Effectiveness of Machine Learning Algorithms in Detecting and Predicting Malware

Dušan Čatloch[1], Eva Chovancová[2], Martin Chovanec[3] and Branislav Madoš[4]

[1] Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia;
dusan.catloch@tuke.sk

[2] Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia;
eva.chovancova@tuke.sk

[3] Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia;
martin.chovanec@tuke.sk

[4] Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia;
branislav.mados@tuke.sk

Abstract: Signature-based malware detection struggles with rapidly evolving threats, motivating machine learning (ML) approaches based on static PE features. We compare common ML classifiers on the EMBER2018 dataset, focusing on four families—Zusy, Ramnit, Sality, and Adposhel. After filtering and relabeling via avclass, we constructed a balanced set of 34,280 benign and 34,280 malware samples, applied a 70/30 train-test split, and injected 4.9% label noise to mimic real-world imperfections. We trained Decision Tree, Random Forest, XGBoost, LightGBM, a PyTorch neural network, and an SVM, and evaluated precision, recall, F1-score per class, and overall accuracy. LightGBM achieved the highest overall accuracy (95%), outperforming XGBoost and Random Forest (93%), the neural network (93%), and Decision Tree and SVM (92%). For the Benign class, LightGBM reached F1 = 0.96. Per-family analysis showed SVM provided the best balance on Zusy (recall 0.90, F1 0.92), while LightGBM best detected Ramnit (precision/recall/F1 = 0.95/0.90/0.92) and Sality (0.94/0.90/0.92); Adposhel was consistently easy to classify across models (F1=0.94). These results indicate that ML-based classification effectively detects malware families from static PE features: LightGBM is the most reliable overall, yet family-specific model selection can further improve deployment performance.

Keywords: classification; machine learning; malware families; malware detection; EMBER dataset; feature engineering; model comparison; precision; recall; F1-score; Random Forest; XGBoost

1 Introduction

The increasing volume and diversity of malware pose significant challenges to conventional cybersecurity systems. Traditional signature-based antivirus solutions rely on known patterns and therefore struggle to detect new, modified, or obfuscated malware variants. As a result, a substantial amount of malicious software may remain undetected, leading to potential financial and operational damage.

Machine learning provides a more adaptive approach by learning patterns from data and generalizing them to unseen samples. This enables the detection and classification of malware families even when their code is partially obfuscated or altered. However, the performance of machine learning models may vary across different malware families, with some models performing better for specific types of threats.

The main objective of this work is to evaluate and compare selected machine learning algorithms for static malware detection and classification, focusing on identifying both the best overall model and the most suitable models for individual malware families.

2 Related Works

Elshan Baghirov states that due to the ever-decreasing effectiveness of static methods in detecting malware, increasing attention is being paid to dynamic analysis, often combined with machine learning algorithms [1].

In dynamic analysis, malware is executed in a controlled environment and its behaviour can be observed, such as function calls, function parameters, transmitted network traffic, changes made to the file system. Akhtar and Feng argue that it is more difficult to hide malicious behaviour of malware during its execution [2]. Therefore, according to them, dynamic analysis is more suitable than static analysis. At the same time, Patil and Deng note that malware versions have consistent patterns of behaviour, which can be detected through dynamic analysis [3]

Although dynamic analysis can reveal more than static analysis, using dynamic analysis alone may no longer be sufficient to deal with the growing number

and diversity of malware. Therefore, classification techniques are often used in combination with dynamic analysis to identify and categorize malware. Kumar and Das point out that these methods help match unknown malware to known types or families, making it easier to react quickly and protect systems from harmful code [4].

It cannot be definitively concluded that one specific machine learning algorithm is the best or most accurate for predicting malware. However, some algorithms perform better than others when it comes to dynamic malware analysis. According to Patil and Deng, the top performing ones include Decision Trees, Random Forests, K-nearest neighbours and Deep Neural Networks with Neural Networks showing the best overall performance [5].

Patil and Deng's results also match those of Shaukat et al., who compared six machine learning algorithms on different datasets [6]. Decision Trees, Random Forests, Deep Belief Networks, which is a type of Deep Neural Network and the Naive Bayes classifier achieved the best results, even when applied to various datasets.

Shaukat et al. pointed out that while machine learning models often perform well on custom datasets, their accuracy can drop significantly when applied to different, non-tailored datasets [7]. This shows that good results on one dataset don't always mean the model will perform just as well elsewhere. Gilbert et al. also highlight the issue with datasets, pointing out that getting quality training data is one of the hardest parts of any machine learning problem. They emphasize that a machine learning model is only as good as the data it's trained on [8].

In malware prediction, Portable Executable (PE) files are commonly used, as they represent the standard format for Windows programs and libraries. Kumar et al. analyzed PE files using six machine learning algorithms, with Decision Trees and Random Forests achieving the best results [5]. Similarly, Tyagi et al. used PE header features such as characteristics, system, version, and image, where Random Forest again performed best [8].

An analysis of existing approaches shows that a major challenge in malware prediction is the limited availability of high-quality, up-to-date datasets covering recent malware families. Prior research also indicates that tree-based models, especially Decision Trees and Random Forests, are among the most effective methods for malware prediction when using PE file–based features.

3 Dataset EMBER

The EMBER (Endgame Malware BEnchmark for Research) dataset is a widely used open-source dataset for malware detection. First released, in 2018 by

Endgame (now Elastic), it supports malware analysis based on features extracted from PE files. The initial version, EMBER2017, contained 1.1 million samples collected in 2017 or earlier. Its successor, EMBER2018, was updated with samples from 2018 or earlier and extended with malware family labels obtained using the avclass tool. EMBER2018 consists of 1 million samples, divided into 800,000 training and 200,000 testing samples.

The dataset includes metadata extracted from Windows executable files using the LIEF tool. These attributes comprise file hashes (SHA256, MD5), an approximate first-seen timestamp, a label indicating benign or malicious files, optional malware family information, and eight groups of raw PE features. The data are stored in JSONL format and further processed into a machine-learning-friendly representation for malware classification.

In this work, the malware families Zusy, Ramnit, Adposhel, and Sality were selected due to their sufficient sample sizes and differing detection difficulty. According to prior analysis, Adposhel and Sality have relatively high detection rates, Ramnit shows moderate difficulty, and Zusy is the most challenging among the selected families.

4 Dataset Preprocessing

The preprocessing pipeline consisted of several steps to ensure data consistency, balance, and reproducibility. First, the original EMBER2018 dataset was filtered to include only the selected malware families (Zusy, Ramnit, Sality, and Adposhel) and benign samples. Malware samples were identified using the avclass attribute, and selected families were mapped to numeric class labels.

To address class imbalance, all classes were downsampled to match the size of the smallest family, Adposhel, which contained 8,570 samples. This resulted in a balanced dataset consisting of 34,280 benign and 34,280 malware samples, with each malware family equally represented.

Feature vectors were extracted using the ember library, which provides a standardized representation of PE file attributes. Since the library expects multiple input files, the merged dataset was adapted to match the required structure before vectorization. The resulting feature matrices were loaded into memory for further processing.

The dataset was split into training and testing sets using a stratified 70/30 split to preserve class distribution. The original EMBER test set was not used due to insufficient representation of the selected malware families.

To simulate real-world conditions where labels may be imperfect, label noise was introduced by randomly modifying 4.9% of class labels. All experiments were conducted using a fixed random seed (42) to ensure reproducibility.

Tree-based models (Decision Tree, Random Forest, XGBoost, LightGBM) were trained directly on the extracted features without normalization, as they are insensitive to feature scaling. In contrast, for the Support Vector Machine and Neural Network models, feature standardization was applied using statistics computed from the training set only, in order to avoid data leakage.

5 Training

We trained and compared six machine learning models: Decision Tree, Random Forest, XGBoost, LightGBM, Neural Network and Support Vector Machine (SVM). To evaluate model performance, we used precision, recall and F1-score for each class, along with overall accuracy. These metrics help us understand how well the models distinguish between actual malware and benign files and how balanced their predictions are.

5.1 Decision Tree

We used the decision tree implementation from sklearn and tuned hyperparameters such as tree depth and minimum samples per leaf using grid search. A depth of 20 and a minimum leaf size of 10 gave the best results. Since the dataset was balanced, class weights were not adjusted. Figure 1 shows the classification results.

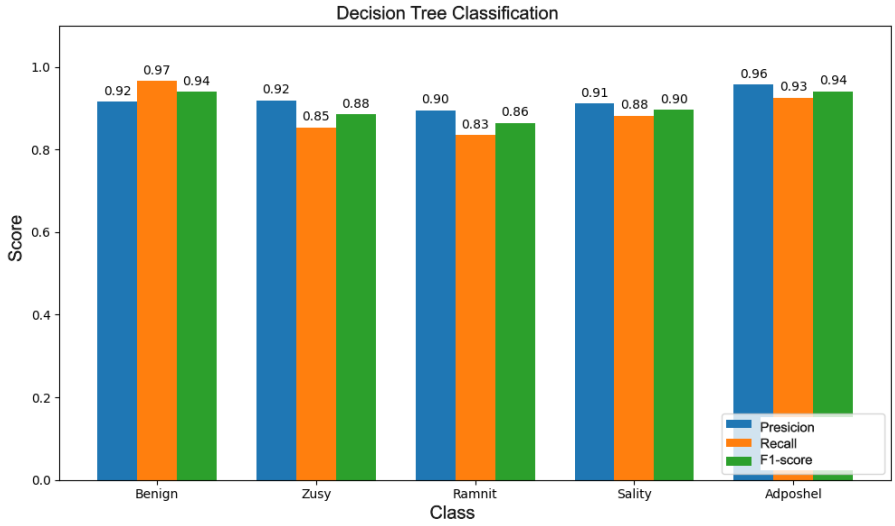

Figure 1
Decision Tree Classification Results.

The model achieved 92% overall accuracy. It performed very well on benign files (recall 0.97) and Adposhel (F1-score 0.94), showing high reliability for these classes. Sality also achieved balanced results, while performance was weaker for Zusy and especially Ramnit due to lower recall, indicating confusion with other malware families. Overall, the model demonstrated solid classification performance with room for improvement in detecting certain malware types.

5.2 Random Forest Model

We trained a random forest using the same library and tuning method. The best setup included 50 trees and a maximum depth of 30, with a minimum of 10 samples per leaf. This gave the model flexibility without overfitting. Figure 2 shows comparison of Random Forest classification results.

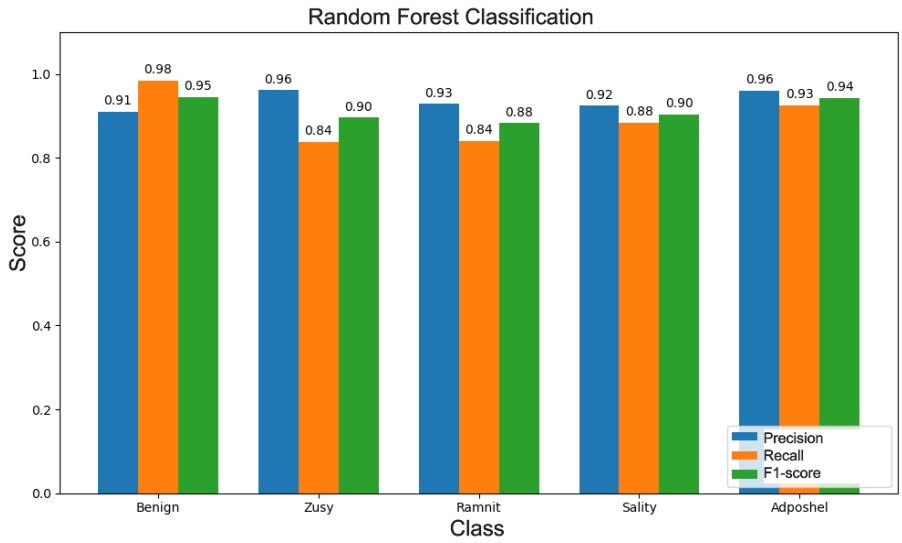

Figure 2
Random Forest Classification Results.

The model achieved 93% accuracy and improved upon the Decision Tree in most cases. It performed very well on benign files (precision 0.91, recall 0.98) and achieved the best results for Adposhel (precision 0.96, recall 0.93). Performance was slightly weaker for Sality, while Zusy and Ramnit showed lower recall, indicating that some samples were missed. Overall, the model demonstrated solid generalization with strong performance across most classes.

5.3 XGBoost Model

XGBoost was chosen for its efficiency and accuracy. We used a maximum depth of 8 and a learning rate of 0.1. During training, only 80% of samples and 70% of features were randomly used per tree, which helped avoid overfitting. Figure 3 shows comparison of XGBoost classification results.

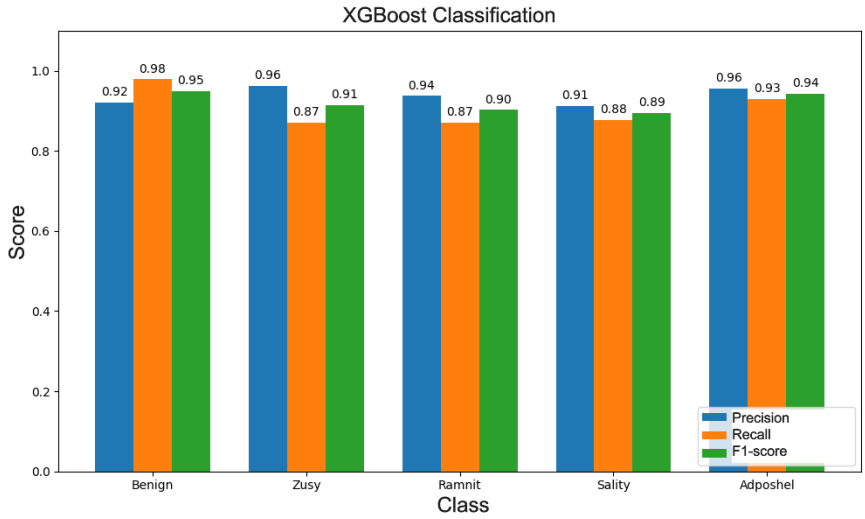

Figure 3
XGBoost Classification Results.

The benign class achieved high precision (0.92) and very high recall (0.98), indicating that the model reliably identified legitimate files. The slightly lower precision suggests a few malicious samples were incorrectly labelled as benign.

The Zusy class showed a precision of 0.96, recall of 0.87, and F1-score of 0.91. While most samples were correctly classified, the lower recall means some true Zusy samples were missed.

For the Ramnit class, the model reached 0.94 precision, 0.90 recall, and 0.92 F1-score. Although some misclassifications occurred, the model maintained a solid balance between precision and recall.

The Sality class had a precision of 0.91, recall of 0.88, and an F1-score of 0.89. Most samples were classified correctly, but the slightly lower recall shows that some confusion with other families occurred. Still, the overall performance remained strong.

The Adposhel class was classified the most accurately, with a precision of 0.96, recall of 0.93, and an F1-score of 0.94. The model had minimal trouble detecting this family, suggesting its features were well-defined and easily distinguishable.

The model reached 93% accuracy and gave very balanced results. It performed especially well with benign and Adposhel, with F1-scores of 0.95 and 0.94 respectively. It did slightly worse on Zusy and Sality, mostly because of lower recall, meaning it missed a few actual malware samples. Still, its performance across all classes was strong and consistent.

5.4 LightGBM Model

LightGBM turned out to be the top performer. It was configured for multiclass classification and evaluated using the multi_logloss metric. The model used moderate tree depth, early stopping, and regularization to avoid overfitting, along with random sampling of data and features. Figure 4 shows comparison of LightGBM classification results.

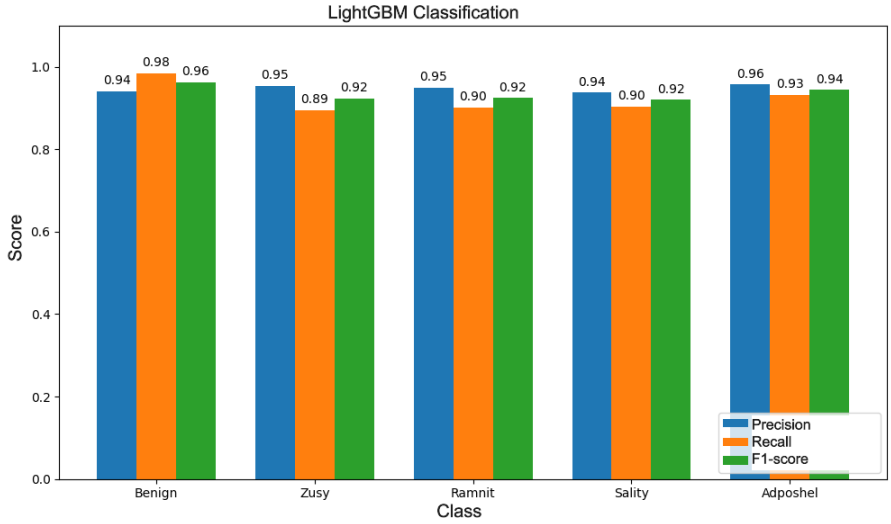

Figure 4
LightGBM Classification Results.

The benign class achieved fairly high precision (0.94) and very high recall (0.98), meaning the model reliably identified nearly all legitimate files.

The model achieved strong performance across all classes. Zusy showed high precision (0.95) with slightly lower recall (0.89), indicating minor confusion with similar malware. Ramnit and Sality were classified reliably, both reaching precision above 0.94 and recall around 0.90. The best results were obtained for Adposhel, with the highest precision (0.96) and F1-score (0.94). Overall, the model achieved the highest accuracy of 95%, excelling in detecting benign files and all malware families. LightGBM proved to be the most reliable and generalizable model.

5.5 Neural Network Model

We created a custom neural network using PyTorch. It had three hidden layers (512, 256, 128 neurons), with ReLU activations and dropout layers to prevent overfitting. The output layer had 5 neurons for each class, and the model was trained

using the Adam optimizer with a learning rate of 0.001 over 20 epochs. Figure 5 shows comparison of Neural Network classification results.

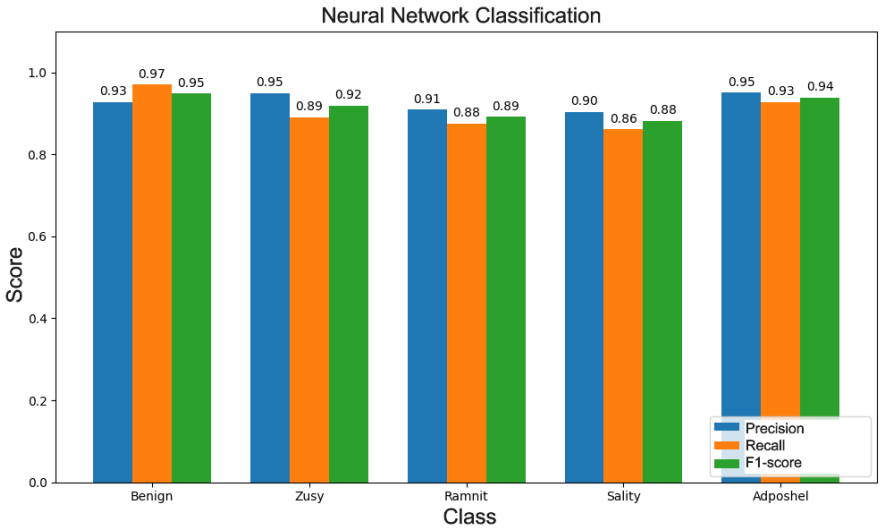

Figure 5
Neural Network Classification Results.

The benign class achieved good precision (0.93) and very high recall (0.97), showing the model reliably identified most legitimate files. An F1-score of 0.95 indicates a strong balance between precision and recall.

The Zusy class had solid precision (0.95) and slightly lower recall (0.89), meaning most samples were correctly classified, though some were missed. An F1-score of 0.92 reflects reliable performance with a few classification errors.

For Ramnit, the model reached lower precision (0.91) and recall (0.88), showing more frequent misclassifications compared to other classes.

The Sality class showed the lowest precision (0.90) and recall (0.86), suggesting the model struggled more with this category. The F1-score of 0.88 confirms this lower performance.

The best results were achieved for Adposhel, with a precision of 0.95 and recall of 0.93. The F1-score of 0.94 shows the model recognized this class accurately and consistently.

The model reached 93% accuracy. It handled benign and Adposhel very well, with strong precision and recall. It struggled more with Ramnit and Sality, where F1-scores dropped a bit. Still, the model was able to learn meaningful patterns and gave reliable results overall.

5.6 Support Vector Machine Model

The SVM was implemented using cuml to allow GPU training, which made it much faster. We used a radial basis function (RBF) kernel with the regularization parameter C set to 50. This made the model prioritize separating the classes more strictly. Figure 6 shows comparison of SVM classification results.

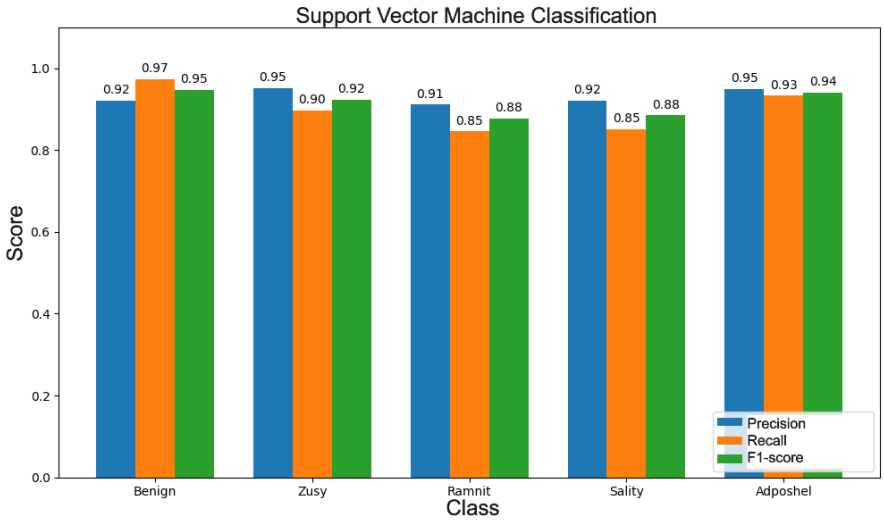

Figure 6
Support Vector Machine Classification Results.

The SVM model achieved 92% accuracy. It performed well on benign files (F1-score 0.95) and Adposhel (F1-score 0.94), showing reliable detection for these classes. Performance was slightly weaker for Zusy, and notably lower for Ramnit and Sality due to reduced recall and higher confusion with other classes. Overall, while SVM was the weakest among the more advanced models, it still delivered solid results given its simpler structure.

6 Results

The LightGBM model achieved the highest overall accuracy, performing best across all three evaluation metrics. This indicates strong generalization capabilities and reliable detection of various malware families. In contrast, the Decision Tree and SVM models showed the lowest accuracy, which is expected due to the simplicity of Decision Trees and the sensitivity of SVMs to high-dimensional data.

In terms of recall, LightGBM again stood out, successfully identifying the majority of true malware cases. The Decision Tree model had the weakest recall, suggesting it may have overlooked a larger portion of actual threats.

When evaluating the F1-score, which balances both precision and recall, LightGBM maintained the strongest performance. Decision Tree and Random Forest showed slightly lower scores, which suggests they weren't as consistent across different classes.

As shown in Figure 7, LightGBM consistently outperformed the other models and proved to be the most robust and well-rounded solution.

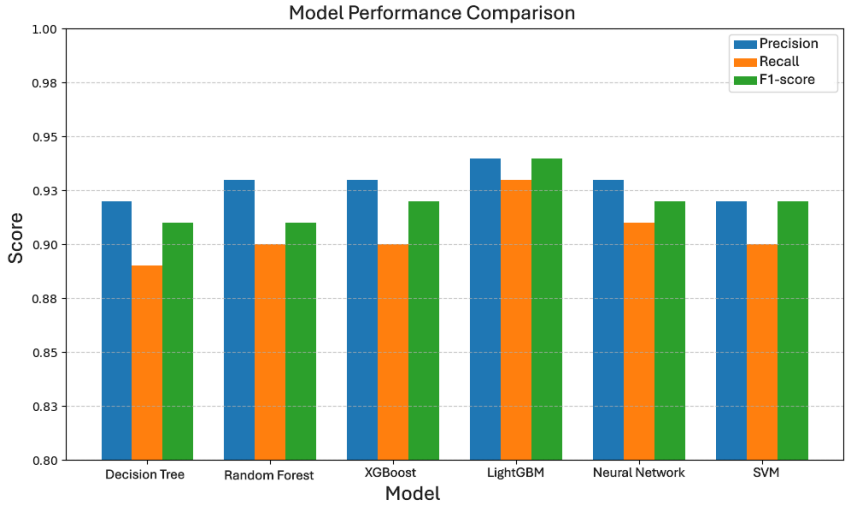

Figure 7
Model performance comparison.

6.1 Benign Class Results

Table 1 compares the performance of all tested algorithms in classifying the Benign class. This class represents legitimate files.

Table 1
Classification results for class Benign.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.92	0.97	0.94
Random Forest	0.91	0.98	0.95
XGBoost	0.92	0.98	0.95

LightGBM	0.94	0.98	0.96
Neural Network	0.93	0.97	0.95
Support Vector Machine	0.92	0.97	0.95

The LightGBM model achieved the best results, with an accuracy of 0.94, recall of 0.98, and F1-score of 0.96, indicating high reliability in detecting benign files. Other models such as XGBoost, Support Vector Machine, and Neural Network showed similar F1-scores (0.94–0.95), with only slight differences in precision. Although Random Forest had slightly lower accuracy (0.91), it still achieved high recall (0.98). Classification results for the Benign class are shown in Figure 8.

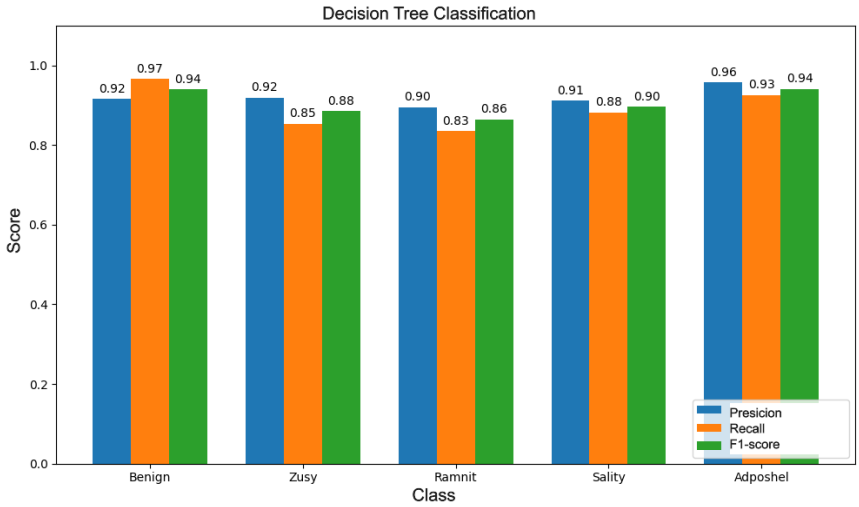

Figure 8
Benign Class Classification Results Comparison.

Overall, all models classified the Benign class very well, which is essential for reliably distinguishing safe files from threats.

6.2 Zusy Class Results

Table 2 shows the classification results for the Zusy malware family. This class was the most difficult to classify, as predicted by its lowest detection rate among the selected families in the dataset.

Table 2

Classification results for class Zusy.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.92	0.85	0.94
Random Forest	0.96	0.84	0.95
XGBoost	0.96	0.87	0.95
LightGBM	0.95	0.89	0.96
Neural Network	0.95	0.89	0.95
Support Vector Machine	0.95	0.90	0.95

The highest precision (0.96) was achieved by Random Forest and XGBoost, indicating a low number of false positives. In terms of recall, Support Vector Machine performed best (0.90), capturing the most Zusy samples. For F1-score, LightGBM, Neural Network, and Support Vector Machine all achieved 0.92, indicating the most balanced performance. Classification results for the Zusy class are shown in Figure 9.

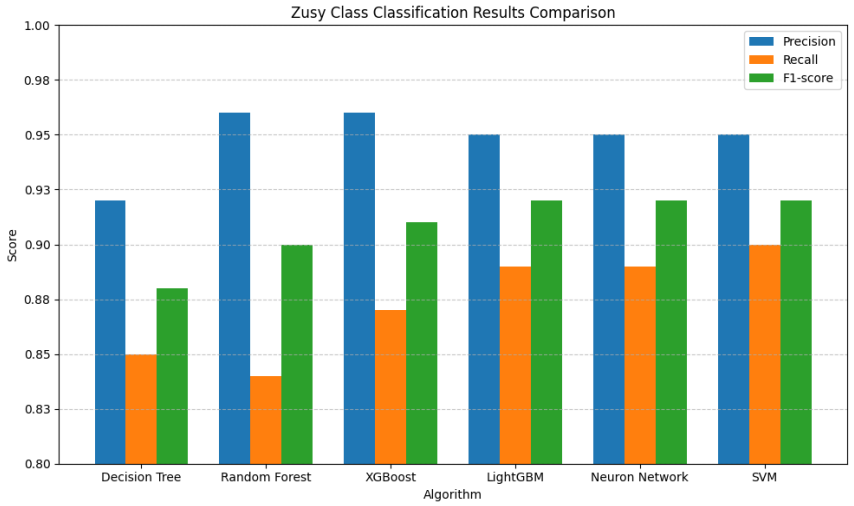

Figure 9
Zusy Class Classification Results Comparison.

6.3 Ramnit Class Results

Table 3 shows the performance of each algorithm in detecting malware from the Ramnit family, which is one of the more widespread and well-known types. The results highlight notable performance differences between models.

Table 3
Classification results for class Ramnit.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.90	0.83	0.86
Random Forest	0.93	0.84	0.88
XGBoost	0.94	0.87	0.90
LightGBM	0.95	0.90	0.92
Neural Network	0.91	0.88	0.89
Support Vector Machine	0.91	0.85	0.88

LightGBM achieved the best results across all metrics, with 0.95 precision, 0.90 recall, and 0.92 F1-score, making it the most reliable model for detecting Ramnit. XGBoost and Random Forest also performed well, with XGBoost scoring just slightly lower. Classification results of different models for class Ramnit can be seen in Figure 10.

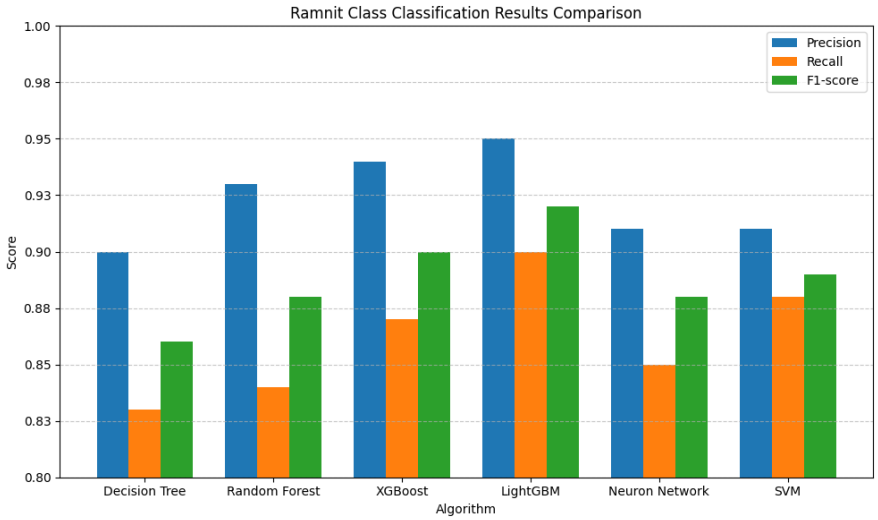

Figure 10
Ramnit Class Classification Results Comparison.

6.4 Sality Class Results

Table 4 presents the classification results for the Sality malware family, known for infecting Windows systems and spreading via botnets.

Table 4
Classification results for class Sality.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.91	0.88	0.90
Random Forest	0.92	0.88	0.90
XGBoost	0.91	0.88	0.89
LightGBM	0.94	0.90	0.92
Neural Network	0.90	0.86	0.88
Support Vector Machine	0.92	0.85	0.88

The best-performing model for this class was LightGBM, achieving a precision of 0.94, recall of 0.90 and the highest F1-score of 0.92, showing strong and balanced performance. Random Forest and Decision Tree followed with an F1-score of 0.90, though slightly lower recall suggests they missed more positive cases. XGBoost, Neural Network and Support Vector Machine performed slightly worse, with F1-scores around 0.88-0.89. Support Vector Machine had the lowest recall (0.85), indicating a higher rate of false negatives in detecting this malware. Classification results of different models for class Sality can be seen in Figure 11.

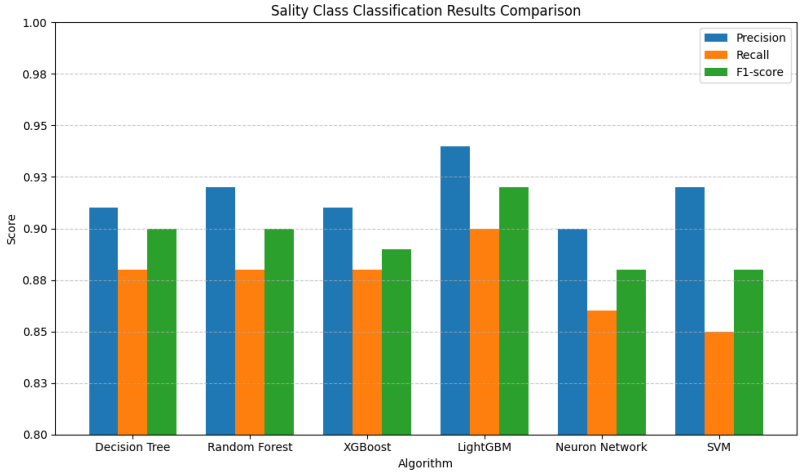

Figure 11
Sality Class Classification Results Comparison.

6.5 Adposhel Class Results

Table 5 shows the performance of all tested algorithms in classifying the Adposhel malware family. The results indicate consistently strong and balanced performance across all models.

Table 5
Classification results for class Adposhel.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.96	0.93	0.90
Random Forest	0.96	0.93	0.90
XGBoost	0.96	0.93	0.89
LightGBM	0.96	0.93	0.92
Neural Network	0.95	0.93	0.88
Support Vector Machine	0.95	0.93	0.88

All models achieved very high scores across all three metrics — precision ranged from 0.95 to 0.96, while recall and F1-score remained consistently high (approximately 0.93–0.94). Decision Tree, Random Forest, XGBoost, and LightGBM produced very similar results, suggesting that Adposhel is a class that can be reliably classified even with simpler algorithms. Classification results of different models for the Adposhel class can be seen in Figure 12.

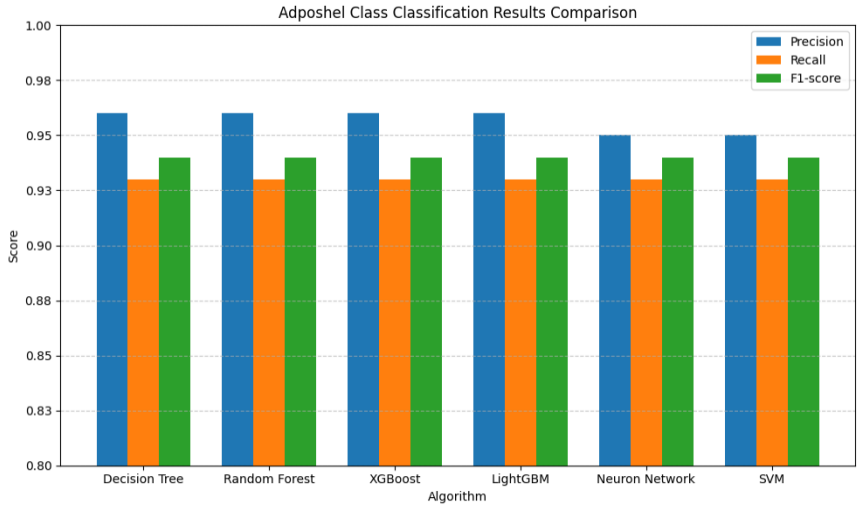

Figure 12
Adposhel Class Classification Results Comparison.

7 Discussion

The experiments indicate that gradient boosted decision trees provide the most reliable performance for static PE malware detection on the evaluated EMBER subset. LightGBM reaches the highest overall accuracy and strong F1 across classes. Random forests and XGBoost are close, which suggests that the main gains come from the shared boosting bias on tabular features. A view by family adds useful detail. LightGBM performs best on Ramnit and Sality and also on the Benign class. The support vector machine achieves the highest recall on Zusy. Adposhel is easier to separate and all models converge to high precision and recall. These patterns are consistent with the feature space: mixed PE metadata and entropy statistics favour tree ensembles that learn combinations of weak cues, while a kernel method can exploit a smoother boundary that appears for Zusy.

The practical takeaway is straightforward. A well regularized LightGBM is a strong default once features are extracted. A simple routing step can raise recall where it matters. Samples that are uncertain near the Zusy decision region can be evaluated by a support vector machine, while the boosted model handles the majority of traffic with low latency. This design keeps inference fast and adds capacity where it helps most. For models that require scaling, the preprocessing must be fit on the training split and applied consistently to held out data to avoid leakage. The results are also stable under a small amount of label noise. Absolute values decrease slightly, yet the ranking among models and the family level trends remain similar, which is encouraging for imperfect annotations.

8 Limitations

The evaluation uses a stratified seventy to thirty split after selecting the target families and balancing class counts. This design supports clear comparisons but can yield optimistic estimates if train and test share time local artifacts that static features can memorize. EMBER contains temporal information, so a time based split that trains on earlier samples and tests on later ones would better reflect deployment where new variants arrive. Repeated stratified cross validation would allow reporting means and confidence intervals for accuracy and macro F1 and would describe the stability of the ranking.

Only static features are considered. Static analysis scales well and avoids the cost and risk of sandboxing, but it is sensitive to packing and obfuscation. The class set is limited to four families plus Benign and the evaluation is balanced, while real world prevalence is skewed and evolves. These choices keep the study tractable. They should be remembered when transferring the conclusions to other settings.

9 Future Directions

Future work should adopt a time-based evaluation strategy to better assess model robustness under distribution shift, particularly by training on earlier samples and testing on newer ones. It would also be beneficial to broaden the taxonomy of malware families to evaluate model performance across a more diverse set of threats.

In addition, future research should explore hybrid approaches that combine static PE features with lightweight dynamic analysis traces, especially in cases where static signals are insufficient due to obfuscation or packing. On the modeling side, a promising direction is to further investigate the combination of LightGBM as a primary model with a focused Support Vector Machine for samples that are difficult to classify, such as those belonging to the Zusy family. Finally, incorporating explainable artificial intelligence (XAI) techniques could improve the interpretability and transparency of model decisions, which is important for real-world cybersecurity applications.

Conclusion

Based on the performance analysis, LightGBM achieved the best overall results, particularly for the Benign, Ramnit, and Sality classes. The Support Vector Machine performed most reliably for the Zusy family, achieving the highest recall and F1-score, while Adposhel was consistently well classified across all models.

These findings provide clear guidance for deployment. A well-regularized LightGBM model serves as a strong default once static features are extracted, offering a favourable balance of accuracy, latency, and memory use. Because the Support Vector Machine achieves the strongest recall on Zusy, a simple two-step design is recommended, where LightGBM handles the majority of samples and SVM is applied to more challenging cases. This approach preserves low latency while improving detection performance where it matters most.

The results remain stable under moderate label noise, suggesting that the ranking of methods is robust to small imperfections in annotations. However, the study is limited to static features and a balanced subset of the EMBER dataset. Real-world scenarios involve distribution shift and class imbalance, which should be addressed in future work through time-based evaluation, broader family coverage, and hybrid approaches that incorporate lightweight dynamic analysis.

In summary, the results support using LightGBM as the primary model for static malware detection, complemented by a Support Vector Machine to improve recall for more challenging malware families such as Zusy.

Acknowledgement

This publication was supported by the Operational Programme Slovakia within the project: Research and development of advanced artificial intelligence

solutions for detection of cyberthreats and defense against sophisticated attacks, ITMS21+: 401101C360.

References

[1] Baghirov, E. Techniques of Malware Detection: Research Review. In 2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT); IEEE: 2021; pp. 1–6. https://doi.org/10.1109/AICT52784.2021.9620415.

[2] Akhtar, M. S.; Feng, T. Evaluation of Machine Learning Algorithms for Malware Detection. Sensors 2023, 23 (2), 946. https://doi.org/10.3390/s23020946.

[3] Patil, R.; Deng, W. Malware Analysis Using Machine Learning and Deep Learning Techniques. In 2020 SoutheastCon; IEEE: 2020; pp. 1–7. https://doi.org/10.1109/SoutheastCon44009.2020.9368268.

[4] Kumar, D. A.; Das, S. K. Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework. Int. J. Intell. Syst. Appl. Eng. 2023, 11 (1), 330–338. Available online: https://www.ijisae.org/index.php/IJISAE/article/view/2543 (accessed on 21 October 2025).

[5] Kumar, A.; Abhishek, K.; Shah, K.; Patel, D.; Jain, Y.; Chheda, H.; Nerurkar, P. Malware Detection Using Machine Learning. In Knowledge Graphs and Semantic Web; Villazón-Terrazas, B.; Ortiz-Rodríguez, F.; Tiwari, S. M.; Shandilya, S. K., Eds.; Springer: Cham, 2020; pp. 61–71. https://doi.org/10.1007/978-3-030-65384-2_5.

[6] Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I. A.; Chen, S.; Liu, D.; Li, J. Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity. Energies 2020, 13 (10), 2509. https://doi.org/10.3390/en13102509.

[7] Gibert, D.; Mateu, C.; Planes, J. The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges. J. Netw. Comput. Appl. 2020, 153, 102526. https://doi.org/10.1016/j.jnca.2019.102526.

[8] Tyagi, S.; Baghela, A.; Dar, K. M.; Patel, A.; Kothari, S.; Bhosale, S. Malware Detection in PE Files Using Machine Learning. In 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON); IEEE: 2023; pp. 1–6. https://doi.org/10.1109/OTCON56053.2023.10113998.

[9] Anderson, H. S.; Roth, P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv 2018, arXiv:1804.04637. Available online: https://arxiv.org/abs/1804.04637 (accessed on 21 October 2025).

[10] Banon, S. Welcome Endgame: Bringing Endpoint Security to the Elastic Stack. Elastic Blog, 5 June 2019. Available online: https://www.elastic.co/blog/endgame-joins-forces-with-elastic (accessed on 21 October 2025).

[11] Thomas, R. LIEF—Library to Instrument Executable Formats. Blog post, 18 April 2017. Available online: https://lief.re/blog/2017-04-18-lief/ (accessed on 21 October 2025).

[12] Roth, P. EMBER Improvements. Conference presentation at the Conference on Applied Machine Learning for Information Security (CAMLIS), November 2019. Available online: https://www.camlis.org/2019/talks/roth (accessed on 21 October 2025).