Contents

Malware Analysis Using Machine Learning Methods to Classify Malware and Malware Type

Abraham Avila Klaus King

Abstract

In our Machine Learning Project, we will be exploring a variety of different machine learning solutions to well-established datasets from the Canadian Institute for Cybersecurity (CIC). Specifically, we will be examining the domain of malware where we will classify a malignant or benign program, as well as, defining what category of malware of the program if it is malicious. The preferred dataset that we will be using for these tasks is the CICMalMem-2022 made and provided by CIC. Methodology will be done by testing the machine learning algorithms to first classify a binary classification task (Benign vs. Malignant) before we test the machine learning models on classifying the types of malware (Ransomware, Trojan, Spyware, vs. Benign). The goal with this project is to determine the performance of each model, and determine the champion model that performed the best from the sought-out tasks. This is in hopes that the models trained and tested here could be used for real-world application, and perform within live environments to be further examined and consider for turning options to improve the model.

1 Introduction

Cybersecurity is a constantly evolving field with new methods or techniques to both defend and attack information systems. As hacking techniques progress and become more sophisticated, we want to find a way to defend ourselves from other potential attack vectors before we become victims to a zero-day or a new type of attack. Therefore, in this paper we want to focus on the potential attacking techniques that malware can embody and successfully infect our infrastructure or personal systems. One such technique that malware can use to effectively evade modern anti-virus or anti-malware software is obfuscation. Obfuscation is an evasion technique used by malware developers or hackers to over complicate or obscure the functionality of the program to confuse modern anti-malware software in letting the malware to pass off as a normal program. As an example, if we wanted to print "Hello, World!" to the console in a high-level programming language we would simply print the string containing this phrase inside of a print function. However, we can obscure this functionality by putting this string inside of a character array then printing each character to the console one at a time. We could further obscure the program by implementing useless or redundant functions so that malware analysts would have a difficult time decompiling or decoding the program's functionality.

Obfuscation is one of many techniques that malicious actors could use to infiltrate or prevail through quality anti-malware software and persist to infect the information system. Therefore, this research is focused on how to detect such techniques without having to use complicated software, such as Ghidra, to decompile and analyze the program to determine its validity. In our research, we used a novel dataset created by the Canadian Institute of Cybersecurity (CIC) to train and test with a plethora of machine learning classifiers to detect malware from a memory dump. The dataset, CICMalMem-2022, is a tabular heterogeneous structure that contains 50 percent benign and 50 percent malicious samples for binary classification, and contains a categorical feature that can be further broken down to sub-classes of the malicious class, such as Ransomware, Spyware, and Trojan. Further details of the dataset are mentioned in the Dataset section. We then use this dataset to train and test with various classification models, provided by the Sci-Kit Learn API, to classify the samples if the program is a benign or malicious program and classify what type of malware the sample may fall into. This is further explained in the Methodology/Experiment section of this paper. Finally, we evaluate the results of the experiments and formulate a conclusion to which model would be best suited for implementation to improve modern anti-malware detection systems. This is further delineated in the Results section. Other sections covered in this paper will go over Feature Engineering/Selection, Limitations, and Future Works to further explain the research done to obtain its results and conclusions.

2 Dataset

The dataset provided by the Canadian Institute of Cybersecurity is a tabular dataset in a form of a comma separated value (CSV) structure. This dataset contains numeric, categorical, and nominal values which contain either integer, floating point, or string/object values. In this research we determined that the nominal feature as the binary target for this experiment. Additionally, we have also determined that the categorical feature is the multi-classification target for this research. The rest of the features within this dataset are used for training or testing the machine learning algorithms that we chose to experiment with. To explain how the dataset was created, the researchers at CIC utilized Cybersecurity tools to extract data from a memory dump file. Specifically, they used a tool called Volatility, which is a digital forensic tool used to extract information about a memory dump file from a machine. In the case of the researcher's data mining process, they used virtual machines (VM) to run well known malware to create said files to be used in creating the dataset. However, before they converted the data into a CSV, they used a feature extraction tool called VolMemLyzer to extract 26 additional learnable features to the dataset

[1]. Once they have extracted these features then they finally converted the data into a CSV via python using the Pandas module. The process of their feature extraction method is visualized in the diagram as seen in Figure 1 [1]. Lastly, the researchers made this dataset balanced where both benign and malicious programs make up half of the dataset. However, instead of running a huge number of benign programs on to the VM, they used the SMOTE algorithm to fill the rest of the benign half in the dataset to merit a balanced dataset [1]. This can be visualized in Figure 2 as the dataset contains half for each binary class [2]. Additionally, they have included different varieties of malware to merit a categorical feature. This can be visualized in Figure 3, where we can see that the benign class makes the bulk of the multiple classes, and the other types of malware covers around an evenly portion within the malignant, or malicious, half of the dataset [2].

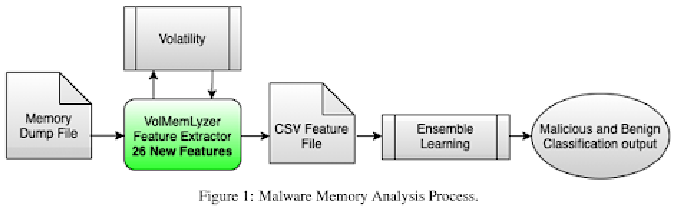
Volatility
Memory Dump File
VolMemLyzer Feature Extractor 26 New Features
CSV Feature File
Ensemble Learning
Malicious and Benign Classification output
Figure 1: Malware Memory Analysis Process.

Figure 1: CICMalMem Feature Extraction Diagram

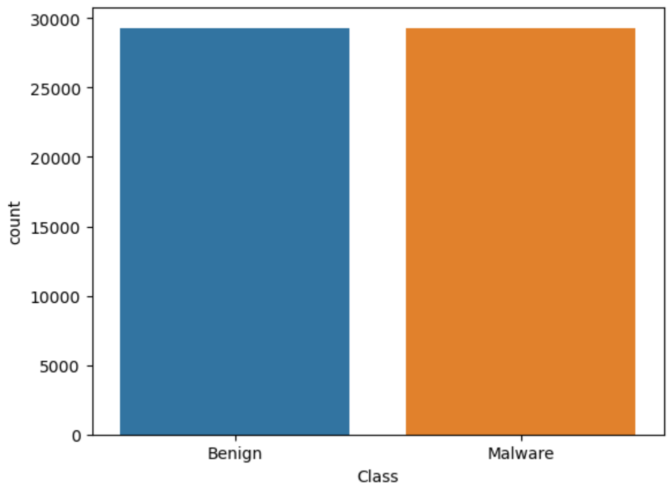

Figure 2: Bar graph of Binary Class Bins

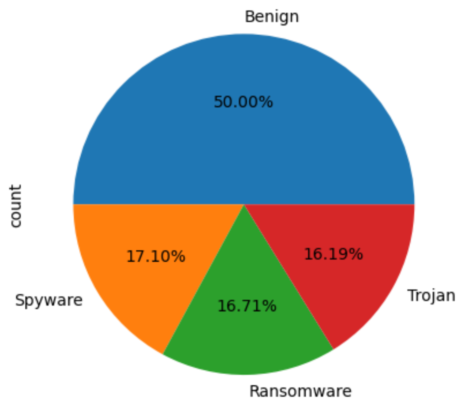

Figure 3: Pie Chart of Multiple Class Percentage

3 Feature Engineering/Selection

In this section, we detail the process of selecting features and reducing dimensionality from the CICMalMem-2022 dataset to enhance the performance of our machine learning models. Feature engineering is a critical step in machine learning as it helps to improve model accuracy by adjusting features that better capture underlying patterns in the data. We started by examining the dataset for any redundant, irrelevant, or otherwise low-importance features. Using the Mutual Information (MI) score, we evaluated the relevance of each feature in predicting the target variables. Features with a low MI score were discarded to reduce noise and computational complexity. This was tested by removing features one at a time, from lowest to highest MI scores and inversely. Also, we removed features that contained strings such as the Class and Category features, as these were our target variables for classification. We handled the target variables by applying label encoding to convert them into numerical values. This encoding is necessary as most machine learning algorithms require numerical output.

4 Methodology/Experiment

For each model, we first grab the dataset and drop the features with a small MI score and the features that contain string objects (our Class and Category features), the rest of the features are kept and assigned to a variable namely "X". Once these features are dropped for our cleaned dataset, we then label encode our targets, which are the Class and Category features, where each class is encoded into numbers for the models to compute the classes as numbers. We then perform a train test split for the cleaned dataset and the target vector.

We then fit the train split dataset into the model where each model either had default parameters or some fixed parameters such as "random state" and "n_estimators". After training the model we then test the model using the prediction function to calculate the essential metrics such as Accuracy, F1-Score, Precision, and Recall. Once these metrics are calculated, we examine these scores to see if the model over-fitted or had satisfactory scores. If over-fitting had occurred, then we put the model through a k-fold cross validation method to further confirm the model's performance. We then generate graphics such as a confusion matrix, ROC, and bar graphs to better understand how the models performed and compare each model's performance.

5 Analysis

During the entire experiment process we faced some problems. The most notable one was realizing the models were over-fitting to the dataset when it comes to multi-classification task, but perfect scores for the binary classification task. With that said, the other problem when using this dataset was the categorical feature containing multiple labels that has the type of malware the sample is in, the family and it's hash was also included in the same sample. This needed to be relabeled, so we used the Microsoft Excel software to view the dataset in its entirety. The benign labels seemed to be already labelled, but for the other malware types we needed to relabel the section of malware type to its respective name, such as "Spyware- Gator-1bdcd3b777965f....692a54a7b5e-1.raw" to "Spyware". The next problem that we encountered was the choice between keeping the category feature with the main dataset or removing it. This may seem obvious, however, to some unsuspecting data analysts or scientists they won't realize that this feature effectively gives the machine learning algorithm the answers to the tests before and during testing. We did a one-hot encoding implementation to this category feature, but because we realized that this feature was causing the machine learning algorithm to perfectly classify the labels it seemed too good to be true. Additionally, if the model were to go out in the field and perform live detection, this model would be severely over-fitting to new samples and might rely on the categorical feature to classify properly. Therefore, we decided to remove this feature and the other features that contained small MI-score values. Speaking of MI-scores, the highest value compared to the other features was a feature called "scancve.nservices". We don't exactly know what this feature entails or signifies in terms of malware analysis, but we noticed that this feature contained a seemingly simple analysis to merit a perfect binary classification. For example, if the integer of that feature indicated a higher number that is above 390 or so then it was considered benign and if it was lower than this then it was considered malicious. Of course, this would indicate a valuable feature within the dataset, but could be in question of causing over-fitting. However, it seemed that this was not the only feature to have a relatively high MI-score, so we must keep this and the other features as it seems that the other features are helping the model to properly classify a given dataset. At this point, we couldn't go too much deeper into this phenomenon as this would merit whole new research of itself to further experiment with combinations of these features to merit a better performance, or to better understand what role each feature plays when the machine learning algorithm uses those features. Therefore, feature engineering would merit a better understanding of this area for future data analysts or scientists that may use this dataset for another research.

6 Results

Our experiments revealed that Random Forest and Extreme Gradient Boosting (XGBoost) classifiers performed exceptionally well in both binary and multi-class classification tasks, indicating the efficiency in detecting and classifying malware. The performance of the models are displayed in Figure 4. For binary classification, most models hit 1.0 for both accuracy and F1-scores. Linear models like Support Vector Classifier (SVC) and Logistic Regression we're only at a 0.99 for both accuracy and F1-scores. Not the biggest difference, but this indicates that our ensemble methods are highly effective in distinguishing between benign and malicious programs. In the multi-class classification task, again, both Random Forest and XGBoost models performed well, although the overall accuracy was slightly lower due to the increased complexity of the task. Both Random Forest and XGBoost classifier achieved an accuracy of 0.87. The F1-scores for these models were also high, demonstrating their ability to correctly classify different types of malware. Again, linear models such as Logistic Regression and SVC showed mediocre performance relative to the other models. To address the issue of over-fitting, particularly noticeable in the classification task, we used k-fold cross-validation. This technique involves splitting the dataset into k subsets(folds) and training the model k times, each time using a different subset as the validation set and the remaining subsets as the training set. In our experiments, we used 10-fold-cross-validation, typical for accurately measuring model performance. The results from the cross-validation process confirmed the initial findings. The ensemble methods maintained their high performance across all folds, with minimal variance in their accuracy and F1 scores (Figure 5). For instance, the Random Forest classifier's accuracy during cross-validation varied between .80 and .86 for multi-classification, indicating stable and reliable performance. Similarly, the XGBoost classifier showed consistency between .75 and .84. The cross-validation results also highlighted the relative under-performance of models like LinearSVC, GaussianNB, AdaBoost, Logistic Regression, MLP, and QDA, which exhibited greater variance in their accuracy scores, further reinforcing the advantages of ensemble methods for this dataset.

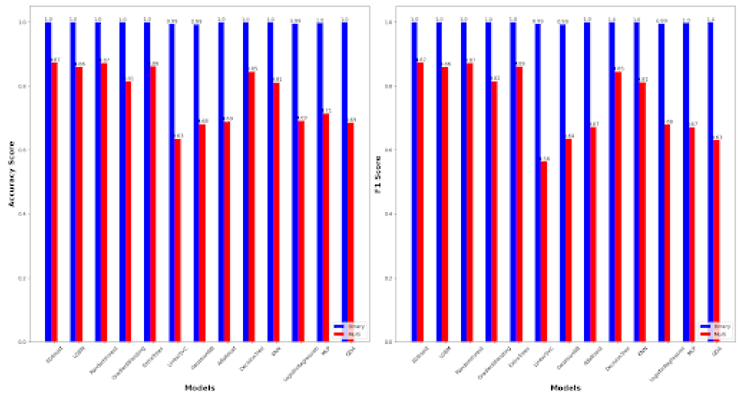

Figure 4: Performance of various models in both classification task

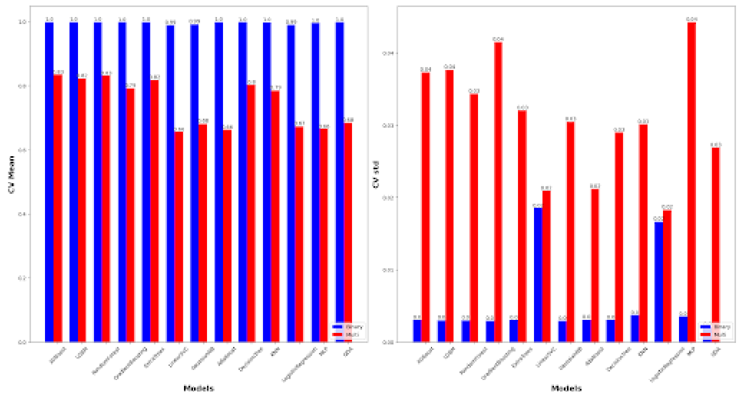

Figure 5: Performance of various models from cross-validation.

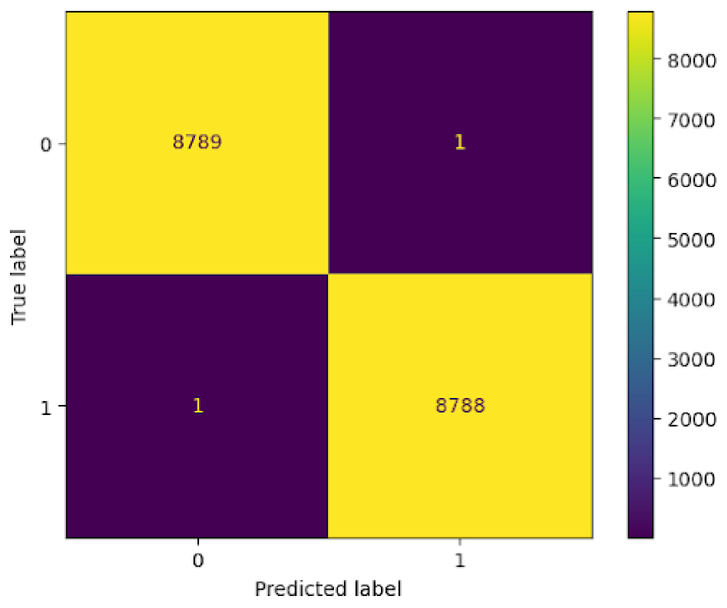

Figure 6: Confusion Matrix for Binary Classification.

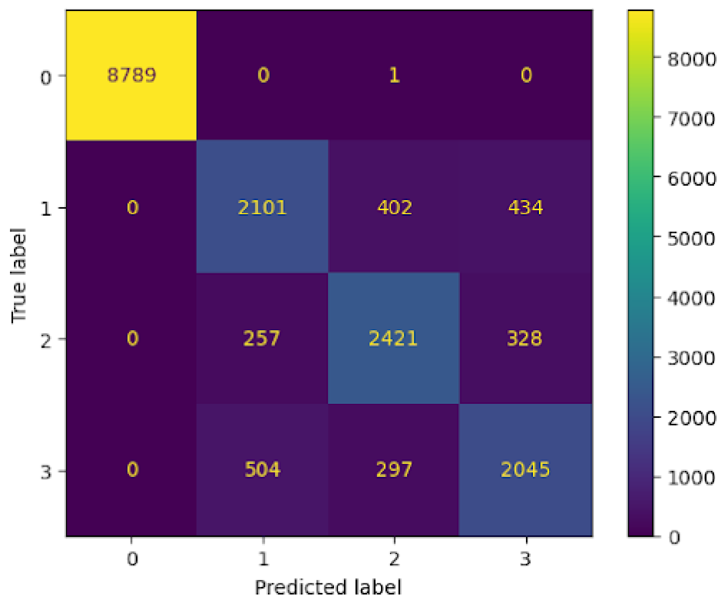

Figure 7: Confusion Matrix for Multiple Classification.

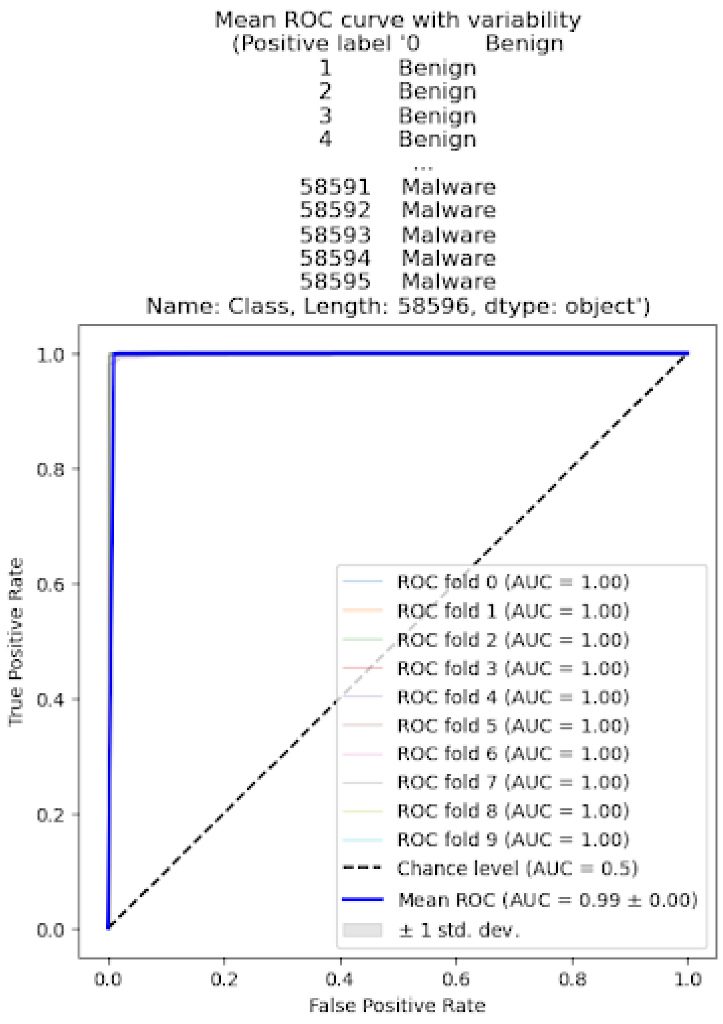

Figure 8: Mean ROC Curve for XGBoost in Binary Task.

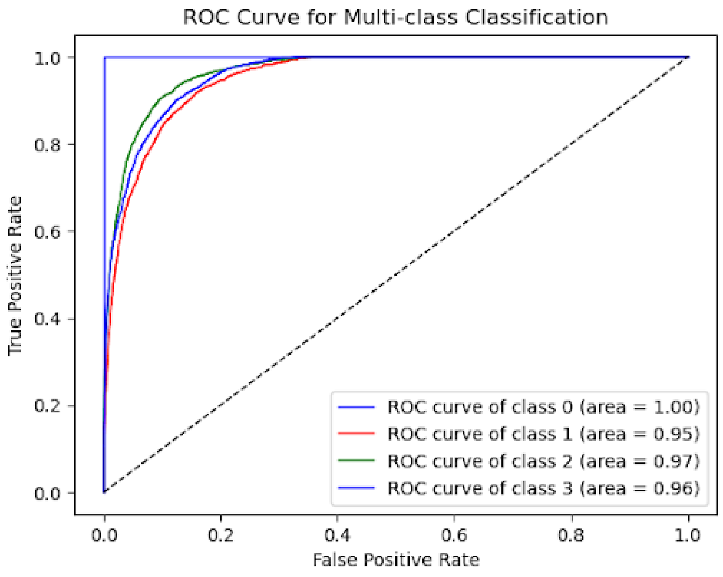

Figure 9: ROC Curve for XGBoost in Multi-Class Task.

7 Limitations

The limitations to this experiment involve the lack of in-depth knowledge of the features within the dataset. To truly understand what features were significant or insignificant we would need to research further into the tools that were used by the researchers and understand what each feature meant according to the VMs memory or processor. Speaking of processors, we seem to have no information about how many processors that the researchers used for the VMs that were infected by the malware. This is problematic as real-world malware that uses obfuscated techniques to evade detection would first check if the environment it's in is, in fact, a legitimate machine (i.e. calculating the specifications of the machine the malware is in). Without this information, we don't know if this dataset would be applicable in real-time detection as new variants of malware could still evade this unique type of detection. Lastly, we could not perform hyper-parameter optimizations due to the constraints of resources to use for this research. If we had more processing power or utilized threading, then we could have also taken the time to perform hyper-parameterization to test the models at their most optimal state, then cross validate their performances. This could have changed the results of models such as Linear Support Vector Classifier, Logistic Regression, Quadratic Discriminant Analysis, and Gaussian Naive Bayes as the dataset had multi-collinearity which could have improved the models scores after further cleaning the dataset of this problem. We did not do this type of cleaning as we wanted to specifically test the models with their default parameters and on a raw dataset with some dropped features due to columns being either 0's or low MI-scores.

8 Future Works

Several areas can be explored to build upon the findings of this research. Hyper-parameter optimization should be conducted for each model to fine-tune their performance. Random search techniques can be used to find the optimal parameters, potentially improving the accuracy of the models. More sophisticated engineering techniques should be explored. This could involve creating new features from the existing ones or applying dimensionality reduction techniques such as Principal Component Analysis (PCA) to reduce the complexity of the dataset while retaining the most informative features. Additional data sources should be incorporated to enhance the dataset. By including more divers samples of malware and benign programs, the models can be trained to recognize a broader range of threats. This will improve their generalization capability and effectiveness in real-world scenarios. Exploring advanced machine learning techniques such as deep learning could provide further improvements in malware detection. Neural networks, particularly those designed for sequential data such as Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, may offer better performance in capturing the temporal patterns of malware behavior. The models should be tested in a live environment to evaluate their real-time detection capabilities, implementing the models in an actual-anti malware system and monitoring their performance against new and emerging threats will provide valuable insights into their effectiveness and areas for further improvement.

9 Conclusion

This research demonstrates the potential of machine learning models in detecting and classifying malware using memory dump data. By leveraging a balanced and well-structured dataset provided by the Canadian Institute of Cybersecurity, we were able to train and evaluate a collection of models, with ensemble methods showing the best performance. Despite some limitations, the results are promising and are indicative of a viable approach to enhancing modern anti-malware systems. Future work should focus on refining these models, incorporating additional data sources, and exploring advanced techniques to further improve malware detection and classification capabilities.

References

[1] Canadian Institute for Cybersecurity. *CICMalMem-2022 Dataset.* Available at: `https://www.unb.ca/cic/datasets/malmem-2022.html`.

[2] Kaggle Malware Memory Analysis. *Malware Memory Analysis*. Available at: `https://www.kaggle.com/code/alperkaraca1/malware-memory-analysis`.

[3] ScienceDirect *Recurrent neural network for detecting malware*. Available at: `https://www.sciencedirect.com/science/article/pii/S0167404820303102`.

[4] Jounral of Machine Learning Research *Random Search for Hyper-Parameter Optimization*. Available at: `https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf`.