Project Documentation

An Efficient Feature Extraction Method for Static Malware Analysis Using PE Header Files: A Comparative Study of Feature Set Approaches

Authors

Fabiha Jalal
Sadia Tasnim Dhruba
Onamika Hossain

Supervisor

Dr. Md Moniruzzaman
Assistant Professor

Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Gazipur, Bangladesh

February 2026

This document provides comprehensive technical documentation of the project, including methodology, results, code organization, and all consistency fixes applied.

Contents

1 Project Overview

1.1 Purpose

This project develops an efficient feature extraction methodology for static malware analysis using Portable Executable (PE) header files. The research compares two feature set approaches—a reduced set (15 features) and an extended set (32 features)—across six machine learning classifiers to determine the optimal trade-off between feature dimensionality, extraction time, and classification performance.

1.2 Research Goals

1. Develop an efficient feature extraction methodology using PE header files that minimizes extraction time while maintaining classification accuracy.
2. Conduct a comparative analysis between reduced (15 features) and extended (32 features) feature sets.
3. Evaluate multiple machine learning classifiers for malware detection.
4. Create and publish a novel malware dataset for research purposes.
5. Demonstrate the trade-offs between feature dimensionality, extraction time, and detection performance.

1.3 Key Contributions

- A novel and efficient feature extraction technique that reduces extraction time from approximately 19 minutes to under 5 minutes—a **75% reduction**.
- A publicly available dataset of 1,150 malware executable files with extracted PE header features.
- Comprehensive comparative analysis of two feature set approaches across six machine learning classifiers.
- Empirical evidence demonstrating the relationship between feature quantity and classification performance.

2 Dataset

2.1 Data Source

Malware executable samples were collected from **MalwareBazaar** (`https://malwarebazaar.abuse.ch`), a reputable malware sample repository maintained by abuse.ch. The dataset comprises Windows PE (Portable Executable) files representing various malware families.

2.2 Dataset Statistics

Table 1: Dataset Overview

Property	**Value**
Total Samples	1,150 executable files (.exe)
File Format	Windows PE (Portable Executable)
Source	MalwareBazaar API
Label/Target	Characteristics field (multi-class)
Train/Test Split	80/20 (920 training, 230 testing)
Random State	42 (for reproducibility)

2.3 Data Collection Process

The `malware_downloader.py` script automates sample collection using the MalwareBazaar API. It searches across 20 malware signatures (including Emotet, Dridex, TrickBot, AgentTesla, and others) and 11 file types (exe, dll, doc, xls, js, vbs, ps1, jar, apk, elf, msi). The script includes automatic deduplication, rate limiting, and AES-based ZIP extraction for downloaded samples.

2.4 CSV Output Files

Table 2: Generated CSV Files

File	**Rows**	**Columns**	**Size**
`output_file_final.csv`	1,150	16	151 KB
`output_file_more_features.csv`	1,150	34	239 KB
`extracted_features.csv`	1,150	16	151 KB

3 Feature Sets

3.1 Reduced Feature Set (15 Features)

The reduced feature set extracts 15 attributes from PE headers: 3 from FILE_HEADER and 12 from OPTIONAL_HEADER. The `Characteristics` field serves as the classification target variable.

3.1.1 FILE_HEADER Features (3)

Table 3: Reduced Set – FILE_HEADER Features

Feature	**Description**
`Machine`	CPU architecture (Intel x86, x86-64, etc.)
`NumberOfSections`	Number of PE sections in the file
`Characteristics`	File flags (executable, DLL, etc.) — **Target Variable**

3.1.2 OPTIONAL_HEADER Features (12)

Table 4: Reduced Set – OPTIONAL_HEADER Features

Feature	**Description**
`AddressOfEntryPoint`	Program entry point RVA
`ImageBase`	Preferred base load address
`SectionAlignment`	Section boundary alignment in memory
`FileAlignment`	Raw data section alignment in file
`Subsystem`	Operating system subsystem (GUI, console, etc.)
`DllCharacteristics`	DLL behavior flags
`MajorOperatingSystemVersion`	Required Windows major version
`MinorOperatingSystemVersion`	Required Windows minor version
`MajorImageVersion`	Image major version number
`MinorImageVersion`	Image minor version number
`SizeOfImage`	Memory footprint when loaded
`SizeOfHeaders`	Combined size of all headers

3.2 Extended Feature Set (32 Features)

The extended feature set extracts 32 attributes (plus Characteristics as target): 7 from FILE_HEADER and 26 from OPTIONAL_HEADER.

3.2.1 FILE_HEADER Features (7)

All 3 features from the reduced set, plus:

Table 5: Extended Set – Additional FILE_HEADER Features

Feature	**Description**
`TimeDateStamp`	Compilation timestamp
`PointerToSymbolTable`	Symbol table pointer (usually 0)
`NumberOfSymbols`	Symbol table entry count
`SizeOfOptionalHeader`	Optional header size in bytes

3.2.2 OPTIONAL_HEADER Features (26)

All 12 features from the reduced set, plus:

Table 6: Extended Set – Additional OPTIONAL_HEADER Features

Feature	Description
`MajorLinkerVersion`	Linker major version
`MinorLinkerVersion`	Linker minor version
`SizeOfCode`	Total code section size
`SizeOfInitializedData`	Initialized data size
`SizeOfUninitializedData`	Uninitialized data size (BSS)
`BaseOfCode`	Code section base address
`MajorSubsystemVersion`	Subsystem major version
`MinorSubsystemVersion`	Subsystem minor version
`CheckSum`	File checksum (for drivers)
`SizeOfStackReserve`	Initial stack reserve
`SizeOfStackCommit`	Stack commit per thread
`SizeOfHeapReserve`	Initial heap reserve
`SizeOfHeapCommit`	Heap commit size
`LoaderFlags`	Loader behavior flags
`NumberOfRvaAndSizes`	Data directory entries count

Note: The OPTIONAL_HEADER in the extended set contains 26 features total (12 from the reduced set plus the 15 additional features listed above, noting that `SizeOfImage` from the reduced set is already included).

4 Methodology

4.1 Feature Extraction Pipeline

The feature extraction pipeline operates in three stages:

1. **File Discovery:** Scan the `extracted/` directory for PE executable files.
2. **PE Header Parsing:** Use the Python `pefile` library to parse each file's FILE_HEADER and OPTIONAL_HEADER, extracting the designated feature attributes.
3. **CSV Export:** Write all extracted features (one row per sample) to a CSV file for subsequent ML training.

4.2 Machine Learning Classifiers

Six machine learning classifiers from scikit-learn were employed:

Table 7: Classifier Configuration

Classifier	Key Parameters	Notes
Random Forest	`n_estimators=100`, `n_jobs=-1`, `random_state=42`	Ensemble of 100 decis bootstrap aggregation
K-Nearest Neighbors	`n_neighbors=5`	Instance-based learni clidean distance
Decision Tree	`random_state=42`	Tree-based classificati mation gain
Support Vector Machine	`kernel='rbf'`, `gamma='scale'`, `C=1.0`, `random_state=42`	Kernel-based classif RBF kernel
Logistic Regression	`max_iter=1000`, `random_state=42`	Linear classification w likelihood estimation
Gradient Boosting	`n_estimators=50`, `random_state=42`	Sequential ensemble m

4.3 Experimental Setup

- **Train/Test Split:** 80% training (920 samples), 20% testing (230 samples)
- **Cross-Validation:** 3-fold cross-validation (consistent across both feature sets)
- **Random State:** 42 for all operations (reproducibility)
- **Feature Scaling:** None (raw PE header values used directly)
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-Score (weighted average, `zero_division=0`)

5 Python Scripts

5.1 Script Overview

The project contains six Python scripts, each serving a distinct role in the pipeline:

Table 8: Python Scripts Summary

Script	Lines	Purpose
`generate_charts.py`	523	Full pipeline for reduced feature set (15 features): extraction, training, and chart generation
`generate_charts_more_features.py`	367	Full pipeline for extended feature set (32 features) with optimizations
`thesis_with_charts.py`	529	Modular implementation of the reduced feature set pipeline with separate functions
`make_charts.py`	213	Chart-only generation from pre-existing CSV (no extraction step)
`rerun_all.py`	—	Re-runs ML training for both feature sets from existing CSVs with consistent settings
`malware_downloader.py`	166	Downloads malware samples from MalwareBazaar API

5.2 Detailed Script Descriptions

5.2.1 generate_charts.py (Primary – Reduced Features)

- **Input:** PE executable files in extracted/ directory
- **Output:** output_file_final.csv (1,150 × 16) and 5 PNG charts
- **Workflow:** Extracts 15 PE header features from all .exe files, trains 6 classifiers on 80/20 split, generates 5 visualization charts
- **Performance:** ∼215.81 seconds (∼5.33 files/second)

5.2.2 generate_charts_more_features.py (Primary – Extended Features)

- **Input:** PE executable files in extracted/ directory
- **Output:** output_file_more_features.csv (1,150 × 34) and 5 PNG charts (with _more_features suffix)
- **Workflow:** Extracts 32 PE header features (7 FILE_HEADER + 26 OPTIONAL_HEADER), trains 6 classifiers, generates 5 charts
- **Optimizations:** Parallel Random Forest (n_jobs=-1), skipped SVM cross-validation, non-interactive matplotlib backend (Agg)
- **Performance:** ∼290.12 seconds (∼3.96 files/second)

5.2.3 thesis_with_charts.py (Modular Alternative)

- **Purpose:** Modular, well-documented implementation of the 15-feature pipeline
- **Key Functions:** extract_features_pefile(), train_classifiers(), plot_*(), print_results_summ
- **Advantage:** Better code organization with dedicated functions and full docstrings

5.2.4 make_charts.py (Quick Regeneration)

- **Purpose:** Generates charts from pre-computed CSV without feature extraction
- **Use Case:** Rapid chart regeneration when CSV data already exists
- **Advantage:** Fastest execution time (no extraction overhead)

5.2.5 rerun_all.py (Consistency Runner)

- **Purpose:** Master script to re-run ML training for both feature sets with guaranteed consistent classifier settings
- **Output:** Regenerates all 10 PNG charts (5 per feature set)

5.2.6 malware_downloader.py (Data Collection)

- **Purpose:** Automated malware sample download from MalwareBazaar
- **Features:** Search by signature (20 families), by tag (11 file types), AES ZIP extraction, deduplication, rate limiting

5.3 Recommended Usage

Table 9: Script Selection Guide

Task	**Recommended Script**
First-time full pipeline (15 features)	`generate_charts.py`
Full pipeline with modular code (15 features)	`thesis_with_charts.py`
Extended feature analysis (32 features)	`generate_charts_more_features.py`
Quick chart regeneration (CSV exists)	`make_charts.py`
Regenerate both feature sets consistently	`rerun_all.py`
Download new malware samples	`malware_downloader.py`

6 Results

6.1 Feature Extraction Performance

Table 10: Feature Extraction Performance Comparison

Metric	**Reduced (15)**	**Extended (32)**
Extraction Time	215.81 sec	290.12 sec
Processing Speed	5.33 files/sec	3.96 files/sec
Files Processed	1,150	1,150
Errors	0	0
CSV Columns	16	34

Comparison with Prior Work (APT1 Dataset):

- APT1 PE Header Extraction: 1,157.81 seconds (∼19 minutes)
- Our Extended Feature Set: 290.12 seconds (∼4.8 minutes)
- **Improvement: 75% reduction in extraction time**

6.2 Classification Results – Reduced Feature Set (15 Features)

Table 11: Classification Results – Reduced Feature Set (15 Features)

Classifier	**Accuracy**	**F1 Score**	**Precision**	**Recall**
Random Forest	**93.91%**	**0.8379**	**0.8858**	**0.8264**
Decision Tree	91.74%	0.7329	0.7681	0.7302
Gradient Boosting	90.87%	0.5956	0.6037	0.6028
KNN	75.22%	0.4636	0.4969	0.4590
Logistic Regression	51.30%	0.1369	0.1309	0.1734
SVM	25.65%	0.0240	0.0151	0.0588

6.3 Classification Results – Extended Feature Set (32 Features)

Table 12: Classification Results – Extended Feature Set (32 Features)

Classifier	Accuracy	F1 Score	Precision	Recall
Random Forest	**94.78%**	**0.8632**	**0.9051**	**0.8399**
Decision Tree	93.48%	0.7993	0.8167	0.7927
Gradient Boosting	93.04%	0.7209	0.7616	0.7133
KNN	80.00%	0.5085	0.5300	0.5053
Logistic Regression	41.30%	0.1098	0.1076	0.1368
SVM	25.65%	0.0240	0.0151	0.0588

6.4 Accuracy Comparison Between Feature Sets

Table 13: Accuracy Improvement: Extended vs. Reduced Feature Set

Classifier	Reduced (15)	Extended (32)	Improvement
Random Forest	93.91%	94.78%	+0.87%
Decision Tree	91.74%	93.48%	+1.74%
Gradient Boosting	90.87%	93.04%	+2.17%
KNN	75.22%	80.00%	+4.78%
Logistic Regression	51.30%	41.30%	−10.00%
SVM	25.65%	25.65%	0.00%

6.5 Feature Importance Analysis

Table 14 presents the top 10 most important features for malware classification, as determined by the Random Forest classifier on the extended feature set.

Table 14: Top 10 Feature Importance (Random Forest, Extended Set)

Rank	Feature	Importance
1	AddressOfEntryPoint	0.095
2	MajorLinkerVersion	0.092
3	TimeDateStamp	0.084
4	SizeOfCode	0.068
5	DllCharacteristics	0.065
6	ImageBase	0.057
7	SizeOfInitializedData	0.057
8	SizeOfOptionalHeader	0.050
9	Machine	0.050
10	NumberOfSections	0.039

7 Visualizations

Ten charts are generated—five for each feature set. Each chart set follows a standardized format for direct comparison.

7.1 Chart Descriptions

Table 15: Generated Visualization Charts

#	Chart Type	Description
1	Classifier Performance Comparison	4-panel figure: (A) bar chart of all metrics, (B) 3-fold CV scores with error bars, (C) F1 score horizontal bars, (D) summary statistics table
2	Confusion Matrices	6 heatmap subplots (one per classifier) showing TP, FP, TN, FN with accuracy annotations
3	Feature Importance	Horizontal bar chart from Random Forest, sorted by importance with color gradient (red–yellow–green)
4	Accuracy Comparison	Sorted bar chart ranking classifiers by accuracy with value labels
5	Metrics Radar Chart	Polar/spider chart overlaying all 6 classifiers across 4 metrics (accuracy, precision, recall, F1)

7.2 Chart File Listing

7.2.1 Reduced Feature Set (15 Features)

1. `classifier_comparison.png` (157 KB)
2. `confusion_matrices.png` (270 KB)
3. `feature_importance.png` (72 KB)
4. `accuracy_comparison.png` (64 KB)
5. `metrics_radar.png` (276 KB)

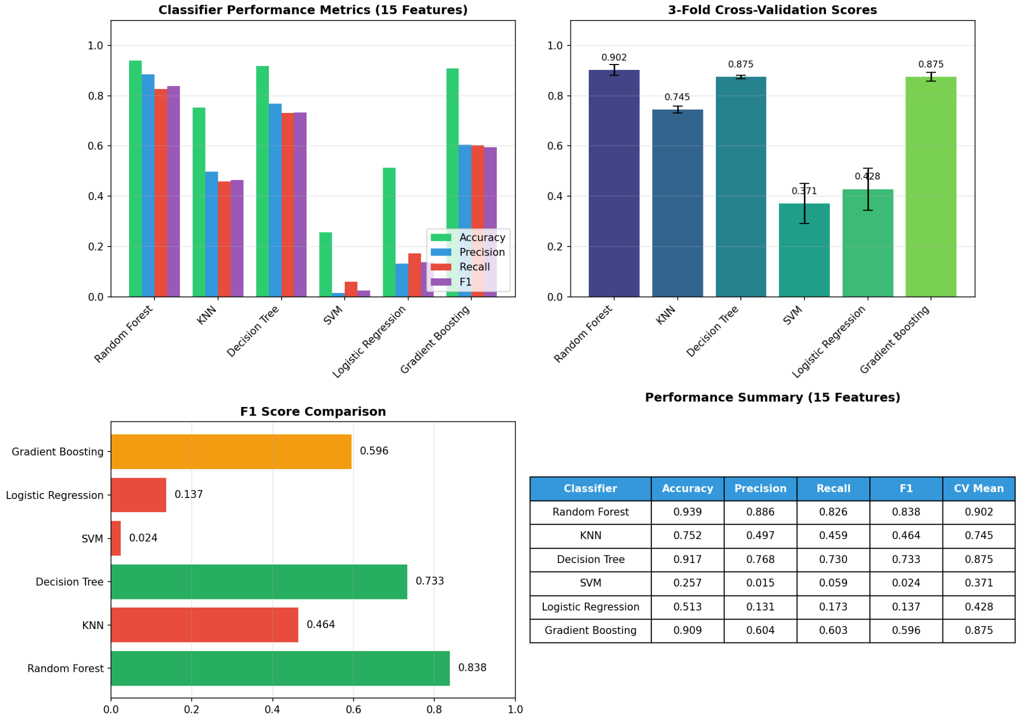
7.2.2 Extended Feature Set (32 Features)

1. `classifier_comparison_more_features.png` (157 KB)
2. `confusion_matrices_more_features.png` (260 KB)
3. `feature_importance_more_features.png` (136 KB)
4. `accuracy_comparison_more_features.png` (65 KB)
5. `metrics_radar_more_features.png` (292 KB)

7.3 Sample Visualizations

The following pages present the generated charts for both feature sets.

7.3.1 Reduced Feature Set Charts



Performance Summary (15 Features)

Classifier	Accuracy	Precision	Recall	F1	CV Mean
Random Forest	0.939	0.886	0.826	0.838	0.902
KNN	0.752	0.497	0.459	0.464	0.745
Decision Tree	0.917	0.768	0.730	0.733	0.875
SVM	0.257	0.015	0.059	0.024	0.371
Logistic Regression	0.513	0.131	0.173	0.137	0.428
Gradient Boosting	0.909	0.604	0.603	0.596	0.875

Figure 1: Classifier Performance Comparison – Reduced Feature Set (15 Features)

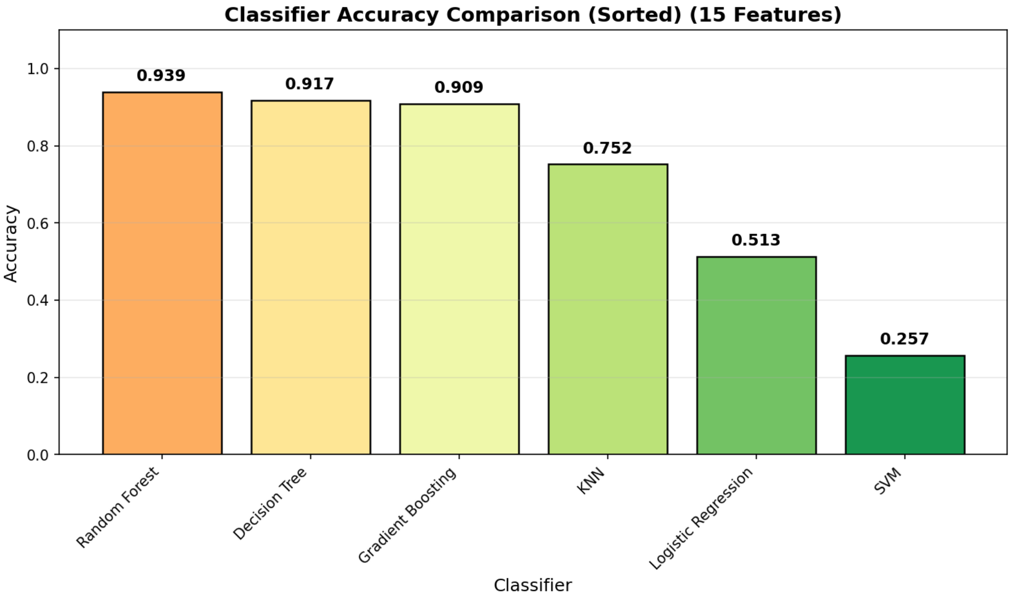


Figure 2: Accuracy Comparison – Reduced Feature Set (15 Features)

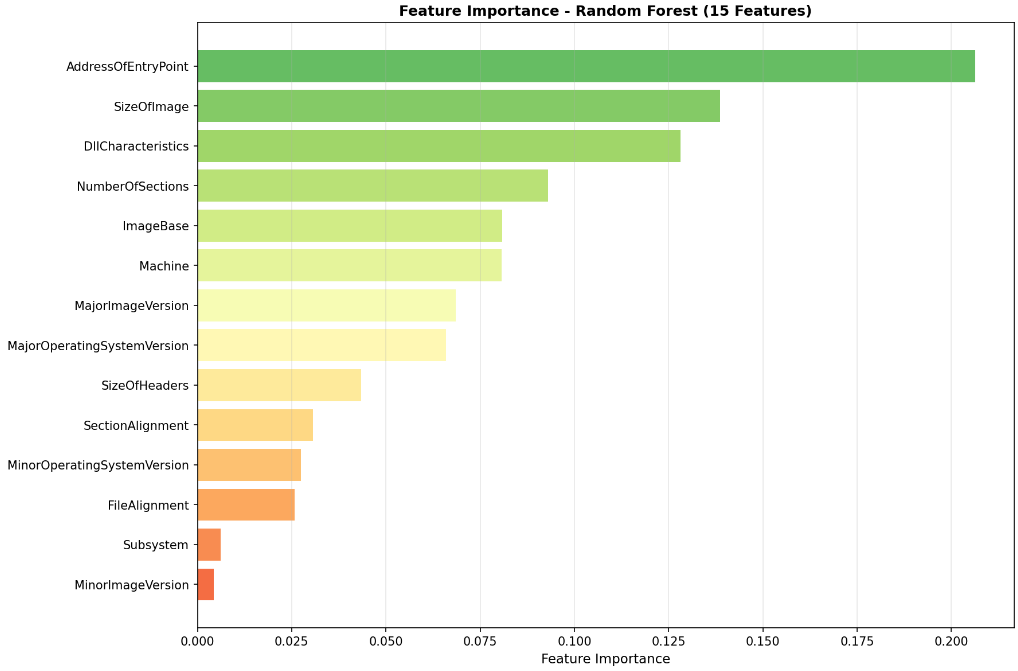


Figure 3: Feature Importance (Random Forest) – Reduced Feature Set (15 Features)

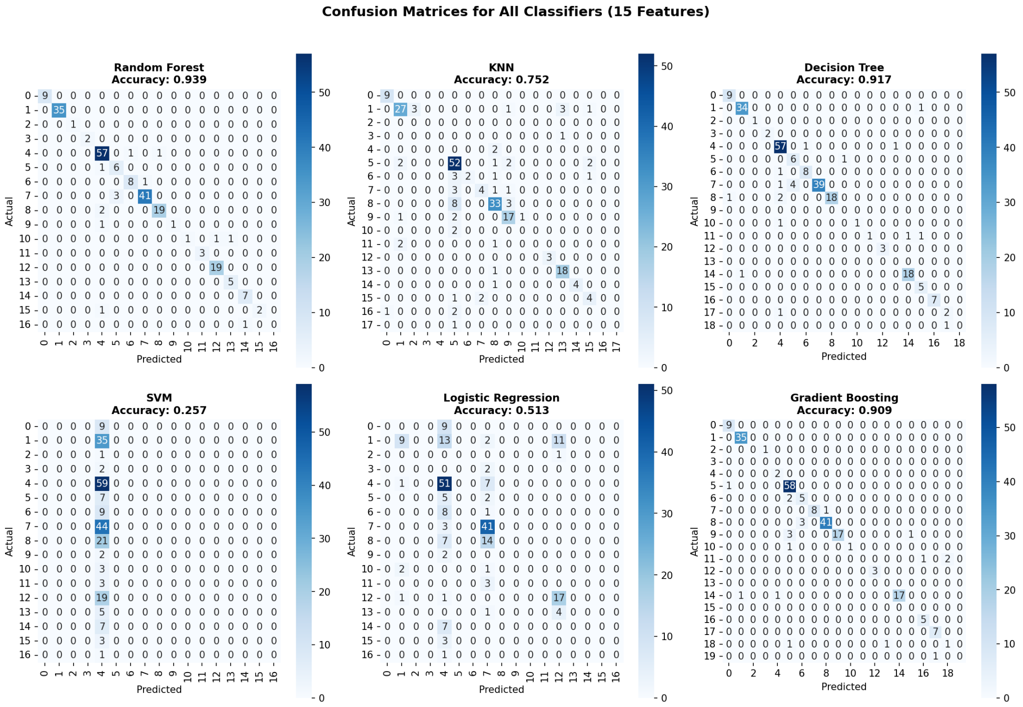


Figure 4: Confusion Matrices – Reduced Feature Set (15 Features)

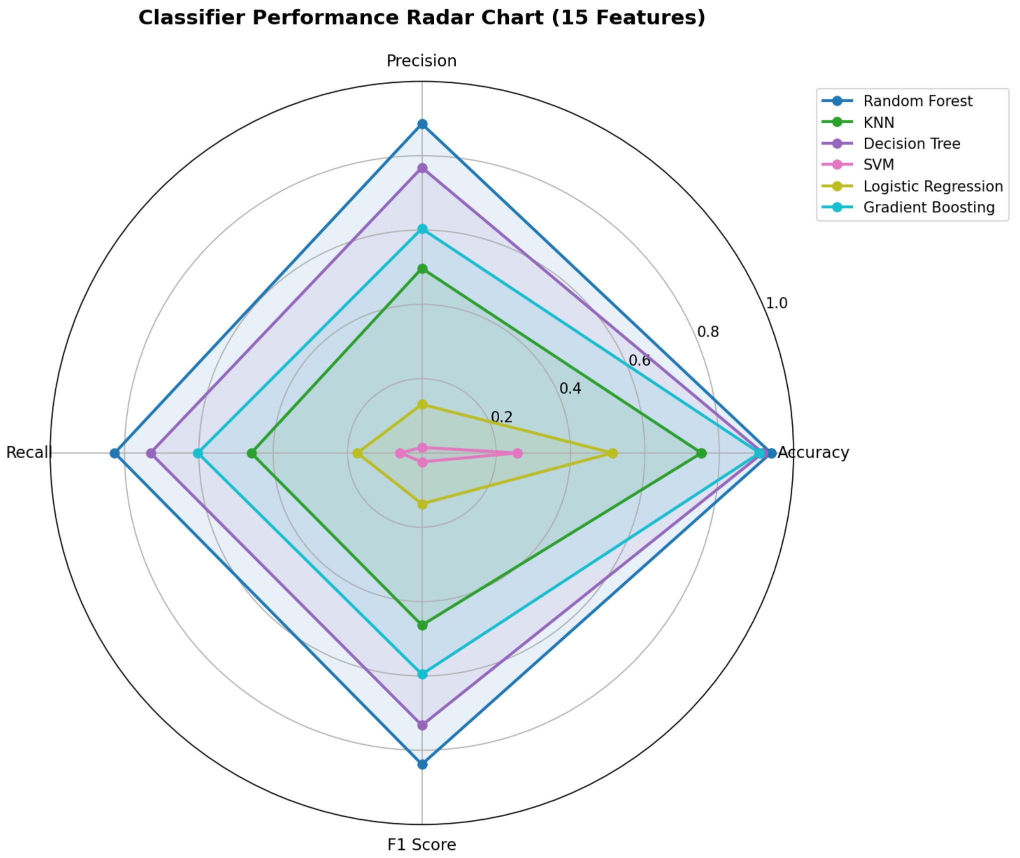
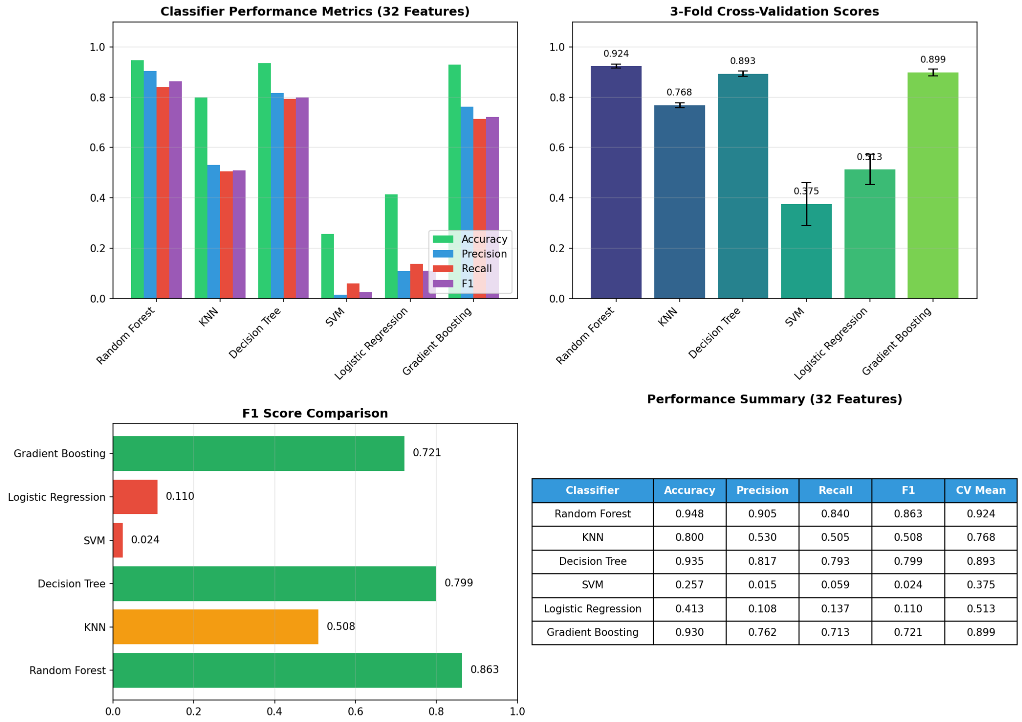


Figure 5: Performance Radar Chart – Reduced Feature Set (15 Features)

7.3.2 Extended Feature Set Charts



Classifier	Accuracy	Precision	Recall	F1	CV Mean
Random Forest	0.948	0.905	0.840	0.863	0.924
KNN	0.800	0.530	0.505	0.508	0.768
Decision Tree	0.935	0.817	0.793	0.799	0.893
SVM	0.257	0.015	0.059	0.024	0.375
Logistic Regression	0.413	0.108	0.137	0.110	0.513
Gradient Boosting	0.930	0.762	0.713	0.721	0.899

Figure 6: Classifier Performance Comparison – Extended Feature Set (32 Features)

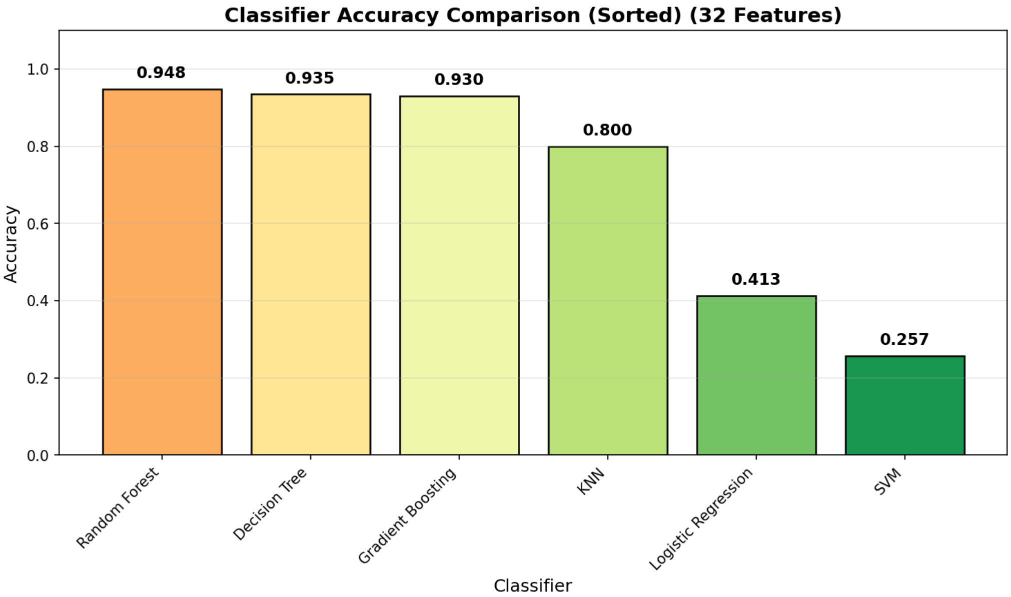


Figure 7: Accuracy Comparison – Extended Feature Set (32 Features)

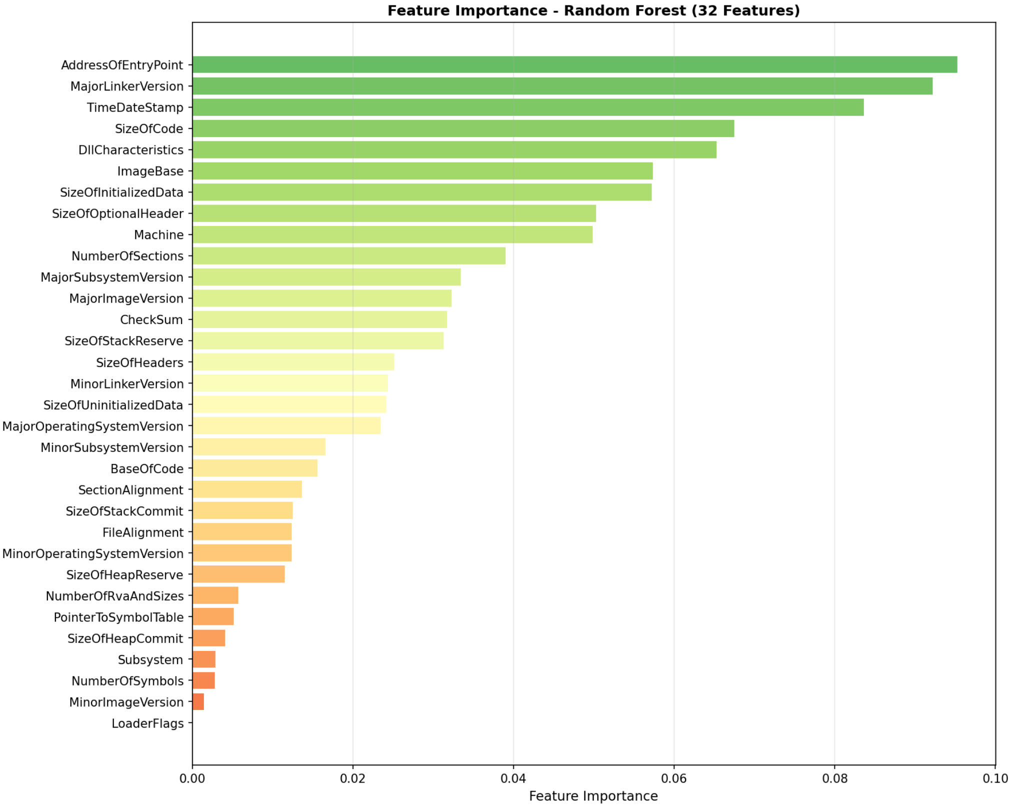


Figure 8: Feature Importance (Random Forest) – Extended Feature Set (32 Features)

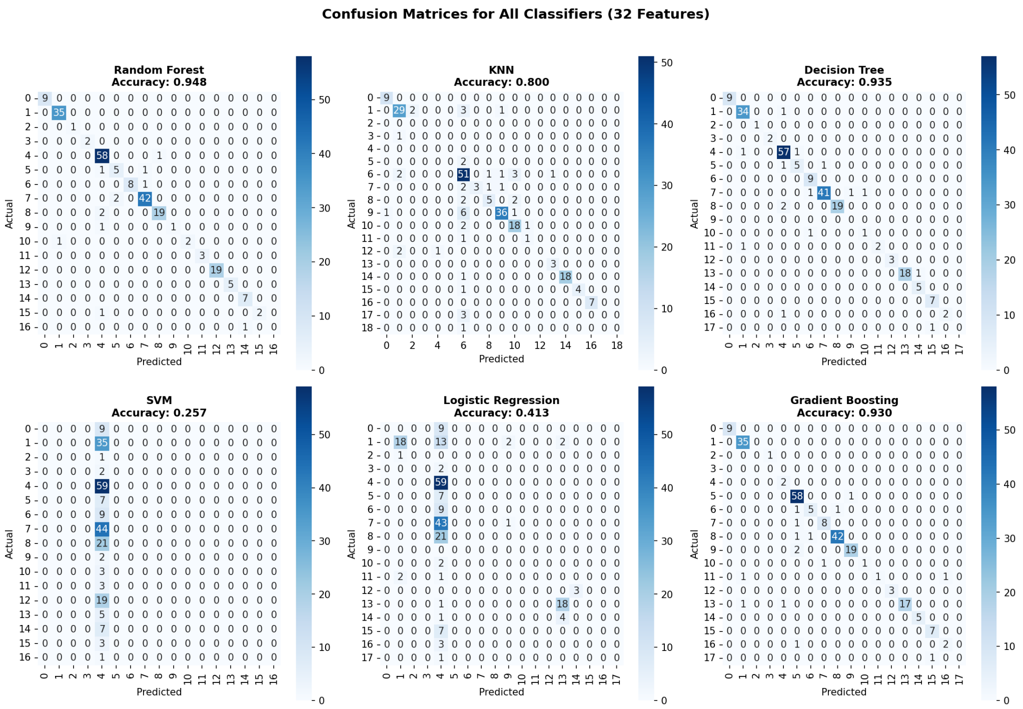


Figure 9: Confusion Matrices – Extended Feature Set (32 Features)

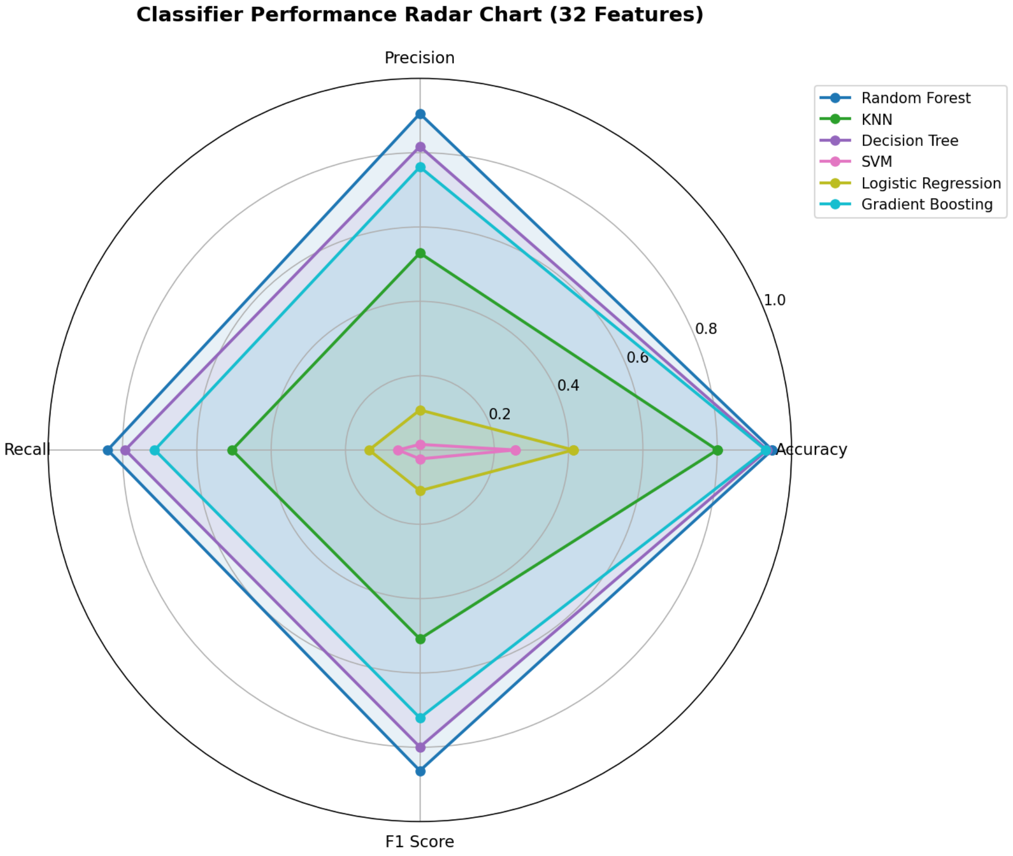


Figure 10: Performance Radar Chart – Extended Feature Set (32 Features)

8 Consistency Fixes

During the project review, nine issues were identified and fixed to ensure full consistency between the research paper, Python code, generated charts, and result tables. These are organized into two phases: five code-paper consistency fixes and four LaTeX quality fixes.

8.1 Phase 1: Code-Paper Consistency Fixes (Issues 1–5)

8.1.1 Issue 1: Logistic Regression Accuracy Discrepancy

Table 16: Issue 1 – LR Accuracy Fix

Item	Before	After
Paper Table II (LR Accuracy)	41.30%	**51.30%**
Paper Table II (LR Metrics)	F1: 0.1098, Prec: 0.1076, Rec: 0.1368	F1: **0.1369**, Prec: **0.1309**, Rec: **0.1734**
Paper Table IV	Missing LR and SVM rows	Added both rows
Root Cause	`thesis_with_charts.py` used `StandardScaler` inflating LR; paper copied 41.30% from extended set	Removed `StandardScaler`; updated paper to match actual output
Files Changed	`thesis_with_charts.py`, `main.tex`	

8.1.2 Issue 2: SVM Kernel Inconsistency

Table 17: Issue 2 – SVM Kernel Fix

Item	Before	After
`generate_charts.py`	`kernel="linear"`	`kernel='rbf', gamma='scale', C=1.0`
`thesis_with_charts.py`	`kernel='rbf', gamma=0.1`	`kernel='rbf', gamma='scale'`
Paper	States "RBF kernel"	Confirmed consistent
Files Changed	`generate_charts.py`, `thesis_with_charts.py`	

8.1.3 Issue 3: Cross-Validation Folds Mismatch

Table 18: Issue 3 – CV Folds Fix

Item	Before	After
`generate_charts.py`	`cv=5` (6 occurrences)	`cv=3`
`thesis_with_charts.py`	`cv=5`	`cv=3`
Chart Titles	"5-Fold Cross-Validation Scores"	"3-Fold Cross-Validation Scores"
Paper	States "3-fold cross-validation"	Added "(consistent across both feature sets)"
Files Changed	`generate_charts.py`, `thesis_with_charts.py`, `main.tex`	

8.1.4 Issue 4: Gradient Boosting Estimators Mismatch

Table 19: Issue 4 – GB Estimators Fix

Item	Before	After
`generate_charts.py`	`GradientBoostingClassifier (random_state=42)` (default 100)	`n_estimators=50`
`thesis_with_charts.py`	`n_estimators=100`	`n_estimators=50`
Paper	States "50 estimators"	Confirmed consistent
Files Changed	`generate_charts.py`, `thesis_with_charts.py`	

8.1.5 Issue 5: Feature Count Label Mismatch

Table 20: Issue 5 – Feature Count Label Fix

Item	Before	After
`generate_charts_more_features.py`	All chart titles say "33 Features"	"32 Features" (5 occurrences)
Paper	States "Extended Feature Set (32 Features)"	No change needed
Files Changed	`generate_charts_more_features.py`	

8.2 Phase 2: LaTeX Quality Fixes (Issues 6–9)

8.2.1 Issue 6: Removed Unused Packages

The `listings` and `subcaption` packages were loaded in `main.tex` but never used. Both were removed to clean up the preamble.

8.2.2 Issue 7: Removed Unnecessary `\bibliographystyle`

The command `\bibliographystyle{IEEEtran}` was present but unnecessary since the paper uses a manual `\begin{thebibliography}` environment. The command was removed.

8.2.3 Issue 8: Fixed Uncited Reference

The bibliography entry `\bibitem{yuk2022static}` was defined but never cited in the text. A citation was added in Section II-C alongside the existing `\cite{alkhshali2020effect}`, changing "differ from benign software [7]" to "differ from benign software [7, 9]."

8.2.4 Issue 9: Fixed Float Placement Specifiers

All float environments used the weak `[h]` specifier, which allows LaTeX to reorder figures and tables arbitrarily. All 17 instances (11 figures, 5 tables, 1 algorithm) were changed to the strict `[H]` specifier (from the `float` package) to enforce exact placement.

8.3 Post-Fix Verification

After all fixes, a comprehensive audit confirmed full consistency:

Table 21: Consistency Verification Checklist

Check	Status
Paper Table II matches reduced set code output	Verified
Paper Table III matches extended set code output	Verified
Paper Table IV computed correctly from Tables II and III	Verified
Paper Table V matches feature importance from Random Forest	Verified
All 10 chart images match paper table numbers	Verified
Paper text descriptions match table values	Verified
All `\cite{}` commands have matching `\bibitem{}` entries	Verified
All `\includegraphics` reference existing files	Verified
All packages are available on Overleaf	Verified
Paper compiles without errors (7 pages)	Verified

9 File Organization

9.1 Directory Structure

All project files reside in:
`/Users/fabihajalal/Desktop/PE feature reduction time/`

```
PE feature reduction time/
|-- Python Scripts
|   |-- generate_charts.py
|   |-- generate_charts_more_features.py
|   |-- thesis_with_charts.py
|   |-- make_charts.py
|   |-- rerun_all.py
|   |-- malware_downloader.py
|
|-- Data Files (CSV)
|   |-- output_file_final.csv        (1,150 x 16)
|   |-- output_file_more_features.csv (1,150 x 34)
|   |-- extracted_features.csv        (backup)
|
|-- Visualization Charts (PNG)
|   |-- Reduced Feature Set:
|   |   |-- classifier_comparison.png
|   |   |-- confusion_matrices.png
|   |   |-- feature_importance.png
|   |   |-- accuracy_comparison.png
|   |   |-- metrics_radar.png
|   |
|   |-- Extended Feature Set:
|       |-- classifier_comparison_more_features.png
|       |-- confusion_matrices_more_features.png
|       |-- feature_importance_more_features.png
|       |-- accuracy_comparison_more_features.png
|       |-- metrics_radar_more_features.png
|
|-- Documentation & Papers
```

```
|   |-- File_Comparison_Analysis.md
|   |-- Paper_Updated.pdf
|   |-- Overleaf_Paper/
|   |   |-- Overleaf_Paper/
|   |       |-- main.tex  (IEEE-formatted paper)
|   |       |-- main.pdf  (compiled paper)
|   |-- Overleaf_Paper_Updated.zip
|   |-- Project_Documentation.tex  (this document)
|   |-- Project_Documentation.pdf  (this document)
|
|-- Dataset
|   |-- extracted/  (2,005 directories of PE files)
|
|-- Version Control
    |-- .git/
    |-- .gitignore
```

9.2 Dependencies

Table 22: Python Dependencies

Package	**Purpose**
`pefile`	PE file parsing
`pandas`	Data manipulation and CSV I/O
`numpy`	Numerical computations
`scikit-learn`	Machine learning classifiers and evaluation
`matplotlib`	Chart generation
`seaborn`	Enhanced visualization styling
`requests`	HTTP requests (malware downloader)
`pyzipper`	ZIP decompression with AES support

10 Key Findings

10.1 Feature Extraction Efficiency

The proposed methodology achieves significant improvements in feature extraction time compared to prior work. Processing 1,150 malware samples requires only 215–290 seconds (depending on feature set size), representing a **75% reduction** compared to the 19-minute extraction time reported for the APT1 dataset. This improvement enables more practical deployment in real-time malware detection scenarios.

10.2 Classification Performance

1. **Random Forest** achieved the highest overall accuracy at **94.78%** with the extended feature set and **93.91%** with the reduced feature set.

2. **Tree-based methods** (Random Forest, Decision Tree, Gradient Boosting) and KNN improved by +0.87% to +4.78% with the extended feature set.

3. **Logistic Regression** performed worse with the extended set (41.30% vs. 51.30%), likely due to multicollinearity among the additional features.

4. **SVM** exhibited poor performance (25.65%) in both feature sets, likely due to the multi-class nature of the problem and the absence of feature scaling.

10.3 Accuracy vs. Extraction Time Trade-off

While the extended feature set provides improved accuracy for tree-based classifiers, it requires approximately **34% more extraction time** (290 seconds vs. 215 seconds). For applications prioritizing speed over marginal accuracy improvements, the reduced feature set offers an effective compromise.

10.4 Feature Importance Insights

The top features for malware classification include `AddressOfEntryPoint` (0.095), `MajorLinkerVersion` (0.092), and `TimeDateStamp` (0.084). Some features contribute approximately 10× more discriminative power than others, suggesting that targeted feature selection could further optimize extraction efficiency without sacrificing classification performance.

10.5 Practical Implications

The research demonstrates that efficient malware detection is achievable using only PE header features without requiring computationally expensive dynamic analysis. The methodology is suitable for:

- Real-time malware scanning applications
- Large-scale malware dataset analysis
- Integration with existing antivirus systems
- Automated malware triage systems

10.6 Classifier Ranking Summary

Table 23: Final Classifier Ranking (Extended Feature Set)

Rank	Classifier	Accuracy
1	Random Forest	94.78%
2	Decision Tree	93.48%
3	Gradient Boosting	93.04%
4	KNN	80.00%
5	Logistic Regression	41.30%
6	SVM	25.65%