Mitigating Malware Threats on Emerging Technology: A Machine Learning Approach

AANMAR ABDOU SALAM[1], MD. ABDUL BASED[1], MOHAMED ISLAM HOUSSAM[2], MOHAMMAD SHORIF UDDIN[3]
[1]Department of Computer Science and Engineering,
Dhaka International University,
Dhaka,
BANGLADESH

[2]Department of Electrical and Electronic Engineering,
Islamic University of Technology,
Gazipur,
BANGLADESH

[3]Department of Computer Science and Engineering,
Jahangirnagar University,
Dhaka,
BANGLADESH

Abstract: - Malicious programs and malware threats lead to a substantial vulnerability and pose a fundamental problem. Nowadays, smart devices are becoming more common, and consequently, the risk of malware intrusion is highly observed. This paper presents a comprehensive exploration from the initial to the final phase of an effective strategy and the deployment of a model to detect malware efficiently. The proposed Mitigating Malware Threats on Emerging Technology framework "MMTET" will help mitigate the risk of intrusion. This study explores the complexity of handling datasets. Random Forests and Decision Trees serve as machine learning algorithms for training and testing. Starting with a data collection method to obtain relevant parameters, this paper highlights the importance of well-curated datasets in training using effective machine learning models. Data analysis follows a statistical approach, and the visualization tools are used for identifying inherent biases, imbalances, and trends in datasets. For boosting the dataset's quality, feature engineering and selection take a central stage to balance the data with new methodologies and detect relevant features with correlation analysis. Experimental result shows that Random Forest has the best performance compared to other methods obtained from different algorithms, with accuracy 98.30%.

Key-Words: - Decision Tree, Random Forest, Dataset, Machine Learning, Threats, MMTET.

Received: July 12, 2023. Revised: February 19, 2024. Accepted: April 13, 2024. Published: May 15, 2024.

1 Introduction

The term "malware", [1], is a combination of malicious files, or code; it can be defined as a form of software that is intentionally used by cybercriminals to obtain unauthorized access to private or sensitive data. This general term refers to a wide range of malicious software variations, all of which are designed for particular, frequent evil objectives.

Numerous well-known threats of malware are ransomware, which encrypts important files and requests a ransom to unlock them; Trojans, which secretly enters a system by standing as a trustworthy application; and spyware carefully gathers private data without the user's awareness. Because malware is constantly changing, it is crucial to have strong cybersecurity methods and measures in place to prevent these threats and protect digital assets from the constant hazards associated with using the internet.

The traditional technique uses a known list of Indicators Of Compromise (IOCs), such as fill hashes, specific bytes' sequences, or network attack patterns, to identify and eliminate malware before infiltrating a system. The signature-based detection method was the most used technique to identify

malware, [2], in the field of cybersecurity. It works by locating patterns or signatures linked to dangerous software. However, as malware has become exponentially more prevalent, this once-dominant approach has run into problems and has become extremely inefficient. The main problem is that it can only identify malware signatures that have been observed, which makes it inefficient for protecting against new potential threats that have never been observed before.

The weakness of signature-based detection becomes more evident since the cybersecurity profile changes dynamically, requiring an investigation and the application of more intelligent and proactive measures to counter the increasing variety of cyber threats. Signature-based detection has been the way of detecting malware until the 1990s. Then machine learning-based malware detection techniques were developed and improved, [3]. Support Vector Machines (SVM), Random Forests (RF), Logistic Regression (LR), Naïve Bayes (NB), and Adaboost , [4], are part of the machine learning methodology proposed to be useful in malware detection and classification techniques, achieving higher performance and accuracy.

The problem nowadays is not just understanding how malware evolves but also understanding effectively how processing large and diverse datasets works and to extract useful information. The first is unable to emphasize the significance of data processing in malware detection. Thoroughly analyzing and preparing a dataset is essential before implementing machine learning models to mitigate, detect, and prevent malware attacks. The complexity of malware behavior and the variety of possible attack avenues necessitate a careful approach to data preparation. This entails correcting problems that can greatly affect the effectiveness of detection algorithms, such as imbalances, biases, and missing or irrelevant data. Considering the above issues, Mitigating Malware Threats on Emerging Technology framework “MMTET” is proposed in this paper that will help mitigate the risk of intrusion.

In this paper, Section 2 summarizes the literature review on the existing methodologies proposed by different authors. Section 3 discusses the methodology, and Section 4 describes different models that are applied to this work. The analysis is done in Section 5, and Section 6 summarizes the findings and the importance of advancing cybersecurity solutions for emerging technologies.

2 Related Work

The DREBIN, a detection system is presented in [5], which allows the identification of malware applications on smart devices. In DREBIN, the authors consider a dataset of 131,611 applications including malware software. Mainly, they apply a broad statistical analysis to extract features from different sources and analyse them in an expressive vector space. The DREBIN worked on 123,453 applications and 5,560 malware samples, and the detection rate was 93,9%.

The Internet of Medical Things (IoMT) method is proposed in [6], to categorize and identify malware. The framework used multidimensional Deep Learning (DL) approaches for an optimal feature analysis to detect malware and perform a classification into categories based on the byte representation of the executable and linkable file. For an excellent outcome, different methods were used for their framework, including Convolutional Neural Network (CNN), bidirectional Long Short-Term Memory (LSTM), and other model for IoMT malware classification comparison. Two separate datasets, Big-2015 datasets, and CDMC-2020-IoMt-Malware were used to evaluate the performance of the framework. D TensorFlow was used on the back end and Keros for the front-end library, and scikit-learn for Machine Learning (ML) algorithm implementation. IoMT framework obtains 95% accuracy which is better than the other DL approaches like RNN, LSTN, GRU, CNN, and bidirectional LSTM. It also gives a better performance in terms of precision (96%), recall (95%), and F1-score (95). The result demonstrates the effectiveness of their framework for malware detection and classification.

Microsoft malware dataset is used in [7], for training and testing of Light Gradient Boosted Machine (LGBM) technique to detect malware attack on Microsoft cloud as a framework. The LGBM decision tree model is used for classification and regression using the AutoML tool and another model to enhance prediction accuracy. Based on that study, LGBD was the perfect model to use for evaluating the framework in [7], on large data. An outcome of 67,78% of F1-score and 66,18% accuracy revealed that the suggested methodology was more accurate in predicting malware than AutoAI and other models.

The authors propose an innovative security framework ‘MobiSentry’ in [8], for detecting malware and mobile categorization with a substantial dataset comprising 184,486 benign and 21,306 malware instances in Android devices.

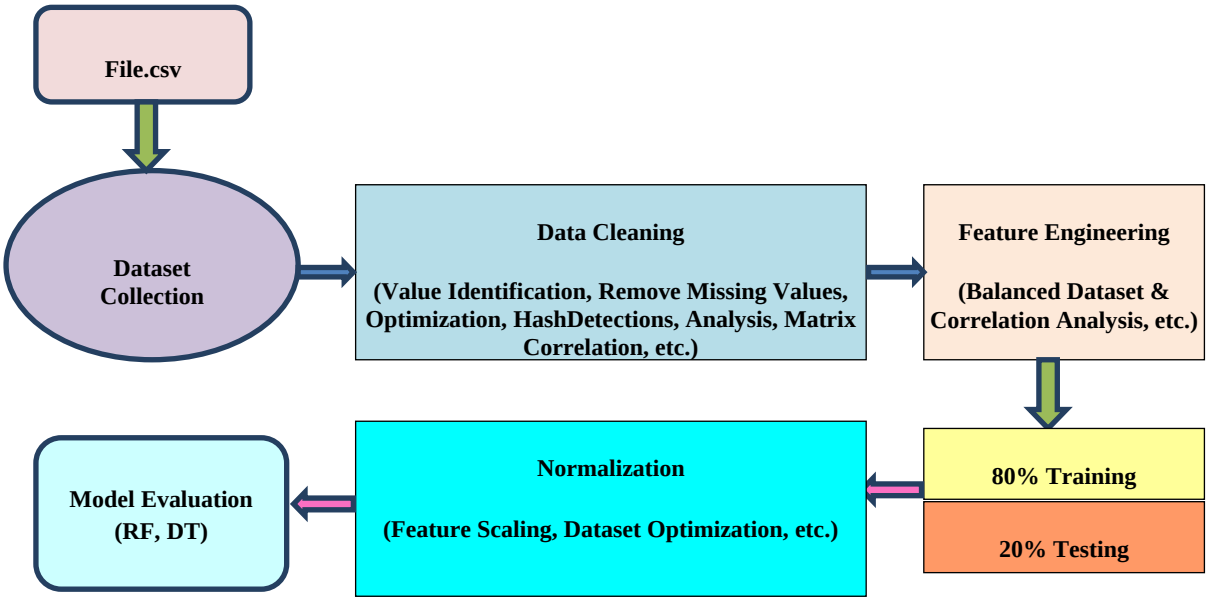

Fig. 1: MMTET Framework

For an extensive examination of their methodology, they split up the dataset into two phases: 80% for training and 20% for testing. For experimentation, five different models have been put into consideration including KNN, RF, SVM, Ada, and GBM for training and testing. The GBM classifier gives a satisfactory result accuracy rate of 96.76% showing the out-performance of the classifier compared to the other algorithms.

Some publications on the detection of malware particularly for the Internet of Things (IoT) are available in [9], [10] and [11]. However, in this paper a new framework MMTET is presented to mitigate the risk of intrusion. The MMTET uses a huge volume of data (600,250) and achieves 98.30% accuracy.

3 Proposed Methodology

The workflow of the proposed MMTET framework for mitigating malware threats on emerging technologies is shown in Figure 1.

3.1 Data Collection and Cleaning

3.1.1 Data Collection

The dataset that has been examined in this paper was taken from Kaggle, [12]. The dataset consists of a very large collection of machine-specific data where each row is uniquely identified by a machine identifier. The initial dataset was very large, with 83 columns and 8 million records. However, a subset of the data was chosen to allow satisfactory processing within the constraints due to practical considerations connected to computational resources. The final dataset which was used to train the machine learning models was reduced down to a higher quality number of records, resulting in a subset of 600,250 records.

3.1.2 Data Cleaning

The dataset required extensive cleaning since it had significant biases when first gathered. There was an extreme incidence of missing values in several columns of the dataset that were selected including '*PuaMode*', *'Census_ProcessorClass'*, *'Census_IsFlightingInternal'*, and '*DefaultBrowsersIdentifier*', often over 90%. A strict threshold of 35% of missing values was set to correct this problem. Columns that exceeded this cutoff were all eliminated since their effectiveness for further classification assignments was judged to be compromised by the wide absence of data.

In addition, columns with unique values in every row were found, which could confuse the machine learning models that are used. To simplify the dataset, these columns were subsequently removed one at a time. After all of this meticulous cleaning, the dataset was improved with columns that included a very limited number of missing values, which are addressed in later steps.

3.2 Data Analysis

Python's Pandas package was used for reading and loading the dataset. Then *'pd.read_csv'* method is used to import the contents of the csv file in the PC defined by *'train_path'* variable to the Pandas. The data frame *'train_df'* is used for efficient data analysis and processing, and it provides an organized and accessible format for subsequent data analysis tasks and training models by allocating the supplied data to *'train_df'*.

Next, to improve the efficiency of the dataset in the training phase a technique for memory optimization is applied. A custom function, *'reduce_mem_usage'* is used to reduce the memory footprint of the DataFrame, which is important when working with a large dataset. Following that, the *Seaborn library* was used to create a categorical plot that showed the number of observations for each category of the variable '*HasDetections*.' The visualization provides an important detail into the distribution of the target variable, increasing the comprehension of the dataset and laying the groundwork for additional investigation.

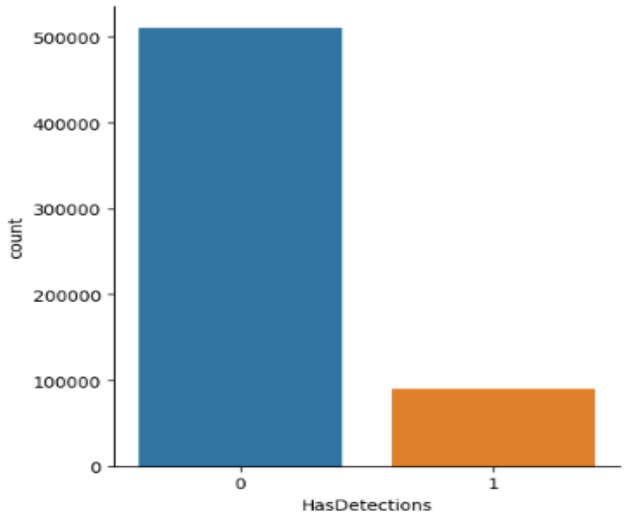

Fig. 2: Plotting Detection

As shown in Figure 2, the *'HasDetections'* variable reveals 510237 samples as detected and 90012 instances as malware detected. Graphically, a distribution of the *'IsBeta'* variable is observed, demonstrating a significant class of imbalance with 600244 cases labeled as 0 and a tiny count of 5 instances labeled as 1. The distribution was then quantified using *'train_df.IsBeta.value_counts()'*.

Figure 3 shows the '*DefaultBrowsersIdentifier*' variable. This variable is the default ID for the machine, to visualize the occurrences of the top ten (10) most common identifiers.

With a large number of unique values, the top frequent identifiers were considered which are important because these allow more focus on the examination of the most influential identifiers, guiding subsequent research and decision-making processes.

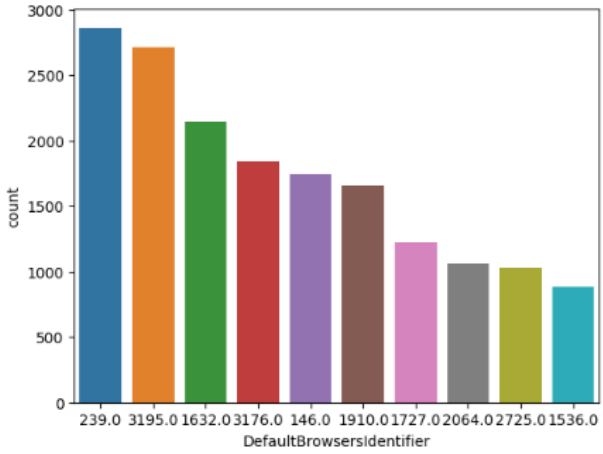

Fig. 3: Top 10 Most Frequent Default Browsers Identifiers Distribution

The distributions of key variables are visually interpreted about the detection outcomes. The first plot analysis '*SmartScreen*,' focusing on the top 5 occurrences is shown in Figure 4. The '*HasDetections'* clarifies possible correlations between '*SmartScreen*' variables and the results of detection. Similarly, '*Census_OSBuildNumber'*, '*AVProductsInstalled*', and '*AVProductsEnabled*' are analyzed with the five most frequent occurrences. In addition, integrating the '*HasDetections*' and '*color.Various'* short illustrations provide useful data about the frequency and correlations of various variables. This improves the capacity to identify patterns and dependencies pertinent to the main research objectives.

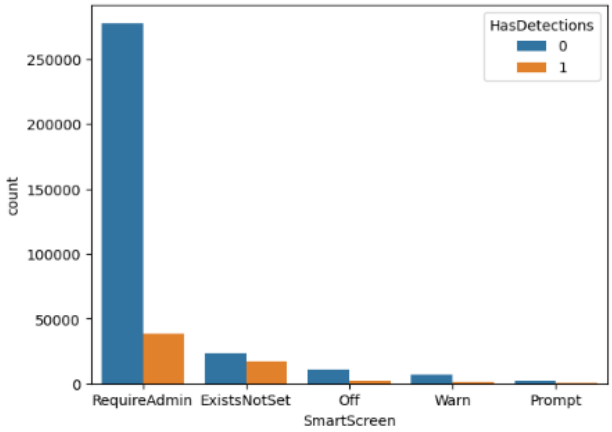

Fig. 4: Top 5 SmartScreen Occurrences with Detection Status Insights

3.3 Feature Engineering

The dataset's properties are examined after data interpretation. It has been observed that out of 600250 records, 510,237 are legitimate and 90,012 are infected files. This indicates an extremely imbalanced distribution. Hence, data balancing is done as described in the following.

3.3.1 Data Balancing

Mitigating the imbalance dataset, an oversampling strategy has been adopted by using Python's *RandomOverSampler* module. The imbalanced data have been reduced by duplicating instances from the minority class. This systematic technique generates a perfectly balanced dataset shown in Figure 5, laying the base for more resilient and dependable model training.

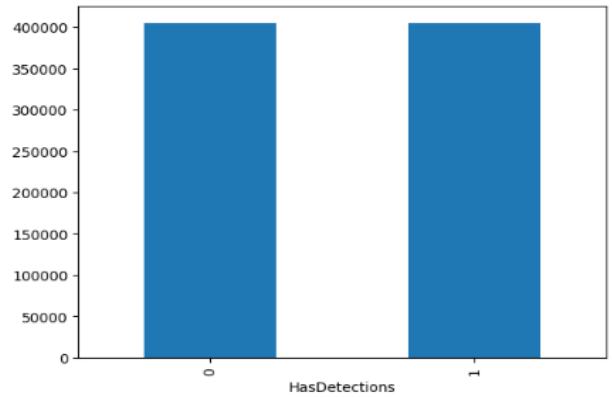

Fig. 5: Balanced dataset

3.3.2 Feature Selection and Correlation Analysis

For a powerful model building, an important step involves a feature selection process through correlation analysis. *'Seaborn library'* is applied to the '*heat map'* for visualization to clarify the correlations within the 83 columns in the dataset. Because it could be difficult to visualize the complete dataset at once, this procedure selectively started by plotting a subset, particularly with 20 columns. However, for a better demonstration of the visualization, the correlation between 1~ 5 columns is shown in Figure 6.

For example, *'IsSxsPassiveMode'* and *'RtpStateBitfield',* demonstrate a strong association, according to the initial analysis. A meticulous decision-making procedure was followed to maximize model efficiency and prevent redundancy. Figure 7 clearly shows that the column *'RtpStateBitfield'* displayed a non-uniform distribution with six distinct values. This led to a prudent deletion, choosing instead to keep the 'IsSxsPassiveMode' column with its two possible values as shown in Figure 8.

The feature evaluation was completed from the 20th to 35th features using the same methodology for the 64 features chosen from the original group of 83, ensuring that there was no association at all between them.

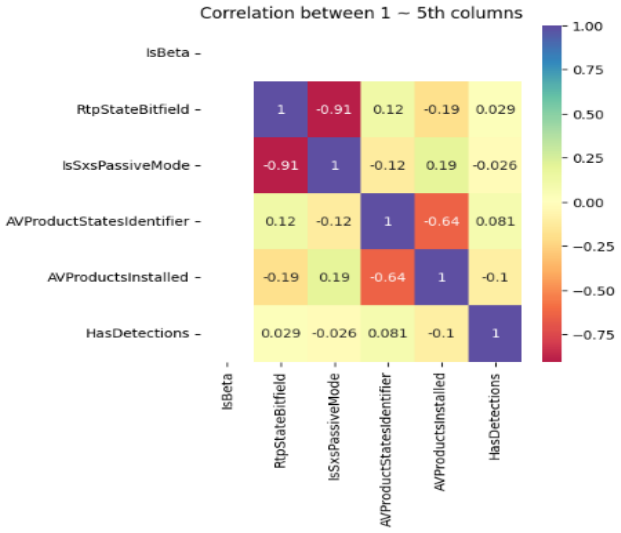

Fig. 6: Matrix correlation between features

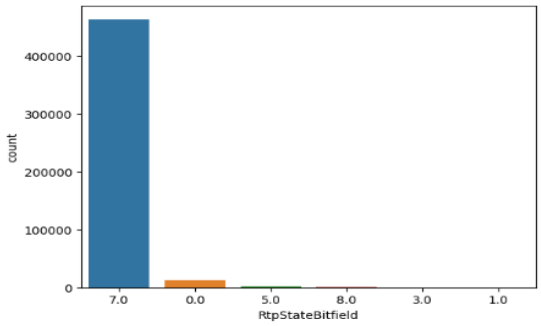

Fig. 7: Non-uniform distribution of RtpStateBitfield

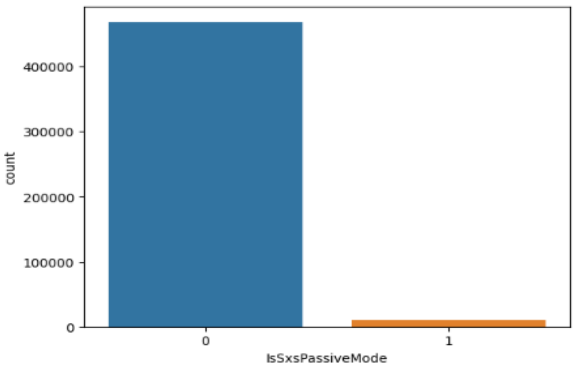

Fig. 8: Distribution of IsSxsPassiveMode

3.4 Normalization

For an efficient training model, normalization is an essential phase to optimize the data, and for that *StandardScaler* module was used. Next, by subtracting the features from their values and dividing the result by the standard deviation, the *StandardScaler* module normalizes the data by

implementing the mathematical transformation, and any divergent scales that may have existed among the original features are eliminated to ensure that all features share a consistent scale.

4 Models

In this paper, after raising the data quality and completing normalization for better performance of the proposed model, the data are split into training and testing phases. 80% of the data have been used to train and the 20% remaining have been used to test purposes.

In the context of regression and classification problems, supervised learning methods like Random Forest (RF) and Decision Tree (DT), KNN, OLGBM, and XGBoost are applied. However, this paper presents the results with RF and DT since these two methods outperform the other methods in terms of performance.

RF is a tree-based classifier that combines multiple classifiers into one using ensemble learning to answer progressively complex problems. In the training phase, multiple internal decision trees are built by RF where each one gives its output. Based on the majority vote, RF chooses the final decision for the problem given. RF does not consider every feature when it builds a tree, so each tree may differ from the other and reduce feature space.

Based on an attribute, every node in the DT denotes the test, each branch indicates the test's result and every label has the class label. DT classifies instances in descending order from the root node of the tree to the last node that offers the instances' classification.

4.1 Performance Metrics

In the MMTET framework, metrics like accuracy, precision, recall, and F1-score are employed. The confusion matrix is characterized by True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The measurements for these are calculated using the following Equations 1 through 5, [13].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \times 100\% \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (3)$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN} \quad (4)$$

$$True\ Positive\ Rate\ (TPR) or\ Recall = \frac{TP}{TP+FN} \quad (5)$$

4.2 Random Forest

For better accuracy, Random Forest has different parameters that can be used for training the model. In the experiment of this paper, the parameters that provide the best results are tested with different values. In the training phase with the default values, the experiment achieved an accuracy of 98,15%, an FPR of 0.028, an F1-score of 0.98 for the malware and legitimate files, and then a TPR value of 0.991. The AUC for Random Forest is shown in Figure 9.

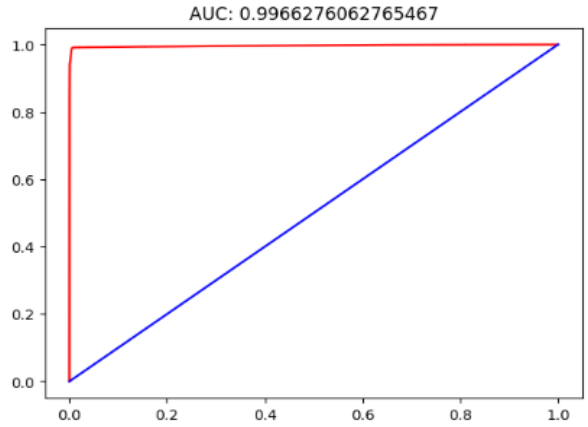

Fig. 9: AUC score with Random Forest Model

Then the parameters are tested with different values and verified if the change could obtain a better outcome. *N_estimators*, *Min_sampler_split*, *Nin_samples_leaf*, *Max_features* and *Boostrap* are the parameters modified for better performance. This change ended up getting better results than the default ones. The parameter *N_estimators* that defines the number of trees was set to 400. *Max_feature* defines the maximum number of features that Random Forest is permitted to try in an individual tree set to Auto. *Max_depth* is set to 2 to control the depth the tree should grow. The *Min_samples_split* and *Min_sample_leaf* both define the minimum number of features needed to split a node were set to 1 and 2 respectively, then the *Bootstrap* was set into Auto.

The result shows the highest accuracy of 98.30%, an FPR of 0.025, an F1-score of 0.98 for malware and valid files, and a TPR of 0.991.

4.3 Decision Tree

The same approach was applied for training with Decision Tree. First, the experiment was done with the default values, and then different values for each parameter are changed and verified to observe the

result. After the experiment, it was found that the result in the default values was better than the changed values. To ameliorate the result, "Max_depth" was set to 20 for controlling the maximum depth of the tree, and the max_features was specified as 'auto' to control feature selection within each tree. Additionally, min_samples_leaf was set to 40, which lead to an Accuracy of 63,13 % which is very low compared to the default one.''
Finally, the result achieved an accuracy of 91.41%, an FPR of 0.0164, an F1-score of 0.92 for malware and 0.91 for begin files, and a TPR of 0.99 (in default setting) under the standard value.

The Area Under Curve (AUC) for the Decision Tree and Random Forest is shown in Figure 10. The experimental results of the Decision Tree model and Random Forest model are shown in Table 1. The AUC score offers important insights into the classifier's performance. An indicator frequently used to assess the effectiveness of binary classification algorithms is AUC. It shows the likelihood that a randomly chosen instance will be ranked by the model.

In Table 1, the AUC score using the Decision Tree model and Random Forest model provides an additional indicator of the classifier's success. Greater discrimination between positive and negative instances by the model is indicated by a higher AUC value. A thorough grasp of this model's capacity may be obtained to discriminate between malicious and benign files over a range of criteria by examining the AUC score in Figure 10. AUC values of 0.983 and 0.914 indicate a great performance of the models.

Table 1. Accuracy, precision, recall, and F-1 score of RF and DT

	Acc	Preci_0	Preci_1	Recall_0	Recall_1	F1_S_0	F1_S_1
RF	0.983	0.99	0.97	0.97	0.99	0.98	0.98
DT	0.914	0.99	0.86	0.84	0.99	0.91	0.92

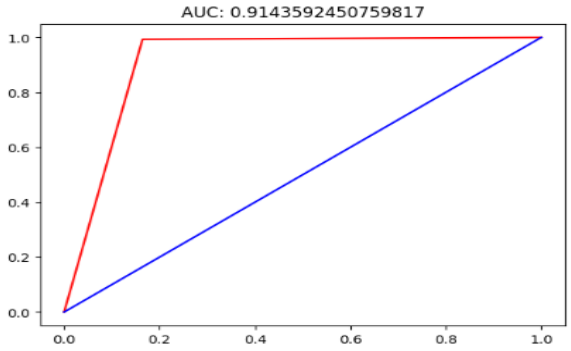

Fig. 10: AUC score with Decision Tree Model

5 Analysis

In this section, a comparative analysis is done between the proposed framework of this paper and other approaches to detect malware using different algorithms. The DREBIN [5], IoMT [6], LGBM [7], and MoBilSentry [8], approaches are considered. The comparison was made on the volume of data, the accuracy, the F1-score, and TPR with FPR records founded on the optimal result.

As shown in Figure 11, Random Forest has largely a better accuracy than the other frameworks and Decision Tree has competed with other approaches. Though DREBIN gets a better accuracy (93.9%) than MMTET RF (98.3%) and MMTET DT (91.41%), DREBIN applies a dataset with a total of 131611 samples. In MMTET, the size of the used dataset is 600250. Similarly MoBilSentry achieves better accuracy 96.76% than MMTET DT with 205792 records in the dataset. Again the dataset size is much lower than that of MMTET.

IoMT gets an accuracy of 95% with fewer size of datasets than MMTET and the same happens with LGBM.

The effectiveness of malware detection with an accuracy rate of 98.30% in detecting malware threats on a large dataset of over 600,000 records, demonstrates the effectiveness of the framework. It presents a comprehensive data pre-processing strategy that elaborates on data cleaning and feature engineering for high-quality data for model training. The selection of Random Forest (RF) and Decision Tree (DT) classifiers demonstrate a robust model selection based on optimization and evaluation, and oversampling strategies for imbalanced data to balanced data enhances the robustness of the trained model.

The study works on a specific dataset from Kaggle which may introduce biases when working with a huge dataset. Computational resources may influence the model affecting the scalability of the framework.

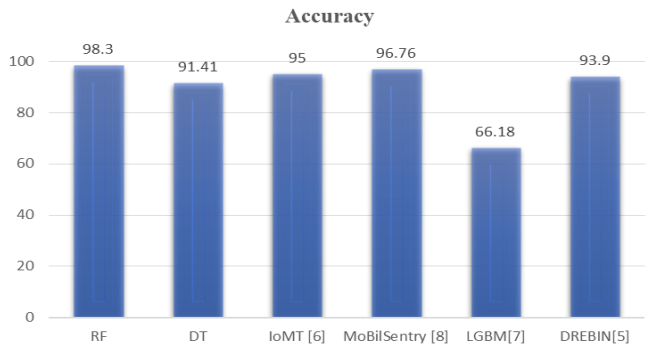

Fig. 11: Comparison of Results

The existing literature's approaches are designed for detecting malware within a limited dataset. The MMTET framework has been trained on a large and higher-quality dataset, which contributes to a comprehensive analysis and well-trained model. MMTET framework prefers the use of RF and DT classifiers due to their superiority in terms of performance in the experimental setup and excels by handling larger datasets while maintaining a better performance than the others.

6 Conclusion

In this paper, an effective framework MMTET to mitigate malware threats on emerging technology is presented. MMTET is proven as an efficient model for detecting malware threats better than the other approaches investigated in this paper such as DREBIN, IoMT, LGBM, and MoBilSentry approaches. These approaches have proposed different ways of detecting malware threats. However, their accuracies and strategies do not compete with the framework proposed in this paper, which provides a higher accuracy of 98.30% in a very large dataset using the Random Forest model. Finding an efficient way to mitigate cybersecurity threats on emerging technology will be a good future work as cybersecurity threats are a real danger for multiple users.

References:

[1] A. P. Namanya, A. J. Cullen, I. Awan, & J.P. Diss (2018). "The World of Malware: An Overview". *IEEE 6th International Conference on Future Internet of Things and Cloud*, Barcelona, Spain, September 2018, pp. 420-427. DOI: 10.1109/FiCloud.2018.00067.

[2] J. Scott (2017). "Signature Based Malware Detection is Dead". *Institute for Critical Infrastructure Technology*, February 2017.

[3] Y. Baychev, & L. Bilge, (2018). "Spearphishing Malware: Do we really know the unknown?" *In C. Giuffrida, S. Bardin, G. Blanc (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2018*. Lecture Notes in Computer Science, Vol. 10885. Springer, Cham, https://doi.org/10.1007/978-3-319-93411-2_3.

[4] I. Narendra, I. Dawar, N. Kumar, S. Negi, S. Pathan, & S. Layek (2023). "Text Categorization using Supervised Machine Learning Techniques". *2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU)*, Riyadh, Saudi Arabia, 2023, pp. 185-190, DOI: 10.1109/WiDS-PSU57071.2023.00046.

[5] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, & K. Rieck (2014). "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket". *Network and distributed system security symposium (NDSS)*, September 2017, Vol. 14. pp.23-26.

[6] V. Ravi, T. D. Pham, & M. Alazab (2023). "Attention-Based Multidimensional Deep Learning Approach for Cross-Architecture IoMT Malware Detection and Classification in Healthcare Cyber-Physical Systems". *In IEEE Transactions on Computational Social Systems*, Vol. 10, no. 4, pp. 1597-1606, Aug. 2023, DOI: 10.1109/TCSS.2022.3198123.

[7] M. Sokolov, & N. Herndon (2021). "Predicting Malware Attacks using Machine Learning and AutoAI". *ICPRAM 10th International Conference on Pattern Recognition Applications and Methods*. pp. 295-301. DOI: 10.5220/0010264902950301.

[8] R. Bungfei, L. Chuanchang, C. Bo, G. Jie, & C. Junliang (2018). "MobiSentry: Towards Easy and Effective Detection of Android Malware on Smartphones". *Mobile Information Systems*. Vol. 2018, Article ID 4317501, pp. 1-14, DOI: 10.1155/2018/4317501.

[9] QD. Ngo, HT. Nguyen, VH. Le, & D.H. Nguyen (2020). "A survey of IoT malware and detection methods based on static features". *ICT Express*. Vol. 6. Issue: 4, pp. 280-286, DOI: 10.1016/j.icte.2020.04.005.

[10] HT. Nguyen, QD. Ngo, & VH. Le (2020). "A novel graph-based approach for IoT botnet detection". *International Journal of Information Security*, Vol. 19. pp. 567–577, DOI: 10.1007/s10207-019-00475-6.

[11] R. Chagantia, V. Ravib, & TD. Pham (2022). "Deep Learning based Cross Architecture Internet of Things malware Detection and Classification". *Computers & Security*, May 2022. Vol. 120, No. 102779, pp. 1-22, DOI: 10.1016/j.cose.2022.102779.

[12] Kaggle. Microsoft Malware Prediction, [Online]. https://www.kaggle.com/c/microsoft-malware-prediction/overview (Accessed Date: November 30, 2023).

[13] H. Dalianis (2018). Evaluation Metrics and Evaluation. *Clinical Text Mining. Chapter 6*, pp.45-53. Springer, Cham, DOI: 10.1007/978-3-319-78503-5_6.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.