Team 2: Data Mining

[FA23\_DATS6103](https://github.com/ning-rui/FA23_DATS6103/tree/main) : Summary Report

Professor Ning Rui

Dec 11, 2023

**Microsoft Malware Prediction**

**Introduction**

**Our proposed project aims to predict the probability of Windows machines becoming infected by malware by analysing telemetry data, including various machine properties and infection records, from Microsoft's endpoint protection solution, Windows Defender.** The goal of this competition is to predict a Windows machine’s probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender. The notebook is structured to process and analyze data for malware classification, specifically in the context of Microsoft Malware Prediction. It includes steps for data pre-processing, feature extraction, and initial exploratory data analysis. The focus is on understanding the characteristics of the malware samples through feature analysis and class distribution, which are critical steps in building a predictive model for malware detection.

The methodology employed for data collection in this study was tailored to align with specific operational limitations, including the safeguarding of user data privacy and the operational duration of the equipment involved. The challenge of malware detection, a task that fundamentally relies on chronological data, is further heightened due to factors such as the introduction of new devices, fluctuating device connectivity, application of updates, and installation of different operating systems. The dataset presented here is divided based on timeframes, but due to the aforementioned complexities and data collection constraints, there might be inconsistencies observed in the alignment of your cross-validation, public, and private scores. It's also important to note that this dataset does not accurately reflect the typical distribution of malware on Microsoft customers' devices; it intentionally contains a higher ratio of malware-infected machines for study purposes.

Where we can predict a high-level overview based on the initial cells of the notebook, each cell's code and output must be analysed for a more comprehensive report.

**SMART Questions**

Intending to focus the project on relevant aspects of our cyber threats, we focused on recent data for properties in Malware detection in Microsoft. In particular, we inquired into two aspects of byte files and asm files and tested various aspects with them:

1. What are the key features in the dataset that can be used to predict malware infections?
2. How does this project's outcome contribute to enhancing cybersecurity efforts?
3. Are asm files and. bytes file enough for malware detection?
4. The dataset has an imbalance, can we handle it through hyper-parameter tuning, which model will stand out?

**Literature Review**

From the available information, it's known that the Microsoft Malware Classification Challenge, related to this Kaggle competition, was launched in 2015 with a significant dataset of nearly 0.5 terabytes, including disassembly and bytecode of over 20,000 malware samples. This dataset has become a standard benchmark in malware behaviour modelling research and has been cited in more than 50 research papers. These studies likely explore various aspects of malware detection, machine learning models for prediction, and the challenges in classifying and predicting malware threats in diverse computing environments [18†source].

Earlier work on malware prediction explored gradient-boosting decision trees, specifically focusing on LightGBM, a sophisticated and relatively new technique. This showed how effective our approach was at predicting malware attacks, with an accuracy of 73.89%. This outperformed traditional machine learning techniques and highlights our approach's appropriateness for big datasets in the cybersecurity domain. (V. Patel; S. Choe; T. Halabi, 2020)

Further, some papers demonstrated creating a semantic transition matrix to illustrate the true links between API functions by simulating the behaviour of API call sequences for both malware and goodware, thus showing a false positive rate of 0.010 and an average detection precision of 0.990. This showed the malicious nature of the API call sequence having an average accuracy of 0.997, based on the original API calling functions, hence providing a proactive means of preventing harmful payloads as opposed to depending on damage repair after post-execution detection. (Eslam Amer; Ivan Zelinka, 2020)

**Description Of Data**

The dataset is comprised of a multitude of rows, each representing a unique machine, identified by its "Machine Identifier." The "Has Detections" column provides the ground truth, indicating whether malware was detected on the machine. The data is sourced from both heartbeat and threat reports collected by Windows Defender. The nature of this data offers a rich source of information for the development of our predictive model. key data files, `asmoutputfile.csv` and `byteoutputfile.csv`, are loaded, each containing features extracted from malware samples.

**Asm file:** The file contains 10,868 entries and 52 columns. It also includes an 'ID' column followed by various feature columns representing counts or metrics extracted from assembly language files of the software. These features include counts of different sections (.text, .data, etc.), assembly instructions (like jmp, mov, retf), and register usages (eax, ebx, ecx, etc.). The first few rows show varying counts across different features, indicating diverse characteristics in the assembly files of different software.

**Byte file:** This file also has a wide format with 258 columns, including an 'ID' column and 257 feature columns. The features are likely byte code frequencies or byte-level components extracted from the software files. Each column, named with hexadecimal values (0, 1, 2, ..., ff), represents a different byte code. The data in the first few rows show a wide range of values across other byte codes, suggesting significant variability in the byte-level characteristics of the software samples

**Analysis**

The code in the Jupyter Notebook starts by importing the necessary libraries and suppressing warnings. This setup is crucial for data handling and visualization. The files provide a rich set of features, capturing different aspects of the software's structure and content at the assembly and byte code levels. The variability in feature values across different samples indicates a diverse dataset, which is beneficial for training robust malware prediction models. The high dimensionality of the data, especially in `byteoutputfile.csv,` might pose challenges regarding computational resources and model complexity. These datasets are comprehensive bases for building and training machine-learning models for malware prediction. The variety and depth of the features are likely to contribute significantly to the accuracy of such models. Thus, careful feature selection and pre-processing are carried out to manage the complexity and optimize the performance.

**Data Preparation and Cleaning**

The ‘For’ loop is utilized for the iteration of data. For every file, the occurrences of every byte are counted and then stored in the form of a NumPy array. The first column of the matrix is left as 0s. After that, the data frame is created using ‘Pandas’ and indexed by file names ‘ID’. This data frame is saved in the form of a ‘CSV’ file as well.

**Feature Selection**

The notebook prints the count of unigram byte features (257) and unigram asm features (51). This indicates a focus on feature extraction from assembly and byte files, which are typical in malware analysis. A visualization is generated to show the distribution of these features. This is important for understanding the dataset's structure and the diversity of malware samples.

**Data Loading**

The PE file features are loaded into a Pandas data frame from the CSV file, thus displaying the fundamental information about the data frame.

**Data Preprocessing**

This process is carried out by handling missing values, dropping the unnecessary columns, and extracting the features and labels.

**Data Training**

‘train\_classifier is utilized for splitting the data into testing and training tests (20:80). Further, a Random Forest classifier is initialized, along with the training of the classifier and the determination of its accuracy on the test set. Preferably, MlawareClassfier is used for the classification of PE files as benign or malware which follows the Random Forest algorithm. The trained classifier is saved into the file ‘malware\_classifier.joblib’ employing the library ‘joblib’ so that the classifier’s loading can be possible for making predictions without any retraining requirement.

**Data Prediction:**

‘predict\_sample’ is employed for predicting a particular sample utilizing a trained classifier.

**Visualization:**

The visualization of the first tree of a Random forest is also carried out using the function ‘visualize\_tree’.

**Testing and Evaluation**

Cross-validation is done to assign the generalization feature of our model for an independent dataset. For this, the dataset is split into many folds, thus, training the model on various combinations of the folds, as well as evaluating its performance. These results are displayed as well using the ‘print’ function which provides the score in terms of various performance metrics like precision, recall, accuracy, and F1-score across several folds.

**Results and Discussion:**

While using the Gaussian Naïve Byes classifier, alpha = 1e-09’ is found to be the best parameter, being able to handle the issues of 0 variances for continuous data. Further, the best accuracy score is got as 66.33% during the process and after tuning, the accuracy of the test set is determined as 66.39%. Likewise, on looking at the output of testmetrics multiclass, very low accuracy is obtained. Also, the log loss is higher. This suggested that our model is not efficient in doing well for many classes. Also, it indicates that the model needed further tuning, feature engineering, or other adjustments.

Likewise, while working with pixels, it is seen that with an accuracy score of 0.899 on the training set and 0.897 on the test set, the Bernoulli Naive Bayes classifier outperforms the others on both sets of data. Further, in terms of accuracy, the Gaussian Naive Bayes classifier performs second best, scoring 0.723 on the training set and 0.729 on the test set. 0.652 on the training set and 0.631 on the test set is the accuracy score of the Multinomial Naive Bayes classifier, which performed the worst. Thus, the Bernoulli Naïve Byes classifier is determined to have the best performance. When XGboost was used, the best value of alpha was 10 at which the log loss and the number of misclassified data are largely reduced, thus indicating the better performance of the model.

Insert the different heatmaps and explain them as well

**Limitations & Other Considerations**

Though the dataset presented here is divided based on timeframes, there are inconsistencies observed in the alignment of the cross-validation, public, and private scores due to the various complexities and data collection constraints. Further, this dataset can not accurately reflect the typical distribution of malware on Microsoft customers' devices; thus showing a higher ratio of malware-infected machines for study purposes.

**Conclusion**

We have interspersed with comments and documentation, which explain the rationale behind each step and make the notebook easier to understand for others. The notebook likely ends with a transformed dataset, ready for further analysis or as input to a machine learning model.

The analysis serves as a comprehensive guide for preparing a dataset for analysis or modelling. It covers essential steps from data cleaning to feature engineering, ensuring the data is in an optimal state for subsequent processes. The inclusion of EDA also suggests a thorough approach to understanding the dataset before any advanced analysis or model building.

**References**

# - V. Patel, S. Choe and T. Halabi, "Predicting Future Malware Attacks on Cloud Systems using Machine Learning," 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), <https://ieeexplore.ieee.org/abstract/document/9123037>

# 2020. Retrieved in October, 2023.

-Amer, Eslam, and Ivan Zelinka. *Computers & Security* 92. "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence."  <https://www.sciencedirect.com/science/article/abs/pii/S0167404820300444>.

2020. Retrieved in October, 2023.

- <https://www.kaggle.com/competitions/microsoft-malware-prediction/data>

# -**ZHUANG JIA. “**Forked LightGBM MSFT Malware predictor”, <https://www.kaggle.com/code/jiazhuang/forked-lightgbm-msft-malware-predictor>. 2018. Retrieved in October, 2023