Performance Benchmarking of H2O AutoML and Individual Models on Malware Detection Tasks

Minakshi Arya
Department of Computer Science
North Dakota State University
Fargo, USA
minakshi.arya@ndsu.edu

Shubhavi Arya
Luddy School of Informatics,
Computing and Engineering
Indiana University Bloomington
Bloomington, USA
aryas@iu.edu

Saatvik Arya
Information School
University of Washington
Seattle, USA
arya3@uw.edu

Abstract— **This paper presents a comprehensive comparative analysis of H2O AutoML and individual machine learning models including Distributed Random Forest (DRF), Gradient Boosting Machine (GBM), XGBoost, and Deep Learning, applied to malware detection. We evaluate these models using key performance metrics such as accuracy, AUC, log loss, precision, recall, and F1 score across different time frames. Our findings highlight the efficiency and effectiveness of H2O AutoML in identifying optimal models, providing insights into its potential advantages and limitations compared to traditional manual model selection.**

Keywords—H2O AutoML, malware, ensemble learning

I. Introduction

The rapid evolution of malware poses a significant threat to the security and integrity of digital systems. Malware, encompassing a wide range of harmful programs, is designed to infiltrate, disrupt, or steal data from computer systems and networks. Detecting and mitigating malware attacks is crucial for safeguarding sensitive information and ensuring the smooth operation of digital infrastructure.

Machine learning techniques have emerged as powerful tools for malware detection, offering the potential to automate the identification and categorization of malicious software based on patterns extracted from large datasets. In this paper, we leverage H2O AutoML, a robust automated machine learning framework, to develop practical models for malware detection using the TUNADROMD dataset.

Limited research has utilized H2O AutoML for malware detection. This study represents one of the pioneering efforts to apply H2O AutoML in this domain. By experimenting with various machine learning algorithms available in the H2O AutoML framework, such as Gradient Boosting Machines (GBM), Random Forest, Deep Learning, Generalized Linear Models (GLM), and XGBoost, we aim to evaluate the efficacy of these models in identifying malware.

Key questions addressed in this study include:

- How does the performance of H2O AutoML compare to traditional machine learning models on the TUNADROMD dataset in terms of accuracy, precision, recall, and F1-score?
- Can H2O AutoML efficiently handle the scale of the TUNADROMD dataset?
- What are the computational requirements and training/inference time for H2O AutoML?
- How interpretably does H2O AutoML produce the models?

The novelty of using H2O AutoML for malware detection lies in its automated approach to feature engineering and model selection, which are traditionally manual processes. This approach significantly reduces the time and expertise required to develop effective malware detection models, providing a more streamlined and accurate solution. Additionally, H2O AutoML is open source, making it a cost-effective solution for individuals and organizations. Its ability to reduce the time required to develop machine learning models is crucial in cybersecurity, where rapid response to new threats is essential.

This paper is organized into several vital sections to provide a comprehensive overview of our research. The Introduction outlines the motivation and objectives of the study. The Literature Review explores existing research in malware detection and the use of automated machine learning. The Dataset Description details the TUNADROMD dataset used for training and evaluation. The Methodology section explains the H2O AutoML framework, and the specific techniques applied in our approach. The Experimental Results present the performance of various models and insights derived from the data. The Conclusion and Future Work summarize our findings and discuss potential directions for further research. Finally, the References section lists the sources and related works cited throughout the paper.

II. Literature Review

Traditional Machine Learning Models

X. Xing et al. [2] utilized traditional machine learning techniques for malware detection, achieving high accuracy and F-score. This work provides a baseline for comparing traditional and deep learning-based methods.

M Arya et al. [12] conducted a comparative study on real-time malware detection in IoT devices, focusing on botnets such as Mirai, Okiru, and Torii. They evaluated various machine learning algorithms using RapidMiner, providing insights into effective detection techniques for enhancing cybersecurity in IoT environments.

Deep Learning Models

H. J. Zhu et al. [1] introduced MSAE and SHLMD frameworks, integrating unsupervised and deep learning techniques to enhance malware detection accuracy.

H. Rodriguez-Bazan et al. [5] used convolutional neural networks (CNN) to transform APK files into grayscale images for malware classification, achieving high accuracy.

X. Jin et al. [10] applied autoencoders to malware detection, focusing on reconstruction error for classification.

A. A. Mustafa Majid et al. [7] reviewed various deep learning methods, including CNN, RNN, LSTM, and autoencoders, highlighting their effectiveness in malware detection.

Federated Learning

N. Subramanian et al. [3] compared federated learning with traditional deep learning, emphasizing its accuracy, scalability, and privacy advantages.

Automated Machine Learning (AutoML)

R. Purwanto et al. [6] compared AutoML frameworks to manually crafted models, demonstrating AutoML's superiority in handling complex classification tasks.

Di. F. Isinngizwe et al. [11] conducted encrypted malware traffic classification using AutoML, showing the benefits of automated hyperparameter tuning and model assembling.

Hybrid Approaches

S. Mahdavifar et al. [8] utilized a semi-supervised learning technique (PLSAE) combining dynamic and static analysis for feature extraction, achieving high accuracy on recent malware samples.

Ensemble Methods

M. Sokolov et al. [4] proposed an ensemble method using Light Gradient Boosted Machines to improve malware detection accuracy.

M. A. Khan et al. [9] developed OE-IDS, an ensemble intrusion detection system validated on multiple datasets, showing high detection rates.

III. DATASET DESCRIPTION

The TUNADROMD dataset, a renowned dataset used in Android malware classification, is significant due to its comprehensive and diverse nature. It contains a wide variety of malware instances and attributes, making it a robust benchmark for evaluating the performance of machine-learning models in malware detection.

The TUNADROMD dataset [18] is a preprocessed TUANDROMD version containing 4465 instances and 241 attributes. Each instance represents a sample of potentially malicious software, and the attributes capture various features and characteristics of the samples. The target attribute for classification indicates whether an instance is categorized as malware or goodware, making this a binary classification task.

IV. METHODOLOGY

We split the data into training and testing sets and applied the following models:

H2O AutoML: Automated machine learning process that runs multiple algorithms and selects the best-performing model.

H2O Distributed Random Forest (DRF): An ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

H2O Gradient Boosting Machine (GBM): An iterative optimization algorithm that minimizes a loss function by adding weak learners.

H2O XGBoost: An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.

H2O Deep Learning: A neural network-based approach for learning from data with multiple layers of abstraction.

We evaluated these models using the following metrics:

Accuracy: The ratio of correctly predicted instances to the total instances.

AUC (Area Under the Curve): Measures the ability of the model to distinguish between classes.

Log Loss: Measures the performance of a classification model where the prediction is a probability value.

Precision: The ratio of true positive predictions to the total predicted positives.

Recall: The ratio of true positive predictions to the total actual positives.

F1 Score: The harmonic mean of precision and recall.

The models were evaluated at different time intervals: 3 seconds, 100 seconds, 300 seconds, 600 seconds, 1000 seconds, and 3600 seconds.

V. EXPERIMENTAL RESULTS

We experimented with various machine learning algorithms including H2O AutoML, Distributed Random Forest, Gradient Boosting Machines (GBM), Deep Learning, and XGBoost. The performance of the trained models was evaluated using several metrics.

A. Model Accuracy Comparison

The figure below illustrates the accuracy comparison of H2O AutoML and individual models over different execution times.

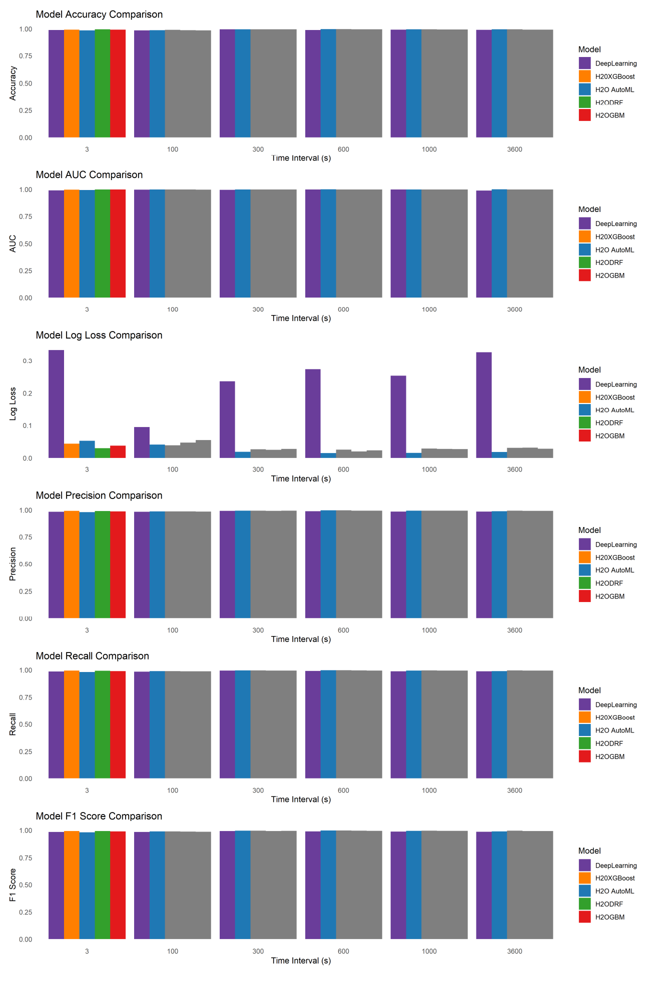

Fig. 1. The different performance metrics for different models

The accuracy generally increases with longer execution times for H2O AutoML and individual models. H2O AutoML consistently identifies high-accuracy models, with the best accuracy observed at the 600-second mark (0.996681).

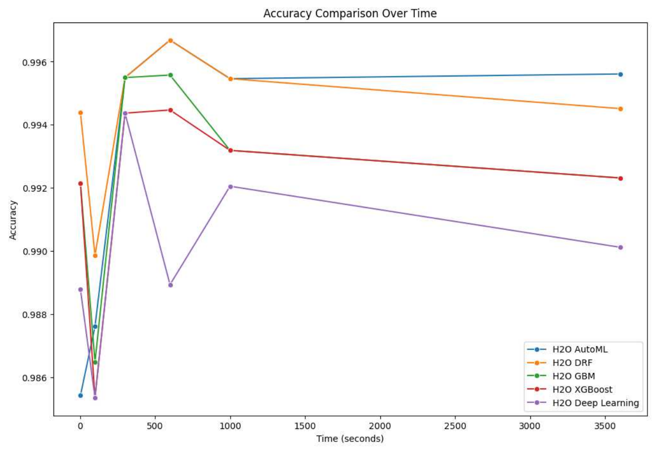

Fig. 2. Accuracy Comparison over time

H2O AutoML and H2ODRF consistently show high accuracy across all time intervals, with H2O AutoML slightly outperforming H2ODRF at higher intervals.

H2OGBM and H2OXGBoost also perform well, but their accuracy is generally slightly lower than H2O AutoML and H2ODRF.

Deep Learning has lower accuracy compared to other models, especially at lower time intervals, but improves significantly with more time.

B. AUC (Area Under the Curve)

A higher AUC represents better model performance. AUC values also show a general increase with longer execution times. H2O AutoML achieves the highest AUC at 600 seconds (0.999757), demonstrating its effectiveness in identifying models with superior discrimination ability.

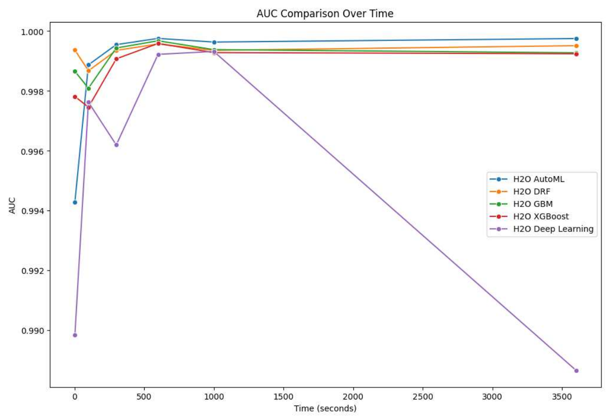

Fig. 3. AUC Comparison Over Time

H2O AutoML and H2ODRF achieve very high AUC scores across all intervals, indicating excellent discriminative ability.

H2OGBM and H2OXGBoost also have high AUC values, closely following H2O AutoML and H2ODRF.

Deep Learning shows lower AUC values compared to the other models, particularly at shorter time intervals, but it improves with more time.

C. *Log Loss*

A lower log loss indicates better model confidence. Log loss decreases with longer execution times, indicating improved model confidence. H2O AutoML consistently outperforms individual models in minimizing log loss, with the lowest log loss at 600 seconds (0.01466).

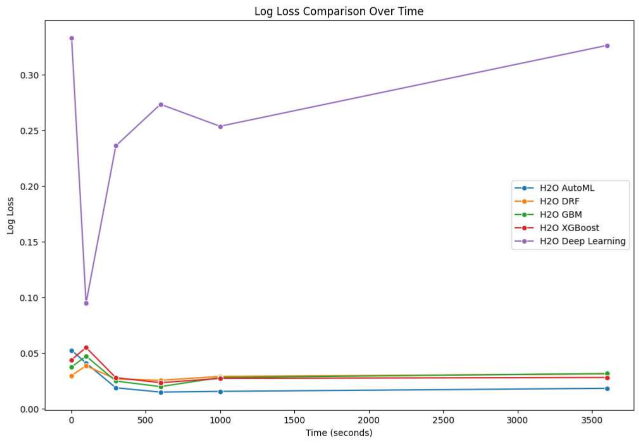

Fig. 4. Log Loss Comparison Over Time

H2O AutoML and H2ODRF consistently achieve low log loss values, indicating better model calibration.

H2OGBM and H2OXGBoost have slightly higher log loss values but still perform well.

DeepLearning shows significantly higher log loss values, particularly at lower time intervals, indicating poorer model calibration.

D. *Precision*

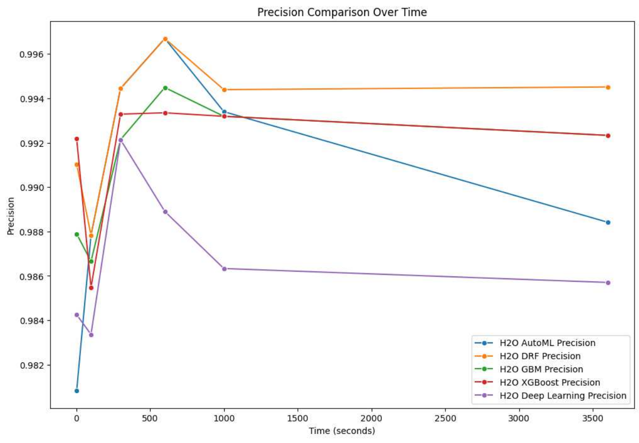

Fig. 5. Precision Comparison Over Time

H2O AutoML and H2ODRF exhibit high precision across all intervals.

H2OGBM and H2OXGBoost have slightly lower precision but are still competitive.

DeepLearning shows relatively lower precision, especially at shorter intervals, but improves with more time.

E. *Recall*

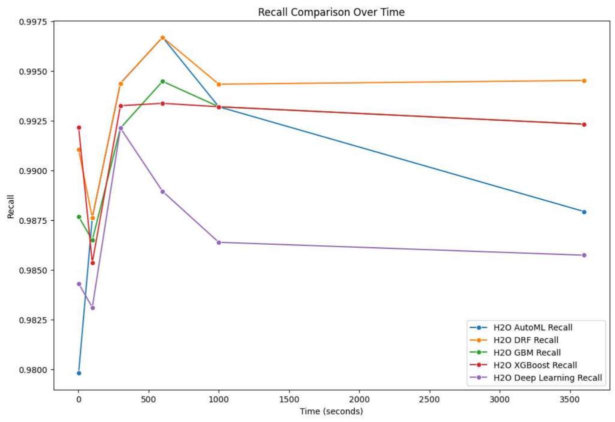

Fig. 6. Recall Comparison Over Time

H2O AutoML and H2ODRF maintain high recall values, indicating their ability to identify most positive cases.

H2OGBM and H2OXGBoost also have high recall values, though slightly lower than H2O AutoML and H2ODRF.

DeepLearning shows lower recall at shorter intervals, improving with longer training times.

F. *F1 Score*

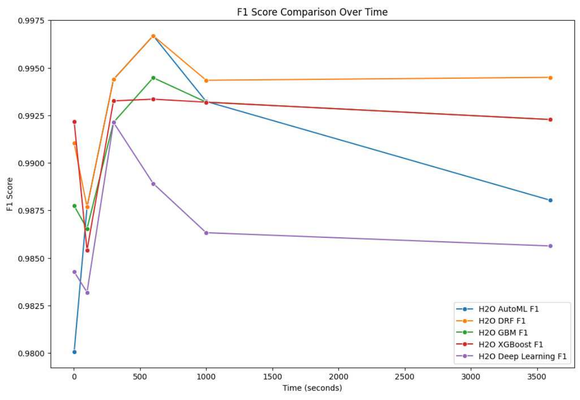

Fig. 7. F1 Score Over Time

H2O AutoML and H2ODRF achieve high F1 scores, reflecting a good balance between precision and recall.

H2OGBM and H2OXGBoost also perform well, with F1 scores close to H2O AutoML and H2ODRF.

Deep Learning has lower F1 scores at shorter intervals but shows improvement with longer training times.

G. *Time Taken (Time (s))*

DeepLearning models take significantly more time to train compared to other models.

H2O AutoML has a higher number of models run, leading to longer total times at higher intervals.

H2ODRF, H2OGBM, and H2OXGBoost are relatively quicker to train, maintaining lower time values

H. *Learning Curve Plot for the Best H2O AutoML model*

This learning curve plot for the "GBM grid model" shows the model's performance as the number of trees increases.

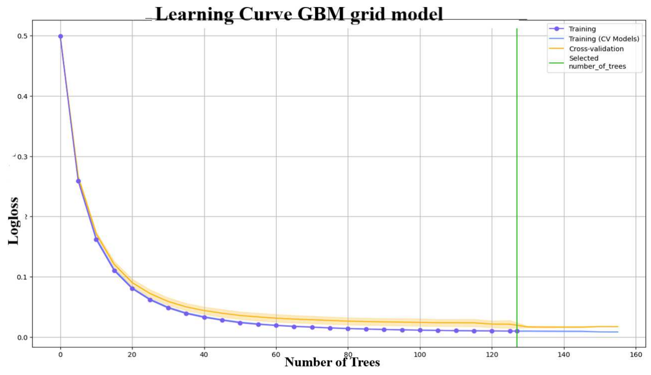

Fig. 8. Learning Curve Plot *for the Best H2O AutoML model*

Training (blue line) represents the log loss on the training data, showing a sharp decrease initially and then plateauing, indicating effective learning up to a certain point.

Training (CV) Models (yellow band) represent the cross-validation log loss with a confidence interval, closely following the training line initially, then plateauing, and the confidence interval narrowing, indicating stable performance across cross-validation folds.

Cross-validation (orange line) represents the cross-validation log loss, closely following the training log loss, demonstrating good generalization to unseen data.

No of trees (green line) represents the out-of-fold log loss. Beyond the green line, the models overfits.

I. *Variable Importance for the Best H2O AutoML model*

This variable importance plot visualizes the relative importance of various variables in a model.

Most Influential Factor: "Receiving Boot Completed Broadcast".

Moderately Important Factors: "Open Connection," "Get running tasks," "Keep the device awake," and "Get Last Known Location"

Less Critical Factors: "Receive SMS messages," "Load Library," and "Terminate Background Processes"

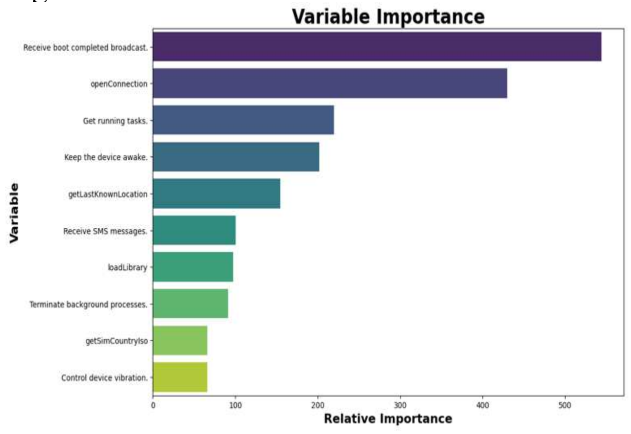

Fig. 9. Variable importance *for the Best H2O AutoML model*

J. *Shap Summary Plot for the Best H2O AutoML model*

The SHAP summary plot provides insights into feature impact on the model's predictions. It orders features by importance, with the most important feature at the top. Positive SHAP values push the prediction higher, while negative values push it lower. The color gradient from blue to red shows the value of the feature.

Most Impactful Features: "Open Connection" and "Receive Boot Completed Broadcast"

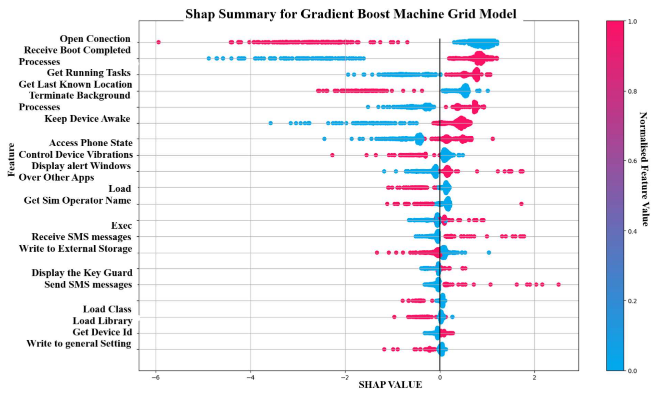

Fig. 10. Shap Summary Plots *for the Best H2O AutoML model*

Notable Impacts: Permissions like "Get Running Tasks," "Keep the Device Awake," and "Get Last Known Location"

Lesser Impacts: Features like "Rebooting the Device" and "Control Device Vibration".

K. *H2O AutoML Model Prediction Metrics*

The performance metrics chart depicts the performance metrics of various models, identified by their IDs on the x-axis.

Logarithmic Loss (blue line): Measures classification performance with probability outputs between 0 and 1. The decreasing trend indicates improved model performance.

Root Mean Square Error (green line): Measures error in predicting quantitative data. The slight decreasing trend suggests a small improvement in prediction accuracy.

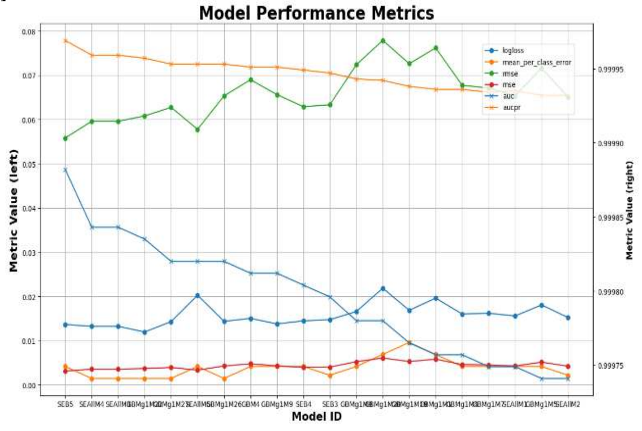

Fig. 11. H2O AutoML Models Performance metrics

Mean Absolute Error (orange line): Measures average error magnitude in predictions. The steady trend with minor fluctuations indicates consist**ent performance.**

Mean Squared Error (red line): Similar to RMSE but without square-rooting. The mostly flat line with slight decreases suggests minor improvements in prediction accuracy.

R-squared (purple line): Indicates the proportion of variance in the dependent variable explained by the independent variables. The flat line indicates consistent variance explanation across models.

Number of Estimators (yellow line with points): Shows the count of estimators (or trees) used, with fluctuations indicating variations across models.

VI. Conclusion and Future Work

This study presents a thorough evaluation of H2O AutoML compared to traditional machine learning models in the domain of malware detection. Our results demonstrate that H2O AutoML outperforms individual models in terms of accuracy, AUC, and log loss, particularly when given longer execution times.

H2O AutoML and H2ODRF are the top-performing models across most metrics, demonstrating high accuracy, AUC, precision, recall, and F1 scores while maintaining low log loss values.

H2OGBM and H2OXGBoost are strong performers, trailing slightly behind H2O AutoML and H2ODRF in most metrics.

Deep Learning models show potential but require more training time to reach competitive performance levels. They improve significantly with longer training times but generally lag behind other models in shorter intervals.

The automated feature engineering and model selection capabilities of H2O AutoML significantly streamline the model development process, making it a valuable tool for cybersecurity applications where rapid and accurate detection is critical.

Key Findings:

Superior Performance: H2O AutoML consistently identifies high-accuracy models with better overall performance metrics compared to traditional models.

Efficiency: The ability of H2O AutoML to handle large and complex datasets efficiently makes it suitable for real-world applications in cybersecurity.

Interpretability: H2O AutoML provides valuable insights into model predictions through variable importance and SHAP summary plots, enhancing the interpretability of the results.

Scalability: The framework demonstrates robustness in scaling to larger datasets, maintaining performance while reducing manual effort in model selection and hyperparameter tuning.

Future Work: Future research will focus on extending this study by evaluating H2O AutoML and individual models on larger and more complex datasets to further validate the findings. Additionally, exploring the integration of H2O AutoML with other cybersecurity tools such as intrusion detection and network security could provide a comprehensive solution for real-time threat detection and mitigation.

In conclusion, H2O AutoML offers a robust, scalable, and interpretable solution for developing high-performance malware detection models. Its automated approach addresses the challenges of manual model selection and feature engineering, providing a cost-effective and efficient tool for enhancing cybersecurity. The insights gained from this study can inform both academic research and practical implementations, contributing to the development of more secure digital infrastructures.

References

[1] H. J. Zhu, L. M. Wang, S. Zhong, Y. Li, and V. S. Sheng, “A Hybrid Deep Network Framework for Android Malware Detection,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 12, pp. 5558–5570, 2022, doi: 10.1109/TKDE.2021.3067658.

[2] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, “A Malware Detection Approach Using Autoencoder in Deep Learning,” IEEE Access, vol. 10, pp. 25696–25706, 2022, doi 10.1109/ACCESS.2022.3155695.

[3] N. Subramanian et al., “Securing Mobile Devices from Malware: A Faceoff Between Federated Learning and Deep Learning Models for Android Malware Classification,” J. Comput. Sci., vol. 20, no. 3, pp. 254–264, 2024, doi: 10.3844/jcsp.2024.254.264.

[4] M. Sokolov and N. Herndon, “Predicting Malware Attacks using Machine Learning and AutoAI,” Int. Conf. Pattern Recognit. Appl. Methods, vol. 1, no. Icpram, pp. 295–301, 2021, doi: 10.5220/0010264902950301.

[5] H. Rodriguez-Bazan, G. Sidorov, and P. J. Escamilla-Ambrosio, “Android Malware Classification Based on Fuzzy Hashing Visualization,” Mach. Learn. Knowl. Extr., vol. 5, no. 4, pp. 1826–1847, 2023, doi: 10.3390/make5040088.

[6] R. Purwanto, A. Pal, A. Blair, and S. Jha, “Man versus Machine: AutoML and Human Experts’ Role in Phishing Detection,” pp. 1–28, 2021, [Online]. Available: http://arxiv.org/abs/2108.12193

[7] A. A. Mustafa Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, “A review of artificial intelligence based malware detection using deep learning,” Mater. Today Proc., vol. 80, pp. 2678–2683, 2023, doi: 10.1016/j.matpr.2021.07.012.

[8] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, “Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder,” J. Netw. Syst. Manag., vol. 30, no. 1, p. 10922, 2022, doi: 10.1007/s10922-021-09634-4.

[9] M. A. Khan, N. Iqbal, Imran, H. Jamil, and D. H. Kim, “An optimised ensemble prediction model using AutoML based on soft voting classifier for network intrusion detection,” J. Netw. Comput. Appl., vol. 212, no. December 2022, p. 103560, 2023, doi: 10.1016/j.jnca.2022.103560.

[10] X. Jin, X. Xing, H. Elahi, G. Wang, and H. Jiang, “A malware detection approach using malware images and autoencoders,” Proc. - 2020 IEEE 17th Int. Conf. Mob. Ad Hoc Smart Syst. MASS 2020, pp. 631–639, 2020, doi: 10.1109/MASS50613.2020.00009.

[11] Di. F. Isingizwe, M. Wang, W. Liu, D. Wang, T. Wu, and J. Li, “Analyzing Learning-based Encrypted Malware Traffic Classification with AutoML,” Int. Conf. Commun. Technol. Proceedings, ICCT, vol. 2021-Octob, pp. 313–322, 2021, doi: 10.1109/ICCT52962.2021.9658106.

[12] M. Arya, S. Arya and S. Arya, "An Evaluation of Real-time Malware Detection in IoT Devices: Comparison of Machine Learning Algorithms with RapidMiner," 2023 IEEE International Conference on Electro Information Technology (eIT), Romeoville, IL, USA, 2023, pp. 077-082, doi: 10.1109/eIT57321.2023.10187265.

[13] D. Escudero García and N. DeCastro-García, “Optimal feature configuration for dynamic malware detection,” Comput. Secur., vol. 105, 2021, doi: 10.1016/j.cose.2021.102250.

[14] L. Dhanya and R. Chitra, “A novel autoencoder based feature independent GA optimised XGBoost classifier for IoMT malware detection,” Expert Syst. Appl., vol. 237, no. PC, p. 121618, 2024, doi: 10.1016/j.eswa.2023.121618.

[15] A. B. De Neira, A. M. Araujo, and M. Nogueira, “Early botnet detection for the internet and the internet of things by autonomous machine learning,” Proc. - 2020 16th Int. Conf. Mobility, Sens. Networking, MSN 2020, pp. 516–523, 2020, doi: 10.1109/MSN50589.2020.00087.

[16] G. D’Angelo, M. Ficco, and F. Palmieri, “Malware detection in mobile environments based on Autoencoders and API-images,” J. Parallel Distrib. Comput., vol. 137, pp. 26–33, 2020, doi: 10.1016/j.jpdc.2019.11.001.

[17] A. Chaudhuri, A. Nandi, and B. Pradhan, “A Dynamic Weighted Federated Learning for Android Malware Classification,” Lecture Notes in Networks and Systems, vol. 627 LNNS. pp. 147–159, 2023. doi: 10.1007/978-981-19-9858-4_13.

[18] P. Borah, “TUANDROMD (Tezpur University Android Malware Dataset,” 2023, doi: https://doi.org/10.24432/C5560H.

[19] A. Brown, M. Gupta, and M. Abdelsalam, “Automated machine learning for deep learning based malware detection,” Comput. Secur., vol. 137, no. October 2023, p. 103582, 2024, doi: 10.1016/j.cose.2023.103582.

[20] Y. D. Bromberg and L. Gitzinger, “DroidAutoML: A Microservice Architecture to Automate the Evaluation of Android Machine Learning Detection Systems,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020. doi: 10.1007/978-3-030-50323-9_10.

[21] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, “Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey,” HORA 2020 - 2nd Int. Congr. Human-Computer Interact. Optim. Robot. Appl. Proc., 2020, doi: 10.1109/HORA49412.2020.9152840.

[22] H. Bakır and R. Bakır, “DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms,” Comput. Electr. Eng., vol. 110, no. June, p. 108804, 2023, doi: 10.1016/j.compeleceng.2023.108804.

[23] O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” IEEE Access, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[24] N. Aslam et al., “Explainable Classification Model for Android Malware Analysis Using API and Permission-Based Features,” Comput. Mater. Contin., vol. 76, no. 3, pp. 3167–3188, 2023, doi: 10.32604/cmc.2023.039721.

[25] A. M. Araujo, A. Bergamini de Neira, and M. Nogueira, “Autonomous machine learning for early bot detection in the internet of things,” Digit. Commun. Networks, vol. 9, no. 6, pp. 1301–1309, 2023, doi: 10.1016/j.dcan.2022.05.011.

[26] Z. A. Abbood, I. Khaleel, and K. Aggarwal, “Challenges and Future Directions for Intrusion Detection Systems Based on AutoML,” Mesopotamian J. CyberSecurity, vol. 2021, pp. 16–21, 2021, doi: 10.58496/MJCS/2021/004.