

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/394880807>

# Malware Detection in Portable Executable (PE) Files Using Supervised Machine Learning (Applied Artificial Intelligence in Cyber Security)

Technical Report · February 2025

DOI: 10.13140/RG.2.2.25846.66883

---

CITATIONS

0

READS

5

1 author:



Muhammad Qasim Raza

University of South Wales

11 PUBLICATIONS 133 CITATIONS

SEE PROFILE



Faculty of Computing, Engineering and Science

National Cyber Security Academy

University of South Wales

MSc Applied Cyber Security

“Malware Detection Using Supervised Machine Learning”

(Applied Artificial Intelligence in Cyber Security)

IY4S734

Submitted To:

Muhammad Abdullah Awais

Submitted By:

Muhammad Qasim Raza

30115377

## Table of Contents

Table of Figures:.....	5
1 Introduction to AI in Cybersecurity.....	6
1.1 How AI Helps in Cybersecurity .....	6
1.2 Challenges in Identifying Malware .....	6
1.3 Supervised Learning for Malware Detection .....	7
2 Understanding PE File Structure.....	8
2.1 What is the PE File Structure?.....	8
2.2 Key Features for Malware Analysis .....	8
2.2.1 Number of Sections:.....	8
2.2.2 Section Sizes: .....	9
2.2.3 Imports:.....	9
2.2.4 Exports: .....	9
2.2.5 Entropy:.....	9
2.2.6 File Metadata: .....	9
2.3 How These Features Help Machine Learning Models.....	9
3 Data Set .....	11
3.1 Where and How to Find Malware and Normal PE Files.....	11
3.1.1 Malware Samples:.....	11
3.1.2 Normal PE Files:.....	11
3.2 Step-by-Step Guide to Extract Features from PE Files Using Python .....	11
3.2.1 Set Up the Environment: .....	11
3.2.2 Write a Python Script to Extract Features:.....	12
3.2.3 Validate the Extracted Features: .....	13
3.3 Issues and Solutions .....	13
3.3.1 Small Dataset: .....	13
3.3.2 Unknown Extensions: .....	13
3.3.3 Imbalance Between Normal and Malicious Files:.....	13
4 Model Training and Testing .....	15

4.1	Supervised Learning Algorithm .....	15
4.2	Accuracy and F1-Score .....	16
4.3	How the Model Classifies Files .....	17
4.3.1	Feature Extraction .....	17
4.3.2	Training the Model .....	17
4.3.3	Making Predictions .....	17
4.3.4	Output the Result .....	18
5	Evidence: .....	19
5.1	Complete python script in our report .....	19
5.1.1	Step 1: Feature extraction from PE Files .....	19
5.1.2	Step 2: Dataset Preprocessing:.....	20
5.1.3	Step 3: Training and Evaluation Pipeline: .....	21
5.1.4	Step 4: Malware Detection Function: .....	22
5.1.5	Step 5: Testing Normal and Malicious file: .....	23
5.2	Malware Samples:.....	23
5.2.1	Malware Sample Files .....	23
5.3	Normal Samples: .....	24
5.3.1	Normal Sample Files .....	24
5.4	Malicious file as a malware .....	25
5.4.1	Output:.....	25
5.5	Normal file as a Normal .....	26
5.5.1	Output:.....	26
6	Model as a Cybersecurity Tool.....	28
6.1	Integration into a Cybersecurity System .....	28
6.1.1	Malware Detection Module: .....	28
6.1.2	Preprocessing and Feature Extraction: .....	28
6.1.3	Threat Mitigation Pipeline:.....	28
6.1.4	Behavioral Monitoring:.....	28
6.1.5	Integration with Threat Intelligence: .....	29

6.2	Advantages in a Larger Cybersecurity Ecosystem .....	29
6.2.1	Real-Time Threat Detection: .....	29
6.2.2	Scalability:.....	29
6.2.3	Automation:.....	29
6.2.4	Adaptability: .....	29
6.3	Enhancements and Features for Practical Deployment .....	29
6.3.1	User-Friendly Dashboard: .....	29
6.3.2	Integration with Endpoint Protection: .....	30
6.3.3	Regular Updates: .....	30
6.3.4	Cross-Platform Support: .....	30
6.3.5	Incident Reporting:.....	30
6.4	Example Workflow in a Cybersecurity Tool .....	30
6.4.1	File Upload or Detection: .....	30
6.4.2	Feature Extraction: .....	30
6.4.3	Classification:.....	30
6.4.4	Response: .....	30
6.4.5	Post-Incident Learning:.....	31
7	Conclusion.....	32
7.1	Summary of Key Points .....	32
7.2	Reflection and Real-World Applications .....	32
8	References .....	34

## Table of Figures:

Figure 5.1.1: Feature Extraction from PE Files .....	19
Figure 5.1.2: Dataset Preprocessing.....	20
Figure 5.1.3: Training and Evaluation Pipeline .....	21
Figure 5.1.4: Malware Detection Function .....	22
Figure 5.1.5: Testing Normal and Malicious file .....	23
Figure 5.2.1: Malware Samples .....	24
Figure 5.3.1: Normal Samples.....	25
Figure 5.4.1: Malware File Output .....	26
Figure 5.5.1: Normal File Output.....	27

## 1 Introduction to AI in Cybersecurity

Artificial intelligence (AI) is transforming cybersecurity by delivering innovative tools and approaches for mitigating online attacks. Finding and stopping malware, malicious software intended to damage networks, steal data, or interfere with services, is one of the biggest issues in cybersecurity. Because AI makes threat identification quicker, more accurate, and more flexible, it is proving to be extremely useful in the fight against such threats.

### 1.1 How AI Helps in Cybersecurity

AI plays a crucial role in enhancing cybersecurity by analyzing vast amounts of data and identifying patterns that humans might miss. In the context of malware detection, AI can spot unusual behaviors in files, network traffic, and system activities that might indicate a cyberattack. Machine learning, a branch of AI, is particularly effective in detecting new, unknown malware, often referred to as zero-day threats.

For instance, traditional cybersecurity systems rely on databases of known malware signatures to identify threats. However, cybercriminals constantly create new malware variants that escape from these systems. AI overcomes this limitation by recognizing suspicious activities or anomalies, even if the malware is entirely new. This proactive approach allows organizations to detect and respond to threats much faster than traditional methods.

### 1.2 Challenges in Identifying Malware

Identifying malware is not an easy task, as cybercriminals use sophisticated techniques to disguise their attacks. Some malware is designed to remain dormant until triggered by a specific event, making it harder to detect. Others use polymorphic techniques to change their code structure constantly, evading signature-based detection methods.

Another challenge is the sheer volume of data that cybersecurity systems must analyze. Large organizations often process millions of files, emails, and network logs daily. Manually reviewing this data for signs of malware is nearly impossible. Moreover, distinguishing between legitimate activities and malicious ones can be tricky, as attackers often mimic normal user behavior to avoid detection.

### 1.3 Supervised Learning for Malware Detection

Supervised learning, a type of machine learning, can be a powerful tool in detecting malicious files. In supervised learning, a model is trained on labeled data, which means it learns from examples where the outcome (e.g., whether a file is malicious or safe) is already known. This training process enables the model to recognize patterns and features associated with malware. (Priyanka Jadav, April 27, 2024)

For example, a supervised learning model might analyze features such as the size of a file, its code structure, or the type of system calls it makes. By comparing these features with its training data, the model can predict whether a new file is likely to be malicious. This approach is highly effective because it can learn from past examples and improve over time.

Supervised learning also reduces the likelihood of false positives, which occur when a system incorrectly flags safe files as malicious. By continuously refining its understanding of what constitutes malware, the model becomes more accurate and reliable.

AI, and particularly supervised learning, is transforming the way cybersecurity professionals detect and respond to malware. Despite the challenges posed by evolving cyber threats, AI provides tools to identify even the most sophisticated attacks. As technology advances, AI-driven cybersecurity solutions will continue to play a vital role in protecting individuals and organizations from malicious actors.

## 2 Understanding PE File Structure

Portable Executable (PE) files are a fundamental part of Windows operating systems, serving as the format for executable files, dynamic link libraries (DLLs), and other binary programs. These files contain essential information for the system to load and execute programs. Understanding the structure of PE files is critical in malware analysis because many malicious programs disguise themselves as legitimate PE files to bypass security measures.

### 2.1 What is the PE File Structure?

The PE file structure is like a roadmap that tells the operating system how to execute a program. It is divided into sections, each serving a specific purpose, such as holding executable code, data, or information about external libraries. Key components of a PE file include: (Microsoft, 02/29/2024)

- **Headers:** Contain metadata about the file, such as its type (executable or DLL), timestamp, and entry point address.
- **Sections:** Hold the main content of the program, including code, resources, and data. Common sections include .text (executable code), .data (initialized data), and .rdata (read-only data).
- **Imports and Exports:** Lists of functions and libraries the file depends on (imports) or provides to other programs (exports).

These structural elements are crucial for malware analysis because they provide insights into the file's behavior and purpose.

### 2.2 Key Features for Malware Analysis

Several features of PE files can be analyzed to determine whether a file is malicious or legitimate. These features are used by machine learning models to detect patterns indicative of malware. Some of the most common features include:

#### 2.2.1 Number of Sections:

- a. Legitimate PE files typically have a standard number of sections.

- b. Malware authors often add unusual sections or modify existing ones to hide malicious code.

#### 2.2.2 Section Sizes:

- a. Normal PE files have sections with predictable sizes.
- b. Suspiciously large or tiny sections can indicate packed or encrypted malware.

#### 2.2.3 Imports:

- a. Malware often imports functions related to suspicious activities, such as CreateRemoteThread or VirtualAlloc, which are used to inject code into other processes.
- b. The absence of expected imports (e.g., standard Windows API functions) can also be a red flag.

#### 2.2.4 Exports:

- a. Files with unusual or excessive exports may be designed to interact with multiple programs in an unintended way, often seen in malicious DLLs.

#### 2.2.5 Entropy:

- a. Entropy measures the randomness of data within a file.
- b. High entropy sections might indicate that the file is packed or encrypted, a common tactic to avoid detection.

#### 2.2.6 File Metadata:

- a. Information like compile timestamps and file signatures can be checked against known malware databases.
- b. Invalid or unusual metadata can hint at tampering.

### 2.3 How These Features Help Machine Learning Models

Machine learning models use these features to identify patterns that distinguish malware from normal files. For instance:

- By analyzing the combination of section sizes and entropy, the model can detect files that use packing techniques to hide malicious code.
- Unusual imports or exports can signal the presence of functions commonly used by malware, prompting further investigation.
- The number and structure of sections provide clues about whether the file was altered from its original state.

By feeding the model large amounts of labeled data (e.g., known malware and legitimate files), it learns to recognize the subtle differences and make predictions about new, unseen files. This automated analysis is especially valuable in detecting polymorphic or previously unknown malware.

Understanding the PE file structure is a cornerstone of malware analysis. Features like section sizes, imports, exports, and entropy provide vital clues about a file's behavior, enabling machine learning models to make informed decisions. By leveraging these insights, cybersecurity professionals can detect and mitigate threats more effectively, ensuring a safer computing environment.

### 3 Data Set

Building a dataset for malware analysis requires finding reliable sources for both malware and normal PE (Portable Executable) files. Ensuring ethical practices and maintaining safety are crucial throughout this process, as mishandling malware can cause unintentional harm. (Iryna Sydorenko, December 21, 2023)

#### 3.1 Where and How to Find Malware and Normal PE Files

##### 3.1.1 Malware Samples:

- a. **Online Repositories:** Reputable repositories like VirusTotal, MalwareBazaar, and CIC Andmal2020 provide malware samples for research purposes.
- b. **Access Requirements:** These platforms usually require you to register and agree to ethical usage policies. Some may only be accessible to verified researchers or organizations.
- c. **Safety Measures:** Always download and handle malware samples within a controlled environment, such as a virtual machine (VM) isolated from your main network.

##### 3.1.2 Normal PE Files:

- a. **Windows System Files:** Obtain legitimate PE files from your own Windows operating system or open-source software projects like Sysinternals.
- b. **Software Repositories:** Platforms like GitHub and SourceForge provide executable files from various open-source projects.
- c. **Safety Measures:** Verify the authenticity and integrity of the files to ensure they are not tampered with.

#### 3.2 Step-by-Step Guide to Extract Features from PE Files Using Python

Feature extraction is essential for analyzing PE files. Below is a step-by-step guide to extract useful features:

##### 3.2.1 Set Up the Environment:

- a. Install Python and required libraries:

- import pandas as pd
- import numpy as np
- from sklearn.model\_selection import train\_test\_split
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.metrics import classification\_report, accuracy\_score
- from sklearn.pipeline import Pipeline
- from sklearn.preprocessing import StandardScaler
- import pefile
- import os

### 3.2.2 Write a Python Script to Extract Features:

```
# Feature extraction from PE files

def extract_pe_features(file_path):

    """Extract basic features from a PE file."""

    try:

        pe = pefile.PE(file_path)

        features = {

            "NumberOfSections": len(pe.sections),

            "ImageBase": pe.OPTIONAL_HEADER.ImageBase,

            "EntryPoint": pe.OPTIONAL_HEADER.AddressOfEntryPoint,

            "SizeOfCode": pe.OPTIONAL_HEADER.SizeOfCode,

            "SizeOfInitializedData": pe.OPTIONAL_HEADER.SizeOfInitializedData,

            "SizeOfUninitializedData": pe.OPTIONAL_HEADER.SizeOfUninitializedData,

            "NumberOfImports": len(pe.DIRECTORY_ENTRY_IMPORT) if hasattr(pe,
"DIRECTORY_ENTRY_IMPORT") else 0,
```

```
    }

    return features

except Exception as e:

    print(f"Error processing {file_path}: {e}")

    return None
```

### 3.2.3 Validate the Extracted Features:

- a. Check the dataset for consistency and correctness.
- b. Visualize data distributions to identify anomalies or patterns.

## 3.3 Issues and Solutions

### 3.3.1 Small Dataset:

- a. **Problem:** A limited number of samples can lead to poor model generalization.
- b. **Solution:**
  - i. Use data augmentation techniques, such as adding noise to features or using oversampling methods like SMOTE (Synthetic Minority Oversampling Technique).
  - ii. Leverage open-source datasets and collaborating with cybersecurity organizations to access more data.

### 3.3.2 Unknown Extensions:

- a. **Problem:** Some files might not have the .exe or .dll extensions but are still valid PE files.
- b. **Solution:** Use file signature checks to identify PE files based on their internal structure rather than extensions.

### 3.3.3 Imbalance Between Normal and Malicious Files:

- a. **Problem:** Having more normal files than malware can bias the model.
- b. **Solution:**

- i. Use under sampling for normal files or oversampling for malicious files to balance the dataset.
- ii. Apply advanced techniques like weight loss functions in machine learning models to account for the imbalance.

Building a dataset of PE files and extracting features is a foundational step in malware analysis. By ensuring ethical and safe practices when handling malware, using Python scripts to extract meaningful features, and addressing issues like data imbalance, researchers can create effective models for detecting malicious software.

## 4 Model Training and Testing

Training a machine learning model to detect malware involves selecting an appropriate supervised learning algorithm, training it on extracted features, and evaluating its performance. Let's explore these aspects in detail.

### 4.1 Supervised Learning Algorithm

In our Python script we used the **Random Forest Classifier**, which is a supervised learning algorithm from the Decision Trees family.

#### How Random Forest Works:

- Random Forest is an ensemble learning algorithm that creates multiple decision trees during training. (Niklas Donges, Nov 26, 2024)
- Each tree makes a prediction, and the majority vote determines the final classification.
- This method helps reduce overfitting and improves accuracy.

We have used **RandomForestClassifier(n\_estimators=100)**, which means it creates 100 decision trees and takes a vote to decide if a file is malware or normal.

#### Why Random Forest?

Random Forest is a method of group learning that uses several decision trees to produce predictions. It functions by dividing the data into decision trees, which are smaller sections that each concentrate on distinct patterns. Either the majority vote (classification) or average (regression), the outputs of all the trees are combined to provide the final forecast. Even if the dataset contains noise or unimportant features, this approach performs well and is resistant to overfitting.

## 4.2 Accuracy and F1-Score

**Accuracy:** Basically Accuracy measures the percentage of correctly classified instances out of all predictions. (Huigol, Aug 24, 2019)

It is calculated as

$$\text{Accuracy} = (\text{TruePositives} + \text{TrueNegatives}) / \text{TotalSamples}$$

A high accuracy (close to 1.0 or 100%) means the model is correctly detecting malware and normal files most of the time.

where:

- **TP (True Positives)** = Malware correctly classified as malware
- **TN (True Negatives)** = Normal files correctly classified as normal
- **FP (False Positives)** = Normal files incorrectly classified as malware
- **FN (False Negatives)** = Malware incorrectly classified as normal

**F1-Score:** F1-score is a balance between **precision** and **recall**:

- **Precision:** Out of all files predicted as malware, how many were actually malware?
- **Recall:** Out of all actual malware files, how many were correctly detected?

F1-score helps when the dataset is **imbalanced** (e.g., more normal files than malware). It ensures the model isn't just biased towards the majority class.

### Accuracy & F1-Score in our Script

our script evaluates the model using:

```
print(classification_report(y_test, y_pred))

print("Accuracy:", accuracy_score(y_test, y_pred))
```

After training, the model prints accuracy and an F1-score based on the test dataset. As accuracy is **too high (1.0 or 100%)**, it might indicate **overfitting**, so it means the model is memorizing rather than generalizing.

### 4.3 How the Model Classifies Files

Our script extracts features from PE (Portable Executable) files and uses them to classify the file.

#### Steps the Model Follows to Classify a File:

##### 4.3.1 Feature Extraction

In machine learning and data analysis, feature extraction is the process of finding and removing pertinent features from unprocessed data. A more informative dataset is then produced using these attributes, and it can be used for several purposes, including: (Domino, 2025)

- Classification
- Prediction
- Clustering
- Extracts information like NumberOfSections, ImageBase, SizeOfCode, etc., from the file.
- These are stored as numerical values in a DataFrame.

##### 4.3.2 Training the Model

- The dataset is split into **training (80%)** and **testing (20%)** using:  

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```
- The RandomForestClassifier is trained on these extracted features.

##### 4.3.3 Making Predictions

- When a new file is tested:  

```
prediction = model.predict(feature_df)
```
- The model uses patterns learned during training to decide if the file is **Malicious (1)** or **Normal (0)**.

#### 4.3.4 Output the Result

```
return "Malicious" if prediction[0] == 1 else "Normal"
```

- If the model predicts 1, it labels the file as **Malware**.
- If the model predicts 0, it labels the file as **Normal**.

## 5 Evidence:

## 5.1 Complete python script in our report

### 5.1.1 Step 1: Feature extraction from PE Files

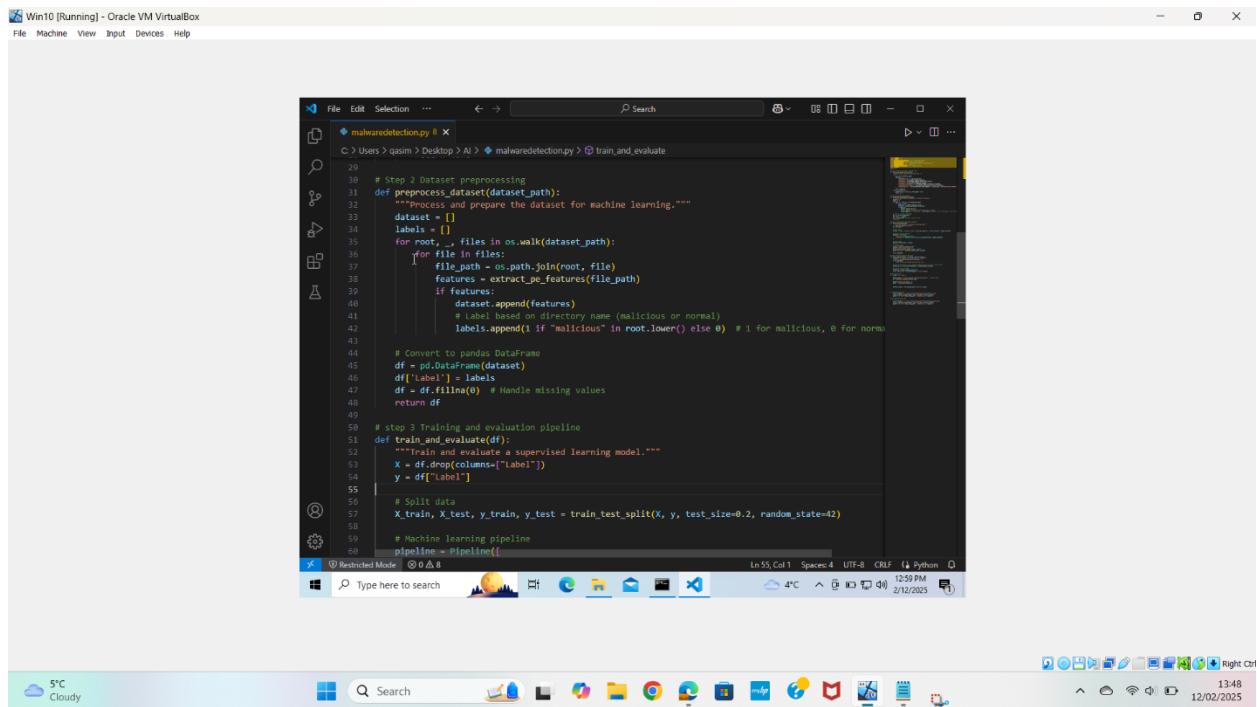
## Import Libraries and **Feature extraction from PE Files**

The screenshot shows a Windows desktop environment with a code editor window open. The title bar of the code editor reads "Win10 [Running] - Oracle VM VirtualBox". The menu bar includes File, Edit, Selection, View, Input, Devices, Help, and a search bar. The code editor displays a Python script named "malwaredetection.py". The script contains functions for extracting features from PE files and preprocessing datasets. The status bar at the bottom shows "In 12, Col 36" and "13:47 12/02/2025". The taskbar at the bottom has icons for File Explorer, Task View, Start, Search, and several pinned applications. The system tray shows battery level (5% C), signal strength, and a network icon.

```
C:\>Users>qasim>Desktop>AI> malwaredetection.py > extract_pe_features
  1 import pandas as pd
  2 import numpy as np
  3 from sklearn.model_selection import train_test_split
  4 from imblearn.over_sampling import RandomOverSampler
  5 from sklearn.metrics import classification_report, accuracy_score
  6 from sklearn.pipeline import Pipeline
  7 from sklearn.preprocessing import StandardScaler
  8 import pefile
  9 import os
 10
 11 # Step 1 Feature extraction from PE files
 12 def extract_pe_features(file_path):
 13     """Extract basic features from a PE file."""
 14     try:
 15         pe = pefile.PE(file_path)
 16         features = {
 17             "NumberSections": len(pe.sections),
 18             "ImageBase": pe.OPTIONAL_HEADER.ImageBase,
 19             "EntryPoint": pe.OPTIONAL_HEADER.AddressOfEntryPoint,
 20             "SizeOfCode": pe.OPTIONAL_HEADER.SizeOfCode,
 21             "SizeOfInitializedData": pe.OPTIONAL_HEADER.SizeOfInitializedData,
 22             "SizeOfUninitializedData": pe.OPTIONAL_HEADER.SizeOfUninitializedData,
 23             "NumberImports": len(pe.DIRECTORY_ENTRY_IMPORT) if hasattr(pe, "DIRECTORY_ENTRY_IMPORT") else 0
 24         }
 25         return features
 26     except Exception as e:
 27         print(f"Error processing {file_path}: {e}")
 28         return None
 29
 30 # Step 2 Dataset preprocessing
 31 def preprocess_dataset(dataset_path):
 32     """Process and prepare the dataset for machine learning."""
 33
```

*Figure 5.1.1: Feature Extraction from PE Files*

### 5.1.2 Step 2: Dataset Preprocessing:



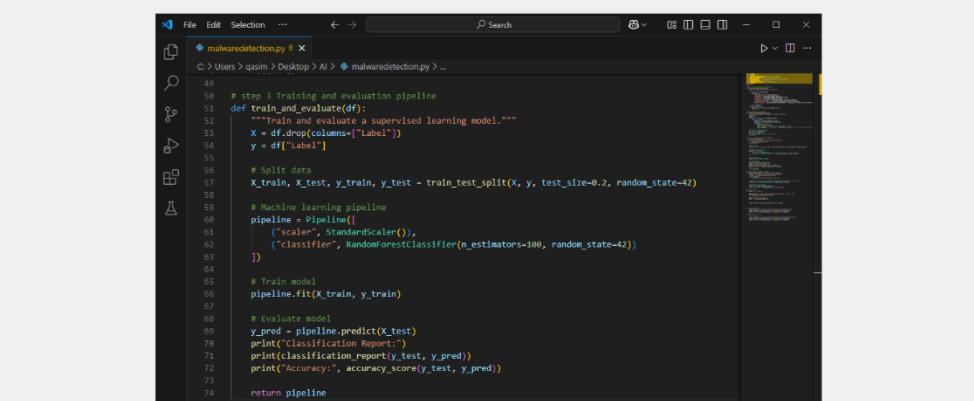
The screenshot shows a Windows 10 desktop environment. In the center, there is a code editor window titled "malwaredetection.py" with the following Python code:

```
29 # Step 2: Dataset preprocessing
30 def preprocess_dataset(dataset_path):
31     """Process and prepare the dataset for machine learning."""
32     dataset = []
33     labels = []
34     for root, _, files in os.walk(dataset_path):
35         for file in files:
36             file_path = os.path.join(root, file)
37             features = extract_pe_features(file_path)
38             if features:
39                 dataset.append(features)
40                 # Label based on directory name (malicious or normal)
41                 labels.append(1 if "malicious" in root.lower() else 0) # 1 for malicious, 0 for normal
42
43     # Convert to pandas DataFrame
44     df = pd.DataFrame(dataset)
45     df['Label'] = labels
46     df = df.fillna(0) # Handle missing values
47
48     return df
49
50 # Step 3: Training and evaluation pipeline
51 def train_and_evaluate(df):
52     """Train and evaluate a supervised learning model."""
53     X = df.drop(columns=['Label'])
54     y = df['Label']
55
56     # Split data
57     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
58
59     # Machine learning pipeline
60     pipeline = Pipeline([
61         ('Feature Selection', SelectKBest(k=10)),
62         ('SVM Model', SVC())
63     ])
64
65     pipeline.fit(X_train, y_train)
66     accuracy = pipeline.score(X_test, y_test)
67
68     print(f"Model accuracy: {accuracy * 100:.2f}%")
69
70     return pipeline
```

The code implements a dataset preprocessing function that walks through a specified directory, extracts features from PE files, and creates a labeled dataset. It then defines a training and evaluation pipeline using a Support Vector Machine (SVM) model.

Figure 5.1.2: Dataset Preprocessing

### 5.1.3 Step 3: Training and Evaluation Pipeline:



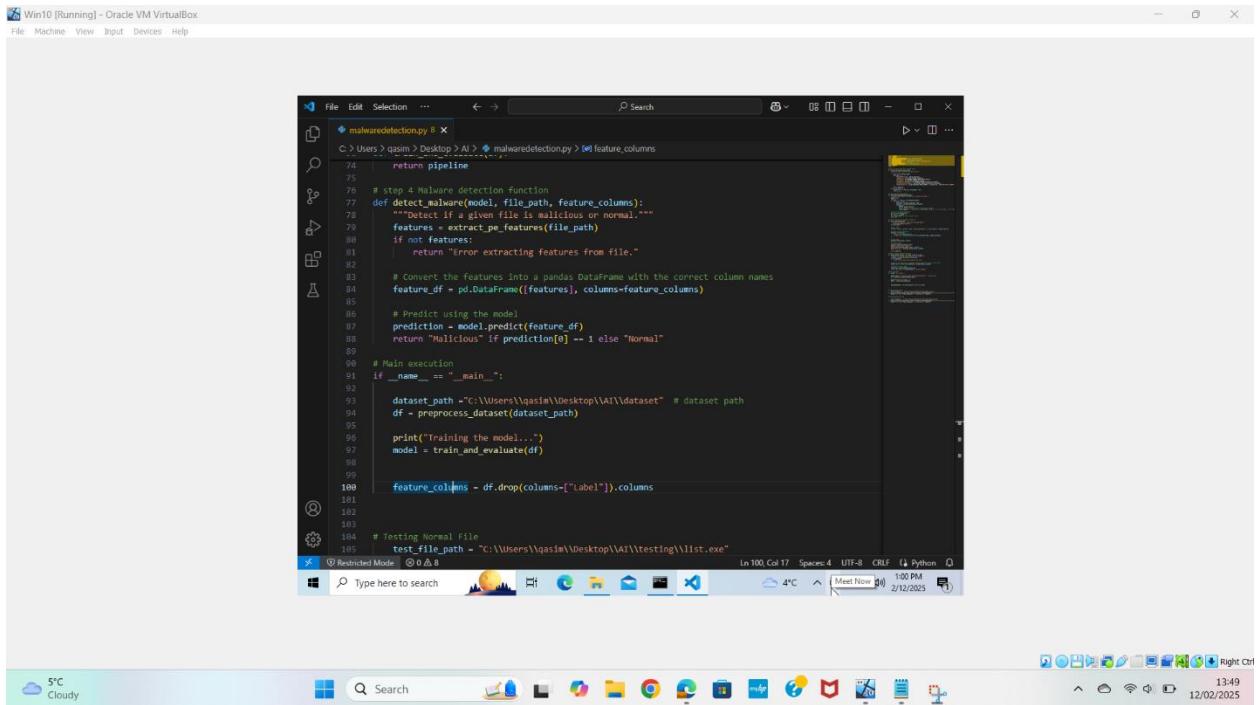
Win10 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
File Edit Selection ... ← → Search
C:\Users\qasim\Desktop\AI> malwaredetection.py ...
49
50 # step 3 Training and evaluation pipeline
51 def train_and_evaluate(df):
52     """Train and evaluate a supervised learning model."""
53     X = df.drop(columns=["Label"])
54     y = df["Label"]
55
56     # Split data
57     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
58
59     # Machine learning pipeline
60     pipeline = Pipeline([
61         ("scale", StandardScaler()),
62         ("classifier", RandomForestClassifier(n_estimators=100, random_state=42))
63     ])
64
65     # Train model
66     pipeline.fit(X_train, y_train)
67
68     # Evaluate model
69     y_pred = pipeline.predict(X_test)
70     print("Classification Report:")
71     print(classification_report(y_test, y_pred))
72     print("Accuracy: ", accuracy_score(y_test, y_pred))
73
74     return pipeline
75
76 # step 4 Malware detection function
77 def detect_malware(model, file_path, feature_columns):
78     """Detect if a given file is malicious or normal."""
79     features = extract_pe_features(file_path)
80     if not features:
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
527
527
528
529
529
530
531
532
533
534
535
535
536
537
537
538
539
539
540
541
542
543
544
544
545
546
546
547
547
548
549
549
550
551
552
553
553
554
555
555
556
556
557
557
558
559
559
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
164
```

*Figure 5.1.3: Training and Evaluation Pipeline*

#### 5.1.4 Step 4: Malware Detection Function:



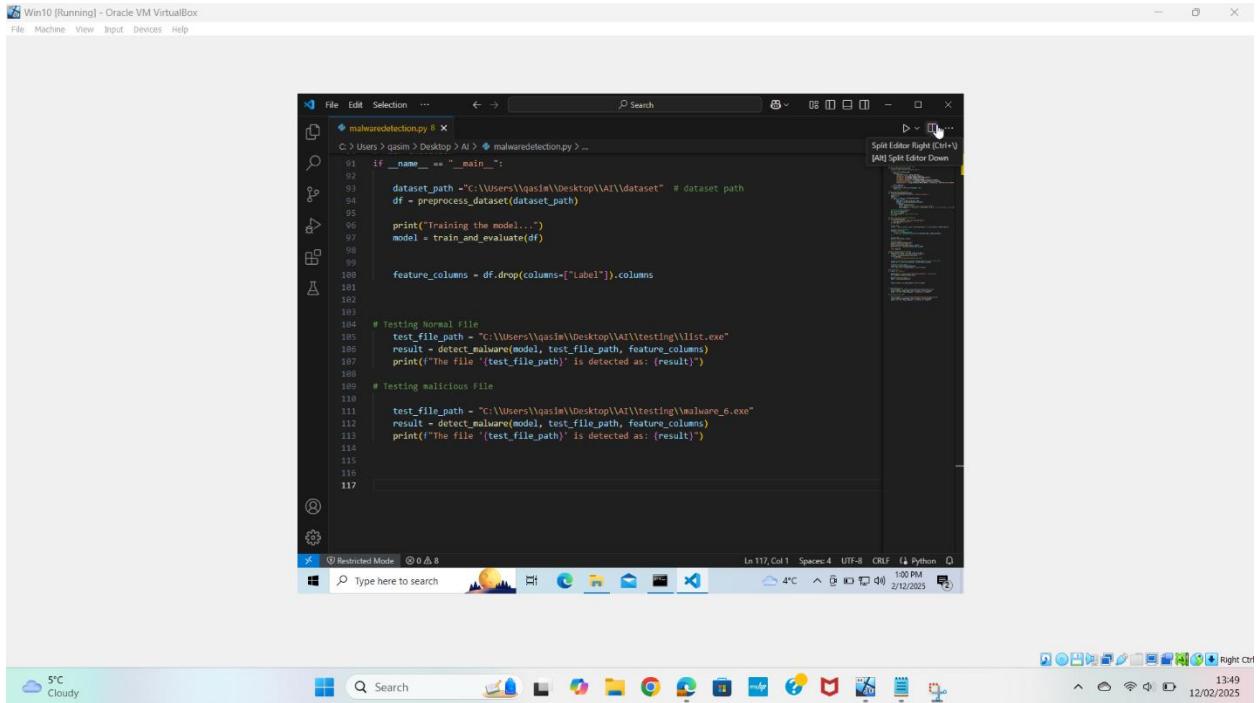
The screenshot shows a Windows 10 desktop environment. In the center, there is a code editor window titled "malwaredetection.py" with the following Python code:

```
74     return pipeline
75
76 # step 4 Malware detection function
77 def detect_malware(model, file_path, feature_columns):
78     """Detect if a given file is malicious or normal."""
79     features = extract_pe_features(file_path)
80     if not features:
81         return "Error extracting features from file."
82
83     # Convert the features into a pandas DataFrame with the correct column names
84     feature_df = pd.DataFrame([features], columns=feature_columns)
85
86     # Predict using the model
87     prediction = model.predict(feature_df)
88     return "Malicious" if prediction[0] == 1 else "Normal"
89
90 # Main execution
91 if __name__ == "__main__":
92
93     dataset_path = "C:\\Users\\qasim\\Desktop\\AI\\dataset" # dataset path
94     df = preprocess_dataset(dataset_path)
95
96     print("Training the model...")
97     model = train_and_evaluate(df)
98
99
100    feature_columns = df.drop(columns=["label"]).columns
101
102
103    # Testing Normal File
104    test_file_path = "C:\\Users\\qasim\\Desktop\\AI\\testing\\list.exe"
105
```

Below the code editor is a terminal window titled "cmd" with the command "python malwaredetection.py" entered and running. The terminal output shows the program's execution and the prediction for the test file.

Figure 5.1.4: Malware Detection Function

### 5.1.5 Step 5: Testing Normal and Malicious file:



The screenshot shows a Windows 10 desktop environment. In the center, there is a code editor window titled "malwaredetection.py" with the following Python code:

```
C:\Users\qasim\Desktop\AI> python malwaredetection.py
91 if __name__ == "__main__":
92     dataset_path = "C:\\Users\\qasim\\Desktop\\AI\\dataset" # dataset path
93     df = preprocess_dataset(dataset_path)
94     print("Training the model...")
95     model = train_and_evaluate(df)
96
97     feature_columns = df.drop(columns=["Label"]).columns
98
99
100    # Testing Normal file
101    test_file_path = "C:\\Users\\qasim\\Desktop\\AI\\testing\\list.exe"
102    result = detect_malware(model, test_file_path, feature_columns)
103    print(f"The file '{test_file_path}' is detected as: {result}")
104
105    # Testing malicious file
106    test_file_path = "C:\\Users\\qasim\\Desktop\\AI\\testing\\malware_6.exe"
107    result = detect_malware(model, test_file_path, feature_columns)
108    print(f"The file '{test_file_path}' is detected as: {result}")
109
110
111
112
113
114
115
116
117
```

The code editor has a status bar at the bottom showing "Ln 117, Col 1" and "Python". Below the code editor is the Windows taskbar with various pinned icons like File Explorer, Edge, and Task View. The system tray shows the date and time as "12/02/2023 13:49".

Figure 5.1.5: Testing Normal and Malicious file

## 5.2 Malware Samples:

### 5.2.1 Malware Sample Files

We have collected 5 samples from the malware bazaar:

- Malware\_1.exe
- Malware\_2.exe
- Malware\_3.exe
- Malware\_4.exe
- Malware\_5.exe

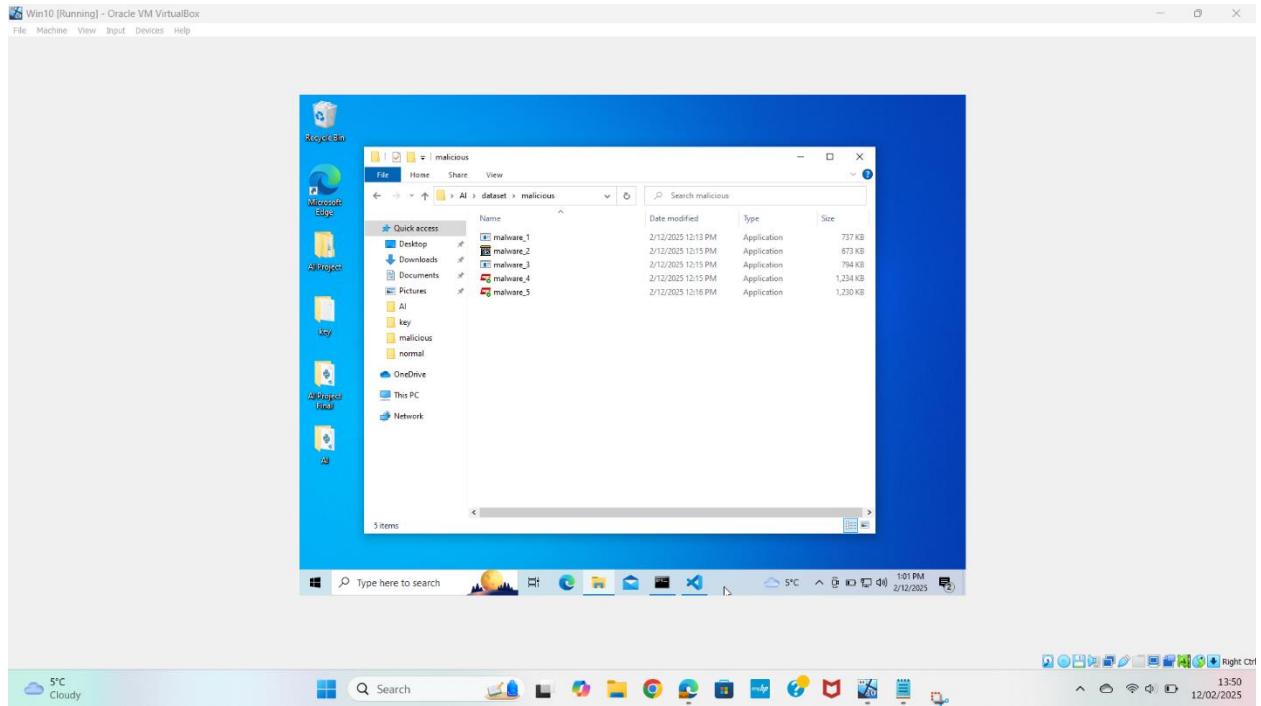


Figure 5.2.1: Malware Samples

## 5.3 Normal Samples:

### 5.3.1 Normal Sample Files

We have created 5 different programs in python language as normal files.

- add.exe
- division.exe
- input.exe
- multiply.exe
- sub.exe

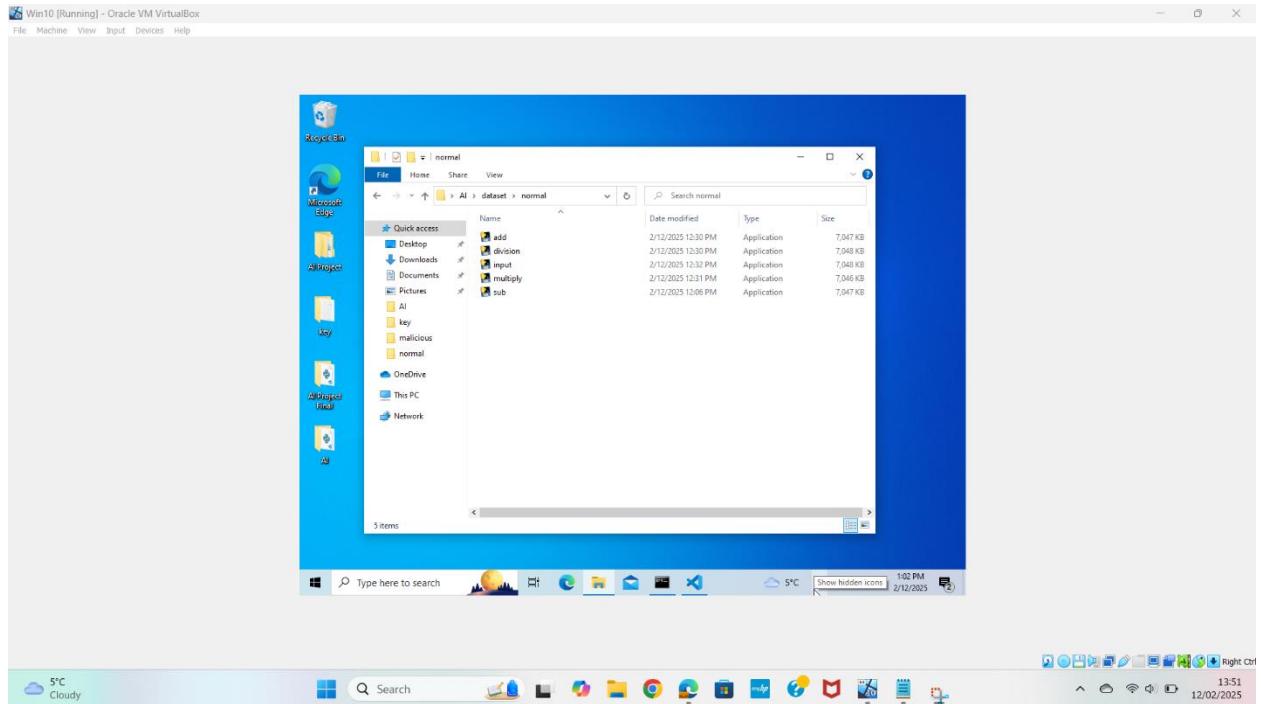
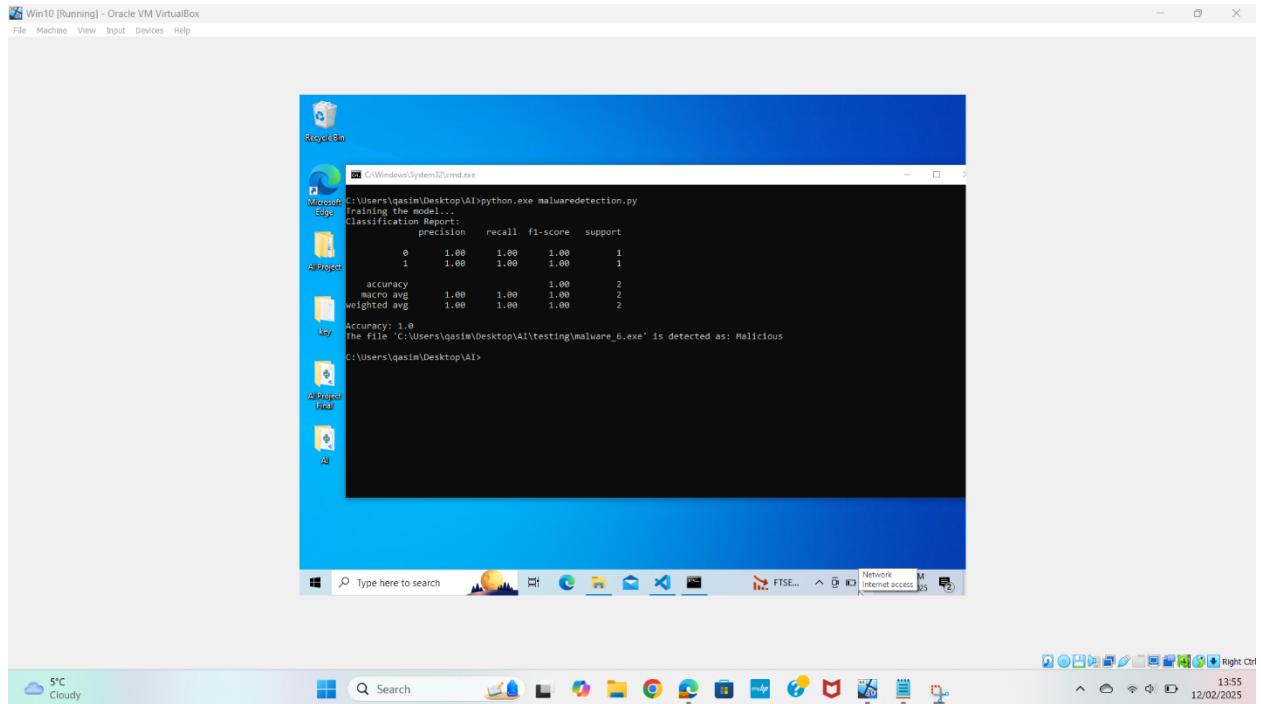


Figure 5.3.1: Normal Samples

## 5.4 Malicious file as a malware

### 5.4.1 Output:

We have provided a malicious file (malware\_6.exe) to our trained model for testing purpose, Here is output.



*Figure 5.4.1: Malware File Output*

## 5.5 Normal file as a Normal

### 5.5.1 Output:

We have provided a normal file (list.exe) to our trained model for testing purpose, Here is output.

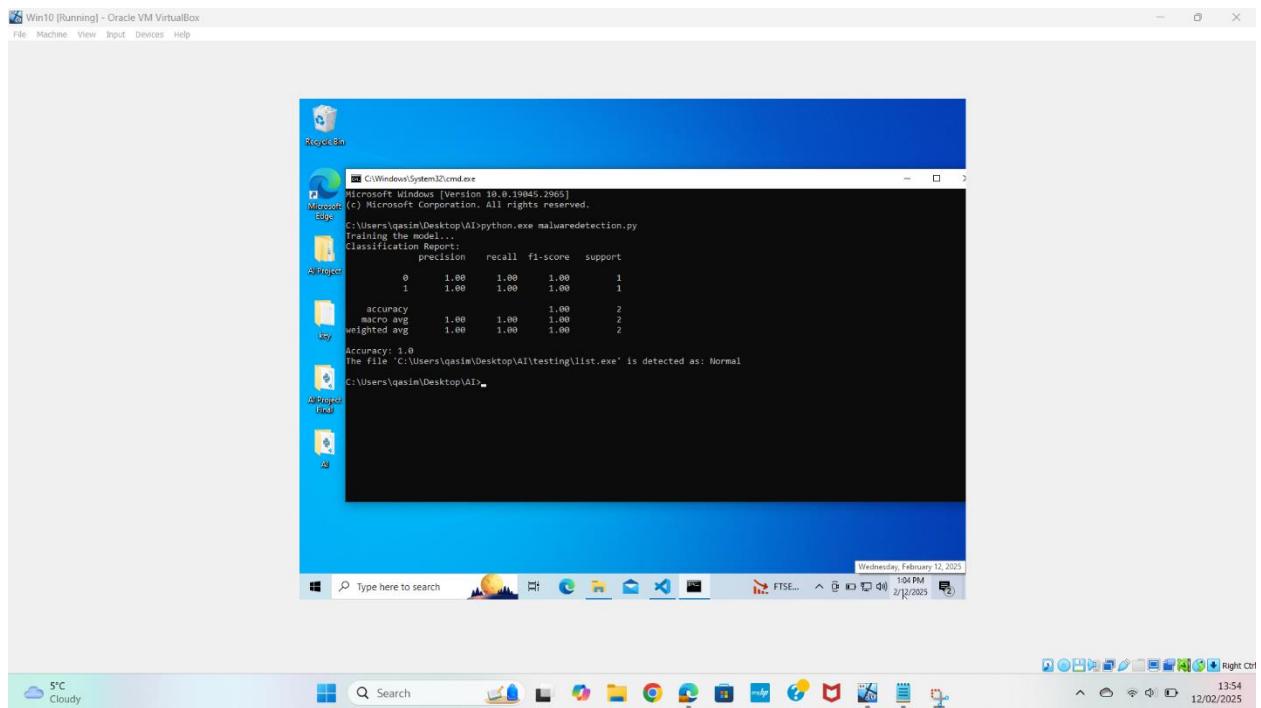


Figure 5.5.1: Normal File Output

## 6 Model as a Cybersecurity Tool

Machine learning models, like the Random Forest-based malware detection system, can be an integral component of larger cybersecurity tools. By automating the analysis and classification of potential threats, such a model enhances the efficiency, accuracy, and adaptability of cybersecurity measures.

### 6.1 Integration into a Cybersecurity System

Here's how the model could fit into a comprehensive cybersecurity tool:

#### 6.1.1 Malware Detection Module:

- a. The model serves as the core engine for identifying malicious files.
- b. It processes incoming files in real-time, extracting features like entropy, imports, and section sizes, then classifies them as either malware or normal.

#### 6.1.2 Preprocessing and Feature Extraction:

- a. A preprocessing module prepares the input data for the model by extracting relevant features from PE files using tools like pefile.
- b. This ensures consistent and accurate feature representation, critical for the model's performance.

#### 6.1.3 Threat Mitigation Pipeline:

- a. When a file is flagged as malware:
  - i. The system quarantines files to prevent execution.
  - ii. It alerts the cybersecurity team or triggers automated responses, such as blocking the file on all connected systems.
- b. For normal files, the system allows safe execution, minimizing disruptions to users.

#### 6.1.4 Behavioral Monitoring:

- a. The model could work alongside behavioral analysis tools that monitor runtime actions of executables.

- b. If the model misses a threat initially, unusual behavior during execution can trigger additional scans.

#### 6.1.5 Integration with Threat Intelligence:

- a. The model can leverage external threat intelligence feeds to update its feature set, improving detection of new malware variants.
- b. For example, if a new malware family is detected globally, its features can be incorporated into the model's training pipeline.

### 6.2 Advantages in a Larger Cybersecurity Ecosystem

#### 6.2.1 Real-Time Threat Detection:

- a. The model can analyze files as they are downloaded, opened, or transferred, providing immediate feedback on potential risks.

#### 6.2.2 Scalability:

- a. The model can handle large volumes of data, making it suitable for enterprise-level deployments where thousands of files are analyzed daily.

#### 6.2.3 Automation:

- a. Automating malware detection reduces the workload for cybersecurity teams, allowing them to focus on more complex threats.

#### 6.2.4 Adaptability:

- a. As new malware emerges, retraining or fine-tuning the model ensures it remains effective.

### 6.3 Enhancements and Features for Practical Deployment

#### 6.3.1 User-Friendly Dashboard:

- a. A graphical interface displays detection statistics, flagged files, and threat levels in real-time.

### **6.3.2 Integration with Endpoint Protection:**

- a. The model can be embedded into endpoint protection tools like antivirus software or host intrusion detection systems (HIDS). (Redscan, 2025)

### **6.3.3 Regular Updates:**

- a. A mechanism to regularly update the model with new malware data ensures it remains effective against evolving threats.

### **6.3.4 Cross-Platform Support:**

- a. The model can be extended to analyze files on different platforms, such as macOS or Linux, for broader protection.

### **6.3.5 Incident Reporting:**

- a. When malware is detected, the system generates detailed reports, including feature analysis and reasons for classification, aiding in forensic investigations.

## **6.4 Example Workflow in a Cybersecurity Tool**

### **6.4.1 File Upload or Detection:**

- a. A user downloads or opens a file.
- b. The file is intercepted by the cybersecurity tool and sent to the model for analysis.

### **6.4.2 Feature Extraction:**

- a. Features like section count, entropy, imports, and exports are extracted and passed to the model.

### **6.4.3 Classification:**

- a. The model predicts whether the file is malware or normal.

### **6.4.4 Response:**

- a. If malware: The file is quarantined, and a system-wide alert is issued.
- b. If normal: The file is allowed to be executed.

#### **6.4.5 Post-Incident Learning:**

- a. Feedback from the cybersecurity team or additional behavioral analysis helps refine the model.

Integrating the malware detection model into a larger cybersecurity tool creates a robust, automated defense mechanism capable of adapting to new threats. By combining real-time analysis, automation, and continuous learning, this model can significantly enhance the overall security posture of individuals and organizations.

## 7 Conclusion

### 7.1 Summary of Key Points

In this report, we explored how Artificial Intelligence (AI) and machine learning can enhance cybersecurity, specifically in detecting malware. Below are the key takeaways:

1. **AI in Cybersecurity:** AI helps in identifying malware by analyzing large datasets, recognizing patterns, and detecting anomalies that traditional methods might miss. It is particularly effective in identifying zero-day threats and polymorphic malware.
2. **Understanding PE File Structure:** The Portable Executable (PE) file structure provides critical information about executable files. Features such as section count, section sizes, imports, and entropy offer valuable insights for malware detection.
3. **Dataset Preparation:** Safe and ethical practices are essential when collecting malware samples and legitimate PE files. Extracting meaningful features using Python scripts ensures that the dataset is suitable for machine learning applications.
4. **Model Training and Testing:** The Random Forest algorithm was chosen for its ability to handle complex datasets and prevent overfitting. Metrics like accuracy and F1-score were used to evaluate the model's performance, ensuring reliable and balanced predictions.
5. **Model Integration:** The model can be integrated into a larger cybersecurity tool, providing real-time malware detection, scalability, automation, and adaptability to evolving threats.

### 7.2 Reflection and Real-World Applications

This project provided valuable insights into the intersection of AI and cybersecurity, emphasizing the role of machine learning in automating and enhancing threat detection. The process of collecting data, extracting features, and training a model highlighted the importance of meticulous preparation and ethical considerations in cybersecurity research.

Key lessons learned include:

1. **Feature Importance:** Understanding the significance of PE file attributes helped in designing an effective detection model. This knowledge can be applied to analyze other file formats and extend protection to different platforms.
2. **Algorithm Selection:** Choosing the right machine learning algorithm and evaluation metrics ensures robust and reliable solutions. This approach can be generalized to solve other cybersecurity problems, such as phishing detection or network intrusion prevention.
3. **Handling Imbalanced Data:** Techniques to address class imbalance, such as oversampling and class weighting, are crucial in real-world datasets where malicious files are relatively rare.
4. **Automation and Scalability:** The integration of machine learning models into larger tools demonstrates how AI can streamline cybersecurity operations, reducing the manual workload for analysts while improving detection rates.

This knowledge has practical applications in real-world scenarios, such as:

- **Enterprise Security:** Organizations can deploy similar models to scan incoming files, emails, or network traffic for potential threats.
- **Incident Response:** Security teams can use the model's outputs to prioritize and respond to high-risk incidents.
- **Threat Intelligence:** Continuous updates to the model using global threat intelligence can ensure resilience against emerging malware.

This report underscores the transformative potential of AI in cybersecurity. By leveraging machine learning models, we can build proactive and adaptive systems that keep pace with the ever-evolving threat landscape. The knowledge gained from this project can contribute to designing innovative solutions that protect individuals and organizations from sophisticated cyberattacks.

## 8 References

- Breiman, L. (2001) ‘Random forests’, *Machine Learning*, 45(1), pp. 5–32. Available at: <https://link.springer.com/article/10.1023/A:1010933404324> (Accessed: 12 February 2025).
- Domino, 2025. *Feature Extraction*. [Online]  
Available at: <https://domino.ai/data-science-dictionary/feature-extraction> [Accessed 11 02 2025].
- Huilgol, P., Aug 24, 2019. *Accuracy vs. F1-Score*. [Online]  
Available at: <https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2> [Accessed 09 02 2025].
- Iryna Sydorenko, December 21, 2023. *What Is a Dataset in Machine Learning*:. [Online]  
Available at: <https://labelyourdata.com/articles/what-is-dataset-in-machine-learning> [Accessed 11 02 2025].
- Microsoft, 02/29/2024. *PE Format*. [Online]  
Available at: <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format> [Accessed 08 02 2025].
- Niklas Donges, Nov 26, 2024. *Random Forest: A Complete Guide for Machine Learning*. [Online]  
Available at: <https://builtin.com/data-science/random-forest-algorithm> [Accessed 09 02 2025].
- Priyanka Jadav, April 27, 2024. *Malware Detection Using Machine Learning Techniques*. [Online]  
Available at: <https://www.einfochips.com/blog/malware-detection-using-machine-learning-techniques/> [Accessed 11 02 2025].
- Redscan, 2025. *HIDS - Host-based intrusion detection*. [Online]  
Available at: <https://www.redscan.com/services/hids/> [Accessed 12 02 2025].