MALPRED: Machine Learning for Malware Prediction in Windows

Abstract. Malware infections are a pervasive issue for computers running the Windows operating system. In this study, we present a machine-learning based approach to predict the likelihood of malware infection in Windows machines. Our methodology involves conducting data preprocessing, feature engineering, and selection on the Microsoft Malware Prediction dataset. We then perform extensive experimentation using various machine learning algorithms and identify XGBoost, LightGBM and CatBoost as the 3 best-performing algorithms. Through hyperparameter tuning via the Tree-Structured Parzen Estimator and using a Meta Learner on top of our top 3 best-performing algorithms, our optimal novel model achieves an AUC score of 73.24% across Stratified 5-fold cross-validation, demonstrating the efficacy of our approach. Additionally, we develop a web-based interface enabling users to input their Windows machine specifications and obtain predictions regarding the probability of malware infection.

Keywords: Machine Learning · Windows · Malware · Tree-Structured Parzen Estimator · XGBoost · LightGBM · CatBoost · Meta Learner.

1 Introduction

The Microsoft Windows Operating System (hereby referred to as Windows OS) is one of the most widely-used operating systems in the world, with more than 1.4 billion monthly active devices and 74% of computer users running Windows 10 or Windows 11 [1]. Due to the widespread use, malware is often created specifically for Windows OS. Malware refers to malicious software such as viruses, worms, trojans, etc. An example is WannaCry [6], a form of ransomware that infected 230,000 computers in just hours, leading to up to $4 billion dollars of damage. It is therefore of utmost importance that there is a reliable mechanism that can advise users on the likelihood of infections and prevent malware from causing more significant issues.

Malware is often analysed in three major forms: static, dynamic and a hybrid of both forms. Static analysis involves analysing static features of the malware sample, such as its strings and PE Headers, while dynamic analysis places the malware sample into a sandbox and extracts its behaviour. Machine learning is also increasingly adopted into malware analysis due to its ability to identify patterns based on large data samples. However, these methods largely focus on analysing specific malware samples, rather than the entire system.

In this paper, we aim to focus on using the specifications of the Windows machine to determine the likelihood of malware infection. The contributions of this paper are as follows:

- Extensive feature processing and engineering of the dataset provided in the Microsoft Malware Prediction.
- A comprehensive set of experiments on the post-processed data set for an empirical comparison among various machine learning models, as well as building a Meta Model from the best 3 performing models
- Hyperparameter tuning of the best models to obtain better performance.
- A Novel Web Application Interface to access the finalised model.

2 Related Works

In the past, several teams have presented work aiming to use the specifications of the Windows machine. Most of these works [8, 10, 11] utilise the Microsoft Malware Prediction dataset, publicly available on Kaggle [5], and most authors evaluate their models mainly on the Area under the ROC curve (hereby referred to as the AUC score).

Pan et al [8] preprocessed the aforementioned dataset by reducing the memory of the data (by removing largely null columns and switching data types to less precise forms), then utilised chi-square testing to evaluate the more useful metrics to the actual label. Following this, they trained 3 models on the dataset: Logistic Regression, K-Nearest Neighbours (KNN) and Light Gradient Boosting Machine (LightGBM). The team found that the LightGBM model attained a high AUC score of 0.72.

Similarly, Shahini et al [10] also preprocessed the data similarly by removing largely null columns. The team then trained a LightGBM model due to its lower memory usage and faster training speed. The team attained an AUC score of 0.74. Following this, they also investigated the model to find the features that contributed the most to the model, citing `FirmwareVersionIdentifier`, `CityIdentifier`, and `SystemVolumeTotalCapacity` as the most contributive features to their model.

Sokolov et al [11] conducted a series of experiments to determine the best feature selection method. Following this, he then utilised Automated Artificial Intelligence (AutoAI) methods to evaluate the stated feature selection methods, in addition to LightGBM, Naive Bayes and Logistic Regression. Through the series of experiments, the team proposed an ensemble model composed of five different LightGBM models, each trained on a different fold of the dataset. Sokolov et al evaluated their models using the accuracy, precision, recall and F_1 scores and attained an accuracy score of 0.6799.

3 Methodology

3.1 Overview

An overview of our methodology is presented in Fig.1[1]. Our methodology is split into 2 phases. The first phase involves the model being trained on the Microsoft

[1] Icons from Pixel Perfect, Eucalyp, Paul J., Freepik, and Canva have been used when designing the figures.

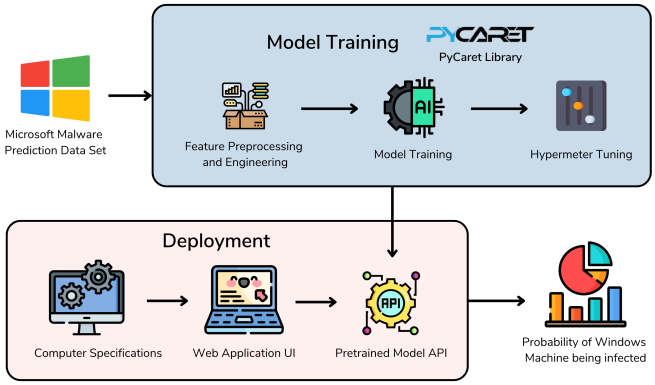

Fig. 1. A simple overview of our approach

Malware Prediction dataset from Kaggle [5]. Using the PyCaret library [3], we perform extensive feature preprocessing and engineering on the dataset. We then train the hypertuned model. The second phase involves the deployment of the model on a Web Application Interface. Our Web Application User Interface (UI) allows users to input the specifications of their personal computers. Following that, inference is conducted and the probability of infection is displayed.

3.2 Microsoft Malware Prediction Dataset

In this paper, the Microsoft Malware Prediction Dataset from Kaggle [5] is used. The dataset contains 8,921,483 samples, each one representing a different Windows Machine, with 83 features extracted from each machine. The `"HasDetections"` column indicates whether the Windows Machine has been previously infected by the malware. As some features have many missing values while other features are not strongly correlated to whether a Windows Machine is likely to get infected, extensive feature preprocessing and engineering is required in order to achieve good results on machine learning models.

3.3 Feature Engineering and Preprocessing

Data Cleaning and Processing An overview of the preprocessing is shown in Fig.2. Firstly, similar to [10], columns with more than 70% null values, (specifically `PuaMode`, `Census ProcessorClass`, `DefaultBrowsersIdentifier`, `Census Is -FlightingInternal`, and `Census InternalBatteryType`) are noted to likely not be helpful in the task at hand and are therefore removed. For other samples that still have null values for other columns, we utilise the simple imputer found in PyCaret to replace the null values. For categorical columns, the null values are replaced by the most frequent value found in the column. For numerical columns, the null values are replaced by the mean of the column. Normalisation

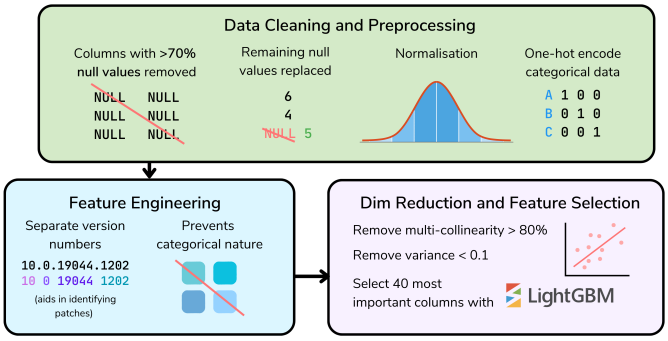

Fig. 2. Feature Engineering and Preprocessing

is also performed on numeric columns such that the difference in values for all numeric columns is not drastically different. Categorical data is one hot encoded as the machine learning model is unable to process categorical data directly.

Feature Engineering In the provided data set, there are 6 fields which are version identifiers. These identifiers have 4 sections separated by dots, showing the major, minor, build and patch numbers. Since the versions that are released closer together are likely to have similar security vulnerabilities, each version field was separated such that each section is a field [8]. This is to allow the models to see how close the versions are instead of having categorical data. This can also help us find which versions may have patches for vulnerabilities.

Dimension Reduction and Feature Selection Before we train our models, we must first investigate the dataset for any highly correlated features, as they do not generally contribute to the performance and add to the variance of the coefficients of the model. This can lead to the model being more unstable in general, thus we remove columns where the multi-collinearity score is found to be higher than 80%, and also columns with a variance lower than 0.1. From here, we perform Feature Selection to select the 40 most important columns using the Light Gradient Boosting Machine (LightGBM) model.

3.4 Machine Learning Models

We evaluate 3 different models to predict the probability of the Windows Machines being infected by Malware: Extreme Gradient Boosting Classifier (XGBoost), Cat Gradient Boosting Classifier (CatBoost), and Light Gradient Boosting Machine (LightGBM). We chose these 3 models as these 3 models are shown to have good performance when on numerical data. Other than these 3 models, we also evaluated Support Vector Machines (SVM), Decision Trees (DT),

Linear Discriminant Analysis (LDA), Multilayer Perceptron (MLP), AdaBoost (ADA), Quadratic Discriminant Analysis (QDA), Gradient Boosting Classifier (GBC), Ridge Classifier (Ridge), and a Dummy Classifier (Dummy) that ignores input features, to compare the effectiveness of our model. We also introduce a novel method using a meta-learner that uses a logistic regression model as Meta Learner and learns the output of the three best-performing models.

3.5 Hyperparameter Tuning

We then tune the hyperparameters of the best-performing model using Tree-Structured Parzen Estimator (TPE), a novel method not highlighted in prior literature. This is because it can exploit the inherent tree structure of models such as Random Forest, Decision Trees and Gradient Boosting Classifiers. TPE balances the exploration and exploitation, efficiently handles high-dimensional spaces and uses Kernel Density Estimation (KDE) to focus on promising hyperparameter configurations, reducing computational resources as well. This led to us using TPE as the search algorithm for hyperparameter tuning.

4 Results

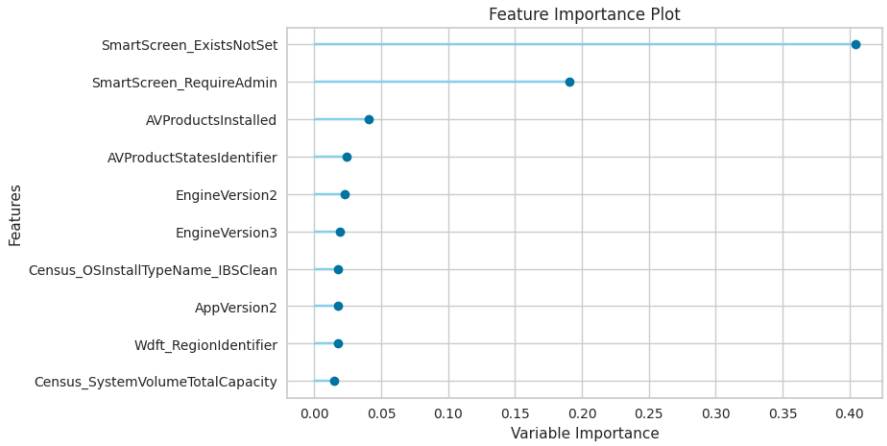

Fig. 3. Feature Importance of Top 10 fields

4.1 Feature Importance

It is important to see which features play the biggest role in the decisions of the model. This is to see what improvements can be made to reduce the chance of

getting infected by the malware the most. As seen in Fig 3, the top 4 features are related to the SmartScreen and Antivirus. These features protect computers from getting infected, so it makes sense that they affect the probability of a computer getting infected greatly. The top features also contain build and patch numbers. A change in these numbers normally connotes a fixed bug or possibly more bugs created by the fix, thus their high importance in the model.

4.2 Experimental Setup and Metrics

We evaluate the models using 5-Fold Cross Validation, where the dataset is split into 5 equal sets, and four sets are used for training the model, while one set is used for the evaluation of the model. The process is repeated until all 5 sets of data have been tested individually and the scores for each fold will be averaged. This reduces any bias in the dataset as all data in the dataset is used for testing. For metrics, we evaluate the performance of our model using Area Under Curve (AUC-ROC), Accuracy, Recall, Precision as well as the F1 score. We mainly optimised on the AUC as the focus of the classifier was to generate probabilities of malware infections. The AUC is also noted to be the main metric evaluated from Microsoft as noticed from the Kaggle competition [5].

4.3 Experiments with Machine Learning Models

As shown in Table 2, the stack model performed the best amongst all models. The Ridge and SVM also do not have AUC-ROC as they do not generate probabilities.

Table 1. Score of Machine Learning Models over 5-fold Cross Validation (Best results bolded)

Models	AUC-ROC	Accuracy	Recall	Precision
Stack Model	**73.03**	**66.45**	**66.25**	**66.49**
XGBoost	72.19	65.79	65.18	65.96
LightGBM	71.25	65.08	64.21	65.33
CatBoost	70.92	64.78	63.50	65.15
MLP	70.72	64.62	63.56	64.94
GBC	69.50	63.57	64.22	63.38
ADA	68.75	63.14	64.87	62.68
LDA	66.72	61.68	57.79	62.64
QDA	66.23	56.98	65.85	61.51
DT	57.60	57.60	57.73	57.56
Dummy	50.00	50.02	00.00	00.00
Ridge	–	61.68	57.79	62.64
SVM	–	61.34	51.93	63.97

4.4 Experiments with Hypertuning

Due to our limited resources, we extracted 10% of the dataset, then utilised the TPE algorithm using PyCaret [3] to tune our model with 100 iterations. We tuned XGBoost, CatBoost and LightGBM using the TPE method. However, the TPE tuning did not return a better model when compared to the LightGBM, as such, we did not include LightGBM in the results below. We then take the best parameters derived from the hypertuning and apply them to our models to test on the whole dataset. We also reconstructed the Meta Model, with the tuned XGBoost [4], CatBoost [9] and the original LightGBM [7], and the tuned XGBoost as a Meta Learner.

Table 2. Comparison of tuned and untuned models (in percentage)

Models	**AUC-ROC**	**Accuracy**	**Recall**	**Precision**
XGBoost	72.19	65.79	65.18	65.96
XGBoost (TPE Tuned)	72.95	64.95	**80.58**	61.38
Normal CatBoost	70.92	64.78	63.50	65.15
CatBoost (TPE Tuned)	72.37	65.95	65.11	66.20
Normal Stacked	73.03	**66.45**	66.25	**66.49**
Stacked (TPE Tuned)	**73.24**	65.36	79.81	61.90

4.5 Discussion of Results

As expected, the gradient boosting models, such as XGBoost, LightGBM and CatBoost, had the best performance on the dataset. MLP is shown to have the next best performance, therefore suggesting the effectiveness of a deep learning approach. Our Meta Model is shown to have the best performance among the untuned models, with the highest AUC score of 73.03%. The TPE tuning also improved the performance of most of our models, with our models seeing an improvement of up to 1.45% in the AUC score. Our finalised model, the TPE-tuned Meta Model, with an AUC score of 73.24% outperforms most existing works, and is only a stone's throw away from the State of The Art (SOTA) model [10], which achieved 74% AUC over 5-Fold Cross-Validation. However, we believe that given more resources, we could have performed TPE tuning across the entire dataset, and matched or even exceed the performance of the SOTA model.

4.6 Web Application

We used PyCaret to create a Gradio [2] app interface for our web application. Gradio is a free and open-source Python library that allows us to develop an easy-to-use customisable component demo for our machine learning model. The novel web app takes in several parameters as inputs and returns the probability

of the presence of malware. The Web Application User Interface can be seen in 4.

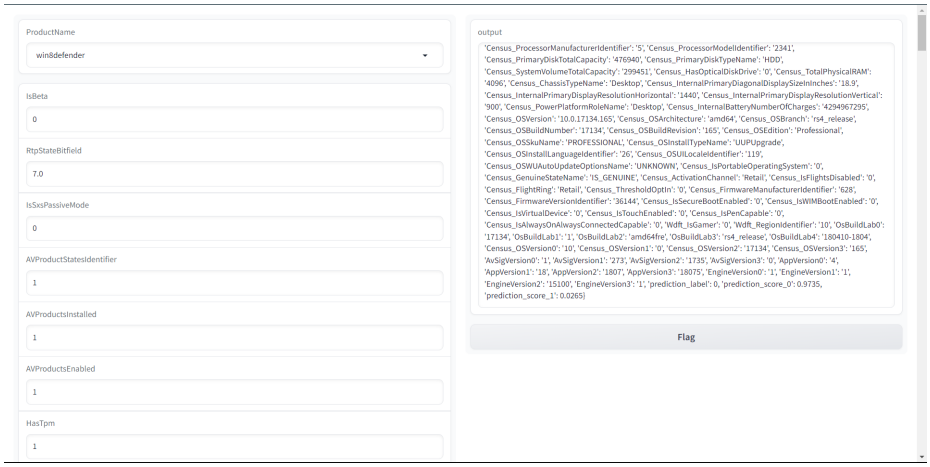

Fig. 4. Gradio Web Application User Interface

5 Conclusion

In this paper, we proposed a method of data cleaning, feature engineering and feature selection for the Microsoft Malware Dataset. We test our feature processing method using various models, such as XGBoost, Cat Boost and LightGBM. We than stack the models together and use a Meta Learner which learns from the output of the best 3 performing models. Finally, we perform hyperparameter tuning using the Tree-Structured Parzen Estimator method. Our model achieves a AUC score of 73.24%.

In the future, we wish to explore the use of Large Language Models (LLMs) on this dataset. We also hope to obtain newer data regarding for this, as the Microsoft Malware Dataset dates back to 2015, where most people were using Windows 8.1. Finally, we hope to develop a desktop application that can automatically collect the specifications of the machine before sending it to the API for inference.

References

1. Microsoft by the Numbers (June 2023), https://news.microsoft.com/bythenumbers/
2. Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., Zou, J.: Gradio: Hassle-free sharing and testing of ml models in the wild. arXiv preprint arXiv:1906.02569 (2019)

3. Ali, M.: PyCaret: An open source, low-code machine learning library in Python (April 2020), https://www.pycaret.org, pyCaret version 1.0
4. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, http://doi.acm.org/10.1145/2939672.2939785
5. Howard, A., Hope, B., Saltaformaggio, B., Eric Avena, M.A., Duncan, M., McCann, R., Cukierski, W.: Microsoft malware prediction (2018), https://kaggle.com/competitions/microsoft-malware-prediction
6. Kaspersky: What is wannacry ransomware? (Feb 2022), https://shorturl.at/acEOT
7. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30**, 3146–3154 (2017)
8. Pan, Q., Tang, W., Yao, S.: The application of lightgbm in microsoft malware detection. Journal of Physics: Conference Series **1684**(1), 012041 (2020). https://doi.org/10.1088/1742-6596/1684/1/012041
9. Prokhorenkova, L.O., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) NeurIPS. pp. 6639–6649 (2018), http://dblp.uni-trier.de/db/conf/nips/nips2018.htmlProkhorenkovaGV18
10. Shahini, M., Farhanian, R., Ellis, M.: Machine learning to predict the likelihood of a personal computer to be infected with malware, https://scholar.smu.edu/datasciencereview/vol2/iss2/9/
11. Sokolov, M., Herndon, N.: Predicting malware attacks using machine learning and autoai. In: ICPRAM. pp. 295–301 (2021)