The Art-of-Hyper-Parameter Optimization with Desirable Feature Selection

Optimizing for Multiple Objectives: Ransomware Anomaly Detection

.

Priynka Sharma, Kaylash Chaudhary, and M.G.M Khan
School of Computing, Information and Mathematical Sciences,
University of the South Pacific, Suva, Fiji.
priynka.sharma@usp.ac.fj, kaylash.chaudhary@usp.ac.fj, mgm.khan@usp.ac.fj

Abstract. The development of cyber-attacks carried out with ransomware has become increasingly refined in practically all systems. Attacks with pioneering ransomware have the best complexities, which makes them considerably harder to identify. The radical ransomware can obfuscate much of these traces through mechanisms, such as metamorphic engines. Therefore, predictions and detection of malware have become a substantial test for ransomware analysis. Numerous Machine Learning (ML) algorithm exists; considering each algorithm's Hyper-parameter (HP) just as feature selection strategies, there exist a huge number of potential options. This way, we deliberate more about the issue of simultaneously choosing a learning algorithm and setting its HPs, going past work that tends to address the issues in isolation. We show this issue determined by a completely automated approach, utilizing ongoing developments in ML optimizations. We also show that modifying the information preprocessing brings about more significant progress towards better classification recalls.

Keywords: HP, Feature Selection, Optimization, Ransomware, ML classification algorithms, Data imbalance.

1 Introduction

The earlier decade has seen a detonation of ML exploration besides applications; particularly, deep learning strategies have empowered key advances in numerous application areas, for example, computer vision, speech processing, and game-playing [1]. In any case, the performance of numerous ML strategies is exceptionally delicate to a plethora of design decisions, which establishes an extensive obstruction for new users. This is especially valid in the booming field of deep learning, where designers' requisite to choose the right models, formulating approaches in addition to tuning HPs of these segments with sufficient executions [1, 3]. Although, this procedure only needs to rehash for individual applications. Even experts are frequently left with monotonous acts of experimentation until they recognize a decent arrangement of decisions for a specific dataset. The field of automated ML (AutoML) plans to settle on choices that are based on information-driven and objectives in an automated way [3]. AutoML makes state-of-the-art ML approaches available to domain researchers who are keen on applying ML yet do not have enough assets to realize the advancements in detail. However, the best-performing models for some modern applications of ML are getting bigger and in this way more computationally costly to organize. Therefore, authorities want to set as many HPs automatically as expected under any circumstances. A vast assortment of learning strategies exists, extending from artificially invigorated neural systems [1, 3] over kernel techniques to ensemble models [1, 11]. A typical attribute in these techniques is parameterization by a lot of HPs λ, which is set appropriately by the user to intensify the usefulness of the learning approaches. HPs are to design different parts of the ML learning algorithms and can have uncontrollably fluctuating consequences for the subsequent model and the situation demo levels [4, 5].

HP combs are usually performed manually, through dependable guidelines, or by testing sets of HPs on a predefined lattice [6]. Automating HP search is accepting total measures of consideration in machine learning, for example using benchmarking suites in addition to different activities. Automated methodologies previously appeared to outflank manual searches through authorities on a few subjects [5]. The limitations call for practical answers for the HPOs enhancement that satisfies numerous desiderata. Consequently, choosing the best arrangement of HP values for an ML model yielding directly with performance level. Although there exist several automatic optimization methods, yet these usually take significant resources, increasing the dynamic complexity to obtain a vast level of accuracy rate. HPO finds a tuple of HPs that yields an optimal model that minimizes a predefined loss function on given independent data [5]. The objective function takes a tuple of HPs and returns the associated loss. Cross-validation is often used to estimate this generalization for performances [5].

Our research displays a review of the quick-moving field of AutoML and precision optimization in the ML algorithm through HP tuning. This curiosity will, in the long run, lead to an ideal isolating hyper-plane realistic in both linear and non-linear classification problems towards ransomware anomaly detection.

1.1 Hyper-parameter Optimization (HPO)

In machine learning, model parameters are the properties of training information that will learn without a person during training by the classifiers. Model HPs are valued in ML models that can require various imperatives, loads, or learning rates to produce various information patterns, for example, the number of neighbors in K-Nearest Neighbors (KNN). HPs are significant by the fact that they legitimately control the practices of the training algorithms and influence the presentation of the models prepared. Selecting appropriate HPs undertakes a basic effort in the performance levels of ML models. HPs improvement is the way forward for a perfect model recognition [7, 8]. Reasonably, HPs tuning is only to streamline over model learning to locate the procedure in prompting the least error on the approval set. Therefore, HPs are the only knobs that can tune when as-assembling the appropriate ML algorithm model for anomaly detection or to any application as in Figure 1.

Figure 1. Highlights the model logic in any ML tuned environment. It shows the logical scheme and confirms on the calculation that Model design added with HP of individual parameters results in enhanced model parameters.

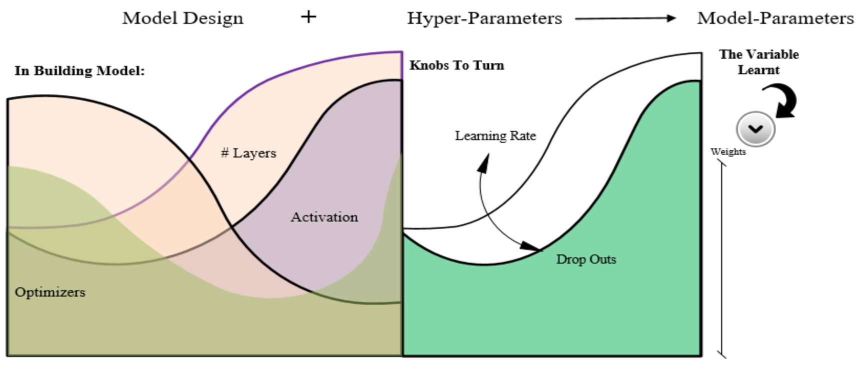


Fig 1: Momentary Portrayal of the HP Scheme.

Following [8], HP λ_p is restrictive on another HP λ_i, if λ_p it is dynamic and HP λ_j takes in approvals from a given set VP (I) $\subseteq \Lambda_i$, then we call λ_i the parent of λ_p. However, the restrictive HPs on the other hand can only be guardians of other dependent HPs, contributing to rising to a tree-organized space otherwise, sometimes, referred to as a directed acyclic graph (DAG) [2, 9]. The objective of HP improvement is to decide the HPs λ^*optimizing hypothetical execution of $A_{\lambda*}$ depends on a restricted measure of training information does = {(x_1, y_1) . . . (Xn, y_n)}. Hypothetical execution is approximated by parting into split training, and approval sets (Ds $_{(p)\,\text{train}}$ and $_{(p)\,\text{valid}}$). The learning volumes can be applied by $A_{\lambda*}$ to Ds $_{(p)\,\text{train}}$ and assessing the presentation of these volumes on Ds $_{(p)\,\text{valid}}$. This permits the HPs improvement into subject composed as:

$$C(\lambda) = \frac{1}{k}\sum_{P=1}^{k} l(\,A_{\lambda},\ Ds_{(p)\,\text{train}},\ Ds_{(p)\,\text{valid}}) \qquad (1)$$

$$\lambda^* \,\epsilon\, \frac{argmin}{\lambda\epsilon\Lambda}\ c(\lambda) \qquad (2)$$

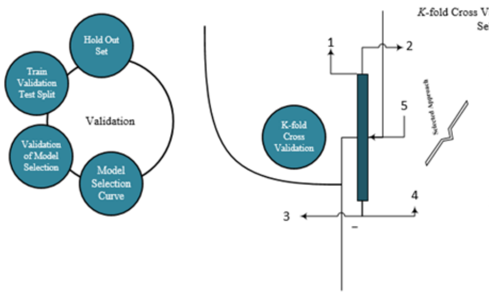
1.2 Model Selection

In model selection accountabilities, we attempt to locate the correct coherence among prediction and estimation of errors. If our learning algorithm ignores to discover an indicator with a little threat, it is imperative to understand over-fitting or under-fitting.

Under-fitting: The classifier learned on the training set is not sensitive enough to account for the data provided. In this case, both the training error and the test error will be high, as the classifier does not account for relevant information present in the training set.

Over-fitting: The classifier learned on the training set is too specific, and cannot be used to infer anything about unseen data accurately. Although training error continues to decrease over time, test error will begin to increase again as the classifier begins to make decisions based on patterns that exist only in the training set and not in the broader distribution.

The over-fitting and under-fitting will result in poor performance in any given model. Therefore, to refrain from these problems during an analysis phase of an ML model it is vital to follow a technique out from the given four techniques as depicted in Figure 2.



K-fold Cross Validation for Model Selection

```
**Input:**
Training set S = $(x_1,y_1)\ldots(x_m, y_m)$
set of parameter values $\Theta$
learning algorithm values $A$
integer $k$
**Partition** S into$S_1,S_2,\ldots S_k$
For each $\theta \in \Theta$
for $i = 1\ldots k$
$h_i, \theta = A(S\backslash S_i;\theta)$
error $(\theta) = \frac{1}{k}\sum_i^k = 1\ Ls_i(h_i,\theta)$
**Output** $\theta^* = argmin_\theta[error(\theta)]$
$h_{\theta^*}\ * = A(S;\theta^*)$
```

Fig 2. Model Selection Approaches.

K-fold Cross-Validation (Selected Method)

In specific applications, information is rare, and we would prefer not to "misuse" information on validation. The k-overlap, cross-validation methods intended to give a precise gauge of the genuine error without squandering an excessive amount of information. In k-overlap cross-validation, the first training set is parceled into k subsets (folds) of size m/k (for straightforwardness, expect that m/k is a number). For each fold, the algorithm prepares for a connection with different overlays thus the error is achieved through overlays. However, K-overlap, cross-validation is often applied for model selection (or parameter tuning).

1.3 The Common Optimization Strategy

A typical optimization procedure defines the possible set of hyper-parameters and the metric to be maximized or minimized for a given problem. Hence, in practice, any optimization procedure follows these classical steps as depicted in Figure 3.

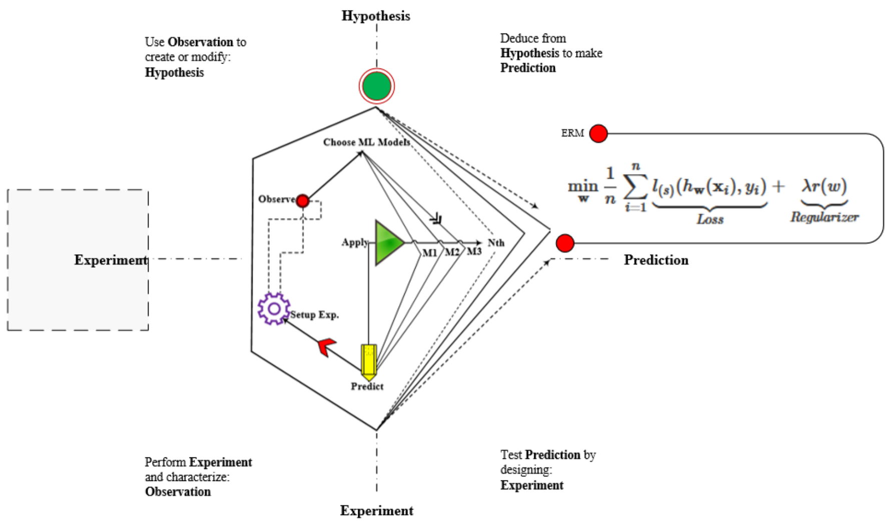

Hypothesis
Use Observation to create or modify: Hypothesis
Deduce from Hypothesis to make Prediction
ERM
$\min_{\mathbf{w}} \frac{1}{n} \underbrace{\sum_{i=1}^{n} l_{(s)}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer}$
Choose ML Models
Observe
Apply
M1
M2
M3
Nth
Experiment
Setup Exp.
Predict
Prediction
Perform Experiment and characterize: Observation
Test Prediction by designing: Experiment
Experiment

Fig 3: Illustrates an Optimization Strategy.

2 Result and Discussion

The experiment dataset was downloaded from VirusShare4, a website that keeps up a continuously updated database of malware for a few [10]. Table 1. Below reports the full list of ransomware families utilized in our research. To analyze the samples, the initial researches used Cuckoo Sandbox to automate the analysis.

Table 1: Data Description for Experiment Test set.

Data set	**Selected Attributes**		
Name: Ransomware	Missing: 0%	Distinct: 12	Unique: 0%
Type: Nominal			
Data set	Ransomware Anomaly Detection		
	Features	*Instances*	*Class*
Selected Attribute	16382	1524	12
After Feature Selection	14631	992	Samples Used
0	942.0 wt.		Goodware
1	50.0 wt.		Critroni
2	107.0 wt.		CryptLocker
3	46.0 wt.		CryptoWall
4	25.0 wt.		Kollah
5	64.0 wt.		Kovter
6	97.0 wt.		Locker

7	59.0 wt.	Matsnu
8	4.0 wt.	PGPCoder
9	90.0 wt.	Reveton
10	6.0 wt.	TeslaCrypt
11	34.0 wt.	Trojan-Ransom

To achieve the objective of this research, the classification methods on ransomware detection datasets were applied, through the WEKA environment. WEKA is an information mining structure made by the University of Waikato in New Zealand that executes information mining algorithms working on the JAVA language [12]. WEKA is the best state-of-the-art facility for making, ML systems, and their application to genuine information mining anomalies. It comprises ML algorithms for information mining assignments [12]. WEKA executes algorithms for information preprocessing, classification, regression, clustering, and association rules. The new plans can similarly be made with this pack. In particular, WEKA is an open-source application given under General Public License [12]. The information record usually used by Weka is in the `ARFF` file-group, which involves labeling to reveal different attributes in the information file. It has many areas, all of which can be used to play out a particular work. At the point when a dataset has been stacked, one of the various panes in the Explorer can be applied to perform further examination.

Most Relevant Features of Each Class Used

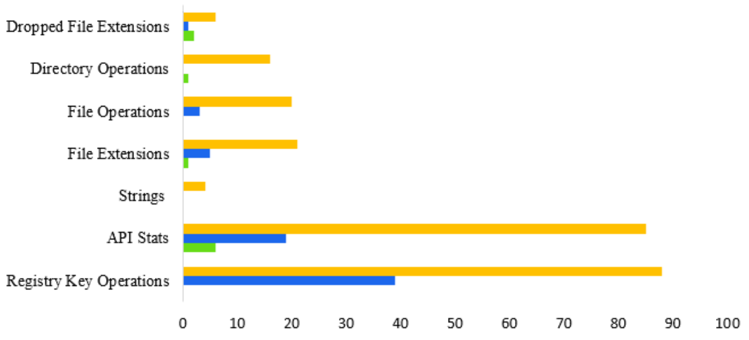


Fig 4. Most Relevant Features of Each Class

We determined the most relevant features through the knowledge based on dataset observation in comparison with feature evaluator and feature model as below:
Dataset Observations: Percentage of the Most Relevant Features for Each Class
Feature Evaluator (supervised, Class (nominal): 16382 Class): WEKA Information Gain Ranking Filter
Evaluation Mode: Evaluate all training data

We determined from Figure 4. Above that Registry Keys and API Stats are the two most pertinent sets, yet different sets are also useful depending on the areas in need. Among every one of these features, several features are for ransomware comportment, together with different features of malware behavior, prompted with an impressive detection rate.

2.1 Finest Features in Descending Approach - Top Twenty

We then again managed to determine our finest features as depicted in Table 2. Table 2. Highlights, top five features ranked in descending order based on the average weights. The finest features turn out to be considered essential when the quantity of features is enormous. From this research, it is evident that the finest feature, giving preferable outcomes over a complete set of features for a similar algorithm. The finest features empower the machine learning algorithm to prepare quicker as well as lessens the complex nature of a model and makes it simpler to interpret.

Search Method: WEKA Feature Ranking
Extracted Features: 14631

Table 2: Top Five Ranked Attributes (Feature Selection)

Ranked Attribute	Abbreviation	Set Id	Avg. Weight
API Stat	API	119	0.431262
Directory Operation	DIR	14265	0.407925
Dropped File Extensions	DROP	330	0.330449
File Extension	FILES_EXT	11684	0.327463
API Stats	API	167	0.275793

2.2 Method Obtained in Tuning ML Algorithms

There are 14631 extracted features with 1524 instances loaded. Hence, in total for each ML algorithm, six algorithm configurations were each evaluated 50 times, or 5-fold cross-validation (CD) multiplied by 10 repeats (R). We are going to compare each algorithm configuration based on the percentage accuracy. All of the default configurations are adjusted as per below control measures.

2.3 Discussion

The results are impressive. Ten ML algorithms were used for this research. Feature ranking and file transformations in `ARFF` file were furthermore performed, with the WEKA tool. In each model analysis, we set `K=10`, where `K` encapsulates the number of base classifiers. Additionally, we took `N=5` for the cross-validation and weight assignments independently. However, ML algorithms can be intended to motivate particular behavior. This is important since it allows the behavior of the model to be acclimated to the main points of our machine learning problem. In this way, one must tune the setup of each ML algorithm to a given problem. This is as often called algorithm tuning or algorithm HPO. For this research, we have chosen the ransomware dataset used to assess the distinctive algorithm configurations. We have additionally included frequent events of all ten algorithms (carried out in Weka) and each with an alternate algorithm arrangement as portrayed in Table 3. To achieve the best outcome. The feature selection method, along with the tuning methodology, has shown an impact on the performance level of the learned model as portrayed in Table 3.

Table 3: Tuning Test Control

ML Algorithms	Common Parameter Tuning Controls
An Iteration Control Set of 10 Repeats; 5-Fold Cross-Validation	
1. IBK 2. J48 3. JRip 4. Naïve Bayes 5. Part 6. Random Tree 7. Random Forest 8. SMO 9. Rep Tree 10. OneR	• Analysis for distance measure: Euclidean or Manhattan • K-values tested for {1,3,7} for both distance measures • Iteration control set to 10 repeats • MinNumObj tested for {2,3,5} • MinNumObj: 2 • NumFolds tested for {3,5,7} • Confidence Factor: 0.25 • Optimization 2 and 5 • MinBucketSize = 6 • NumDecimalPlaces = 2

Table 4. Provides a list of WEKA algorithms with the Receiver Operating Characteristic (ROC) area. Each value on ROC highlights the sensitivity in correspondence with a particular decision threshold. The ROC curve additionally reveals the correctly classified instances as positive values and incorrectly classified instances as a negative value. Whereas, Kappa stats provides the correlation coefficient in our performed experiment. Though the value of Kappa squared is responsible for the accurate amount of data, due to the similarity with our data correctors. Moreover, the False Positive (FP) in our case is in charge of depicting the number of detected ransomware anomaly values and the True Positive (TP) reveals the instances that are effectively anticipated as normal. Finally, after several trials and tuning, we managed to achieve the improved percentage model accuracy performance as in Table 4.

Table 4: Final Results (With and without HPO)

Algorithms		ROC Area	FPR	Recall	Precision	Model (%) Performance	Rank
IBK	*With HPO*	0.977	0.974	0.95	0.967	97.83	2
	Without HPO	0.916	0.047	0.811	0.821	81.1	
PART	*With HPO*	0.89	0.754	0.977	0.821	85.01	
	Without HPO	0.899	0.618	0.804	0.803	80.38	
SMO	*With HPO*	0.989	0.034	0.867	0.961	86.67	3
	Without FS	0.923	0.057	0.841	0.835	84.1	
J48	*With HPO*	0.805	0.457	0.087	0.804	80.72	
	Without FS	0.907	0.076	0.805	0.796	80.51	
Jrip	*With HPO*	0.871	0.071	0.782	0.777	78.37	
	Without FS	0.735	0.317	0.745	0.756	74.54	
OneR	*With HPO*	0.51	0.41	0.512	0.574	65.32	
	Without FS	0.57	0.506	0.646	0.664	64.56	
RF	*With HPO*	0.978	0.981	0.97	0.961	98.01	1
	Without FS	0.959	0.08	0.846	0.812	97.89	
RT	*With HPO*	0.61	0.51	0.57	0.589	79.1	
	Without FS	0.863	0.068	0.775	0.785	77.5	
RepT	*With HPO*	0.615	0.541	0.63	0.714	78.01	
	Without FS	0.919	0.773	0.773	0.751	77.3	

3 Conclusion

The outcomes just indicated that Auto-WEKA is powerful at advancing it's given objective. Though, the amount of HPs of an ML algorithm develops and so does its potential for overfitting. The use of cross-validation significantly increments Auto-WEKA's robustness against overfitting. In this work, we have presented the irresistible issue of simultaneously choosing an algorithm selection in addition to HPOs that can be settled by a completely automated tool. This is made promising by recent optimization techniques that iteratively assemble models of the algorithm HP landscape and influences these models to distinguish new focuses on the space that requires investigation. Auto-WEKA, which draws on the full scope of learning algorithms in WEKA and makes it simple for non-specialists to assemble great classifiers for giving application situations. A broad observational examination of ransomware detection datasets showed that Auto-WEKA regularly beat standard algorithm selection and HPO techniques, particularly on substantial data sets.

4 References

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems,* vol. 24, pp. 2546-2554, 2011.

[2] M. Shahhosseini, G. Hu, and H. Pham, "Optimizing ensemble weights and hyperparameters of machine learning models for regression problems," *arXiv preprint arXiv:1908.05287,* 2019.

[3] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," *arXiv preprint arXiv:1807.01774,* 2018.

[4] M. Claesen, F. De Smet, J. Suykens, and B. De Moor, "EnsembleSVM: a library for ensemble learning using support vector machines," *arXiv preprint arXiv:1403.0745,* 2014.

[5] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127,* 2015.

[6] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research,* vol. 12, pp. 2825-2830, 2011.

[7] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. de Carvalho, "Hyper-parameter tuning of a decision tree induction algorithm," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, 2016: IEEE, pp. 37-42.

[8] K. Bae, "Bayesian model-based approaches with MCMC computation to some bioinformatics problems," Texas A&M University, 2005.

[9] H. J. Escalante, M. Montes, and L. E. Sucar, "Particle swarm model selection," *Journal of Machine Learning Research,* vol. 10, no. 2, 2009.

[10] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020,* 2016.

[11] L. Breiman, J. Friedman, and R. Olshen, "Classification and regression trees Routledge," 2017.

[12] S. R. Garner, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*, 1995, vol. 1995, pp. 57-64.

[13] S. Alsoghyer and I. Almomani, "Ransomware Detection System for Android Applications," *Electronics,* vol. 8, no. 8, p. 868, 2019.