VOL.40 Nº4 2023
https://doi.org/10.22491/rmct.v40i4.12122.pt

Detection of Novelties in Malware Families

Ricardo Sant'Ana[1], Julio Cesar Cardoso Tesolin[1] e Julio Cesar Duarte[1]
[1]Military Institute of Engineering (IME)
Praça General Tibúrcio, 80, 22290-270, Praia Vermelha, Rio de Janeiro, RJ, Brazil
***ricardo.santana@ime.eb.br**

ABSTRACT: Many researches have already presented approaches to the malware detection task. Classifying them into families provides a better understanding of their behavior, allowing companies and researchers to optimize their efforts. Nevertheless, an issue still needs to be proper addressed: how to verify if an artifact detected as a malware belongs to a known family? This work proposes the use of two widely known classifiers - GMM and SVM - for a novelty detection task in malware analysis to redirect proper human and computational efforts for a quick counter measure. The main contribution of this work is the use of features directly extracted from the detected malwares's binary file such as entropy and image's texture for novelty detection.

KEYWORDS: Gaussian Mixture Model. Malware Family Detection. Novelty Detection. Support Vector Machine. Malware as an image. Entropy.

RESUMO: Muitas pesquisas já apresentaram abordagens para a tarefa de detecção de malware. Classificá-los em famílias fornece uma melhor compreensão de seu comportamento, permitindo que empresas e pesquisadores otimizem seus esforços. No entanto, um problema ainda precisa ser tratado corretamente: como verificar se um artefato computacional detectado como malware pertence a uma família já conhecida? Este trabalho propõe o uso de dois classificadores amplamente conhecidos – GMM e SVM – para uma tarefa de detecção de novidade em análise de malware, em que o objetivo é direcionar esforços humanos e computacionais adequados para fornecer uma rápida contramedida. A principal contribuição deste trabalho está no uso de características diretamente extraídas do arquivo binário do malware detectado, como entropia e textura de imagem para a tarefa de detecção de novidade.

PALAVRAS-CHAVE: Modelo de Misturas Gaussianas. Detecção de Família de Malware. Detecção de Novidade. Máquina de Vetores de Suporte. Malware como imagem. Entropia.

1. Introduction

Detecting and protecting against malicious computer artifacts constitutes a high-profile topic in cybersecurity. The literature includes various definitions of malicious artifacts (or malware) [12]. In general, these software can damage users, systems, and networks, corrupting their codes, harming their operations, and/or stealing their information.

A 2017 survey by a large company that produces software to combat malicious artifacts [1] involving 1,300 professionals from small and large information technology companies showed that 91% of them were attacked by some type of malware, 45% were ill-prepared for dedicated cyberattacks, and 17% lost financial data as a result of these attacks.

New malicious artifacts usually consist of versions of existing ones, as in [21], in which less than 10% of the malware in 2019 are new. Thus, characterizing them into families accelerates the identification of their behavior, of vital importance for allocating computational and human resources to promote a countermeasure as soon as possible. However, cases in which the malicious artifact configures a novelty in malware sets (rather than a variation of existing software) eludes this characterization. Thus, research initiatives must seek to automatically identify these novelties to accelerate the fight against these threats.

This research aims to evaluate the capacity of models to classify artifacts that configure novelties in malware families by using two common machine learning processes to detect novelties: Gaussian Mixture Models (GMM) and Support Vector Machines (SVM). This task entailed (a) defining a minimum set of characteristics to enable rapid detection; (b) proposing a method to select data sets; and (c) choosing criteria for model hyperparameters considering novelty detection. These constitute the main contributions of this study.

This study is structured as follows: Section 2 describes the theoretical foundations of novelty detection, the used machine learning models, and the publicly available malware. Section 3 reports the main studies on machine learning for malware analysis. Section 4 described the method to define the datasets and choose the most appropriate hyperparameters to fit the novelty detection models. Section 5 shows the experiments and the obtained results. Finally, Section 6 describes the conclusions and suggestions for future research.

2. Theoretical Framework

This section shows the theoretical foundations of malware analysis, anomaly and novelty detection, the used machine learning methods, and the dataset for the experiment.

2.1 Static and dynamic analysis

Malware analysis has two fundamental methods: the static and the dynamic approaches. Static analysis examines malware without running it, whereas dynamic analysis runs it [22]. Both techniques have advantages and disadvantages and can be used to complement analysis.

2.2 Novelty detection

According to [13], anomalies constitute data patterns that fail to fit a well-defined notion of normal behavior. Bringing this definition to the detection of malware, a malicious artifact that fails to fit into any known family is an anomaly. However, unprecedented malicious artifacts (anomalies) are quite common. Thus, following [13], this type of anomaly is defined as a novelty.

The use of one term or another exceeds mere semantics: the methods to identify anomalies presuppose anomalous samples (albeit in a very small quantity) in the training set. Novelty detection methods lack these anomalous samples for parametrization, making the detection of an unprecedented malware even more challenging. The anomaly detection problem can obviously use supervised machine learning methods, whereas the novelty detection problem commonly uses unsupervised machine learning methods.

2.3 Gaussian mixture models

The Gaussian mixture model (GMM) is a probabilistic model that assumes that all data points stem from a mixture of a finite number of Gaussian distributions with unknown parameters. Thus, a GMM-1 uses only one Gaussian distribution to model the distribution of data, whereas a GMM-32 uses 32 weighted Gaussian distributions. Traditional statistical techniques with Gaussian models to detect novelties may define their probability threshold in two ways: (i) as $\mu \pm 3\ \sigma$, in which μ is the mean and σ is the standard deviation and (ii), as per the Grubbs' test [13]. However, machine learning avoids establishing the threshold for novelties a priori.

Novelty detection by unsupervised learning mainly involves estimating densities. Clearly, prior knowledge of the density of the data points would enable us to solve any problem based on the data [14].

2.4 Support vector machine

Support vector machines (SVM) constitute a set of supervised learning methods to classify, regress, and detect anomalies. It can be specifically found for novelty detection in [14], which proposed a method to solve the following problem: once a dataset is extracted from an underlying probability distribution P, it is desired to estimate a subset S so the probability of a test sample from P lying outside S is equal to V. The use of SVM in this type of task is known as OneClassSVM. It has the advantage of avoiding assumptions about the form of the distribution of known data [15], i.e., it applies to any distribution. According to [23], the idea lies in finding a function that is positive for regions with high point density and negative for small densities.

A disadvantage of OneClassSVM refers to its discrete prediction whether the sample belongs to a

modeled group or not. This limitation can be circumvented by changing OneClassSVM to enable the output of the SVM as conditional class probabilities, as per [16]. A probabilistic approach offers many advantages over the conventional method, including the ease of automatically selecting a probabilistic novelty threshold.

2.5 Database of malicious artifacts for novelty detection

Anomaly and novelty detection techniques based on machine learning clearly depend on databases with a representative volume of samples. Few such databases of malware are available to the public.

Also, according to [20], malicious artifacts for Windows represented, in August 2019, 74.49% of the existing malware universe, which motivated our choice for this platform.

3. Related studies

This section describes the main studies on machine learning and malware analysis.

The extensive use of machine learning to analyze malware includes several objectives, such as the detection of malicious artifacts [2, 3, 4, 5, 6], their variants [7, 8] and categories [9,10,12], among others.

The author in [2] proposed a new graph-based algorithm built from traces of dynamically collected instructions to classify artifacts into malicious or benign. Results shows that dynamic analysis, combined with n-grams, outperforms the antivirus tools at the time (2011). The authors in [3] proposed static and dynamic analyses of artifacts and similarity metrics with several kernels to detect malware. A weight is given to each kernel to find the weight/kernel combination with the best accuracy. Using this approach obtained 99.78% accurate results. The author in [4] used support vector machines to detect malware. The scope of that study totaled 398 samples, obtaining an accuracy from 94 to 95%. The authors in [5] use static and dynamic techniques to detect malicious artifacts. They used the C5.0 and random forest algorithms, which were implemented on the FAME framework. The obtained accuracy totaled 95.75% for binary classification and 93.02% for multiple categorization. An architecture to automatically analyze malicious artifacts (dynamic analysis) is proposed in [6]. The results obtained with random forest (based on ID3) showed an accuracy above 90%.

The author in [7] proposed a new use for hidden Markov models: sets of relevant opcodes are initially chosen, which are subjected to a hidden Markov model. Results improved from 8 to 42% when compared to studies with Hidden Markov models without opcode selection. In [8], a generic structure extracts structural information from malware as graphs of function calls that are encoded as function-level attributes. This approach is evaluated for 11 malware families, obtaining an accuracy of 86.67% in the detection of samples with previously chosen variants.

In [9], decision tree and machine support vector algorithms are used to categorize malicious artifacts into trojans, infectors, backdoors, and worms. Results obtained an accuracy above 98% for the decision tree algorithm and above 97% for the support vector machine algorithm. In [10], the authors propose two machine learning approaches to correctly classify the malicious artifacts in the Microsoft Malware Classification Challenge [11]. It extracted several artifact characteristics: referring to the binary content of the malicious artifact in the deconstructed file (disassembled). By an algorithm to select the best attributes, the system obtained a 99.76% accuracy. Finally, in [12], the authors used unsupervised machine learning (shared nearest neighbor) to classify and detect new malware families. Using a database of 20,000 samples, results show a 3% improvement using random forest and 18% using the Naive Bayes algorithm considering a combination of three attribute types in all cases (grayscale image and n-gram image attributes). In summary, these last articles show two different strategies: [10] classifies malware in existing families and [12] proposes classification and detection by an unsupervised approach.

Thus, despite the extensive research in malware analysis by machine learning, no analyzed article addresses novelty detection, the aim of this study.

4. Research development

This section describes the processes and methods to adjust the hyperparameters of machine learning models to detect malware novelties.

4.1 Analysis of the malware dataset

This study used the dataset provided by Microsoft and hosted on the Kaggle [11] platform to classify its malware families. Originally, two datasets are provided: a training dataset with 10,868 samples from known families and a test dataset with 10,873 samples from unknown families. This research only used the training subset since it was the only one with family labels for each sample. Table 1 shows the distribution of this training subset by family.

Table 1 - Distribution Of Malware Families On The Microsoft Database [11]

Order	Name of the family	Number of samples
1	Ramnit	1,541
2	Lollipop	2,478
3	Kelihos_ver3	2,942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator,ACY	1,228
9	Gatak	1,013

The malware samples in this database are available in two formats: raw binary, which contains the hexadecimal representation of the binary content of the file; and the deconstructed code, the result of disassembling the original code (with the IDA tool [17, 10]). In both cases, the removal of the file header (PE header) made any attempt at dynamic analysis of the code impossible.

Since this research aims to define a classifier that can quickly identify novelties, only a minimum set of attributes from the raw binary data was considered based on two premises: (i) close-to-normal distribution and (ii) successful use in previous studies. Thus, four classes of attributes were chosen from [10]: entropy (ENT), metadata (MD1), Haralick (IMG1), and local binary patterns (IMG2).

Entropy attributes (ENT) serve to measure the clutter in the raw binary file. Entropy is calculated by N sliding windows to represent the malware as a measure of entropy $E = E_i$ with $i = \{1, 2, 3 \ldots N\}$, in which E_i is the entropy measured in window i and N is the number of windows. Next, the statistics of the entropy vector obtained by the sliding window method were considered, i.e., the entropy was calculated for each 10,000-byte window and the statistical measures of the obtained quantiles, percentiles, mean, and variance of the distribution were then considered. The entropy of all malware bytes totaled 202 characteristics. The extracted metadata attributes (MD1) refer to file size and the address of the first byte string, totaling two characteristics.

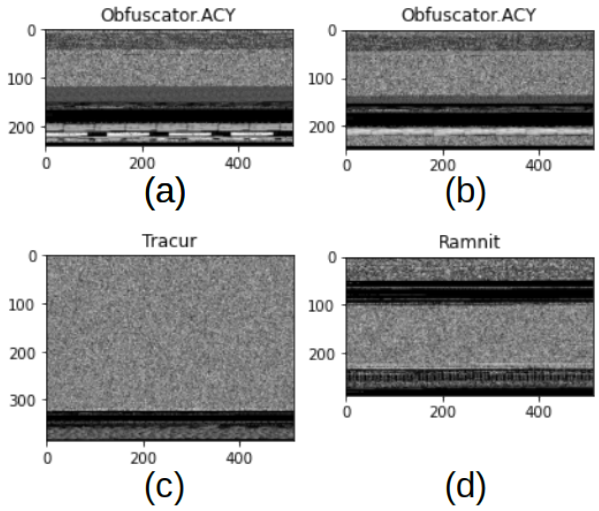
An original way to represent a sample of malware refers to viewing them as a one-pixel grayscale in a 2D image [10]. Figure 1 shows a cutout of the grayscale image of (a) a Obfuscator.ACY malware, (b) another Obfuscator.ACY malware, (c) a Tracur malware, and (d) a Ramnit malware. (A) and (b) show that two malware of the same family have similar visual representations.

In total, two sets of attributes were extracted from the grayscale image of each sample that describe the textures: Haralick (IMG1), with 53 extracted traits; and the local binary standard attributes (IMG2), with 108 extracted characteristics, all of which are easy to process [10]. This extraction used Mahotas as its image processing and computer vision library [18].

Thus, the entropy (ENT), metadata (MD1), Haralick (IMG1), and local binary standard (IMG2) classes

total 365 individual attributes. The source code of the extraction software for these attributes can be found in [10]. No initiatives were carried out to reduce dimensionality since these 365 attributes were selected from the 1,805 attributes in [10].

Fig. 1 - Representation of malware in their respective families by grayscale images.



4.2 Definition of the training and test sets

The definition of the training and testing sets entailed three steps to obtain suitable data subsets for novelty detection.

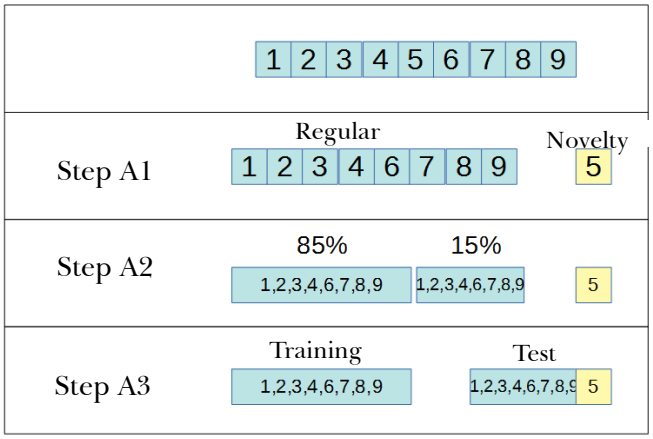
Initially, selecting the malware family to play the role of novelty proved to be challenging due to the absence of formal definition or method for this task to the best of our knowledge. It would be possible, for example, to choose both the family with the smallest number of samples and that with the greatest distance from the centroids of other families (considering that a set of attributes was selected). This research decided to adopt the family with the smallest number of samples as the novelty.

Thus, Step A1 chose samples belonging to family 5 (Simda) (Table 1). From this point on, this study refers to this family as the **novelty** class, grouping the other families into a single **regular** class.

Then, Step A2 divided the regular class into two groups, composed of 85 and 15% of all samples by random extractive partitioning, respectively. Finally, Step A3 defined the first group as the training set (85% of the regular class samples) and the test set as the union of the second (15% of the regular class samples) and the novelty classes (Simda).

As a result of this partitioning (Step 3), no samples of the novelty class belong to the training set. The ratio between the number of samples in the training and test group revolved around 6:1. Figure 2 shows the steps above, with the numbers 1 to 9 representing the nine malware families in Table 1.

Fig. 2 - Representation of the method to partition the training and test sets.



4.3 Definition of the training and validation set

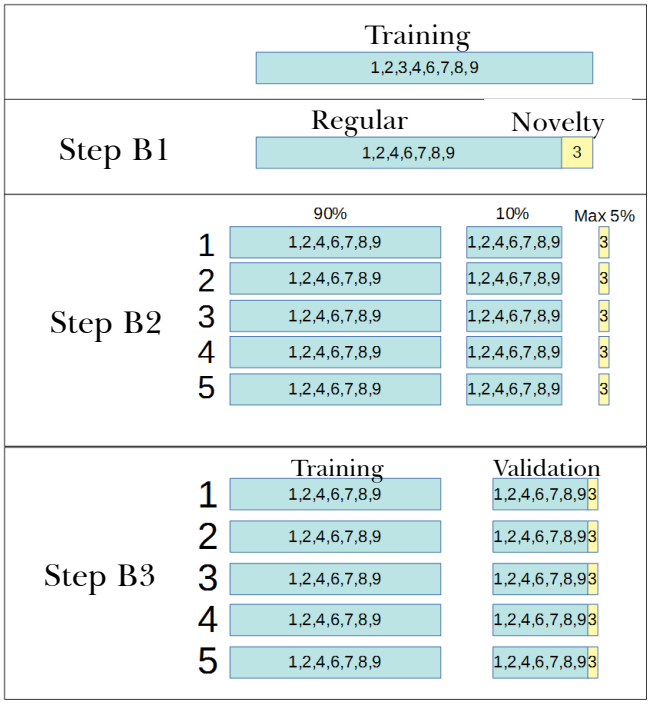
As in Section 2.2, detecting a novelty differs from detecting an anomaly. While the latter includes samples to calibrate the machine learning process, the former offers no such possibility. Thus, the choice of the training and validation set should be adapted to novelty detection so the several families in the training set (Section 4.2) play the role of novelty class in the validation set. Thus, each family in the training set would play the role of novelty class in at least one iteration. As in the previous section, this process entailed three steps (Figure 3).

Step B1 chose a family from the training set to represent the novelty class, whereas the rest of the samples represented the regular class. In Figure 3, as an example, family 3 played the role of the novel-

ty family. Step B2 divided the regular class into five equally sized subsets by random extractive partitioning. The idea of using five subsets is that, during the training and validation of the model, the performance will be the value of the average of the performances in each subset. Next, each subset was divided into two groups: the first with 90% of the samples and the second with 10% of the samples.

Finally, Step B3 designated the first group (90% of the samples) as the training set. On the other hand, the validation set consists of the union of the second group (10% of the samples) with a set of samples from the novelty family defined in Step B1 (family 3 in Figure 3), randomly limiting its volume to 5% of the total amount of the second group. Therefore, the **validation** set has regular and novelty samples. Again, this study sought a ratio between training samples and validation close to 6:1.

Fig. 3 - Method to select the training and validation sets for the novelty detection task. Family 3 served as the novelty class.



As each family will play the role of novelty, a total of eight experiments will be carried out to define the training/validation sets, as described above.

4.4 Classifier parameter selection

This study adjusted the hyperparameters of this model by simultaneously considering the result in the eight training/validation experiments in Section 4.3. Thus, this research created a criterion to accept a certain set of hyperparameters of a model.

Initially, P_i is defined as a set of hyperparameters in which $i = \{1, 2, 3 \ldots n\}$ and n is the number of attempts. To evaluate the learning process based on these hyperparameters, three metrics were established:

- ***Training Error Percentage*** **(TEP)**: Error in the training set, calculated by dividing the number of predictions outside the regular class and the total number of elements of the training class. The training dataset only has regular class elements.
- ***Validation Error Percentage*** **(VEP)**: Error in the validation set, calculated by dividing the number of errors in the prediction of validation set samples by the total number of validation set samples. The validation set has both regular and novelty samples.
- ***Novelty Error Percentage*** **(NEP)**: The novelty error is defined by dividing the number of errors in predicting only the validation set novelty class samples by the total number of novelty class samples in the validation set.

To accept a set of parameters P_i, the following criteria were defined:

$$(TEP < 30\%) \wedge (VEP < 30\%) \wedge (NEP < 40\%)$$

This criterion must be validated considering the average of the TEP, VEP, and NEP for each of the eight experiments in Section 4.2. Note that the percentage thresholds above were established after repeated attempts.

Case P_i meets the established criteria, it is selected. Otherwise, a new set of parameters P_{i+1} is set and the parameter selection process is restarted. This study showed no intention to choose the best possible model. It aimed to choose the one with hyperparameters set to P_i, capable of being used in

all eight experiments. The adopted assumption is that the criterion for choosing hyperparameters P_i can generalize the final model and obtain a good result for unprecedented samples.

Finally, with the chosen set of hyperparameters P_i, a model with the entire original training set (families 1, 2, 3, 4, 6, 7, 8, and 9) is trained. Then, the test set (family 5) is introduced to this model. To estimate the performance of the model in the test suite, the following metrics are defined:

Training Error Percentage **(TEP)**: Error in the training set, calculated by dividing the number of predictions outside the regular class and the total number of elements of the training class.

Test Error Percentage **(TEPer)**: Error in the test set, calculated by dividing the number of errors in the prediction of training set samples by the total number of validation set samples. The test set features regular samples and novelty samples from family 5.

Novelty Error Percentage **(NEP)**: novelty error, i.e., family 5, defined by dividing the number of errors in predicting only the test set novelty class samples by the total number of novelty class samples in the test set.

5. Results

This section describes the results of the hyperparameters that were chosen in the previous section for the support vector machine and Gaussian mixture machine models. As in sections 2.3 and 2.4, this study used the support vector machine and Gaussian mixture models to detect anomalies and novelties despite their rather different approaches, justifying the choice of these models to evaluate their behavior in a specific malware case. Note that this research evaluated their performance both in samples from previously identified families and in unprecedented samples.

5.1 Support Vector Machine Model

This study decided to use the RBF kernel for the SVM model, as per [19]. The rest of the hyperparameters that can be modified are (i) outliers (which define the number of anomalies in the training set) and (ii) gamma (γ) (which defines the chosen kernel). As a probabilistic implementation of OneClassSVM — as in [16] —, this study chose a conventional version of this algorithm.

Then, the algorithm obtained its best results with 13% outliers and a 0.01 γ, following the method in Section 4.4. Table 2 shows the results of the metrics for the various validation sets. In each row, the metric is averaged for five subsets, in which one of the eight families plays the role of novelty (see Section 4.3).

The parameter selection phase finds that the worst result occurs when family 6 (Tracur, with a 74.18% novelty error percentage) is novel. This may stem from the Tracur malware family differing from the others since they can primarily redirect online research, download and execute files (including other malware), and allow for command-and-control actions on the infected computer. Thus, Tracur family artifacts differ from the others for their greater capacity for malicious activities. As a result, the classifier may have erroneously classified them into other families.

Since the average metrics result meets the criteria in Section 4.4, this study chose this set of hyperparameters to create a model that can consider the entire training set in Section 4.2.

Table 2 - Training and validation error for SVM

Name of the family	TEP	VEP	NEP
Ramnit	13.14%	12.48%	6.08%
Lollipop	13.10%	13.97%	7.08%
Kelihos_ver3	13.21%	12.46%	0.00%
Vundo	13.17%	11.63%	1.41%
Tracur	13.18%	33.99%	74.18%
Kelihos_ver1	13.14%	12.48%	6.08%
Obfuscator,ACY	13.27%	12.61%	3.92%
Gatak	13.19%	16.44%	15.73%
Mean:	**13.18%**	**15.86%**	**13.43%**

Next, Table 3 shows the prediction in the test set and the result of the metrics in this set. The SVM model solidly detected samples from the regular set but shows a decreased performance in novelty detection.

Table 3 - SVM test error

Name of the family	TEP	PET	NEP
Simda	13.17%	17.29%	26.19%

5.2 Gaussian mixtures model

Initially, a Gaussian mixture model with only one Gaussian was used, which is equivalent to a simple Gaussian model to detect novelties. Several attempts were made to find the error threshold (ε) using the approach described in Section 4.4, in which values below this threshold indicate that the sample belongs outside the regular class.

After several experiments, an error threshold was reached $\varepsilon = \mu - 0.98\sigma$, in which μ is the average of the weighted logarithms of the probabilities for each training sample and σ is the standard deviation of the weighted logarithms of the probabilities for each training sample. Table 4 shows the results of the metrics for the various validation sets. Each row shows the metric average for the five subsets in which each of the eight families played the role of novelty.

Table 4 - Error training and validation for GMM-1

Name of the family	TEP	VEP	NEP
Ramnit	31.66%	30.91%	40.50%
Lollipop	21.64%	21.66%	10.97%
Kelihos_ver3	31.52%	32.51%	0.01%
Vundo	28.70%	29.04%	98.12%
Tracur	33.76%	35.16%	38.54%
Kelihos_ver1	28.30%	28.72%	13.31%
Obfuscator.ACY	29.40%	28.79%	88.71%
Gatak	26.25%	27.37%	24.37%
Mean:	**28.95%**	**18.97%**	**39.34%**

The average result of the metrics met the criteria in Section 4.4. The families that obtained the worst result were Vundo and Obfuscator.ACY. A possible reason for this error is that Gaussian distributions are unable to approximate the distribution of the attributes of these families. Thus, this set of hyperparameters were selected, creating a model that considers the entire training set, and then the prediction was performed in the set of tests. Table 5 shows the results for the test set.

Table 5 - Test error for GMM-1

Name of the family	TEP	VEP	NEP
Simda	27.79%	27.91%	19.05%

It can be preliminarily concluded that the Gaussian model could detect novelties according to the criteria to select parameters.

In any case, several experiments were carried out varying the number of Gaussians in the GMM. Models with 2, 4, 8, 16, 32, and 64 Gaussians were tested. From 32 Gaussians onward no improvement in training/validation occurred. Thus, Table 6 shows the results of the metrics for the various validation sets for 32 Gaussians.

continues

Table 6 - Training error for GMM-32

Name of the family	TEP	VEP	NEP
Ramnit	16.00%	30.25%	0.08%
Lollipop	8.57%	20.25%	14.98%
Kelihos_ver3	8.14%	21.69%	0.19%
Vundo	8.54%	18.86%	97.92%
Tracur	11.27%	21.09%	29.97%
Kelihos_ver1	18.04%	27.26%	2.84%
Obfuscator.ACY	9.18%	18.84%	88.77%
Gatak	7.87%	17.74%	26.40%
Mean:	**10.95%**	**22.00%**	**39.33%**

Again, a large variation in the percentage of novelty error occurred, as the case of GMM-1. The average error percentage of the model with only one Gaussian resembles the 32-Gaussian model.

The process of choosing and selecting hyperparameters for the models with 2, 4, 8, 16, and 32 Gaussians was carried out and the set of tests was applied to each of these models. Table 7 shows these results.

Table 7 - Test error for 2,4,8,16, and 32 GMM

GMM	Name of the family	TEP	PET	NEP
2	Simda	17.09%	18.13%	30.95%
4	Simda	27.22%	30.79%	28.57%
8	Simda	26.58%	28.09%	28.57%
16	Simda	21.00%	27.01%	28.57%
32	Simda	15.52%	25.15%	26.19%

If we only consider the number of errors in the detection of novelties, GMM-2 showed 13 errors; GMM-4, GMM-8, and GMM-16, 12 (28.57%); and GMM-32, 11 errors (26.19%). Thus, model training and testing evince the slight outperformance of the GMM-32 model. This was expected since the greater the number of Gaussians, the better the model fits the distribution of characteristics.

It is interesting to note that the model with a single Gaussian better detected novelties, i.e., a model that best fits the attribute distribution of the regular set fails to necessarily offer the best model for novelty detection. Also, both models (GMM-1 and GMM-32), during parameter selection, showed the worst results for classes 4 (Vundo) and 8 (Gatak) as novelties.

SVM, GMM-1, and GMM-32 results show that all models adequately detected novelties according to the criteria for choosing parameters and the difficulty of detecting unprecedented samples (novelty).

In other words, SVM obtaining 26.19% NEP and GMM-1, a 19.05% NEP equals claiming that SVM and GMM-1 detected 73.81 and 80.95% of all novelties, respectively, which is far better than no detection.

The SVM model seems to have a greater potential to detect novelties than the GMM model but both showed compatible results.

6. Conclusion

Novelty detection is a complex task as it lacks an obvious relation between parameter selection (by a training and validation set) and testing results (by a test set). Thus, this study used two classifier algorithms (GMM and SVM) that satisfactorily detected novelties. Defining training, validation, and testing sets and criteria for accepting hyperparameters was essential for these results. This study employed an adequate method to choose parameters for each model, although more in-depth studies are needed since some experiments showed poor results during selection (the Vundo and Gatak novelty classes for GMM and the Tracur novelty class for SVM).

As a suggestion for future research, this study proposes other attributes, a further development of poor results in some experiments during parameter selection, datasets with larger numbers of malware families, and a probabilistic novelty SVM classifier, as suggested in [16].

References

[1] – KASPERKY LABS. Ready... or Not? Balancing future opportunities with future risks. Moscou: Kaspersky Labs, 2017. Disponível em: https://media.kaspersky.com/documents/business/brfwn/en/The-Kaspersky-Lab-Global-IT-Risk-Report_Kaspersky-Endpoint-Security-report.pdf. Acesso em: dd mmm aaaa.

[2] - ANDERSON, B.; QUIST, D.; NEIL, J.; STORLIE, C.; LANE, T. Graph-based malware detection using dynamic analysis. Journal in Computer Virology, v. 7, n. 4, p. 247-258, 2011. DOI: 10.1007/s11416-011-0152-x

[3] – ANDERSON, B.; STORLIE, C.; LANE, T. Improving malware classification: Bridging the static/dynamic gap. In: Proceedings of the workshop on security and artificial intelligence, 5., [S.l.: s.n.], 2012. p. 3-14.

[4] - KRUCZKOWSKI, M.; SZYNKIEWICZ, E. N. Support vector machine for malware analysis and classification. In: Proceedings Of The 2014 Ieee/Wic/Acm International joint conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). Washington, DC: IEEE Computer Society, 2014. p. 415-420. DOI: 10.1109/WI-IAT.2014.127

[5] - MANGIALARDO, R. J.; DUARTE, J. C. Integrating static and dynamic malware analysis using machine learning. IEEE Latin America Transactions, v. 13, n. 9, p. 3080–3087, 2015.

[6] - ANDRADE, C. A. B.; MELLO, C. G.; DUARTE, J. C. Malware automatic analysis. In: 2013 BRICS Congress On Computational Intelligence And 11th Brazilian Congress on Computational Intelligence. [S.l.: s.n.], 2013. p. 681-686.

[7] - GHARACHEH, M.; DERHAMI, V.; HASHEMI, S.; HAZRATI FARD, S. M. Detection of metamorphic malware based on hmm: A hierarchical approach. International Journal of Intelligent Systems and Applications, v. 8, n. 4, p. 18-25, 2016.

[8] - KONG, D.; YAN, G. Discriminant malware distance learning on structural information for automated malware classification. In: Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining, 19., New York: ACM, 2013. p. 1357-1365. DOI: 10.1145/2487575.2488219

[9] - CHEN, Z.; ROUSSOPOULOS, M.; LIANG, Z.; ZHANG, Y.; CHEN, Z.; DELIS, A. Malware characteristics and threats on the internet ecosystem. Journal of Systems and Software, v. 85, n. 7, p. 1650-1672, 2012. DOI: 10.1016/j.jss.2012.02.015

[10] - AHMADI, M.; ULYANOV, D.; SEMENOV, S.; TROFIMOV, M.; GIACINTO, G. Novel feature extraction, selection and fusion for effective malware family classification. In: ACM Conference on Data and Application Security and Privacy (CODASPY), 16., 2016. p. 183-194

[11] – PANCONESI, A.; MARIAN; CUKIERSKI, W. Microsoft Malware Classification Challenge (BIG 2015). [S. l.]: Kaggle, 2015. Disponível em: https://www.kaggle.com/c/malware-classification. Acesso em: 17 set. 2018.

[12] - LIU, L.; WANG, B.-S.; YU, B.; ZHONG, Q.-X. Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, v. 18, p. 1336-1347.

[13] CHANDOLA, V.; ARINDAM, B.; VIPIN, K. Anomaly Detection: A Survey. ACM Computing Surveys, v. 41, n. 3, p. 1-58.

[14] – SOLLA, S. A.; LEEN, T. K.; MÜLLER, K.-R. Support vector method for novelty detection. In: Advances in Neural Information Processing Systems, 12., Cambridge: MIT Press, 2000. p. 582-588.

[15] - SHAWE-TAYLOR J.; ŽLIČAR, B. Novelty Detection with One-Class Support Vector Machines. In: MORLINI, I.; MINERVA, T.; VICHI, M. (Ed.). Advances in Statistical Models for Data Analysis. Berlin: Springer, 2015. p. 231-257.

[16] – CLIFTON, L.; CLIFTON, D. A.; ZHANG, Y.; WATKINSON, P.; TARASSENKO, L.; YIN, H. Probabilistic novelty detection with support vector machines. IEEE Transactions on Reliability, v. 63, n. 2, p. 455-467, 2014.

[17] - RONEN, R.; RADU, M.; FEUERSTEIN, C.; YOM-TOV, E.; AH-MADI, M. Microsoft malware classification challenge. [S. l.: s. n.], 2018.

[18] - COELHO, L. P. 2013. Mahotas: Open source software for scriptable computer vision. Journal of Open Research Software, v. 1, n. 1, 2013.

[19] - CHANG, C. C. A practical guide to support vector classification, Technical report, n. 5. Taipei: National Taiwan University, 2005.

[20] – Heightened threat scenario: all the facts in the AV-TEST Security Report 2018/2019. Magdeburg. AVTEST – The Independent IT-Security Institute, 2019. Disponível em: https://www.av-test.org/en/news/heightened-threat-scenario-all-the-facts-in-the-av-test-security-report-2018-2019/. Acesso em: 23 ago. 2019.

[21] – Malware. AVTEST – The Independent IT-Security Institute, 2019. Disponível em: https://www.av-test.org/en/statistics/malware//. Acesso em: 10 set. 2019.
[22] SIKORSKI, M.; HONIG, A. Practical malware analysis: the hands-on guide to dissecting malicious software. San Francisco: No Starch Press, 2012.
[23] SCHÖLKOPF, B.; WILLIAMSON, R.; SMOLA, A.; SHAWE-TAYLOR, J.; PLATT, J. Support vector method for novelty detection. NIPS, v. 12. 1999.