11th May, 2024

4th International Conference on Machine Learning and Big Data Analytics (ICMLBDA 2024)

193: Malware Classification Leveraging NLP & Machine Learning for Enhanced Accuracy

Presented by

Bishwajit Prasad Gond, Rajneekant, Pushkar Kishore and Dr Durga Prasad Mohapatra

National Institute of Technology Rourkela

PLAN OF DISCUSSION

INTRODUCTION

As cyber threats continually evolve, the need for advanced malware detection and classification methods becomes paramount.

- This paper investigates the application of NLP-based n-gram analysis and ML techniques to enhance malware classification.
- We explore how NLP can be used to extract and analyze textual features from malware samples through n-grams, contiguous string or API call sequences.
- This approach effectively captures distinctive linguistic patterns among malware and benign families, enabling finer-grained classification.
- We delve into n-gram size selection, feature representation, and classification algorithms.

INTRODUCTION CONT.

MALWARE: Malware (*malicious software*)[1] is any software intentionally designed to

- cause disruption to a computer, server, client, or computer network,
- leak private information, gain unauthorized access to information or systems,
- deprive access to information, or which unknowingly interferes with the user's computer security and privacy

MALWARE TYPES

1. **Virus:** Virus infects computers and other files by replicating itself.

2. **Worms:** Worms are malicious piece of code that exist independently. they have feature to replicate itself. They propagate through storage devices and emails, also consume network and computer resources which leads to system degradation in performance.

3. **Trojan Horse:** It behaves like a useful program but its harmful. They do not replicate themselves but transferred to a computer by internet interaction like downloading.

4. **Spyware:** Spyware are used to steal someone personal information or keep a watch on user's activities. It is installed without the knowledge of system owner and secretly collect the information and send it back to the creator.

5. **Adware:** Adware are quite annoying most of the time as it plays advertisement on user computer without its permission and interrupt its current activity. Basic purpose of the adware is to get financial benefit. It does as much harmful as other malwares.

MALWARE TYPES

6. Downloaders: Programs that download and install other malware onto a victim's computer, often through deceptive means.

7. Backdoors: Unauthorized access points into a computer system or network, allowing attackers to control it remotely without the user's knowledge.

8. Ransomware: Ransomware are malwares that take control of your PC by encrypting your data, stop some application and don't allow you use your operating system until you fulfil their demands. The demands is mostly in terms of cryptocurrency. After paying ransom, it is still not guaranteed that you will get your control back.

APPROACHES OF MALWARE ANALYSIS

1. **Static Analysis:** This involves examining the malware without executing it. It includes examining the code, file structure, and metadata to identify patterns, signatures, and potential indicators of malicious activity. Tools such as disassemblers, decompilers, and hex editors are commonly used for static analysis.

2. **Dynamic Analysis:** Dynamic analysis involves executing the malware in a **controlled environment, such as a virtual machine or sandbox**, to observe its behavior. Analysts monitor system changes, network activity, file modifications, and any other actions taken by the malware. This approach provides insights into the malware's runtime behavior and potential impact on a system.

3. **Behavioral Analysis:** Behavioral analysis focuses on understanding how malware behaves when executed. Analysts observe the actions taken by the malware, such as file system modifications, registry changes, network communications, and interactions with other processes. This helps in understanding the malware's intentions and potential capabilities.

INTRODUCTION CONT.

N-grams and TF-IDF (Term Frequency-Inverse Document Frequency) are two fundamental concepts in natural language processing (NLP) and information retrieval. They play crucial roles in tasks such as text analysis, document classification etc.

N-grams: N-grams are contiguous sequences of n items (words, characters, or symbols) extracted from a text document. The **"n"** in n-grams represents the number of items in the sequence.

For example, in the sentence "The quick brown fox," some 2-grams (or bigrams) would be "The quick," "quick brown," and "brown fox."

Similarly, 3-grams (trigrams) would include "The quick brown" and "quick brown fox."

N-grams capture the local structure and context of words in a document. They are particularly useful in tasks such as language modeling, spell checking, and predictive text input. By analyzing the frequency and distribution of n-grams, we can derive insights into the linguistic patterns and characteristics of a text corpus.

INTRODUCTION CONT.

TF-IDF: TF-IDF is a statistical measure used to evaluate the importance of a term in a document relative to a collection of documents (corpus). It stands for Term Frequency-Inverse Document Frequency.

Term Frequency (TF) measures the frequency of a term within a document. It indicates how often a particular word appears in a document relative to the total number of words in that document. It can be calculated using the formula:

$$\mathrm{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \qquad \ldots\ldots\ldots (1)$$

Inverse Document Frequency (IDF) measures the rarity of a term across the entire corpus. It is calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term. Rare terms have higher IDF scores, while common terms have lower IDF scores. The IDF of a term (t) is calculated as:

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right) \qquad \ldots\ldots\ldots (2)$$

INTRODUCTION CONT.

TF-IDF is obtained by multiplying the TF and IDF scores:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t) \qquad \ldots\ldots.. (3)$$

TF-IDF helps in identifying the significance of terms in a document within the context of the entire corpus. Terms that are frequent in a document but rare in the corpus will have high TF-IDF scores, indicating their importance in representing the content of that document.

OBJECTIVES

1. **Enhanced Accuracy through NLP and ML Techniques:** Develop a malware classifier that leverages Natural Language Processing (NLP) techniques to analyze API call sequences, coupled with Machine Learning (ML) algorithms, to achieve superior accuracy in classifying malware into different classes.

2. **Dynamic Analysis with API Calls:** Incorporate dynamic analysis techniques using API calls to improve the classifier's ability to detect and classify malware. This can involve analyzing the sequence and patterns of API calls made by malware during execution to enhance the classifier's effectiveness.

3. **Feature Reduction for Efficient Classification:** Utilize feature selection and extraction methods to reduce the number of features used for classification, focusing only on the most relevant and informative features. This approach aims to improve the classifier's efficiency and performance while maintaining high accuracy in malware classification.

RELATED WORK

TITLE	AUTHOR	YEAR	CONTRIBUTIONS	LIMITATIONS
An effective malware detection method using hybrid feature selection and machine learning algorithms	Dabas et al. [2]	2023	The paper extracts API calls information in three forms: usage, frequency, and sequences, creating feature sets. These are enriched using TF-IDF and combined into the API integrated feature set, addressing high-dimensionality.	The proposed methods is not generalize well to all types of malware or might be sensitive to changes in malware behavior over time
Malanalyser: An effective and efficient windows malware detection method based on api call sequences	Dabas et al. [3]	2023	The paper introduces a novel malware detection method for Windows, based on API call sequences. It extracts API call information in specific sequences to create a feature set.	1. is not generalize well to all types of malware 2. The Problem of concept drift

RELATED WORK CONT…

TITLE	AUTHOR	YEAR	CONTRIBUTIONS	LIMITATIONS
Windows Malware Detection using Machine Learning and TF-IDF Enriched API Calls Information	Sharma et al. [4]	2022	The paper investigates the use of API calls in the Windows operating system for identifying and categorizing program behavior as either malware or benign. The study utilizes information from dynamic analysis of both malware and benign samples, focusing on three feature sets: API calls usage, API calls frequency, and API calls sequences, which are then integrated into a single API calls feature set.	1. Dataset is small having 5000 sample. 2. Malware Classes not disclosed
DMalNet: Dynamic malware analysis based on API feature engineering and graph learning	Li et al. [5]	2022	In this paper, authors developed API feature engineering ad API call graph learning with Graph Neural Networks for malware detection and malware classification	1. Robustness to adversarial attacks 2. The Problem of concept drift

PROPOSED ARCHITECTURE

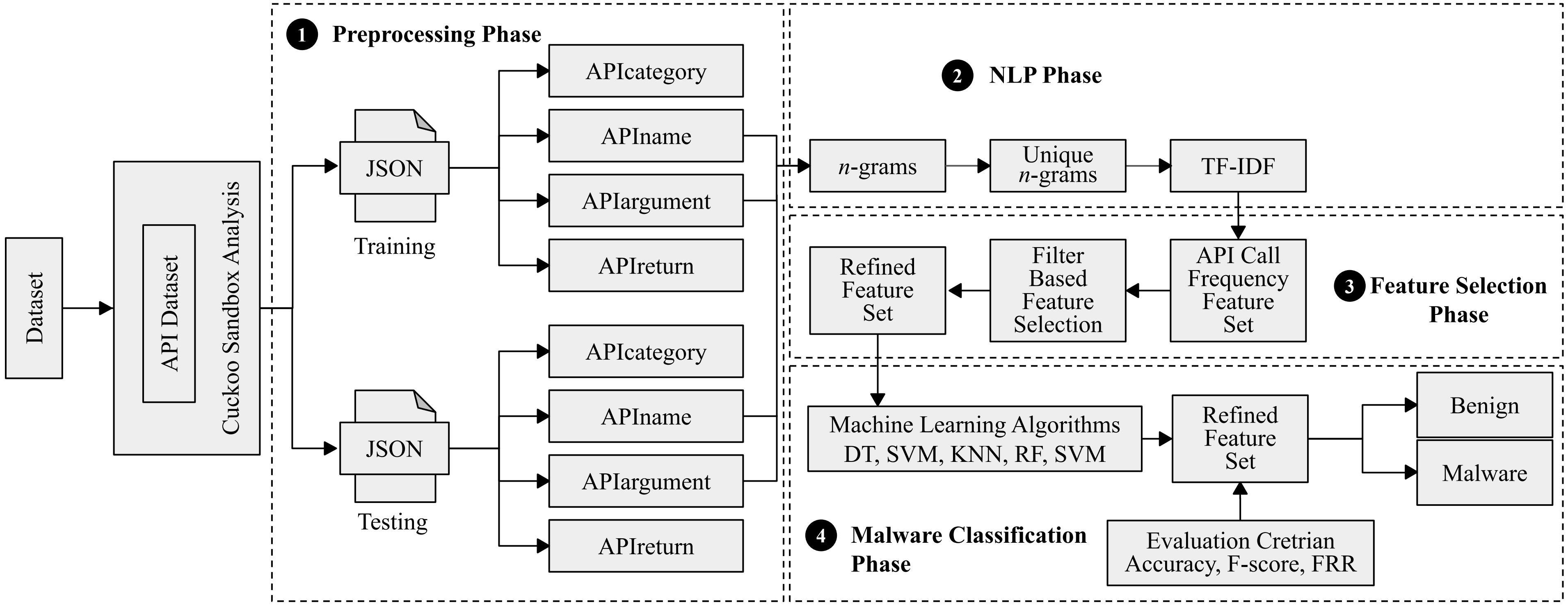
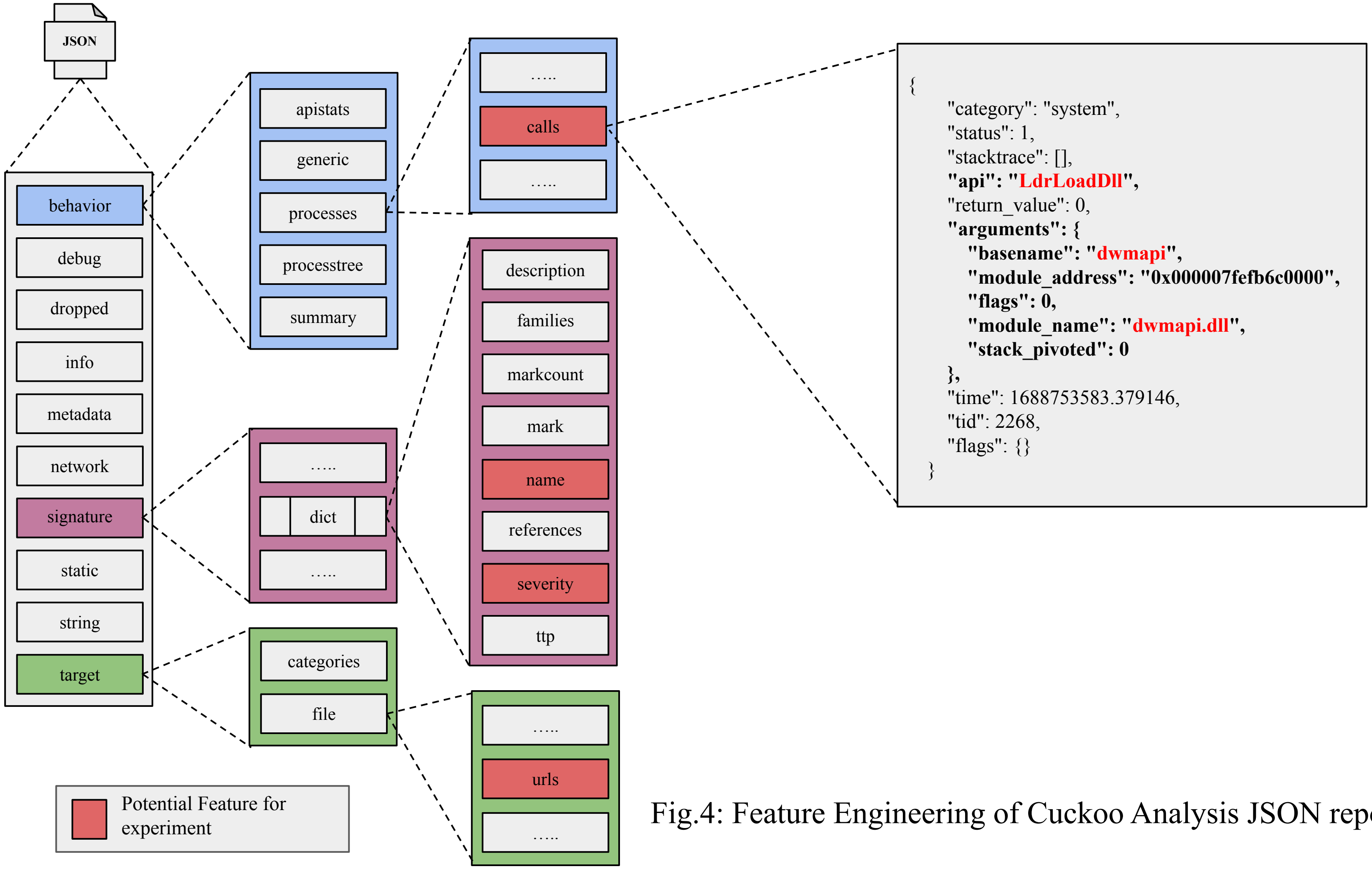


Fig.1: Proposed Architecture for Malware Classification using NLP and ML



```
{
    "category": "system",
    "status": 1,
    "stacktrace": [],
    "api": "LdrLoadDll",
    "return_value": 0,
    "arguments": {
        "basename": "dwmapi",
        "module_address": "0x000007fefb6c0000",
        "flags": 0,
        "module_name": "dwmapi.dll",
        "stack_pivoted": 0
    },
    "time": 1688753583.379146,
    "tid": 2268,
    "flags": {}
}
```

Fig.4: Feature Engineering of Cuckoo Analysis JSON report

METHODOLOGY

PHASE 1: DATA PREPROCESSING

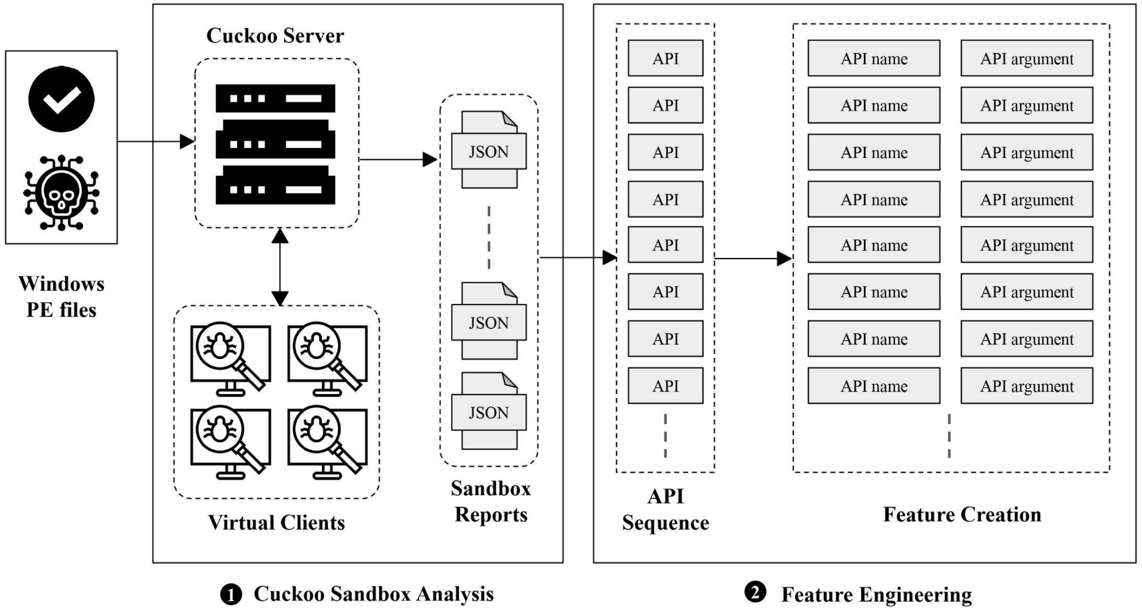


Fig. 2: Data Preprocessing and Feature Engineering

1. **Preprocess the Data:** After obtaining the data for each class, we performed behavioral analysis using Cuckoo Sandbox, resulting in a behavioral report in JSON format.

2. **Data Division:** The dataset is partitioned into training and testing sets, formatted as JSON files.

3. **API Sequence:** We then split the API sequence of JSON into four parts: API category, API name, API argument, and API return.

4. **Feature Engineering:** we selected the API name and argument to create n-grams, where the API name is the first part and the API call argument is added using underscores. Finally, we obtained a CSV file containing n-grams for each category.

METHODOLOGY CONT..

PHASE 2: NLP PART

1. ngrams and Unique ngrams: N-grams and unique N-grams are extracted from the dataset to capture linguistic patterns.
2. n-grams after Processing: The following are examples of unigrams, bigrams, and trigrams that we have used in this paper:

- Unigram: *LdrLoadDll_urlmon_urlmon.dll*
- Bigram: *NtAllocateVirtualMemory_na, LdrLoadDll_ole32_ole32.dll*
- Trigram: *LdrUnloadDll_SHELL32, LdrLoadDll_SETUPAPI_SETUPAPI.dll, LdrGetProcedureAddress_ole32_OleUninitialize*

3. TF-IDF: It tokenizes text, counts occurrences of each token, and computes TF-IDF weights, which reflect the importance of each token in a document relative to the entire corpus. Optional L2 normalization ensures consistent feature vector lengths. TF-IDF is applied to each CSV file of unigram, bigram, and trigram with the unique n-gram CSV file to transform the text data.
4. Refined Feature Set: A refined feature set is obtained after filtering based on frequency, ensuring relevance and significance in subsequent stages.

METHODOLOGY CONT..

$$\underbrace{\text{LdrLoadDll(}}_{\text{API call}} \quad \underbrace{\text{ADVAPI32.dll}}_{\substack{\text{API arguments} \\ \text{(DLL file)}}} \quad \underbrace{\begin{matrix}\text{ADVAPI32.dll} \\ \vdots \end{matrix}}_{\substack{\text{API arguments} \\ \text{(DLL file)}}} \text{)}$$

LdrLoadDll_ADVAPI32_ADVAPI32.dll

LdrLoadDll_urlmon_urlmon.dll

Fig. 3: API call structure

API arguments: An API arguments can be DLL file, URL and memory address etc.

METHODOLOGY CONT...

PHASE 3: FEATURE SELECTION PHASE

1. **API Call Frequency Feature Set:** Features derived from API call frequency are explored to gain insights into system behavior and usage patterns.

2. **Filter Based Feature Selection:** Filter-based techniques such as mutual information and correlation analysis are applied to select the most informative features.

3. **Refine Feature Set:** Redundant or irrelevant features are eliminated to ensure the final feature set is discriminative and predictive.

METHODOLOGY CONT...

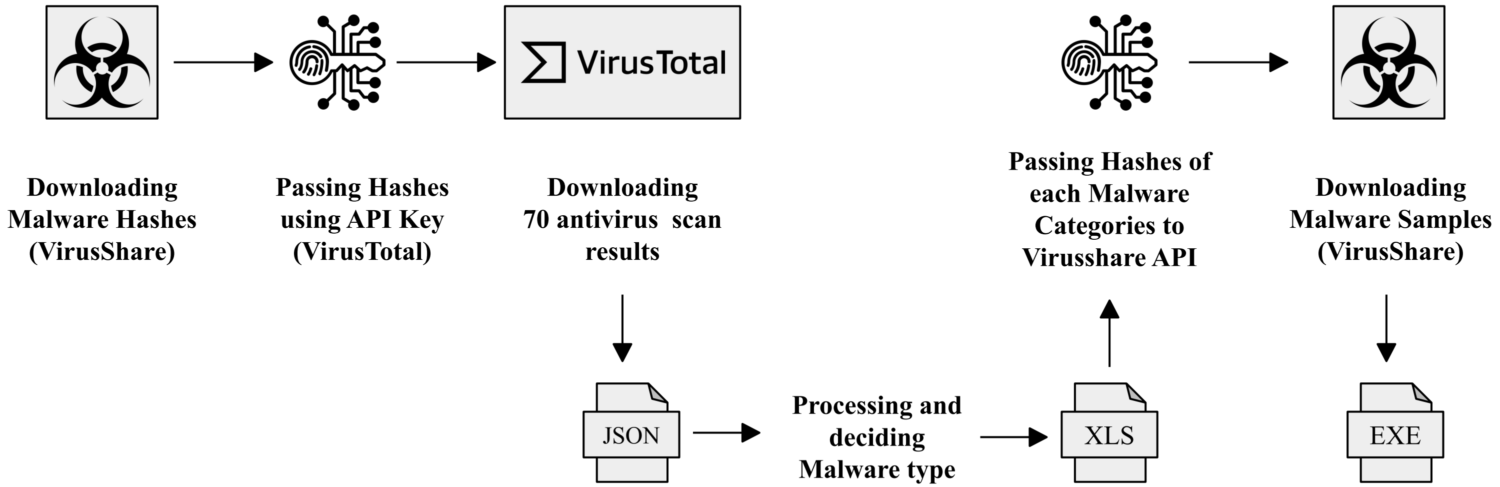
PHASE 4: MALWARE CLASSIFICATION PHASE

1. Machine Learning Algorithms: Various machine learning algorithms including Decision Trees (DT), Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Logistic Regression (LR) are applied to the refined feature set from Phase 2.

2. Evaluation Criteria: Evaluation is based on accuracy, F-score, and False Rejection Rate (FRR) to determine the effectiveness of the chosen algorithms.

3. Malware Detection: This phase involves distinguishing between benign and malware entities based on the features selected and evaluated in the previous phases.

EXPERIMENTS AND RESULTS



VirusTotal
Downloading Malware Hashes (VirusShare)
Passing Hashes using API Key (VirusTotal)
Downloading 70 antivirus scan results
Passing Hashes of each Malware Categories to Virusshare API
Downloading Malware Samples (VirusShare)
JSON
Processing and deciding Malware type
XLS
EXE

Fig. 4: Data Collection Process Using VirusShare and VirusTotal

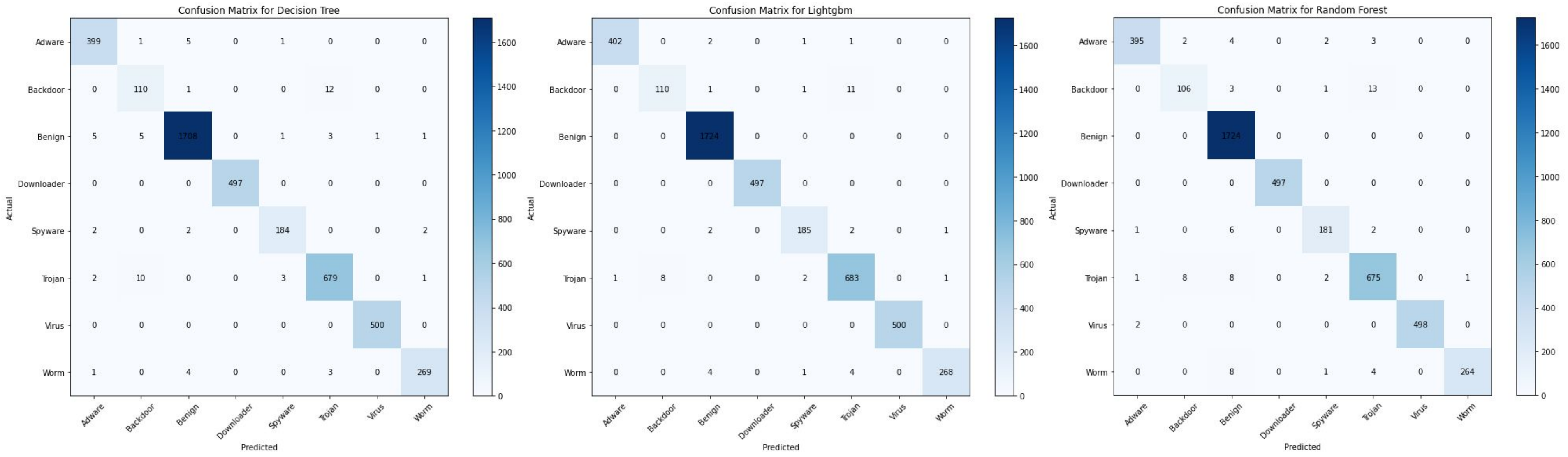
- We obtained an API key from VirusShare, which allowed us to download more than two lakhs (200,000) JSON files for each hash provided by VirusShare. However, there was a constraint of downloading only four JSON files per minute due to API limitations.
- From the downloaded JSON files, we extracted essential information, such as the scan results from 70 antivirus programs.

EXPERIMENTS AND RESULTS

Table 1: Datasets Used

NO.	Types	Test Sample	Train Sample	Total Sample
1	Adware	406	1580	**1986**
2	Backdoor	123	551	**674**
3	Downloader	495	2002	**2497**
4	Spyware	190	756	**946**
5	Trojan	695	2873	**3568**
6	Worm	277	1080	**1375**
7	Virus	500	1892	**2392**
8	Benign	1724	6910	**8634**
	Total	**4410**	**17644**	**22054**

RESULT ANALYSIS



Confusion Matrix for Decision Tree

	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	399	1	5	0	1	0	0	0
Backdoor	0	110	1	0	0	12	0	0
Benign	5	5	1708	0	1	3	1	1
Downloader	0	0	0	497	0	0	0	0
Spyware	2	0	2	0	184	0	0	2
Trojan	2	10	0	0	3	679	0	1
Virus	0	0	0	0	0	0	500	0
Worm	1	0	4	0	0	3	0	269

Confusion Matrix for Lightgbm

	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	402	0	2	0	1	1	0	0
Backdoor	0	110	1	0	1	11	0	0
Benign	0	0	1724	0	0	0	0	0
Downloader	0	0	0	497	0	0	0	0
Spyware	0	0	2	0	185	2	0	1
Trojan	1	8	0	0	2	683	0	1
Virus	0	0	0	0	0	0	500	0
Worm	0	0	4	0	1	4	0	268

Confusion Matrix for Random Forest

	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	395	2	4	0	2	3	0	0
Backdoor	0	106	3	0	1	13	0	0
Benign	0	0	1724	0	0	0	0	0
Downloader	0	0	0	497	0	0	0	0
Spyware	1	0	6	0	181	2	0	0
Trojan	1	8	8	0	2	675	0	1
Virus	2	0	0	0	0	0	498	0
Worm	0	0	8	0	1	4	0	264

Fig. 5: Confusion Matrix for Decision Tree, LightGBM and Random Forest

RESULT ANALYSIS

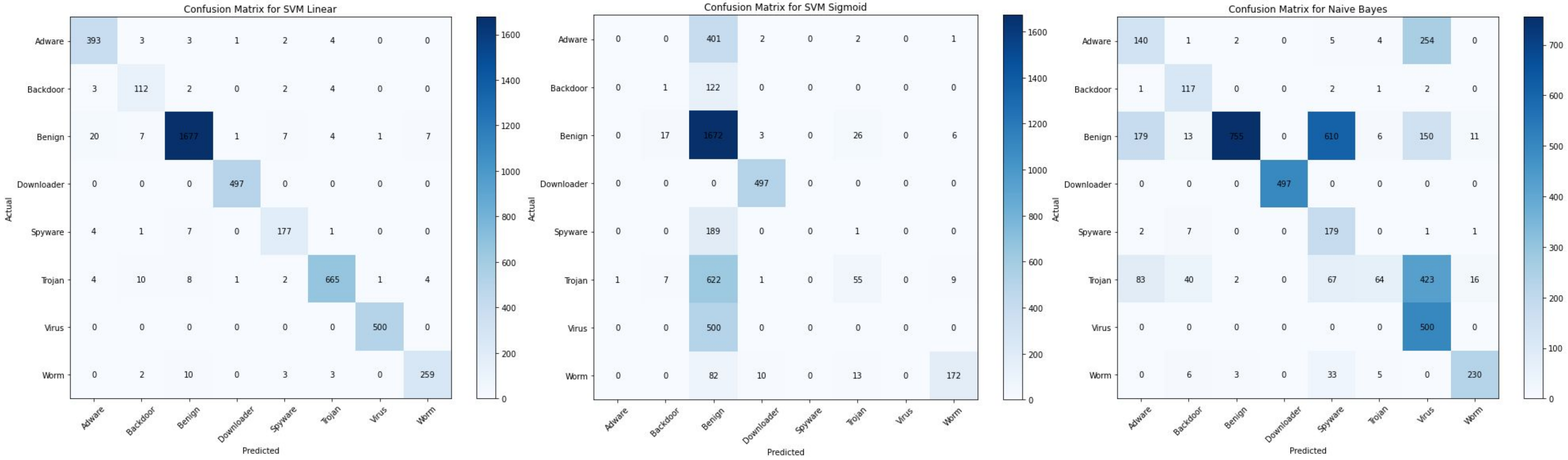
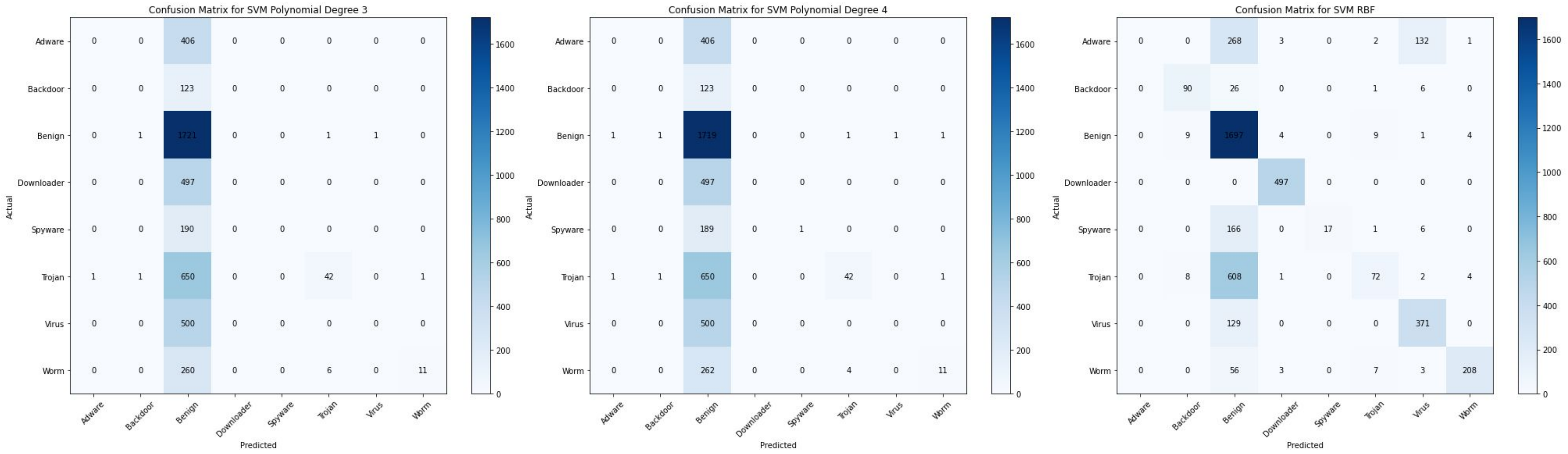


Fig. 6: Confusion Matrix for SVM Linear, SVM Sigmoid and Naive Bayes

RESULT ANALYSIS



Confusion Matrix for SVM Polynomial Degree 3

Actual \ Predicted	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	0	0	406	0	0	0	0	0
Backdoor	0	0	123	0	0	0	0	0
Benign	0	1	1721	0	0	1	1	0
Downloader	0	0	497	0	0	0	0	0
Spyware	0	0	190	0	0	0	0	0
Trojan	1	1	650	0	0	42	0	1
Virus	0	0	500	0	0	0	0	0
Worm	0	0	260	0	0	6	0	11

Confusion Matrix for SVM Polynomial Degree 4

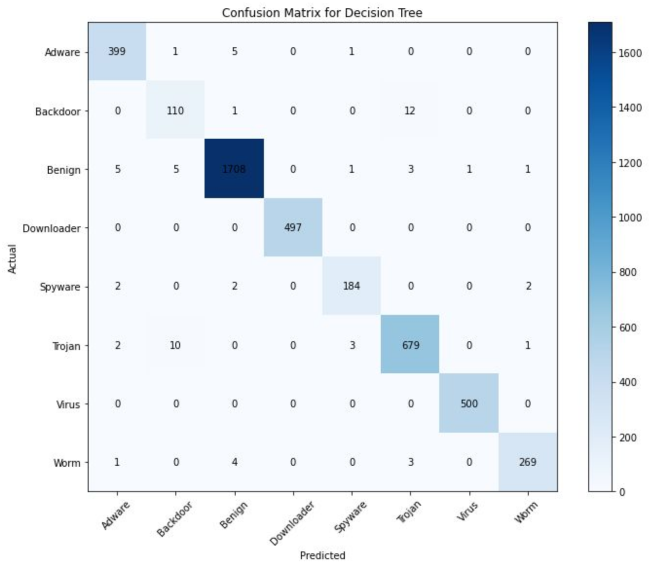
Actual \ Predicted	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	0	0	406	0	0	0	0	0
Backdoor	0	0	123	0	0	0	0	0
Benign	1	1	1719	0	0	1	1	1
Downloader	0	0	497	0	0	0	0	0
Spyware	0	0	189	0	1	0	0	0
Trojan	1	1	650	0	0	42	0	1
Virus	0	0	500	0	0	0	0	0
Worm	0	0	262	0	0	4	0	11

Confusion Matrix for SVM RBF

Actual \ Predicted	Adware	Backdoor	Benign	Downloader	Spyware	Trojan	Virus	Worm
Adware	0	0	268	3	0	2	132	1
Backdoor	0	90	26	0	0	1	6	0
Benign	0	9	1697	4	0	9	1	4
Downloader	0	0	0	497	0	0	0	0
Spyware	0	0	166	0	17	1	6	0
Trojan	0	8	608	1	0	72	2	4
Virus	0	0	129	0	0	0	371	0
Worm	0	0	56	3	0	7	3	208

Fig. 7: Confusion Matrix for SVM Poly 3°, SVM Poly 4° and SVM RBF

RESULT ANALYSIS



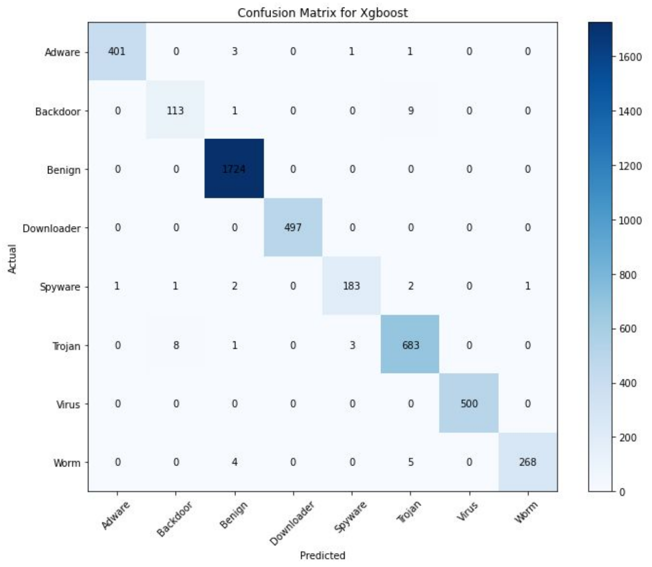
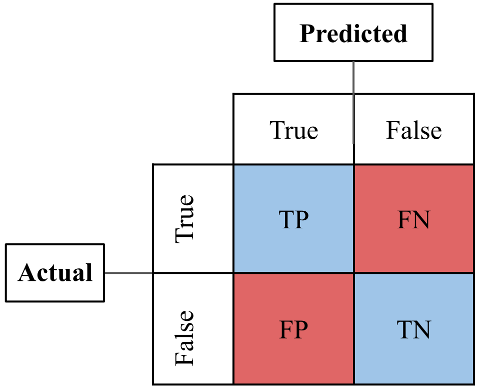


Fig. 8: Confusion Matrix for k-NN and XGboost

RESULT ANALYSIS



$$\textbf{Precision} = \frac{TP}{TP+FP}$$

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

$$\textbf{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$\textbf{F-1 Score} = 2 * \frac{precision * recall}{precision + recall}$$

$$\text{Precision}_{\textit{Macro-average}} = \frac{\text{Precision}_{adware} + \text{Precision}_{backdoor} + \ldots \text{Precision}_{benign}}{N}$$

$$\text{Recall}_{\textit{Macro-average}} = \frac{\text{Recall}_{adware} + \text{Recall}_{backdoor} + \ldots \text{Recall}_{benign}}{N}$$

Tabel 2: Comparison of Performance Metrics of using different ML models

NO.	ML Algo	ACC	PREC	RE	F1
1	DT	98.50	97.23	97.30	97.16
2	RF	98.37	97.03	96.35	97.75
3	KNN	94.40	92.65	92.16	93.16
4	Naive Bayes	56.26	75.61	67.26	65.49
5	SVM Linear	97.01	95.32	95.93	94.77
6	SVM Ploy 3°	40.21	9.45	13.73	27.12
7	SVM Ploy 4°	40.19	9.57	13.78	39.19
8	SVM RBF	66.91	55.17	55.03	73.10
9	SVM Sigmoid	54.33	31.32	33.48	36.96
10	**XGBoost**	**99.02**	**98.04**	**97.74**	**98.35**
11	**LightGBM**	**99.02**	97.95	97.60	98.31

RESULT ANALYSIS

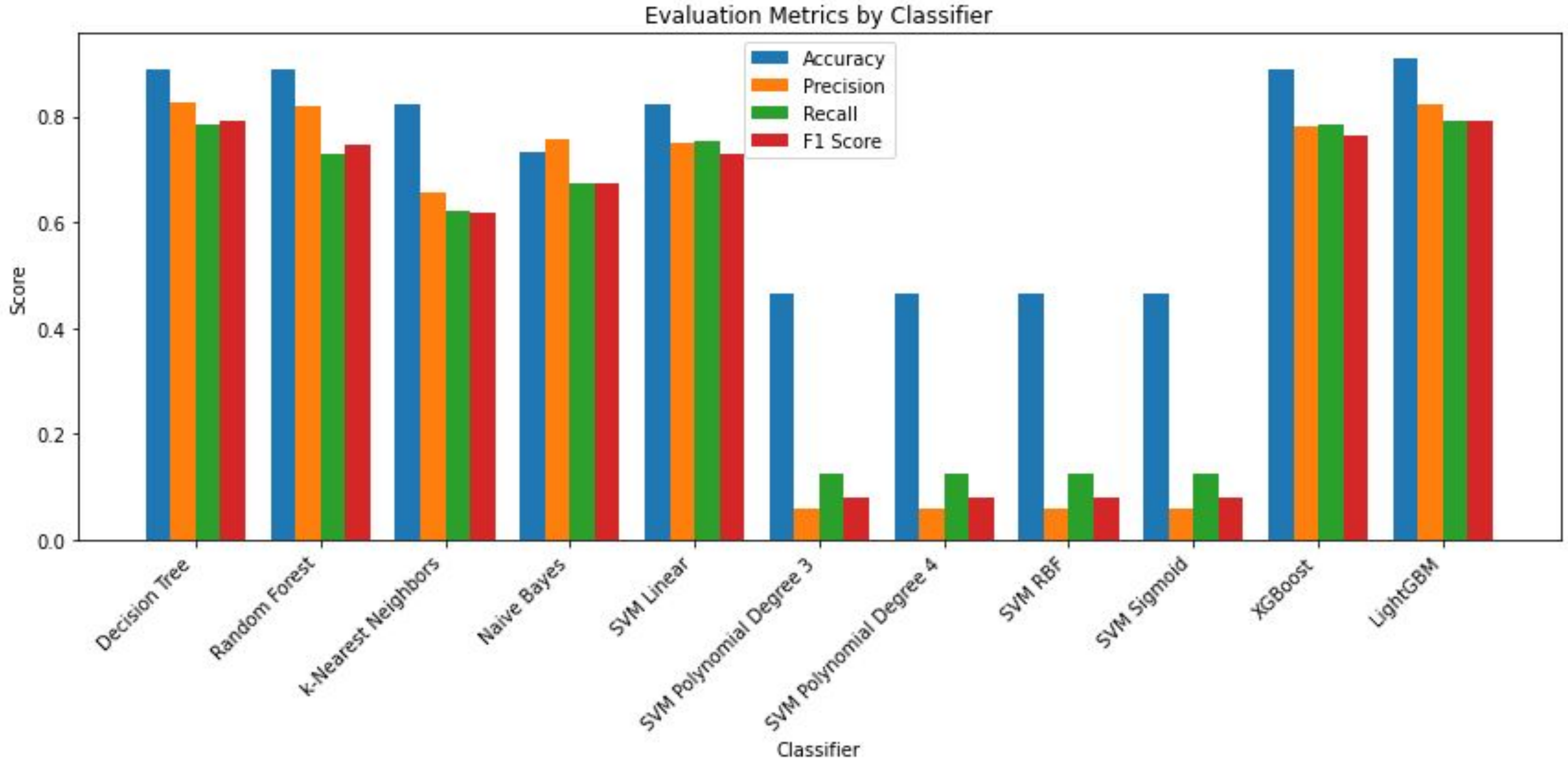


Fig. 9: Comparison of Algorithms Evaluation Metrics using different ML models

RESULT ANALYSIS

Precision is about being accurate when you say something is positive. Recall is about catching all the relevant positive things.

Recall is more important when:
Overlooked cases: (False Negatives) are more costly than False Alarms (False Positives) We cannot afford to miss any detection

Precision is more important when:
False Alarms: (False Positives) are more costly than Overlooked Cases (False Negatives) We cannot afford to have any incorrect detection

We would like to have less False Positives in trade off to have more False Negatives. For example, precision is more important than recall when the cost of buying a bad apple is high, but the cost of passing up a particular good apple is low.

COMPARISON OF OUR WORK

Tabel 3: Comparison with related work

No	Author	n-gram	IF-TDF	Feat. Red	ML Tech	Datasets	Class	Dect	Acc
1	Dabas et al.	✖	✔	9%	✔	VirusShare	✖	✔	99.6
2	Dabas et al.	✖	✔	30%	✔	VirusShare	✖	✔	99.7
3	Sharma et al.	✖	✔	Not Disclosed	✔	VirusShare	✖	✔	99.9
5	Proposed Work	✔	✔	1.6%	✔	VirusShare	✔	✖	99.02

From the 22,054 JSON files, we obtained 5,578,098 features through the feature extraction process. From this extensive set, we selected 88,975 features for our experiment, representing a significant reduction to approximately **1.6 %** of the total features.

HYPERPARAMETER USED

Tabel 4: Hyperparameter used for classifiers

No.	Classifiers	Hyperparameters Used
1	Decision Tree	criterion='gini', max_depth=None, splitter='best', min_samples_split=2, min_samples_leaf=1
2	Random Forest	n_estimators=100, criterion='gini', min_samples_split=2, max_depth=None, min_samples_leaf=1
3	k-Nearest Neighbors (KNN)	n_neighbors=5, weights='uniform', leaf_size=30, p=2 (Euclidean distance), algorithm='auto'
4	Naive Bayes	No hyperparameters for Gaussian Naive Bayes
5	SVM Linear	C=1.0, kernel='linear'
6	SVM Polynomial Degree 3	C=1.0, kernel='poly', degree=3
7	SVM Polynomial Degree 4	C=1.0, kernel='poly', degree=4
8	SVM RBF	C=1.0, kernel='rbf', gamma='scale'
9	SVM Sigmoid	C=1.0, kernel='sigmoid', gamma='scale'

CONCLUSION

- **XGBoost** and **LightGBM** classifiers as the most proficient options among those considered.
- Both classifiers demonstrate exceptional performance across key metrics, achieving the highest accuracy, precision, recall and F_1 score values.
- XGBoost achieves remarkable results with an accuracy of 99.02 %, precision of 98.35 %, recall of 97.74 %, and F_1 score of 98.04 %, closely followed by LightGBM with comparable scores of 99.02 % accuracy, 98.31 % precision, 97.60% recall, and 97.95 % F_1 score.
- These findings show the effectiveness and reliability of both XGBoost and LightGBM for classification tasks, making them the optimal choices for achieving high predictive performance and robustness in similar scenarios.

FUTURE WORK

- Exploring diverse feature selection approaches to enhance the versatility of our analysis.
- Extending the investigation by applying various deep learning approaches such as CNN, RNN, GNN etc
- Extending the investigation by applying genetic algorithm approaches like crossover, mutation etc for feature selection and feature creation, broadening the scope of the research and potentially uncovering new insights.

CODE AVAILABILITY

All code of current implementation is uploaded in GitHub repository:

https://github.com/bishwajitprasadgond/MalwareClassification

REFERENCES

1. Skoudis E, Zeltser L. Malware: Fighting malicious code. Prentice Hall Professional; 2004.
2. Dabas, N., Ahlawat, P., and Sharma, P., 2023. "An effective malware detection method using hybrid feature selection and machine learning algorithms". Arabian Journal for Science and Engineering, 48(8), pp. 9749–9767.
3. Dabas, N., Sharma, P., et al., 2023. "Malanalyser: An effective and efficient windows malware detection method based on api call sequences". Expert Systems with Applications, 230, p. 120756.
4. Sharma, Prabha. "Windows Malware Detection using Machine Learning and TF-IDF Enriched API Calls Information." In 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), pp. 1-6. IEEE, 2022.
5. Li C, Cheng Z, Zhu H, Wang L, Lv Q, Wang Y, Li N, Sun D. DMalNet: Dynamic malware analysis based on API feature engineering and graph learning. Computers & Security. 2022 Nov 1;122:102872.

REFERENCES

6. Tahir R. A study on malware and malware detection techniques. International Journal of Education and Management Engineering. 2018 Mar 1;8(2):20.

ANY QUESTIONS??

thank you!