GMRF-CRISPR: a Genetic RandomForest Classification Algorithm

Ricard Marsal I Castan

2020

Abstract

As we generate more data, IDS (Intrusion Detection Systems) have become important in network security to prevent unauthorized use data. An IDS has to handle massive amounts of data and make predictions. Machine learning algorithms and data mining techniques are perfect for this task. Random Forest (RF) is a classification technique and has shown a higher performance in many applications than other algorithms. RF has been proposed for an IDS before, but combined with a Genetic Algorithm it achieves higher performance. This paper presents a novel algorithm for intrusions classification based on a RandomForest and a Genetic Algorithm.

Contents

List of Tables

List of Figures

1 Introduction

Humans have been altering plants and animals from the beginning of history. First by doing a selective breeding, where we breed together animals or plants that we consider better than the rest, so we only get good spices. A good example is corn. Humans only planted corn that came from the biggest and tastiest parent plants, so it only grows this type of corn. With the new knowledge of genes and DNA new developments for alterations appeared. GMO (genetically modified organisms) are created by adding a gene from another organism or modifying a gene in a lab. With this technique humans made some plants resistant to pesticides. With CRISPR-Cas9 (Clustered Regularly Interspaced Short Palindromic Repeats) scientists can alter the DNA by cutting genome at a desired location, allowing existing genes to be removed and/or add new ones.

For a long time Computer Science techniques helped analyze biology data, and biology helped create new optimization techniques for Computer Science. Many algorithms are inspired by nature, from a neural network to the brain, to genetics algorithms (GA) imitating the process of natural selection.

In this paper we present a new algorithm that is inspired by the ideas of GMO and CRISPR-Cas9. Using multiple Random Forests we will perform a selective breeding of decision trees to create a new Random Forest with better performance. Then during multiple generations, the GMO algorithm will select two trees of the new forest and will mix their nodes at a desired location of the tree to generate new modified trees. At the end of each generation, only the trees with better performance will remain in the Forest. This algorithm will be train and tested detect intrusions on the network.

The rest of this paper is organized in five sections. In section two we provide a background information on GMO Techniques and Machine Learning techniques. In the third section we describe related work in the field. The fourth section is the algorithm itself and the fifth is the experiment's and evaluation of the model.

2 Background

In this section, we present some background information about genetics, intrusion detection, and machine learning with the goal to put it all in context.

2.1 GMO Techniques

A genetically modified organism (GMO) is a plant or animal whose DNA has been altered using genetic engineering techniques. GMOs appear a lot in plants. The first genetically engineered plants for human consumption were introduced in the mid-1990s, and today approximately 90 percent of the corn, soybeans and sugar beets are GMOs [12]. One of the biggest breakthrough in the gene editing techniques was the CRISPR/Cas. This techniques is based in the immune system of a bacteria. CRISPR/Cas system allows bacteria to fight against

invading viruses by getting DNA from the virus and add it by pieces in its own genome. This DNA after being converted to RNA and combined with the protein Cas9, can be used to create a double-stranded breaks at specific points of the DNA sequences and target DNA repair by giving a DNA template with the desired change [16].

2.2 Intrusion Detection Background

Intrusion detection is the process of analyzing a computer network and detecting possible attacks that attempts to bypass the security of the system. The device or software application that monitors the network to detect malicious activity is called Intrusion Detection Systems (IDS)[31]. IDS allows organizations to protect their systems from threads that come with increasing network connectivity[5].

2.3 Machine Learning Background

Machine Learning is the base of data mining, is used to extract information and patterns in data[32]. The data is formed by instances that can be divided between attributes, the characteristics of that instances, and labels, the class or value of those attributes. For example, if we want to train an algorithm to classify snakes by the labels "snake" or "not snake", we would need to select a sequence of measurable features(or attributes) to distinguish a snake from anything else. Some features could be the length, height, color, number of legs, any characteristics that can be defined. After training, the algorithm can tell us if there is a snake or not.

There are three main types of classifications for machine learning algorithms: binary classification, multiclass classification, and multilabel classification.

Binary classification is used when there are only two distinct classes in the labels. Multiclass classification is used when there are three or more classes on the data and the instance that we want to classify only belongs to one of them. Multilabel classification is used when there are two or more classes that can classify the instance.

Algorithms that performs machine learning can be divided into two categories, based on how the data is presented to them: unsupervised, and supervised. Unsupervised algorithms classifies similar data when the class is unknown and supervised algorithms use the labeled data during training to predict the value of unlabeled data.

2.3.1 Unsupervised Machine Learning Algorithms

In a unsupervised environments, the labels of the data are unknown. The algorithm learns to classify by grouping similar instances together. The principal method of unsupervised learning is clustering. Depending on the cluster model, the principal clustering algorithms are the hierarchical clustering and the centroid-based clustering[8].

The idea of hierarchical clustering is that instances that are related to the ones that are around them and not instances that are far away. They don't support well outliers so the user has to go back and choose appropriate clusters[32]. The centroid-based clustering are more popular with the name k-means clustering. In this algorithm we select k instances and use them as centers of clusters and then assign the rest of instances to a nearest center of a cluster using an Euclidean distance[22].

2.3.2 Supervised Machine Learning Algorithms

Supervised learning uses know instances to train the model. In this paper we used this supervised machine learning algorithms to test against our algorithms: artificial neural network, naive Bayes classifier, support vector machines, decision trees, and random forests.

Artificial neural networks (ANNs) are models that are inspired in the neural networks structure of the brain[17]. Ann are a collection of nodes (artificial neurons) imitating the neurons of the bran, that process information before sending to the nodes that are attached to it. The nodes are aggregated in layers and each layer performs different transformations. The first layer (input layer) gets the information and the last layer (output layer) gives the classification output. The layers in between (hidden layers) determine the output of a node by the activation function. The ANN uses back-propagation for classification method and the sigmoid activation function in all the neural nodes. Back-propagation correct the weights of the nodes to compensate for the errors during learning.

Naive Bayes algorithm assumes that all the features of the dataset are completely independent of each other. That is, if a t-shirt is white, size medium, long sleeve, a naive Bayes classifier would consider those features to independently contribute to that clothe to be a t-shirt regardless if there is a correlation between the color, the size, and the type of sleeve. Naive Bayes works well in complex real-world application despite it's simple assumptions. In [25] a successful naive Bayes classifier was used for intrusion detection. The advantage of naive Bayes is that with a few training data, it can start classifying[30].

Support Vector Machines (SVMs) uses hyperplanes for classification. The original input dimensional space is mapped to another dimensional space that makes easier the feature separation.

A decision tree classifier is a tree (data structure) of nodes, branches, and leaves. From the root (top node) are connected other child nodes thought the branches with their own branches and their own children. Each node represents a feature of the training data and the branches the courses of actions that are available to make a decision. The instance is classified when it reaches a leaf, a node with no children.

A RandomForest consist of multiple decision trees and the outputs is the majority prediction of each individual decision tree. The training algorithm used in the trees is called bagging (bootstrap aggregating). Bagging consists of generating new training sets by sampling from the dataset uniformly and with

replacement.

2.3.3 Metrics for Evaluation of Classification Performance

We used a confusion matrix for the overall performance. A confusion matrix is a table of performance where each row represents an instance in a predicted class and the columns represents the instance in the actual class[28]. The resultant classification can be broken down in four boxes:

- True Positives (TP) - number of predictions that were attacks and were correctly classified as attacks.
- False Positives (FP) - number of predictions that were non-attacks and where classified as attacks.
- True Negatives (TN) - number of predictions that were non-attacks and were correctly classified as non-attacks.
- False Negatives (FN) - number of predictions that were attacks, but were incorrectly classified as non-attacks.

For our classification of attacks we were interested in the *F-Measure* or *F1 score*, which is defined by:

$$F - M = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The *Recall* or true positive rate (TPR) is the measurement of how the algorithm correctly classify the positive training instances.

$$Recall = \frac{TP}{TP + FN}$$

A perfect TPR, with score of 1 means that all the instances are properly classified. The complement is the false negative rate (FNR).
Precision is the closeness of the predictions to each other. Is defined as:

$$Precision = \frac{TP}{TP + FP}$$

2.4 Introduction to Genetic Algorithm

A Genetic Algorithm (GA) is a computational model based on evolution and natural selection. GA are used to generate great solutions to optimization and search problems[23]. These algorithms uses chromosome-like data structure. The population of candidate solutions (individuals) evolve to a better solution. On each generation the population's individuals mutate, crossover, and a fitness function evaluates each individual. The fitness function contains the goal of the algorithm. The more fit an individual is, the more chances it has that it will prevail to the next generation.

We consider an individual a Decision Tree and a Random Forest to be the whole population. On each generation two Decision Tree are selected and generate a new Decision Tree by selecting different parts of each tree. Our fitness function is the overall accuracy of each individual tree. Only the trees with greater accuracy are passed to the next generation.

3 Related Work

3.1 Related Work on Intrusion Detection

Studies have shown that machine learning applied to intrusion detection have successful results, including SVM [9], Random Forest [18], and K-Nearest Neighbours [19]. In [27], an evolutionary algorithm with an artificial neural network is used to extract feature subsets of the data and a Random Forest is used to evaluate the efficiency of the feature subsets. In [3], a genetic algorithm with a set of classification rules is used on a network audit data to improve the system security. The different methods for intrusion detection have been studied by researches for over 20 years, authors of [26], compare the performance of neural networks with support vector machine algorithms for intrusion detection.

Authors of [33], propose an intrusion detection model based on a convolutional neural network. By changing the traffic data into a two-dimensional matrix form the experimental results shows a considerably improvement the classification detection performance of the intrusion network traffic and it also improves in the training time.

A new anomaly detection framework was proposed in [21], where multiple deep learning techniques were integrated. They used Damped Incremental Statistics algorithm for feature extraction, Training Auto encoder with small amount of label data, then mark the abnormal score and use it to train a long short-term memory (LSTM) recurrent neural network (RNN).

In [4], a Restricted Boltzmann Machine (RBM) and a deep belief network for detection of new attacks in real time is used. Their method uses on-hidden layer RBM for unsupervised feature reduction, then another RBM for training with a logistic regression classifier with multi-class soft-max.

3.2 Related Work on RandomForests

This section presents a review of RandomForest algorithms. Farnaaz and Jabbar compared in [10] a RandomForest with other classifiers and showed an efficient model with low false alarm rate and high detection rate.

Authors of [14] present a two step approach of feature selection of intrusion network data, based on a Random Forest, where the first step select the features with higher variable importance score and the second step the ones for classification and training. The results show that a RandomForest can select the most important features for classification.

In [29], authors compare the performance of a random forest as an IDS by getting two performance measures, the accuracy and false alarm rate. The results of the random forest outperformed other models like a decision tree, naive Bayes, and a neural network.

RandomForests were proposed in [13] for mobile intrusion detection, where the RF performs best when it has to classify intruders taking as input user data coming from only one service.

3.3 Related Work on Genetic Algorithms

A genetic algorithm (GA) is an algorithm inspired by the process of natural selection used for optimization and search problems by using biologically inspired operators like the mutation, crossover and selection[24].

A GA was implemented in [15] and [11] as a base of an IDS. The approach detects various types of network intrusions and uses evolution theory in order to filter traffic data to reduce complexity.

The authors in [2] present a lightweight machine learning-based intrusion detection technique with high performance for resource limited internet of things wireless networks. This IDS is a based on hybridization of genetic algorithm and grey wolf optimizer (GA-GWO). GA-GWO improved the performance of the IDS computationally and having a low false alarm rate.

Li [20] presented a GA that detects complex anomalous behaviours in the TCP/IP network protocols by considering both temporal and spatial information of network connections in encoding the network connection information into rules of the IDS.

4 GMRF-CRISPR

The Genetic Modified Random Forest-CRISPR (GMRF-CRISPR) is an algorithm with the idea of simulating a genetic modified organism, in this case a RandomForest. The GMRF-CRISPR algorithm creates a GMRF-CRISPR model. This model is a collection of trees that operates like a RandomForest with a difference: the model is formed by doing a selected breeding of Decision-Trees. The trees of the model are the "best trees", based on overall accuracy, of multiple RandomForest models created during the training process. Once the GMRF-CRISPR model is created starts the genetics part of the algorithm. During multiple generations each tree is selected with another randomly chosen tree. Each tree is mixed with the randomly selected tree and if the overall accuracy of the new tree is greater than the overall accuracy of the previous tree, the new tree is added to the GMRF-CRISPR model. The mixing, targets different specific part of the tree, inspired on the CRISPR/Cas technique. At the end of each generation the worst trees are eliminated from the model.

The GMRF-CRISPR model is constructed from two disjoint labeled data sets. One for training the RandomForest models and one for evaluation of the

GMRF-CRISPR model on each generation.

ALGORITHM 1: GMRF-CRISPR: A Random Forest with modified Decision Trees

Input: $\mathcal{L}$: a distributed, labeled data set for training
$\mathcal{E}$: an unseen distributed, labeled data set for testing
$numForests$: the number of RandomForests models to create
$numTrees$: the number of trees for the final GMRF-CRISPR model
$generations$: number of repetitions

Output: $GMRF-CRISPR$: an inductive GMRF-CRISPR Model containing modified Decision Trees

$GMRF-CRISPR \leftarrow$ an empty GMRF-CRISPR model, with space for $numTrees$ Decision Trees;
$size \leftarrow 0$;
$generation \leftarrow 1$;
$accuracy \leftarrow$ an empty list of Double;
`/* Step 1: Train RandomForests on the labeled data and add best trees to GMRF-CRISPR model, see Algorithm 2 */`
$GMRF-CRISPR \leftarrow createForestSelectBestTrees(GMRF-CRISPR, numForests, \mathcal{L}, \mathcal{E})$;
`/* Step 2: Test the model */`
$accuracy \leftarrow$ accuracy score for $GMRF-CRISPR$ model tested on $\mathcal{E}$,;
while $generation < generations$ **do**
  `/* Step 3: Create new modified DecisionTrees add the best ones to the model, see Algorithm 3 */`
  $GMRF-CRISPR \leftarrow gmoAlgo(GMRF-CRISPR, \mathcal{E})$;
  `/* Step 4: Remove worst DecisionTrees from the GMRF-CRISPR model */`
  `/* Step 5: Test the model */`
  $accuracy \leftarrow$ accuracy score for $GMRF-CRISPR$ model tested on $\mathcal{E}$;
end
return $GMRF-CRISPR$

The GMRF-CRISPR model can be broken in two parts, the first part, training, is where all the Random Forest are created (see Algorithm 2) and the best trees of all the Random Forest are added to the GMRF-CRISPR model. The Random Forest are trained increasing the percentage training data set, from 5% to 45%. The data is stratified.From all the trees of all the RandomForest, only an specified numbers of trees will be selected as "best" trees, in this version of GMRF-CRISPR, overall accuracy is used to determine the best trees, but the performance metrics could be changed depending on the situation. Something that might occur is that some RandomForest don't contribute any tree to the GMRF-CRISPR model.

ALGORITHM 2: GMRF-CRISPR-createForestSelectBestTrees

Input: $GMRF-CRISPR$: a partial GMRF-CRISPR Model
$\mathcal{L}$: a distributed, labeled data set for training
$\mathcal{E}$: an unseen distributed, labeled data set for testing
$numForests$: the number of RandomForests models to create
$numTrees$: the number of trees for the final GMRF-CRISPR model
$accuracy$: List of accuracies
Output: $GMRF-CRISPR$: a GMRF-CRISPR Model with Trees
$treeAccuracy \leftarrow 0.0$;
`/* Train RandomForests on the labeled data */`
$\mathcal{R} \leftarrow$ RandomForests $R_1, R_2, \ldots, R_k$ trained on random samples of $\mathcal{L}$;
`/* Test each Forest */`
$accuracy \leftarrow$ accuracy score for $\mathcal{R}$ tested on $\mathcal{E}$,;
`/* Determine the best trees and add them to the GMRF-CRISPR model */`
foreach $R_i \in \mathcal{R}, i = 1, 2, \ldots, k$ **do**
  foreach $tree \in R_i, i = 1, 2, \ldots, k$ **do**
    $treeAccuracy \leftarrow$ accuracy score for $tree$ tested on $\mathcal{E}$;
    if $GMRF-CRISPR.IsEmpty()$ **then**
      $GMRF-CRISPR \leftarrow tree$;
    end
    if $GMRF-CRISPR.IsNotEmpty()$ **then**
      `/* Sort by accuracy` $GMRF-CRISPR$ `*/`
      $lastTreeAcc \leftarrow$
       $GMRF-CRISPR(GMRF-CRISPR.size-1).accuracy$;
      if $treeAccuracy > lastTreeAcc$ **then**
        $GMRF-CRISPR \leftarrow tree$;
      end
      `/* Sort by accuracy` $GMRF-CRISPR$ `*/`
      if $GMRF-CRISPR.size > numTrees$ **then**
        $GMRF-CRISPR.remove(GMRF-CRISPR.size-1)$;
      end
    end
  end
end
return $GMRF-CRISPR$

Once the model is created and the best trees are selected, a genetic algorithm starts. Considering each tree of the model as an individual of the population, for each individual we select another one randomly and from the modification of the left and right Nodes of the original individual (see Algorithm 3). From the modification we create a new individual, if this one performs better (based on overall accuracy in an unseen data set) than the original individual then is added to the list of individuals.

ALGORITHM 3: GMRF-CRISPR-gmoAlgo

Input: $GMRF-CRISPR$: a partial GMRF-CRISPR Model
$\mathcal{E}$: an unseen distributed, labeled data set for testing
Output: $GMRF-CRISPR$: GMRF-CRISPR Model
$randTree \leftarrow$ spark.mllib.tree.model.DecisionTreeModel;
$originalRightNode \leftarrow$ spark.mllib.tree.model.Node;
$treeAccuracy \leftarrow 0.0$;
`/* For each tree create a new tree that is a mix with a random selected tree if the new tree has greater accuracy than the original tree, add the new tree to GMRF-CRISPR model */`
foreach $tree \in GMRF-CRISPR$ **do**
    `/* Right modification of the tree */`
    $randTree \leftarrow$ Random selected tree from $GMRF-CRISPR$;
    $originalRightNode \leftarrow tree.topNode.rightNode$;
    $tree.topNode.rightNode \leftarrow randTree.topNode.rightNode$;
    $treeAccuracy \leftarrow$ accuracy score for $tree$ model tested on $\mathcal{E}$;
    if $treeAccuracy > tree.accuracy$ **then**
        $GMRF-CRISPR \leftarrow tree$;
        $tree.topNode.rightNode \leftarrow originalRightNode$;
    end
    `/* Left modification of the tree */`
    $randTree \leftarrow$ Random selected tree from $GMRF-CRISPR$;
    $originalLeftNode \leftarrow tree.topNode.LeftNode$;
    $tree.topNode.LeftNode \leftarrow randTree.topNode.LeftNode$;
    $treeAccuracy \leftarrow$ accuracy score for $tree$ model tested on $\mathcal{E}$;
    if $treeAccuracy > tree.accuracy$ **then**
        $GMRF-CRISPR \leftarrow tree$;
        $tree.topNode.LeftNode \leftarrow originalLeftNode$;
    end
end
return $GMRF-CRISPR$

Steps 4 and 5 of Algorithm 1 sort all the individuals by its accuracy and remove the worst trees to maintain the specified number of individuals in the model. Steps 3,4,5 are repeated during multiple generations.
The end result is a GMRF-CRISPR model containing "genetically modified" best DesicionTrees. The main novel aspects of the GMRF-CRISPR that it set it appart from the other algorithm in the literature are:

1. We select the best performing trees as in [7] but we select only the best ones out of all the trees, not the best ones from each forest.

2. We create new trees by combining the nodes of two different trees.

3. The GMRF-CRISPR model performs better than other state of the art algorithms.

The GMRF-CRISPR algorithm could also be used in any type of classification problem, here we used a network intrusion dataset to test its effectiveness.

5 Experiments and Evaluation

This section covers the experimental results obtained. Our code is available at this GitHub repository [6]. We check these points to evaluate our model:

1. What parameters allow our model achieve the best performance.
2. How does our algorithm perform compared with other state of the arts algorithms.

5.1 Dataset

This dataset is the result of a collaborative project between the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC)[1].

5.1.1 Attack and Feature Description of the dataset

The dataset CSE-CIC-IDS2018 is organized by days, every day the data is recorded using Pcaps and using CICFlowMeter-V3 is saved into a CSV file with 80 traffic features per machine of network traffic. The dataset is generated by profiles which contain detailed descriptions of intrusions and abstract distribution models for applications, protocols, or lower lever network entities[1]. This profiles generate events on the network and multiple network protocols are applied to them. There are two types of profiles: B-profiles, and M-profiles.

1. B-profile: This profile is the behaviours of machine learning and statistical analysis techniques that simulated the following protocols: HTTPS, HTTP, SMTP, POP3, IMAP, SSH, and FTP[1].
2. M-Profiles: This profiles tries to simulate different types of attacks. There are six different types of scenarios:
    - Infiltration of the network from inside: where an email is send to a victim, if the exploitation is successful, then a backdoor will be created in the victims computer that will be used to exploit other vulnerabilities.
    - HTTP denial of service: using Slowloris and LOIC a TCP a connection to the server is generated and starts sending valid, but incomplete HTTP requests to keep sockets from closing.
    - Collection of web application attacks: here Damn Vulnerable Web App (DVWA) is used to scan a website for vulnerabilities and then conduct different web attacks such us SQL injection, command injection, and unrestricted file upload.
    - Brute force attacks: this scenario runs a SHH and MySQL dictionary attack against the main server.
    - Last updated attacks: This attacks are based on famous vulnerabilities like Heartbleed.

5.2 Experimental results

In this section we evaluate the effectiveness of GMRF-CRISPR model using the CSE-CIC-IDS2018 dataset (described in section 5.1).

5.3 Serial Distributed Computing Environment

The classification algorithms were evaluated in serial form on a single desktop workstation. The hardware and software configuration are shown in Table 1.

Resource Type	Configuration
Software environment	Spark 2.3.2
	Scala 2.11
Development environment	Scala for Eclipse
	Maven
	Apache Spark MLlib
CPU	Intel(R) Core(TM) i7-7500U
Memory	16G

Table 1: Software and hardware configurations

5.4 GMRF-CRISPR Results

We wanted to check in what conditions the model performs the best. We changed the number of trees to construct a forest, the number of initial forests, and we wanted to see how the algorithm performed with less training data. The number of trees on each forest range from 5 to 10 and the number of forest range from 1 to 10. The training data started at 5% and went all the way up to 45%(increments of 10%). In each trial, we stratified random sampling to select the subsets for training and testing.

Figure 1 shows the average overall accuracy of each RandomForest, the initial GMRF-CRISPR model and each generation after that. We can see how the accuracy increments faster on the beginning and stabilizing towards latter generations. Compared with a RandomForest our model performs a lot better after a couple of generations.

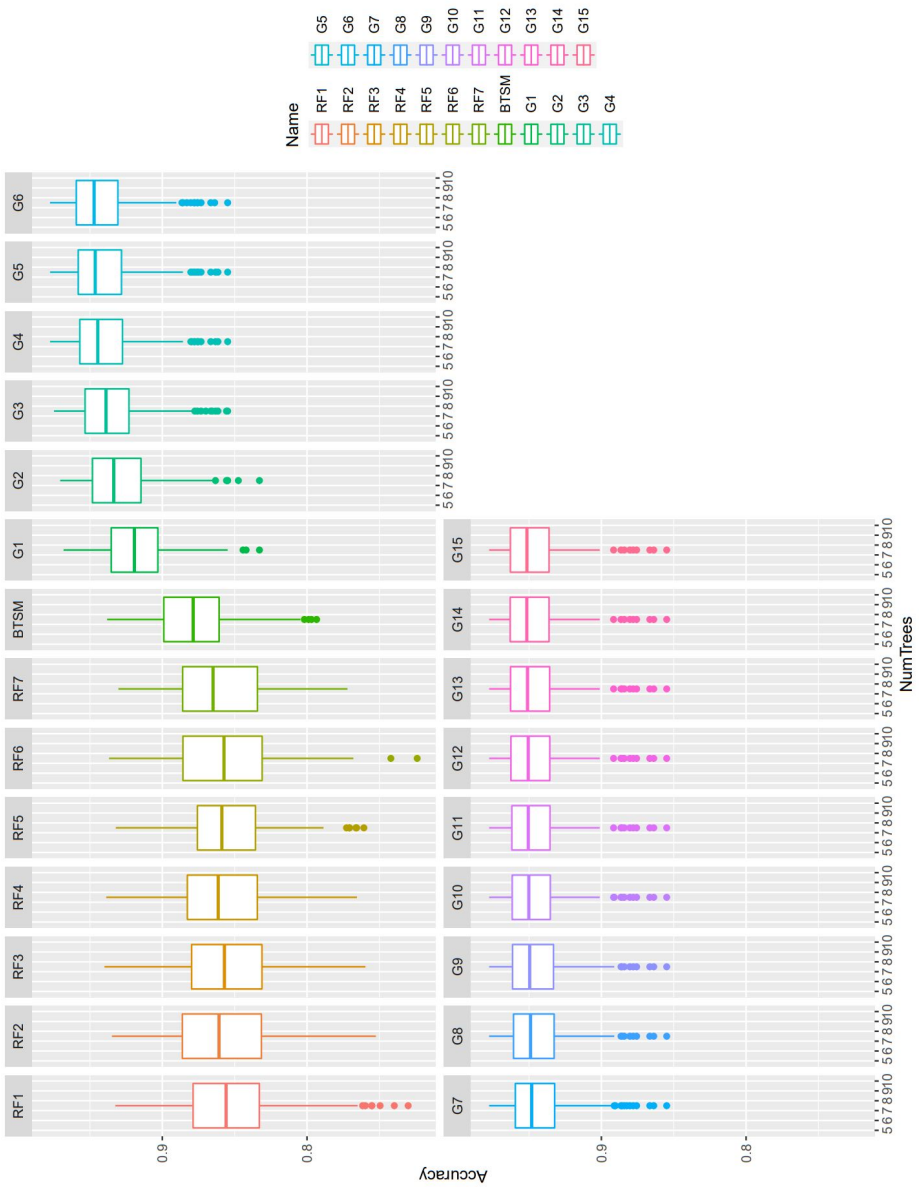

Figure 1: Overall Accuracy of GMRF-CRISPR on each generation of the model. Each panel shows a step on the generation where the $x - axis$ is the number of trees

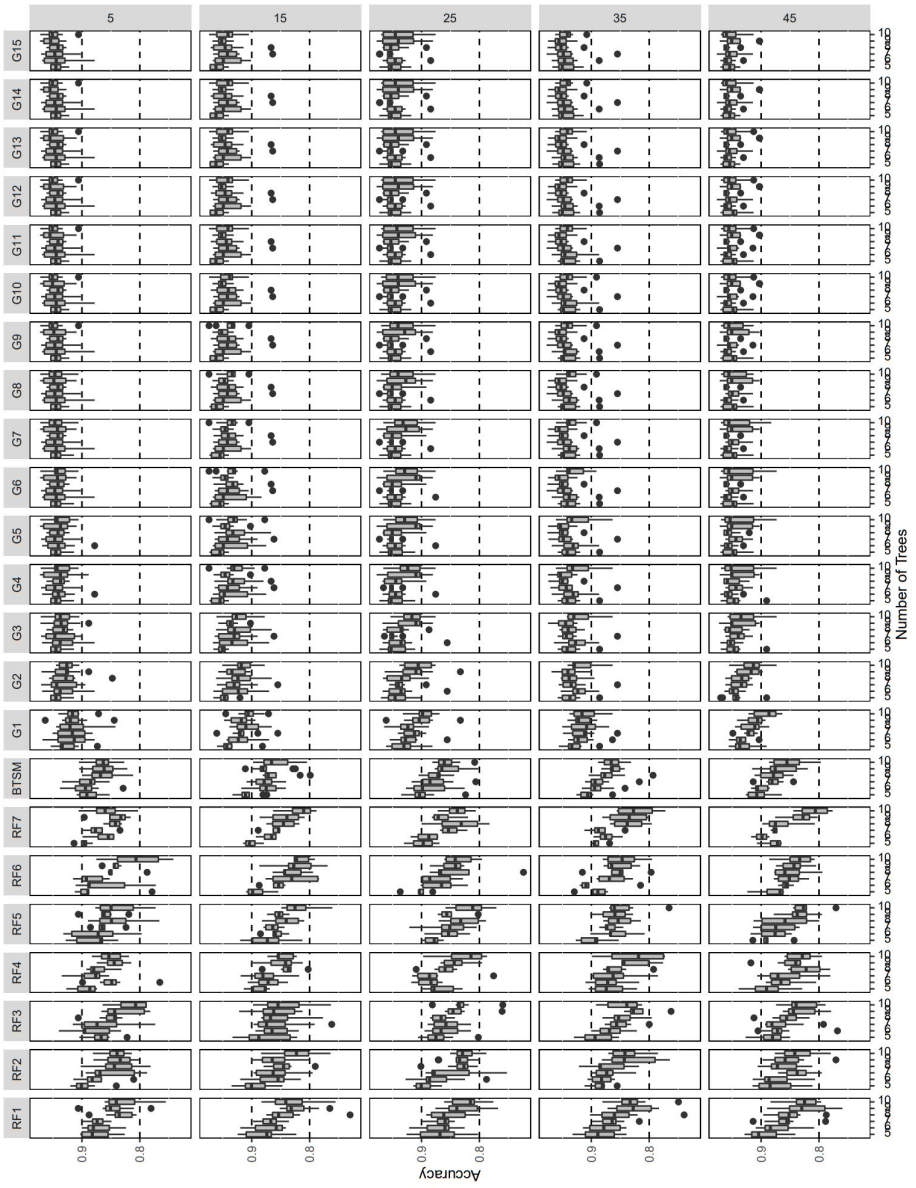

Figure 2: Accuracy of the GMRF-CRISPR model, and the RandomForests that build on training sets raging from 5% to 45% of the data. The x-axis shows the number of trees used to construct the model.

We can see the same trend in Figure 2. We can also see that the percentage of training data doesn't influence much on the outcome. The model trained with 5% of the data performs as good as the model trained on the 45%. The more data is used to train, the accuracy remains uniform thru generations versus the number of forests (see Figure 3).

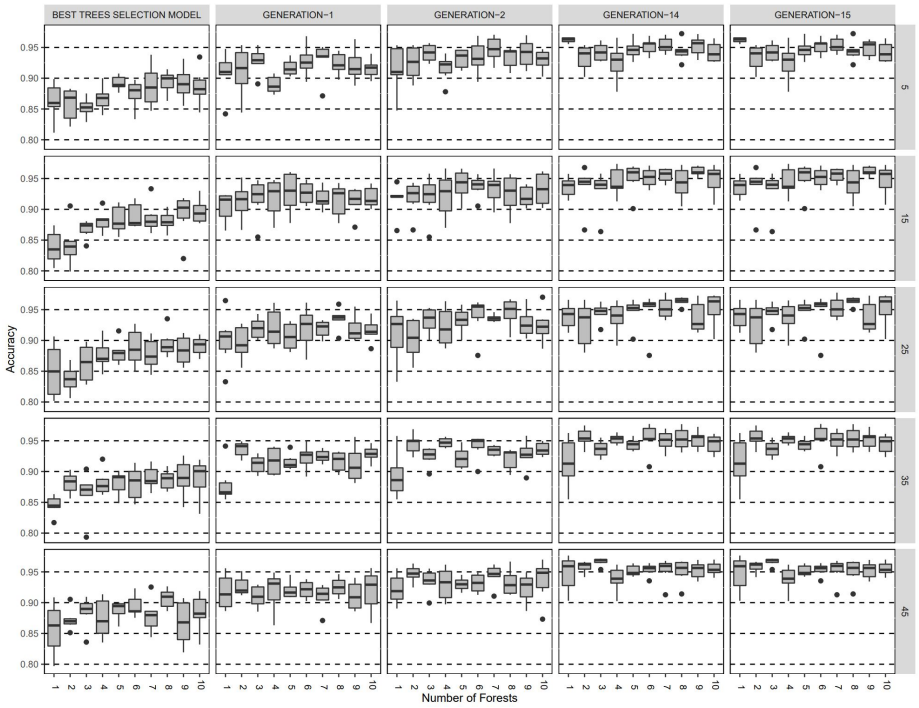

Figure 3: Accuracy of the GMRF-CRISPR model, the first,the second, the fourth-teen, and the fifth-teen generations build on training sets raging from 5% to 45% of the data. The x-axis shows the number of forests used to construct the model.

On the other hand, the greater the number of trees, the initial generations tend to perform worse, but it later generations tend to do better, possible because there are more variety of individuals to mix (see Figure 4).

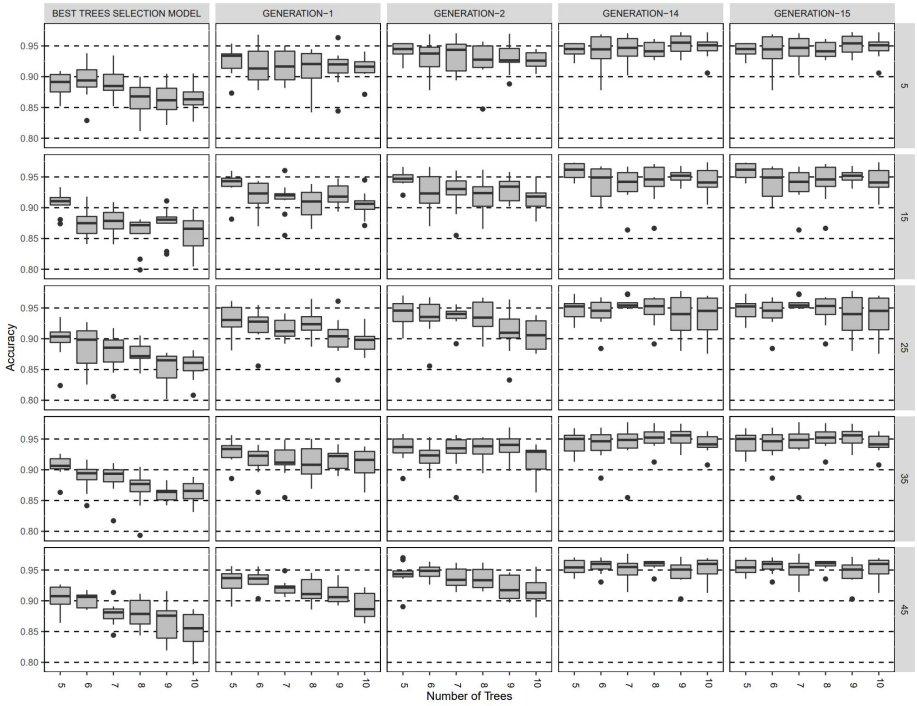

Figure 4: Accuracy of the GMRF-CRISPR model, the first,the second, the fourth-teen, and the fifth-teen generations build on training sets raging from 5% to 45% of the data. The x-axis shows the number of trees on each model.

The best performing model achieved an accuracy of 0.97757 starting with the best trees selection model having an accuracy of 0.87719. The top 5 models are on the Figure 5 below.

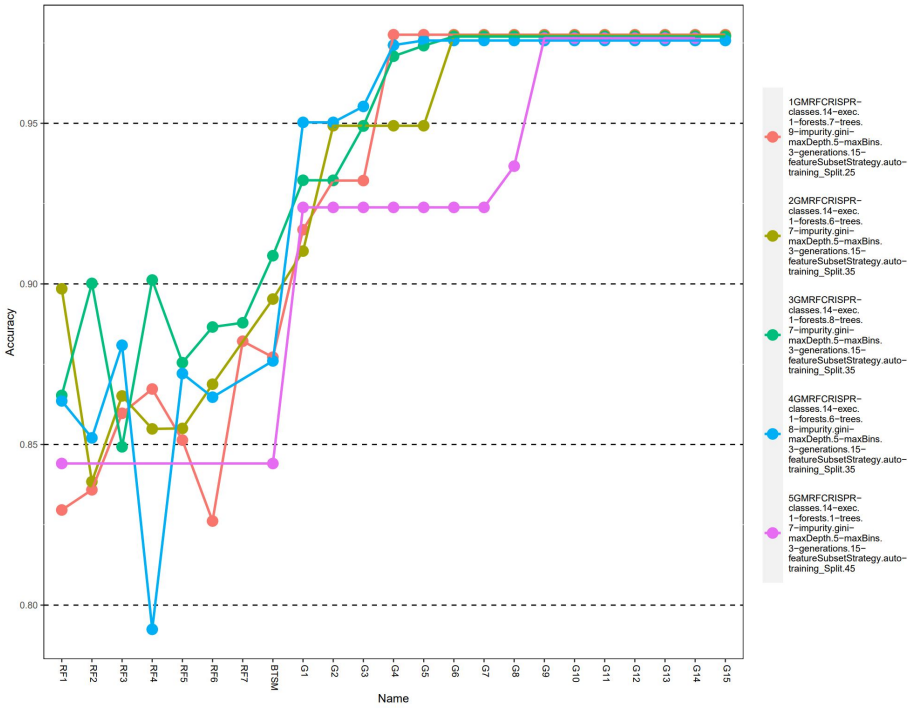

Figure 5: Accuracy of BTSM, each generation, and the RandomForests of the Top 5 models

6 Conclusion

Network data has been increasing in complexity during the last few years. More and more data is being transferred over the internet every day and that makes it very susceptible to attacks and privacy infringement. In this paper, a new algorithm inspired in nature and in gene editing techniques was proposed as an anomaly classifier over network data. To deal with the complexity, multiple RandomForests were created to establish an initial data prediction. The trees with greater accuracy were selected to create the GMRF-CRISPR model. During multiple generations new trees were added to the model to further increase accuracy of it.

The initial simulations of the GMRF-CRISPR have proven to be very effective for detecting intrusions and anomalies on the CSE-CIC-IDS2018. The GMRF-CRISPR has proven to perform better than well established machine learning algorithm like the Random Forests. Further work will be needed to test the model on other data sets and against other algorithms.

References

[1] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) - Registry of Open Data on AWS. A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018).* URL: https://registry.opendata.aws/cse-cic-ids2018/.

[2] Davahli; Shamsi; Abaei. "Hybridizing genetic algorithm and grey wolf optimizer to advance an intelligent and lightweight intrusion detection system for IoT wireless networks." In: *J Ambient Intell Human Comput* (2019). DOI: https://doi.org/10.1007/s12652-020-01919-x.

[3] Ojugo Arnold; Eboka Andrew; Emmanuel Okonta; Yoro Rume; Aghware. "Genetic Algorithm Rule-Based Intrusion Detection System." In: *Journal of Emerging Trends in Computing and Information Sciences.* 3 (2012), pp. 1182–1194.

[4] K. Alrawashdeh and C. Purdy. "Toward an Online Anomaly Intrusion Detection System Based on Deep Learning". In: *15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2016), pp. 195–200. DOI: 10.1109/ICMLA.2016.0040.

[5] Rebecca Bace and Peter Mell. "Intrusion Detection Systems". In: *NIST Special Publication on Intrusion Detection Systems* ().

[6] Ricard Marsal I Castan. *Cluster.* URL: https://github.com/ricardmarsalcastan/cluster.

[7] Thomas Ryan Devine. "Searching for Needles in the Cosmic Haystack". In: (2020).

[8] Brian; Dr Sabine Landau; Dr Morven Leese; Dr Daniel Stahl Everitt. *Cluster analysis.* Wiley Series in Probability and Statistics. Wiley, 2011. ISBN: 9780470749913.

[9] W. Xu F. Kuang and S. Zhang. "A novel hybrid KPCA and SVM with GA model for intrusion detection". In: *Appl. Soft Comput.* 18 (2004), pp. 178–184. DOI: https://doi.org/10.1016/j.asoc.2014.01.028.

[10] Nabila Farnaaz and M.A. Jabbar. "Random Forest Modeling for Network Intrusion Detection System". In: *Procedia Computer Science* 89 (2016), pp. 213–217. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.06.047. URL: http://www.sciencedirect.com/science/article/pii/S1877050916311127.

[11] Belal Al-fuhaidi et al. "Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System". In: *International Conference on Aerospace Sciences and Aviation Technology* 13 (May 2009). DOI: 10.21608/asat.2009.23490.

[12] *Genetically Modified Organisms. National Geographic Society.* URL: https://www.nationalgeographic.org/encyclopedia/genetically-modified-organisms/.

[13] Dimitrios Damopoulos; Sofia A. Menesidou; Georgios Kambourakis; Maria Papadaki; Nathan Clarke; Stefanos Gritzalis. "Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers". In: 5.1 (2011). DOI: https://doi.org/10.1002/sec.341.

[14] M.; Ahmad S. Hasan M.; Nasser and Molla. "Feature Selection for Intrusion Detection Using Random Forest". In: *Journal of Information Security* 7 (2016), pp. 129–140. DOI: 10.4236/jis.2016.73009.

[15] Mohammad Sazzadul Hoque, Md Mukit, and Md. Abu Naser Bikas. "An Implementation of Intrusion Detection System Using Genetic Algorithm". In: *International Journal of Network Security Its Applications* 4 (Mar. 2012), pp. 109–120. DOI: 10.5121/ijnsa.2012.4208.

[16] Philippe Horvath and Rodolphe Barrangou. "CRISPR/Cas, the immune system of bacteria and archaea." In: *Science (New York, N.Y.)* 327,5962 (2010), pp. 167–70. DOI: 10.1126/science.1179555.

[17] Chen Yung-Yao; Lin Yu-Hsiu; Kung Chia-Ching; Chung Ming-Han; Yen I.-Hsuan. "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes". In: *Sensors* 19 (2019). DOI: 10.3390/s19092047.

[18] M. Zulkernine J. Zhang and A. Haque. "Random-forests-based network intrusion detection systems". In: *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* 38 (2008), pp. 649–659. DOI: https://doi.org/10.1109/TSMCC.2008.923876.

[19] W. Li; P. Yi; Y. Wu; L. Pan; J. Li. "A new intrusion detection system based on KNN classification algorithm in wireless sensor network". In: *J. Elect. Comput. Eng.* (2014). DOI: https://doi.org/10.1155/2014/240217.

[20] Wei Li. "Using genetic algorithm for network intrusion detection". In: (Jan. 2004).

[21] Ying Zhong; Wenqi Chen; Zhiliang Wang; Yifan Chen; Kai Wang; Yahui Li; Xia Yin; Xingang Shi; Jiahai Yang; Keqin Li. "HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning". In: *Computer Networks* 169.107049 (2020). DOI: https://doi.org/10.1016/j.comnet.2019.107049.

[22] J. Sander; M. Ester; H.-P. Kriegel and X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *AAAI Press* (1996), pp. 226–231.

[23] Mitchell Melanie. *An Introduction to Genetic Algorithms*. MIT Press. Cambridge, 1996. ISBN: 9780585030944.

[24] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996. ISBN: 9780585030944.

[25] Neelam Mukherjee Saurabh Sharma. "Intrusion Detection using Naive Bayes Classifier with Feature Reduction". In: *Procedia Technology* 4 (2012), pp. 119–128. DOI: 10.1016/j.protcy.2012.05.017.

[26] S. Mukkamala, G. Janoski, and A. Sung. "Intrusion detection using neural networks and support vector machines". In: 2 (2002), 1702–1707 vol.2. DOI: 10.1109/IJCNN.2002.1007774.

[27] Golrang A.; Golrang A.M.; Yildirim Yayilgan S.; Elezaj O. "A Novel Hybrid IDS Based on Modified NSGAII-ANN and Random Forest." In: *Electronics 2020* 9(4) (2020). DOI: https://doi.org/10.3390/electronics9040577.

[28] David Martin Ward Powers. "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation". In: (2007).

[29] R. Primartha and B. A. Tama. "Anomaly detection using random forest: A performance revisited". In: (2017), pp. 1–6. DOI: 10.1109/ICODSE.2017.8285847.

[30] Peter Russell Stuart; Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003. ISBN: 978-0137903955.

[31] Hung-Jen Liao; Chun-Hung Richard Lin; Ying-Chih Lin; Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review". In: 36.1 (2013), pp. 16–24. DOI: https://doi.org/10.1016/j.jnca.2012.09.004.

[32] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. na. Morgan Kaufmann Publishers Inc., 2005. ISBN: 0-12-088407-0.

[33] Yihan Xiao; Cheng Xing; Taining Zhang; Zhongkai Zhao. "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks". In: 7 (2019), pp. 42210–42219. DOI: 10.1109/ACCESS.2019.2904620.