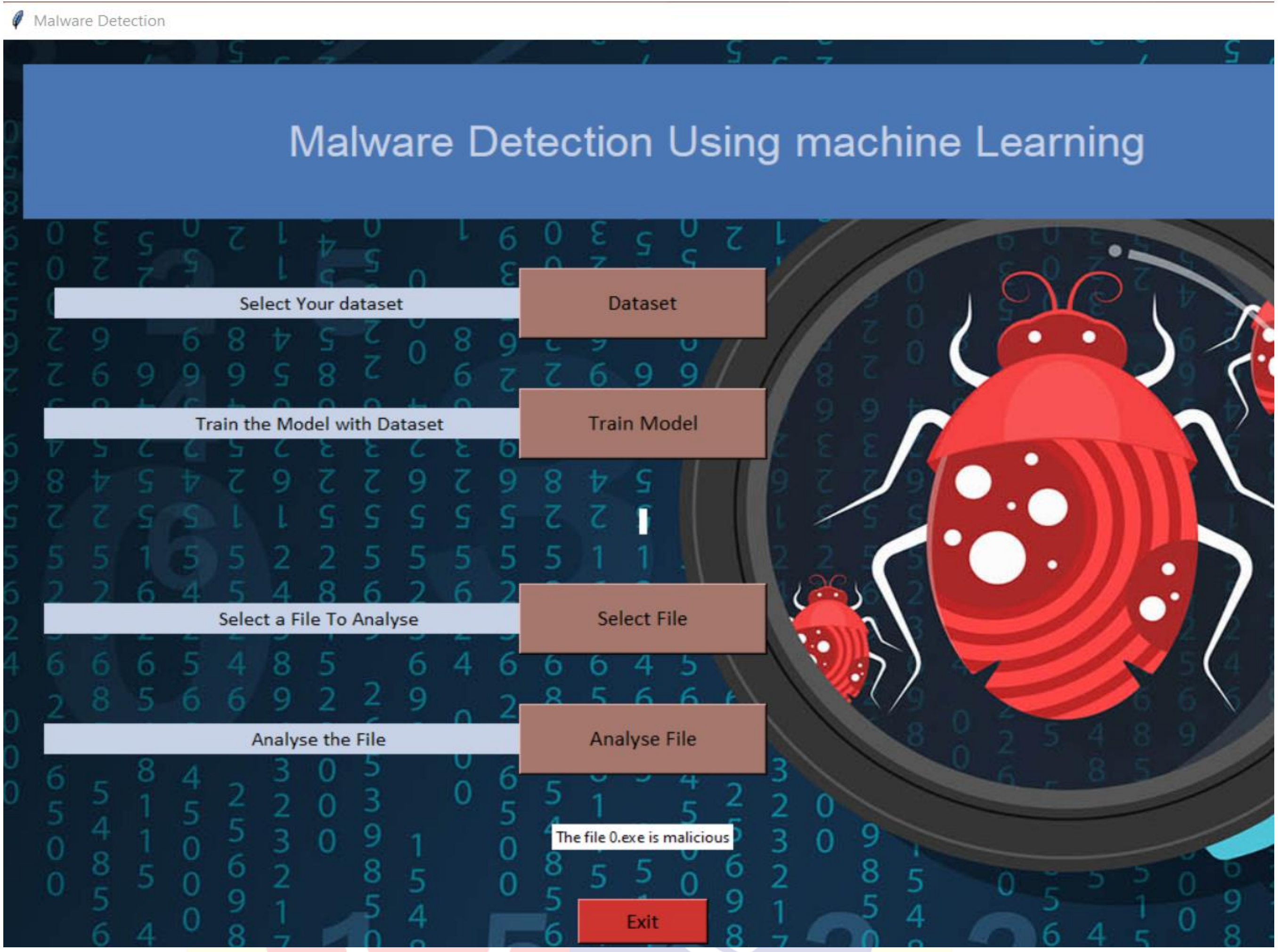
screenshot 3:Training

5.4 Select EXE File



screenshot 4:File Selection EXE

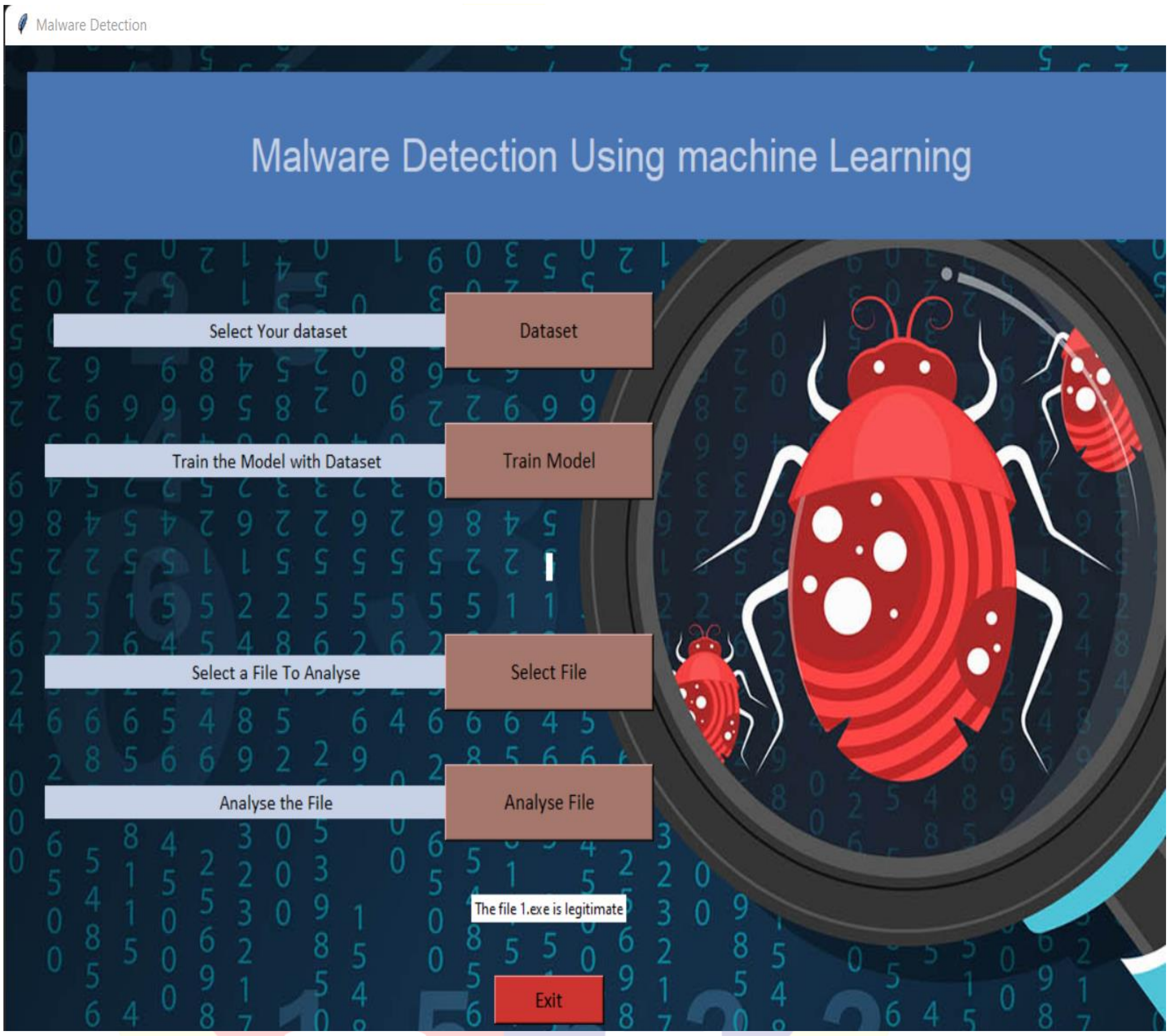
5.5 **Result – Malware**



The file 0.exe is malicious

screenshot 5:Result Malware

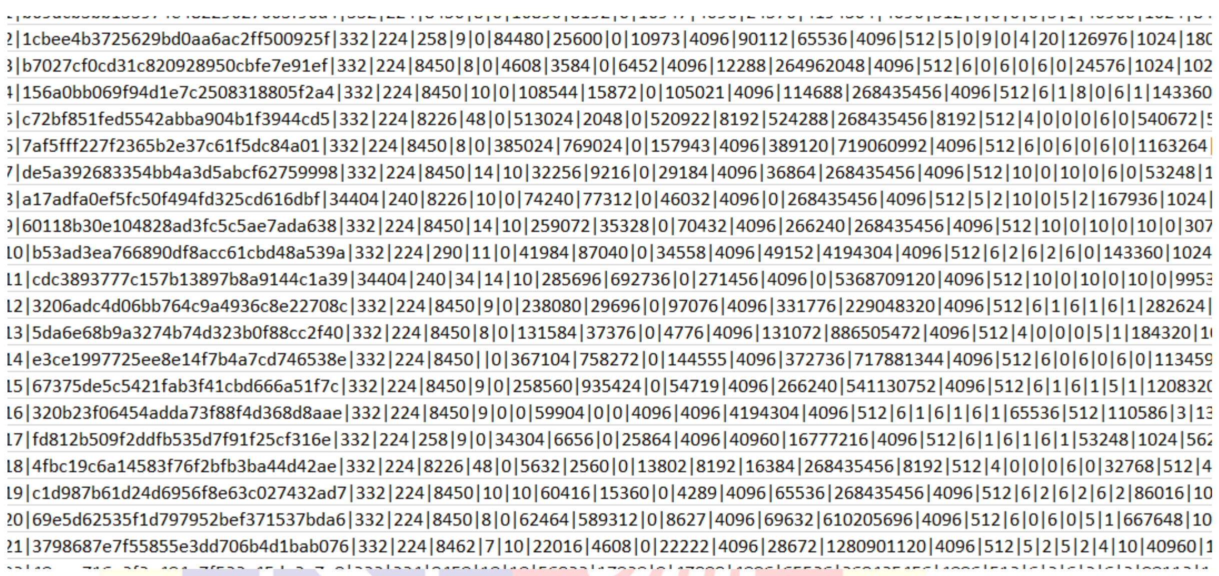
5.6 Result – Legitimate



The file 1.exe is legitimate

screenshot 6: Result Legitimate

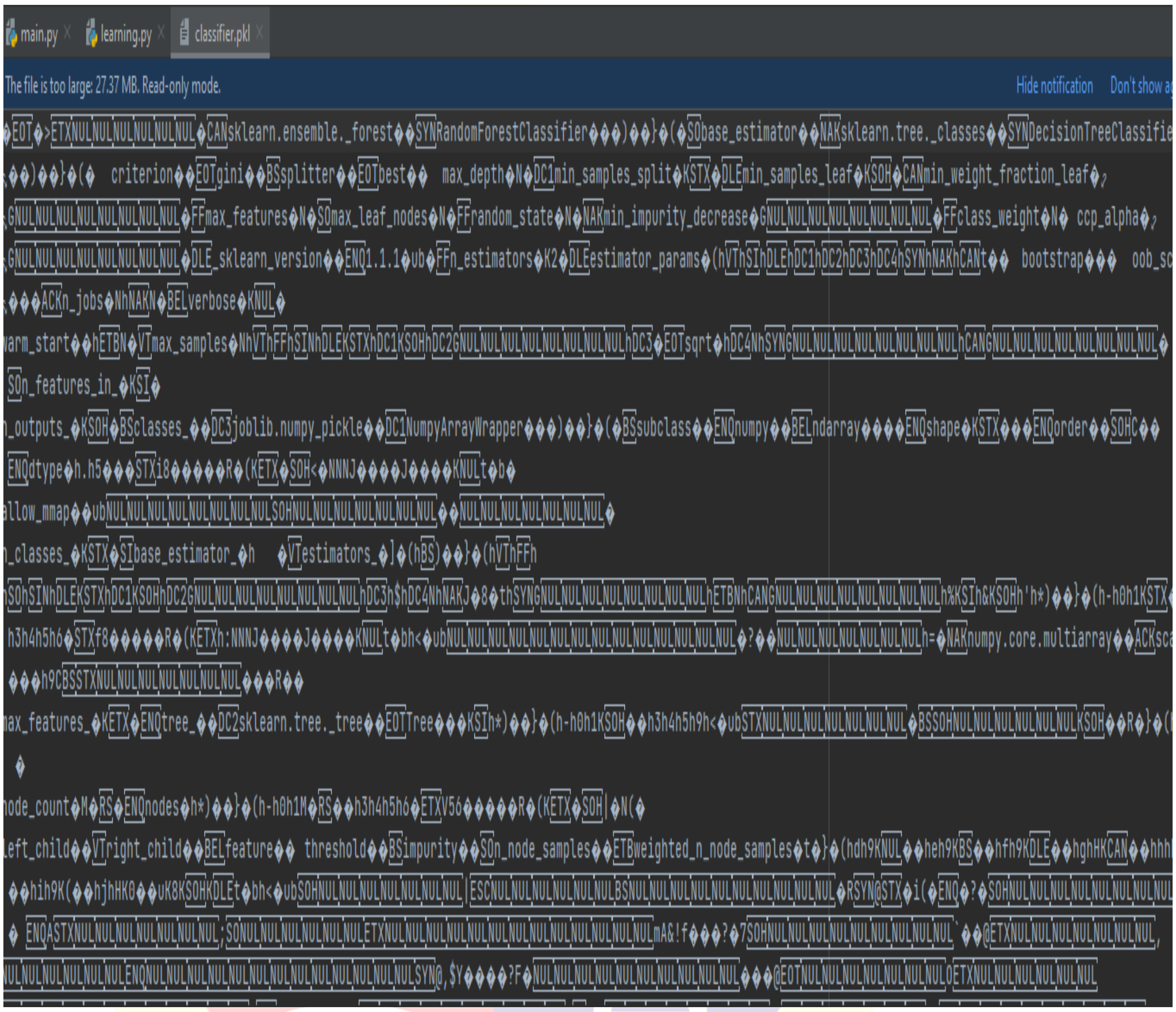
5.7 Dataset



```
2|1cbee4b3725629bd0aa6ac2ff500925f|332|224|258|9|0|84480|25600|0|10973|4096|90112|65536|4096|512|5|0|9|0|4|20|126976|1024|180
3|b7027cf0cd31c820928950cbfe7e91ef|332|224|8450|8|0|4608|3584|0|6452|4096|12288|264962048|4096|512|6|0|6|0|6|0|24576|1024|102
4|156a0bb069f94d1e7c2508318805f2a4|332|224|8450|10|0|108544|15872|0|105021|4096|114688|268435456|4096|512|6|1|8|0|6|1|143360
5|c72bf851fed5542abba904b1f3944cd5|332|224|8226|48|0|513024|2048|0|520922|8192|524288|268435456|8192|512|4|0|0|0|6|0|540672|5
6|7af5fff227f2365b2e37c61f5dc84a01|332|224|8450|8|0|385024|769024|0|157943|4096|389120|719060992|4096|512|6|0|6|0|6|0|1163264|
7|de5a392683354bb4a3d5abcf62759998|332|224|8450|14|10|32256|9216|0|29184|4096|36864|268435456|4096|512|10|0|10|0|6|0|53248|1
8|a17adfa0ef5fc50f494fd325cd616dbf|34404|240|8226|10|0|74240|77312|0|46032|4096|0|268435456|4096|512|5|2|10|0|5|2|167936|1024|
9|60118b30e104828ad3fc5c5ae7ada638|332|224|8450|14|10|259072|35328|0|70432|4096|266240|268435456|4096|512|10|0|10|0|10|0|307
10|b53ad3ea766890df8acc61cbd48a539a|332|224|290|11|0|41984|87040|0|34558|4096|49152|4194304|4096|512|6|2|6|2|6|0|143360|1024
11|cdc3893777c157b13897b8a9144c1a39|34404|240|34|14|10|285696|692736|0|271456|4096|0|5368709120|4096|512|10|0|10|0|10|0|9953
12|3206adc4d06bb764c9a4936c8e22708c|332|224|8450|9|0|238080|29696|0|97076|4096|331776|229048320|4096|512|6|1|6|1|6|1|282624|
13|5da6e68b9a3274b74d323b0f88cc2f40|332|224|8450|8|0|131584|37376|0|4776|4096|131072|886505472|4096|512|4|0|0|0|5|1|184320|1
14|e3ce1997725ee8e14f7b4a7cd746538e|332|224|8450||0|367104|758272|0|144555|4096|372736|717881344|4096|512|6|0|6|0|6|0|113459
15|67375de5c5421fab3f41cbd666a51f7c|332|224|8450|9|0|258560|935424|0|54719|4096|266240|541130752|4096|512|6|1|6|1|5|1|120832
16|320b23f06454adda73f88f4d368d8aae|332|224|8450|9|0|0|59904|0|0|4096|4096|4194304|4096|512|6|1|6|1|6|1|65536|512|110586|3|13
17|fd812b509f2ddfb535d7f91f25cf316e|332|224|258|9|0|34304|6656|0|25864|4096|40960|16777216|4096|512|6|1|6|1|6|1|53248|1024|562
18|4fbc19c6a14583f76f2bfb3ba44d42ae|332|224|8226|48|0|5632|2560|0|13802|8192|16384|268435456|8192|512|4|0|0|0|6|0|32768|512|4
19|c1d987b61d24d6956f8e63c027432ad7|332|224|8450|10|10|60416|15360|0|4289|4096|65536|268435456|4096|512|6|2|6|2|6|2|86016|10
20|69e5d62535f1d797952bef371537bda6|332|224|8450|8|0|62464|589312|0|8627|4096|69632|610205696|4096|512|6|0|6|0|5|1|667648|10
21|3798687e7f55855e3dd706b4d1bab076|332|224|8462|7|10|22016|4608|0|22222|4096|28672|1280901120|4096|512|5|2|5|2|4|10|40960|1
```

screenshot 7: Dataset

5.8 **Classifier Pickle**

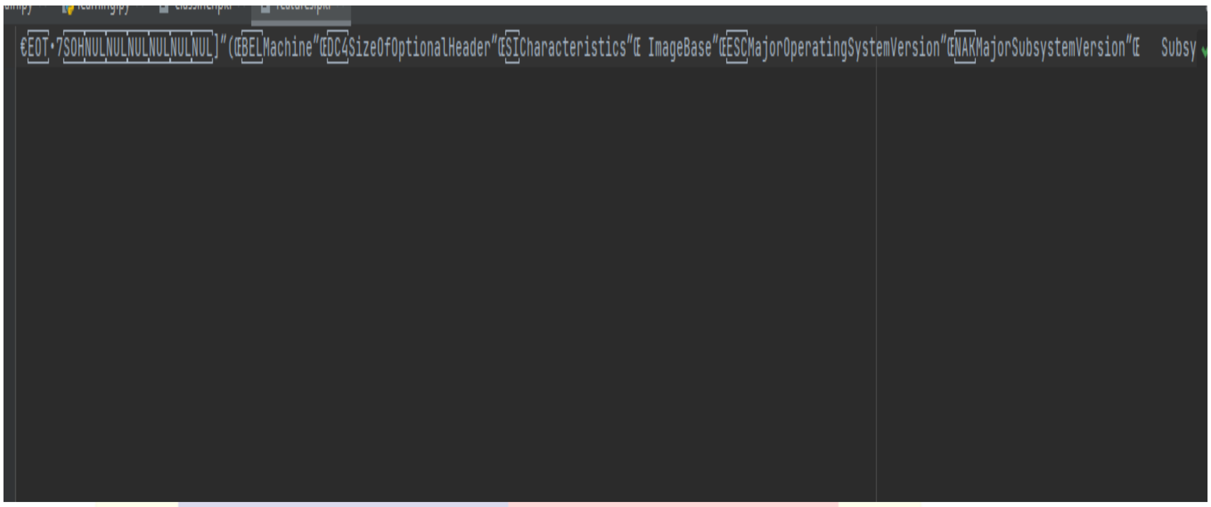


The file is too large: 27.37 MB. Read-only mode.

```
�EOT�>ETXNULNULNULNULNULNUL�CANsklearn.ensemble._forest��SYNRandomForestClassifier���)��}�(�SObase_estimator��NAKsklearn.tree._classes��SYNDecisionTreeClassifie
��)��}�(�  criterion��EOTgini��BSsplitter��EOTbest��  max_depth�N�DC1min_samples_split�KSTX�DLEmin_samples_leaf�KSOH�CANmin_weight_fraction_leaf�
GNULNULNULNULNULNULNULNUL�FFmax_features�N�SOmax_leaf_nodes�N�FFrandom_state�N�NAKmin_impurity_decrease�GNULNULNULNULNULNULNULNUL�FFclass_weight�N� ccp_alpha�
GNULNULNULNULNULNULNULNUL�DLE_sklearn_version��ENQ1.1.1�ub�FFn_estimators�K2�DLEestimator_params�(hVThSIhDLEhDC1hDC2hDC3hDC4hSYNhNAKhCANt��  bootstrap���  oob_sc
���ACKn_jobs�NhNAKN�BELverbose�KNUL�
warm_start��hETBN�VTmax_samples�NhVThFFhSINhDLEKSTXhDC1KSOHhDC2GNULNULNULNULNULNULNULNULhDC3�EOTsqrt�hDC4NhSYNGNULNULNULNULNULNULNULNULhCANGNULNULNULNULNULNULNULNUL�
SOn_features_in_�KSI�
n_outputs_�KSOH�BSclasses_��DC3joblib.numpy_pickle��DC1NumpyArrayWrapper���)��}�(�BSsubclass��ENQnumpy��BELndarray����ENQshape�KSTX���ENQorder��SOHC��
ENQdtype�h.h5���STXi8�����R�(KETX�SOH<�NNNJ����J����KNULt�b�
allow_mmap��ubNULNULNULNULNULNULNULNULSOHNULNULNULNULNULNULNUL��NULNULNULNULNULNULNUL�
n_classes_�KSTX�SIbase_estimator_�h   �VTestimators_�]�(hBS)��}�(hVThFFh
hSOhSINhDLEKSTXhDC1KSOHhDC2GNULNULNULNULNULNULNULNULhDC3h$hDC4NhNAKJ�8�thSYNGNULNULNULNULNULNULNULNULhETBNhCANGNULNULNULNULNULNULNULNULh%KSIh&KSOHh'h*)��}�(h-h0h1KSTX
 h3h4h5h6�STXf8�����R�(KETXh:NNNJ����J����KNULt�bh<�ubNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL�?��NULNULNULNULNULNULNULh=�NAKnumpy.core.multiarray��ACKsca
 ���h9CBSSTXNULNULNULNULNULNULNUL���R��
max_features_�KETX�ENQtree_��DC2sklearn.tree._tree��EOTTree���KSIh*)��}�(h-h0h1KSOH��h3h4h5h9h<�ubSTXNULNULNULNULNULNULNUL�BSSOHNULNULNULNULNULNULKSOH��R�}�(
 �
node_count�M�RS�ENQnodes�h*)��}�(h-h0h1M�RS��h3h4h5h6�ETXV56�����R�(KETX�SOH|�N(�
left_child��VTright_child��BELfeature��  threshold��BSimpurity��SOn_node_samples��ETBweighted_n_node_samples�t�}�(hdh9KNUL��heh9KBS��hfh9KDLE��hghHKCAN��hhh
 ��hih9K(��hjhHK0��uK8KSOHKDLEt�bh<�ubSOHNULNULNULNULNULNULNUL|ESCNULNULNULNULNULNULBSNULNULNULNULNULNULNULNULNULNUL�RSYN@STX�i(�ENQ�?�SOHNULNULNULNULNULNULNULNUL
� ENQASTXNULNULNULNULNULNULNUL;SONULNULNULNULNULNULETXNULNULNULNULNULNULNULNULNULNULNULmA&!f���?�7SOHNULNULNULNULNULNULNULNULNUL`��@ETXNULNULNULNULNULNULNUL,
NULNULNULNULNULNULENQNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULSYN@,$Y����?F�NULNULNULNULNULNULNULNULNULNUL���@EOTNULNULNULNULNULNULNUL0ETXNULNULNULNULNULNUL
```

screenshot 8: Classifier Pickle

5.9 Features Pickle



```
€EOT·7SOHNULNULNULNULNULNUL]"(ŒBELMachine"ŒDC4SizeOfOptionalHeader"ŒSICharacteristics"Œ ImageBase"ŒESCMajorOperatingSystemVersion"ŒNAKMajorSubsystemVersion"Œ   Subsy
```

screenshot 9: Features Pickle

6. Future Scope of Work

- Finding new dataset to train the model with good number of features.
- Training the model with different algorithms.
- Implementation Dynamic malware analysis using machine learning.
- Try to implement scanner for malicious URL.
- Increase the accuracy of training model by modifying dataset.
- Discover and analyze different forms of malware.

7. Conclusion

- We came to know about static malware analysis using machine learning.
- This method is not completely accurate.
- Selection of dataset and reduction of field is necessary to reduce overfitting.
- This Method is just for static analysis using PE file so one has to implement advanced static analysis and dynamic analysis Before concluding anything.
- We came to know about different forms of malware.

8. References

- Paper 1: - https://www.sciencedirect.com/science/article/abs/pii/S1383762120301442
- Paper 2: - https://link.springer.com/article/10.1631/FITEE.1601325
- Paper 3: -https://link.springer.com/chapter/10.1007/978-3-030-04780-1_28
- Paper 4: - https://pdfs.semanticscholar.org/2fb8/c4d2a203bc918f5d936036428579ecb09d92.pdf
- Paper 5: -https://iopscience.iop.org/article/10.1088/1757-899X/846/1/012036/meta
- PyCharm: -https://www.jetbrains.com/pycharm/
- Kaggle: -https://www.kaggle.com/c/malware-detection
- Practical Malware Analysis: - https://www.oreilly.com/library/view/practical-malware-analysis/9781593272906/
- Mastering Machine Learning for Penetration Testing: - https://www.packtpub.com/product/mastering-machine-learning-for-penetration-testing/9781788997409