IOT Traffic Malware Detection Using AutoML

Karina Chichifoi

Master in Computer Engineering - University of Bologna

A.Y. 2022-2023

Abstract

In the previous report we developed machine learning models, using random forest and support vector machines (SVMs), to detect specific malware in IoT devices. We improved them through grid search cross-validation, which, given a range of hyperparameters, provides the optimal combination. In this report we discuss an alternative approach based on automated machine learning (AutoML) and, more specifically, AutoGluon: such approach requires few lines of code to train highly accurate machine learning models, finding the best configuration and best model for a given task.

1 Introduction

In the project "IoT Traffic Malware Detection Using Machine Learning models" we explored possible machine learning approaches, such as random forest and support vector machines, for detecting specific malware affecting IoT devices. We improved the previous models using grid search cross-validation to find the best configuration of hyperparameters for each model. We noticed that:

- the process of labelling and preprocessing in general is expensive;
- in order to obtain the best model we have to try different hyperparameters and models;
- even if we employ grid search cross-validation, we still need to provide a suitable range for each hyperparameter, which may be non-trivial;
- cybersecurity is a very dinamic field, and new challenges and bypasses often occur (e.g., adversarial attacks);
- we have to deal with a great quantity of data.

In this report we will discuss about an alternative approach, automated machine learning, which could improve model deployment and, at the same time, make machine learning accessible to people who do not work in the AI field (e.g., cybersecurity experts). The source code is available at this repository: https://github.com/TryKatChup/ML-IOT-malware-analysis.

2 AutoML

Automated machine learning (AutoML) automates the process of applying machine learning to real-world problems. AutoML generally covers:

- preprocessing of raw data, such as column type detection, labeling and task detection (i.e., classification, regression, clustering, ranking);
- feature engineering (i.e., feature selection, feature extraction, transfer learning and detection of missing values);
- model selection;
- ensembling, namely using a combination of multiple models, as it generally performs better than using a single model;
- hyperparameter optimization using, for instance, grid search, Bayesian optimization, gradient optimization, early stopping, etc.

There are several libraries and frameworks that implement AutoML functionality. Some popular open-source libraries include:

- PyCaret: low-code machine learning library in Python that automates machine learning workflows.
- Auto-sklearn: this library is built on top of Scikit-learn and automates the selection of algorithms and hyperparameters for classification and regression tasks.
- AutoGluon: developed by Amazon, it provides easy-to-use interface for training, deploying, and evaluating machine learning models with little or no code.

AutoML does not rely on machine learning models only, but also on deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). It has several advantages, including:

- efficiency: AutoML can automate time-consuming tasks;
- increased accuracy: AutoML can often achieve better results than a human by systematically trying different combinations of algorithms and hyperparameters, preventing overfitting and underfitting;
- accessibility: AutoML can make advanced machine learning techniques more accessible to non-experts, allowing a broader range of users to build predictive models.

3 AutoGluon

Since the IoT-23 dataset comprises tabular data, we chose the AutoGluon-Tabular framework for our experiments. Via a simple `fit()` call, it can produce highly-accurate models for classifying tabular data. AutoGluon with tabular data is used for both classification and regression problems.

Furthermore, tests on a suite of 50 classification and regression tasks from Kaggle and the OpenML AutoML Benchmark reveal that AutoGluon is faster, more robust, and much more accurate than its competitors. Its core principles are:

- simplicity: a user can train a model on the raw data directly, without knowing the details about the data and ML models;
- robustness: the framework can handle a large variety of structured datasets and ensures training succeeds even when some of the individual ML models fail;
- fault tolerance: the training can be stopped and resumed at any time;
- predictable timing: it returns the results within the time range specified by users.

4 Training machine learning models

4.1 Preprocessing

As we said before, AutoGluon can be used to preprocess data. However, as we have 48 GB of different files, we cannot afford this expensive operation due to limited memory availability. Therefore, we employ the same subset of our previous project "IoT Traffic Malware Detection Using Machine Learning Models", which is already balanced and preprocessed.

4.2 Automated steps

The following operations were automated by AutoGluon:

- model selection;
- hyperparameter optimization;
- analysis of obtained results.

4.3 Models

AutoGluon uses both machine learning and custom deep learning models in a predefined order. This ensures that reliably performant models (e.g, random forest) are trained prior to more expensive and less reliable models (e.g., k-nearest neighbor). The models trained include random forest and decision trees, ensembles and neural networks.

4.4 Metrics

AutoGluon employs the following metrics:

- `score_test`: accuracy score on test set
- `score_val`: accuracy score on validation set
- `pred_time_test`: time required to compute prediction on test set
- `pred_time_val`: time required to compute prediction on validation set
- `fit_time`: time required to train the model
- `pred_time_test_marginal`: time required to compute predictions on the test set (ignoring inference times for base models).
- `pred_time_val_marginal`: time required to compute predictions on the validation set (ignoring inference times for base models).
- `fit_time_marginal`: time required to train the model (ignoring base models).
- `stack_level`: stack level of the model (a model with stack level N can take any set of models with stack level less than N as input, with stack level 1 models having no model inputs)

4.5 Results

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	ExtraTreesEntr_BAG_L2	0.999921	0.999762	4.769219	11.467763	232.962383	0.109097	1.479923	1.124924	2	True	22
1	ExtraTreesGini_BAG_L2	0.999921	0.999762	4.774835	11.432402	233.052523	0.114714	1.444561	1.215064	2	True	21
2	RandomForestEntr_BAG_L2	0.999921	0.999723	4.776412	11.389512	234.959031	0.116291	1.401671	3.121572	2	True	19
3	RandomForestGini_BAG_L2	0.999921	0.999703	4.780225	11.385758	234.891113	0.120103	1.397918	3.053654	2	True	18
4	WeightedEnsemble_L2	0.999842	0.999802	1.052884	1.688675	26.858392	0.004531	0.003320	5.666573	2	True	14
5	NeuralNetFastAI_BAG_L2	0.999842	0.999822	5.876441	10.940377	292.935756	1.216320	0.952536	61.098296	2	True	15
6	WeightedEnsemble_L3	0.999842	0.999842	6.103206	11.267824	310.615211	0.002182	0.003327	4.812466	3	True	26
7	RandomForestGini_BAG_L1	0.999762	0.999624	0.109604	0.897619	1.492563	0.109604	0.897619	1.492563	1	True	6
8	LightGBM_BAG_L2	0.999762	0.999762	4.731534	10.060448	244.550537	0.071412	0.072607	12.713078	2	True	17
9	LightGBMLarge_BAG_L2	0.999762	0.999782	4.772942	10.157421	263.538974	0.112820	0.169580	31.701515	2	True	25
10	LightGBMXT_BAG_L2	0.999762	0.999762	4.884704	10.311961	244.704448	0.224582	0.324121	12.866989	2	True	16
11	XGBoost_BAG_L2	0.999762	0.999663	4.970886	10.147380	248.716908	0.310764	0.159539	16.879449	2	True	23
12	CatBoost_BAG_L2	0.999683	0.999822	4.749632	10.080818	279.429843	0.089510	0.092978	47.592384	2	True	20
13	NeuralNetTorch_BAG_L2	0.999683	0.999525	5.788403	10.862667	287.679106	1.128281	0.874826	55.841646	2	True	24
14	RandomForestEntr_BAG_L1	0.999604	0.999723	0.099203	0.755432	1.305125	0.099203	0.755432	1.305125	1	True	7
15	XGBoost_BAG_L1	0.999604	0.999564	0.439709	0.260700	7.068130	0.439709	0.260700	7.068130	1	True	11
16	LightGBMLarge_BAG_L1	0.999445	0.999604	0.190340	0.284174	6.659400	0.190340	0.284174	6.659400	1	True	13
17	LightGBM_BAG_L1	0.999445	0.999544	0.319102	0.385049	6.159164	0.319102	0.385049	6.159164	1	True	5
18	CatBoost_BAG_L1	0.999366	0.999227	0.076305	0.079099	18.450965	0.076305	0.079099	18.450965	1	True	8
19	ExtraTreesEntr_BAG_L1	0.999366	0.999386	0.118302	0.798300	0.973575	0.118302	0.798300	0.973575	1	True	10
20	ExtraTreesGini_BAG_L1	0.999208	0.999366	0.122299	0.794334	0.969857	0.122299	0.794334	0.969857	1	True	9
21	LightGBMXT_BAG_L1	0.999049	0.999049	2.024941	4.014495	13.374473	2.024941	4.014495	13.374473	1	True	4
22	NeuralNetTorch_BAG_L1	0.998653	0.998910	0.242557	0.160240	105.400009	0.242557	0.160240	105.400009	1	True	12
23	NeuralNetFastAI_BAG_L1	0.995959	0.995602	0.605527	0.462065	69.890762	0.605527	0.462065	69.890762	1	True	3
24	KNeighborsDist_BAG_L1	0.919889	0.921649	0.157709	0.541724	0.045432	0.157709	0.541724	0.045432	1	True	2
25	KNeighborsUnif_BAG_L1	0.912441	0.913546	0.154525	0.554609	0.048004	0.154525	0.554609	0.048004	1	True	1

Figure 1: All models tested by AutoML

The best model, based on average accuracy on both validation and test set, was `WeightedEnsemble_L3`, yielding an accuracy of 99.98% on the two subsets. There are models, like `ExtraTrees` family, which yield a higher test accuracy (99.99%), but a lower validation accuracy (99.97%). Of course, the best model depends on the task and the time given. If we consider a compromise between inference time and accuracy, the best model is `RandomForestGini_Bag_L1` (one order of magnitude faster than `WeightedEnsemble_L3`, with 0.01s inference time). Neural networks perform better with two stack level, but machine learning models are generally faster and more accurate.

Notice the best model in accuracy is from Ensemble family, as we know they make better predictions and achieve better performance than any single contributing model.

5 Conclusions

In conclusion, the utilization of automated machine learning (AutoML) and AutoGluon specifically, offers a comprehensive and effective solution for detecting malware in IoT devices. With its focus on simplicity and accessibility, AutoGluon automates time-consuming tasks such as preprocessing, feature engineering, model selection, and hyperparameter optimization. Our experiment on the IoT-23 dataset showcases the potential of AutoGluon as a convenient tool for IoT traffic malware detection, reducing the cost and effort required for manual preprocessing and hyperparameter optimization, allowing non-experts to improve malware detection in IoT devices for more challenging events, such as evasive targets and new classes of malware.

References

[1] Sebastian Garcia, Agustin Parmisano, and Maria Jose. "IoT-23: A labeled dataset with malicious and benign IoT network traffic". Version 1.0.0. In: (2020). DOI: 0.5281/zenodo.4743746.

[2] Nick Erickson et al. *AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data*. 2020. DOI: 10.48550/ARXIV.2003.06505. URL: https://arxiv.org/abs/2003.06505.

[3] *AutoGluon Tabular Predictor*. URL: https://auto.gluon.ai/dev/_modules/autogluon/tabular/predictor/predictor.html#TabularPredictor.leaderboard.

[4] *The Growing Role of Machine Learning in Cybersecurity*. URL: https://web.archive.org/web/20230108145359/https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity.