International Journal on Computational Modelling Applications
Vol. 02, Iss. 04, S. No. 04, pp. 32–40, Dec 2025
ISSN (Online): 3048-8516

Received: 17 Dec 2025, Accepted: 30 Dec 2025, Published: 04 Jan 2026
DOI: https://doi.org/10.63503/j.ijcma.2025.208

Research Article

An Optimized LightGBM Framework for Static Malware Classification Based on Windows Portable Executable File Attributes

Amit Verma
School of Computer Science, University of Petroleum and Energy Studies Dehradun, Uttarakhand, India
Senior Member, IEEE
Email: amit.uptu2006@gmail.com
*Corresponding Author: Amit Verma, amit.uptu2006@gmail.com

Abstract

Malware poses a significant threat to the digital world, as weak detection systems can enable attackers to steal data or corrupt critical files. Numerous researchers have contributed to this domain by developing highly accurate malware classification systems using various machine learning and deep learning techniques. According to the literature, most machine learning models have been extensively explored for accurate malware classification, with LightGBM consistently demonstrating superior performance in most cases. However, there remains scope for developing more efficient, lightweight, and robust malware classification models through machine-learning-based ensemble techniques. In this paper, we conduct experiments using the balanced and extensive EMBER 2018 dataset, comprising 799,876 samples with 2,382 features, to ensure robust training and obtain an unbiased model. We fine-tune the LightGBM model and additionally implement various machine learning models including Random Forest (RF), ExtraTrees Classifier (ET), XGBoost Classifier (XGB), and a soft-voting ensemble stacking RF, ET, and LightGBM classifiers. Our results show that the proposed optimized fine-tuned LightGBM model outperforms other approaches, achieving an accuracy of 96%.

Keywords: machine learning, LGBM, security, malware

1 Introduction

Digitalization has significantly improved the convenience and efficiency of modern life; however, it has also introduced substantial risks related to data security and privacy. As digital systems have become more widely adopted, various forms of malware have emerged, capable of infiltrating computer systems to corrupt or steal sensitive information. Certain malware variants can even encrypt an entire system's data, rendering it unreadable to the legitimate owner. Although numerous anti-malware tools have been developed to mitigate such threats, these solutions are often insufficient because new and more sophisticated malware variants continue to appear daily. This growing challenge highlights the need for highly accurate, generalized, and robust malware classification systems. With advancements in artificial intelligence and machine learning, many researchers have proposed effective malware classifiers using machine learning and deep learning techniques [1, 2].

Liu et al. [3] used a SNN clustering approach with 20,000 samples for malware classification and achieved 98.9% accuracy. However, the limited sample size increases the risk of overfitting [4] and may result in a biased model. In [5], the authors used a small dataset of 400 samples and implemented popular machine learning algorithms including SVM [6], random forest [7], decision tree [8], naive Bayes [9], and AdaBoost [10]. The SVM approach outperformed the other algorithms. In [11], the authors implemented various machine learning algorithms and ensemble models, proposing a weighted-voting based ensemble approach that achieved an accuracy of 95%. The dataset used for experimentation contained only 141 features, suggesting potential for improvement by increasing

the number of features to produce a more efficient model. In [12], the authors discussed the computational challenges of the large EMBER dataset. Considering these challenges, they reduced the features from the original 2381 to 384 features. After exploring multiple machine learning approaches, the LightGBM ensemble technique showed the best result with an accuracy of 97.52%. In [13], the authors implemented nearly all popular machine learning approaches for malware classification using the EMBER dataset. However, only 49 out of 2381 features were used for the training process. Based on the results, random forest and extra trees models achieved equal and highest accuracy of 97%. In [14], the authors used AutoML for deep learning models with various datasets including EMBER 2018. For the EMBER dataset, the proposed work achieved an AUC of 0.98 and an F1-score of 0.96. In [15], the authors implemented various popular machine learning algorithms on the EMBER 2018 dataset. The results show that the LightGBM model outperformed others with an accuracy of 94.8%. In [16], the authors compared two AutoML frameworks using the EMBER 2018 dataset and found that LightGBM provided the best results with 90% accuracy. In [17], the authors implemented a convolutional neural network (CNN) on the EMBER 2018 dataset and reported that the basic CNN achieved 93.45% accuracy. In [18], the authors implemented an artificial neural network (ANN) using the raw EMBER 2018 dataset and compared its performance with popular machine learning algorithms such as logistic regression, random forest, LightGBM, KNN, and SVM. It was found that ANN outperformed others with an accuracy of 95%. In [19], the authors used EMBER 2017 and 2018 datasets for malware classification using deep learning. The proposed work achieved accuracies of 97.53% and 94.09% for EMBER 2017 and 2018 datasets, respectively. However, the model was trained for only 10 epochs with a learning rate of 0.01.

Based on the reviewed literature, we conclude that the LightGBM model has provided better results for malware classification using the EMBER dataset. However, there remains scope for hyperparameter tuning to optimize LightGBM rather than using the base model. In this paper, we fine-tune hyperparameters such as the number of estimators and learning rate. The optimized LightGBM has shown improvement across nearly all evaluation metrics. The main contributions of this study are as follows:

- Efficient fine-tuning of the LightGBM model, which improves its performance.
- Implementation of five tree-based ensemble models with comparative performance analysis.
- Demonstration of the performance of a stack-based ensemble model using tree-based machine learning algorithms.

The remainder of the paper is organized as follows: details of dataset preprocessing are presented in Section 2, the methodology is discussed in Section 3, performance evaluation and comparative analysis are presented in Section 4, and finally, conclusions and future work are discussed in Section 5.

2 Data set description

For experimentation, we use a cleaned version of the benchmark EMBER 2018 dataset in parquet format [20]. The original dataset was pre-split into training and testing sections, each containing three classes: clean (0), malware (1), and unknown (-1).

First, the training and testing subsets are concatenated. During this process, rows with unknown labels are removed to maintain binary classification and avoid ambiguity. The resulting labels are:

- **0** - Clean
- **1** - Malware

After preprocessing, we obtain a complete binary dataset with no unknown labels, as shown in table 1.

The process of preparing the dataset is shown in the below algorithm 1.

Table 1: Class-wise distribution of samples in the dataset.

Label	Count
Clean file(0)	399,976
Malware file(1)	399,900

Algorithm 1 Dataset Preparation Pipeline

Require: Train file T, Test file S
Ensure: Filtered and merged dataset D
1: $X_T \leftarrow \text{readParquet}(T)$
2: $X_S \leftarrow \text{readParquet}(S)$
3: Remove unlabeled samples:

$$X_T \leftarrow X_T[\text{Label} \in \{0, 1\}], \quad X_S \leftarrow X_S[\text{Label} \in \{0, 1\}]$$

4: Merge datasets:

$$D \leftarrow \text{concat}(X_T, X_S)$$

5: Shuffle dataset (optional)
6: Return D

In lines 1 and 2, the training (T) and testing (S) parquet files are read and loaded into variables X_T and X_S, respectively. During this process, unknown labeled data are removed in line 3. The updated datasets are then concatenated to form the complete binary EMBER dataset D.

2.1 Dataset Visualization

For better understanding of the dataset, multiple visual representations are provided. Figure 1 shows that both benign and malware classes are balanced, containing nearly equal numbers of samples, which ensures an unbiased trained model.

Due to the large size of the dataset, 3,000 random samples are selected to generate a t-SNE visualization, as shown in Figure 2. The overlapping between sample points illustrates that the data are not pre-separated, necessitating an effective model for accurate malware classification.

To analyze dependencies among dataset features, a heatmap is generated using the first 20 features, as shown in Figure 3. The heatmap displays predominantly light blue colors, indicating weak correlations between features. This suggests that the dataset contains minimal duplication or similarity among features, which reduces the risk of overfitting and contributes to a more generalizable and robust model

3 Methodology

This section describes the complete workflow, from data cleaning to training the malware classification model. After preparing the data as outlined in the previous section, we implement several popular lightweight machine learning models. Addressing the research gap regarding unexplored ensemble approaches, we propose a novel structure that stacks Random Forest, Extra Trees, and LightGBM classifiers using soft voting. The workflow is visually represented in Figure 4.

3.1 Data Sampling

Experiments were performed on Kaggle using a GPU P100 to efficiently utilize the available RAM and processing speed. Given the impracticality of using the complete EMBER dataset with 799,876 samples (after removing un-

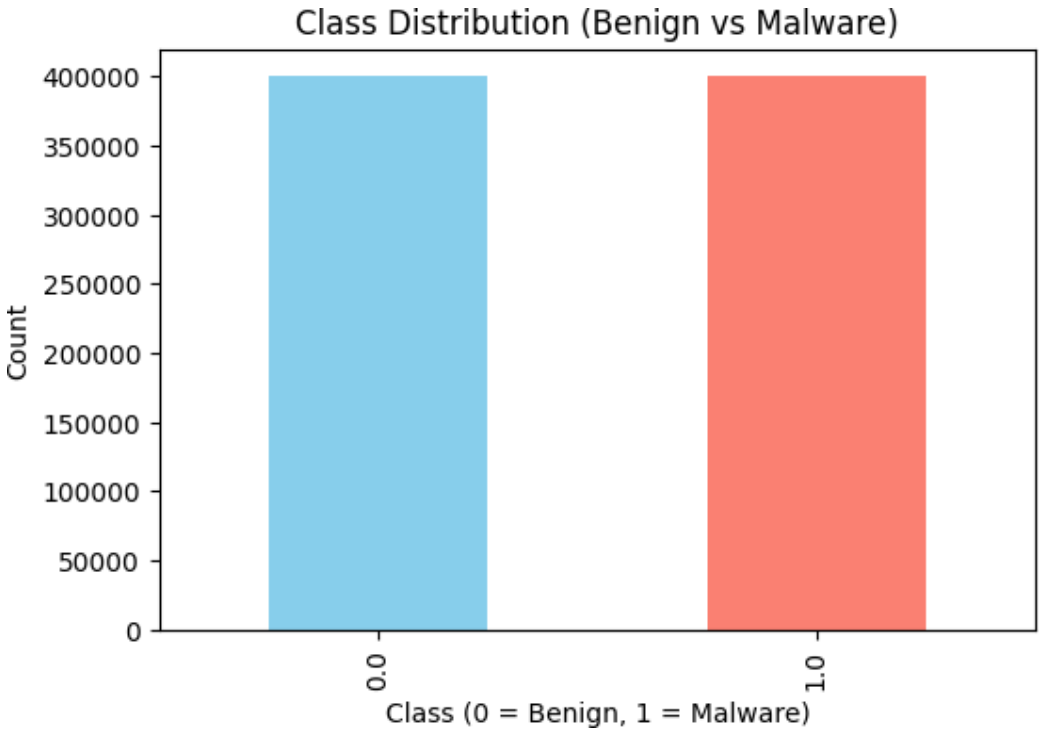

Figure 1: Class-wise distribution of samples in the dataset.

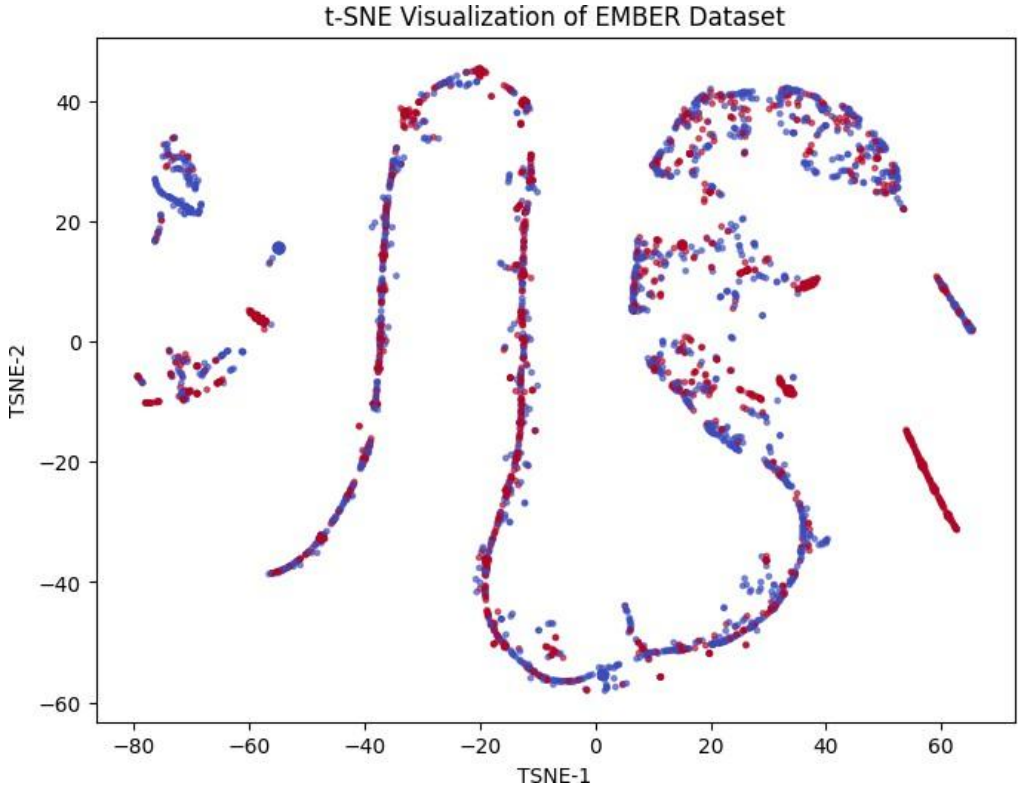

Figure 2: t-SNE to visualize the structure of the data.

known labels, as shown in Table 1), 50,000 random samples with all 2,381 features were selected for model training without compromising the training process.

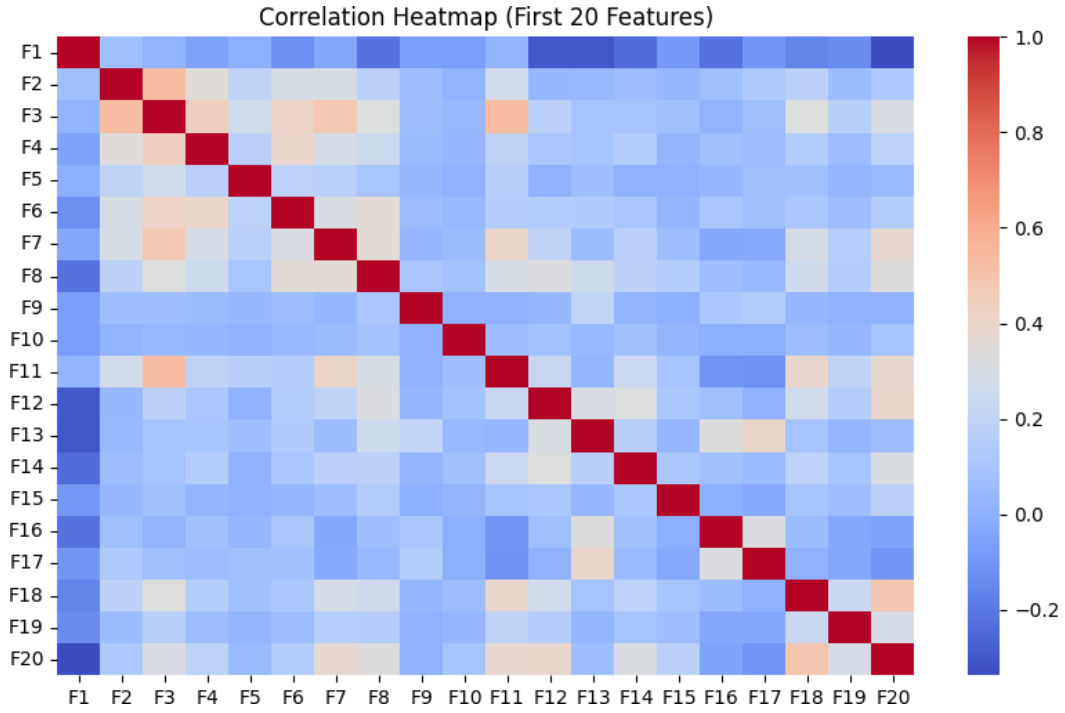

Figure 3: Heatmap showing the corelation among first 20 features.

3.2 Dataset Partitioning

For model evaluation, the dataset was partitioned in an 8:2 ratio into training and testing sets. The class-wise distribution is shown in Table 2. Stratified sampling was employed to ensure balanced class representation during training. The training data were used for model training, while the testing data were reserved for performance evaluation.

Table 2: Distribution of samples in training and testing data.

Class	Training Samples	Testing Samples
Clean file (0)	20,030	5,007
Malware file (1)	19,970	4,993

3.3 Training Lightweight Machine Learning Models

In this study, we selected tree-based machine learning models that have demonstrated strong performance in malware classification across various datasets, based on the existing literature. Models with poor performance were excluded. Four models were trained using the dataset:

- **Random Forest (RF)**: This model uses 200 decision trees with parallel processing, combining their outputs for final prediction.
- **Extra Trees (ET)**: For more generalized training, 200 extremely randomized decision trees were used to generate the final prediction.
- **LightGBM (LGBM)**: Instead of building trees independently, 300 decision trees are created sequentially with a learning rate of 0.1 to improve accuracy. Each tree learns from the misclassifications of the previous tree.
- **XGBoost (XGB)**: This gradient boosting model constructs 150 trees sequentially, each with a maximum depth of 8. A learning rate of 0.1 and row subsampling of 80

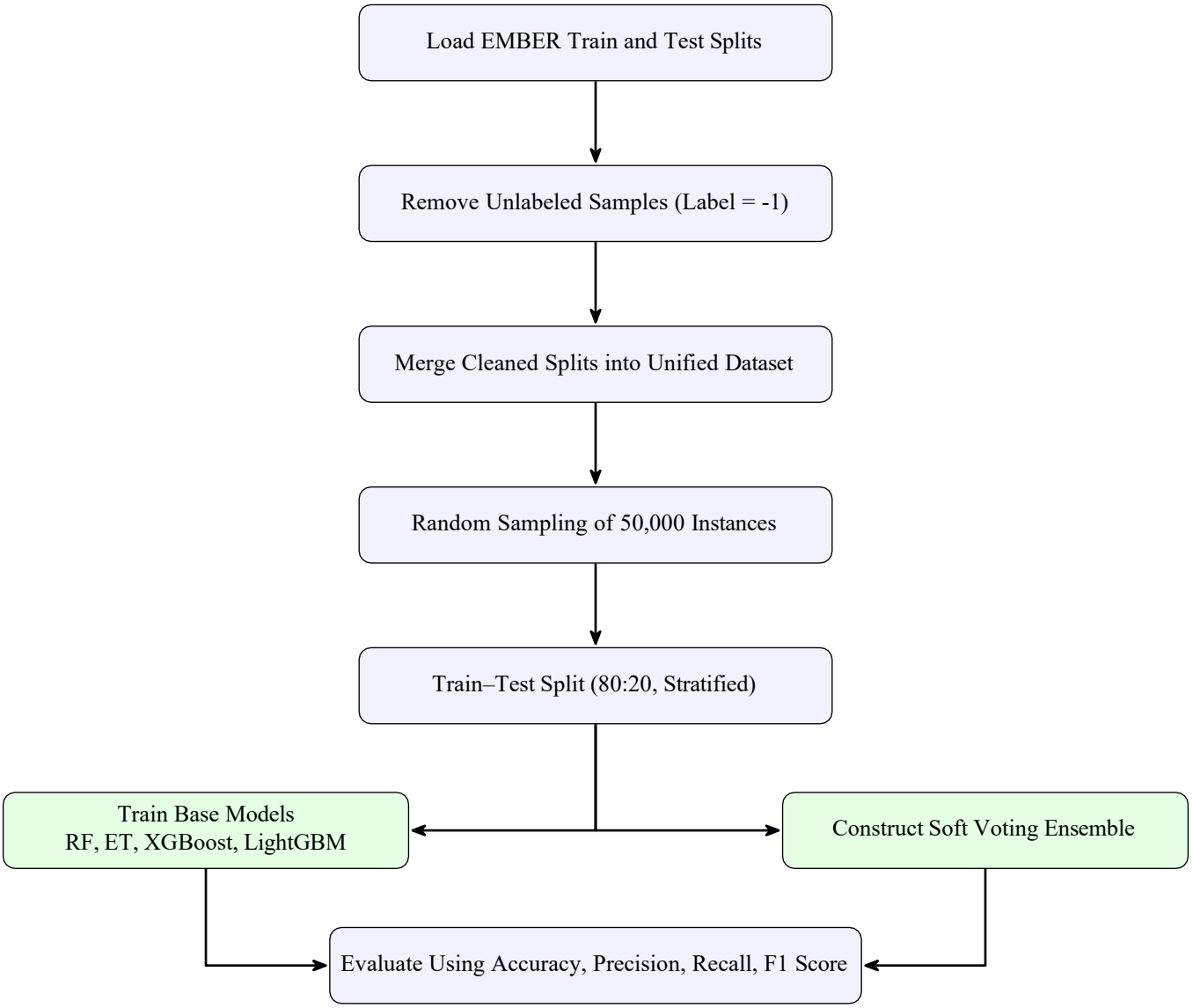

Figure 4: Workflow diagram for dataset preprocessing, model training, ensemble creation, and evaluation.

All models were trained using the training dataset and evaluated with the testing dataset. Among these, the LightGBM model achieved the highest performance across evaluation metrics.

3.4 Stack-Based Ensemble Model

In addition to individual tree-based ensemble models, a stack-based ensemble model was created by combining Random Forest, Extra Trees, and LightGBM. In this approach, each model is trained individually, and the probability distributions for each sample are averaged to obtain the final classification. The training dataset was used for model development, while the testing dataset was used for performance evaluation. However, the individual LightGBM model ultimately outperformed this ensemble approach.

4 Results and Discussion

This section presents the performance of the proposed approach using accuracy, precision, recall, AUC, and F1-score as evaluation metrics. In the comparative analysis of tree-based machine learning models—including fine-tuned LightGBM, RF, ET, XGB, and a soft-voting stack-based ensemble (RF, ET, and LGBM)—the fine-tuned LightGBM model outperformed all others, as shown in Table 3.

Table 3: Comparative analysis of performance of tree-based machine learning models with the proposed fine-tuned LGBM algorithm.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random Forest	0.9359	0.9423	0.9285	0.9353
Extra Trees	0.9464	0.9554	0.9363	0.9458
XGBoost	0.9580	0.9603	0.9553	0.9578
Stack-based Ensemble	0.9566	0.9617	0.9509	0.9563
Fine-tuned LightGBM	**0.9607**	**0.9626**	**0.9585**	**0.9606**

For better understanding of the proposed fine-tuned LightGBM model's performance, the confusion matrix is shown in Figure 5. Additionally, ROC curves for all models are presented in Figure 6.

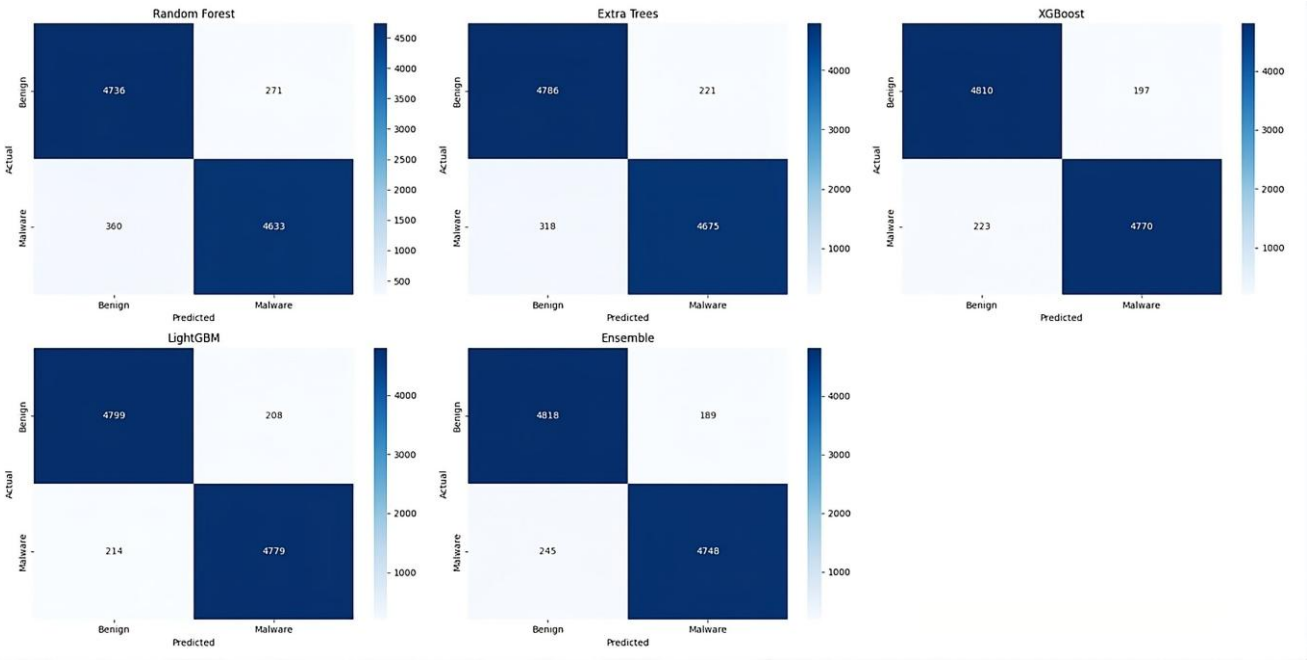

Figure 5: Confusion matrix of the proposed Fine-tuned LightGBM model on EMBER 2018 dataset.

To strengthen the proposed study, the performance of the fine-tuned LightGBM model is compared with existing works. All comparative studies were published in reputable journals and conferences between 2020 and 2025. The results demonstrate that our model outperforms all considered existing works across all evaluation metrics. The comparative analysis is presented in Table 4.

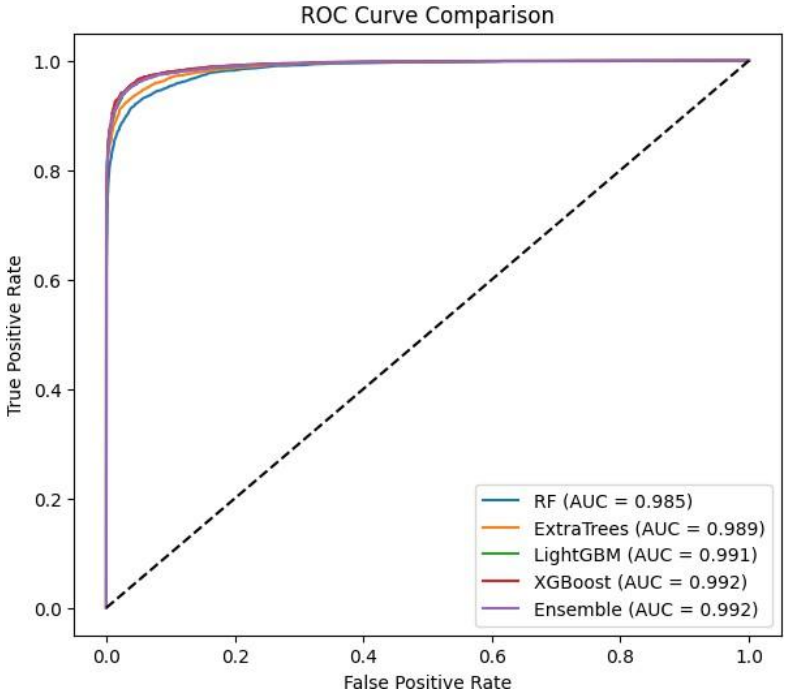

Figure 6: ROC curves for all evaluated models.

Table 4: Comparision of proposed work with the existing models on ember 2018 dataset.

S. No.	Reference	Model	# Features	AUC	Precision	Recall	F1 score	Accuracy(%)
1	Manjaly et al. [13]	Random forest	49	-	0.97	0.97	0.97	97
		Extra tree	49	-	0.97	0.97	0.97	97
2	Brown et al. [14]	AutoML	Not specified	0.98	-	-	0.96	-
3	Galen et al. [15]	LightGBM	2381	0.99	0.95	0.95	-	94.8
4	Kundu et al. [16]	LightGBM	Not Specified	-	-	-	-	90
5	Thosar et al. [17]	CNN	Not Specified	0.99	0.94	0.93	0.93	93.45
6	Connors et al. [18]	ANN	2381	0.98	0.96	0.94	0.95	95
7	Lad et al. [19]	CNN	2381	0.91	0.90	0.89	0.89	94.09
8	**Proposed work**	**FineTune-Light GBM**	2381	**0.99**	**0.96**	**0.96**	**0.96**	**96**

5 Conclusion & future scope

In this work, we consider the rarely explored, large raw EMBER 2018 dataset, which was preprocessed to obtain binary classes with 799,876 total samples and 2,382 features. We implemented various tree-based machine learning algorithms, among which LightGBM performed best. We then fine-tuned the hyperparameters of the LightGBM model to enhance its performance. This optimization improved the baseline LightGBM accuracy to 96%. Additionally, we developed a stack-based ensemble model using soft voting to combine Random Forest, Extra Trees, and LightGBM predictions. However, the proposed optimized LightGBM still achieved slightly better results than the stack-based ensemble model.

For future work, researchers can develop memory-optimized methods to experiment with the full EMBER 2018 dataset. There also remains considerable scope to explore additional ensemble machine learning models and further optimize the LightGBM architecture.

References

[1] M. H. Al-Adhaileh, A. Verma, T. H. Aldhyani, and D. Koundal, "Potato blight detection using fine-tuned cnn architecture," *Mathematics*, vol. 11, no. 6, p. 1516, 2023.

[2] T. H. Aldhyani, A. Verma, M. H. Al-Adhaileh, and D. Koundal, "Multi-class skin lesion classification using a lightweight dynamic kernel deep-learning-based convolutional neural network," *Diagnostics*, vol. 12, no. 9, p. 2048, 2022.

[3] L. Liu, B.-s. Wang, B. Yu, and Q.-x. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, 2017.

[4] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[5] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[6] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[7] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] B. De Ville, "Decision trees," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 6, pp. 448–455, 2013.

[9] T. Bayes, "Naive bayes classifier," *Article Sources and Contributors*, pp. 1–9, 1968.

[10] R. E. Schapire, "Explaining adaboost," in *Empirical inference: festschrift in honor of vladimir N. Vapnik*. Springer, 2013, pp. 37–52.

[11] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 100–111, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667345223000202

[12] A. Choudhary, S. Pawar, and Y. Haribhakta, "Efficient malware detection with optimized learning on high-dimensional features," *arXiv preprint arXiv:2506.17309*, 2025.

[13] J. Sunny Manjaly, R. CR, and L. Jose, "Evaluating the efficacy of machine learning models in predictive malware detection," *SSRN Electronic Journal*, 2025.

[14] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Computers & Security*, vol. 137, p. 103582, 2024.

[15] C. Galen and R. Steele, "Evaluating performance maintenance and deterioration over time of machine learning-based malware detection models on the ember pe dataset," in *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2020, pp. 1–7.

[16] P. P. Kundu, L. Anatharaman, and T. Truong-Huu, "An empirical evaluation of automated machine learning techniques for malware detection," in *Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics*, 2021, pp. 75–81.

[17] K. Thosar, P. Tiwari, R. Jyothula, and D. Ambawade, "Effective malware detection using gradient boosting and convolutional neural network," in *2021 IEEE Bombay Section Signature Conference (IBSSC)*. IEEE, 2021, pp. 1–4.

[18] C. Connors and D. Sarkar, "Machine learning for detecting malware in pe files," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 2194–2199.

[19] S. S. Lad and A. C. Adamuthe, "Improved deep learning model for static pe files malware detection and classification," *International Journal of Computer Network and Information Security*, vol. 12, no. 2, p. 14, 2022.

[20] Kaggle Dataset: dhoogla, "Ember-2018-v2-features: Elastic malware benchmark feature set (version 2)," https://www.kaggle.com/datasets/dhoogla/ember-2018-v2-features, 2025, accessed: 2025-12-10.