FeSA: Feature Selection Architecture for Ransomware Detection Under Concept Drift

Damien Warren Fernando, Nikos Komninos

Department of Computer Science, School of Mathematics, Computer Science and Engineering, City, University of London, UK

Abstract

This paper investigates how different genetic and nature-inspired feature selection algorithms operate in systems where the prediction model changes over time in unforeseen ways. As a result, this study proposes a feature section architecture, namely FeSA, independent of the underlying classification algorithm and aims to find a set of features that will improve the longevity of the machine learning classifier. The feature set produced by FeSA is evaluated by creating scenarios in which concept drift is presented to our trained model. Based on our results, the generated feature set remains robust and maintains high detection rates of ransomware malware. Throughout this paper, we will refer to the true-positive rate of ransomware as detection; this is to clearly define what we focus on, as the high true positive rate for ransomware is the main priority. Our architecture is compared to other nature-inspired feature selection algorithms such as evolutionary search, genetic search, harmony search, best-first search and the greedy stepwise feature selection algorithm. Our results show that FeSA displays the least degradation on average when exposed to concept drift. FeSA is evaluated based on ransomware detection rate, recall, false positives and precision. The FeSA architecture provides a feature set that shows competitive recall, false positives and precision under concept drift while maintaining the highest detection rate from the algorithms it has been compared to.

Keywords: Ransomware, Concept-Drift, Detection, Learning-Algorithms, Features

1. Introduction

In recent years ransomware has emerged as one of the most potent malware threats out there. Ransomware uses tactics to reduce the victim's access to their system or prevent files by encrypting them. Victims pay for various reasons, whether it is a business that needs access to its files and does not have sufficient backups [4] or a single person who has "lost" personal files due to a ransomware attack. There are two types of ransomware, the first being locker ransomware. Locker-ransomware will stop users from accessing their systems by displaying a lock screen when they log into their systems. The second type of ransomware is crypto-ransomware that our research is focused on. Crypto-ransomware is a highly sophisticated malware type, a more common form of ransomware used in ransomware attacks. Crypto-ransomware will use complex encryption schemes to encrypt a victim's files, rendering them unusable and unrecoverable unless the ransom is paid and the attacker provides the subsequent decryption keys with a decryption tool. A popular example of crypto-ransomware CryptoWall appeared in 2014, and it has generated approximately $320million [1]. Overall it is estimated that in 2020, organisations will pay up to $11 billion in paying ransoms or dealing with the damage caused by ransomware attacks [20]. Popular and infamous ransomware like Petya encrypts the Master Boot Record of a Windows system in terms of behavioural diversity. Modern

ransomware variants like Maze encrypts files, steals sensitive information from companies and then exposes it if organisations and individuals do not pay the ransom [2]. Ransomware malware evolves to become more dangerous and damaging, as history has shown us. In the context of machine-learning detection systems, the constant evolution of malware can be classed as concept drift, a phenomenon that means the rules and logic learned by the classifier to classify malware becomes outdated and incorrect.

1.1. Malware Evolution

According to DataProt [21], around 980 million malware programs on the internet today and 350,000 new malware pieces are detected every day. The recent boom in malware evolution is traced back to 2013, in which the number of malicious files on the web doubled, this growth may have slowed, but it has not stopped. The statistics show that malware is not only emerging at a rapid rate; this is also acknowledged in [23] recognised the diversity in malware in 2008, which implies malware has been evolving and changing for years. Singh et al. described three types of malware evolution, the first being a natural evolution, the second being environmental evolution, and the third being polymorphic evolution. Most aspects of malware evolution are due to adaption to avoid anti-virus (AV) detection. Environmental evolution occurs when software development changes, such as compiler changes. If malware uses different libraries to fulfil its goals, its behaviour may appear significantly different from what detection systems expect. The definition of environmental change depends heavily on compiler and library changes as defined in [22], which means these changes will be far less frequent than natural evolution. Polymorphic evolution occurs in the form of transformation and obfuscation [22]. The use of packers and protectors create an artificial diversity that is designed to evade detectors. Packing will not help track drift, as the packers will encrypt and compress code; drift tracking should be carried out on unpacked malware. Malware evolution poses a large threat to systems due to the rate of evolution not slowing down, according to Symantec [24]. Enterprise ransomware like SamSam and Dharma are coordinated hits on organisations using a manual attack methodology [25]. Doxxing is also a new methodology in ransomware attacks, threatening to expose sensitive data of attack victims [3], yet another example of ransomware's dangerous evolution. When high diversity and evolution rates exist in a destructive malware type like ransomware, the consequences for victims become severe.

1.2. Motivation

Our main motivation for this research is the need for robust features which will allow ransomware detection systems to remain effective when exposed to concept drift. It is observable that features can quickly be rendered ineffective by concept drift; therefore, this creates the need for an architecture that can create robust feature sets for ransomware detection under which will not degrade excessively under concept drift. A zero-day vulnerability is a software vulnerability that attackers discover before the software vendor is aware of it. A zero-day exploit is a method of exploiting a zero-day vulnerability [41]. A zero-day malware threat is a threat that has not been seen by the detection system before and can be a variant or malware type for which no signatures exist [42]. Machine learning classifiers have been proven effective in detecting zero-day malware threats, as shown in [7] and [33]; however, zero-day ransomware is not necessarily an example of ransomware that has evolved and will be difficult for a classifier to identify correctly. Machine learning detection systems like ransomwall [7] and the system designed by Sgandurra et al. use dynamic features like API calls to differentiate ransomware from benign files; therefore, machine learning detection systems effectively detect ransomware rapidly without relying on signatures or heuristics. Machine learning systems will be able to identify patterns and statistical properties of malware that distinguish them from benign files, hence why they are effective at detecting zero-day threats. Zero-day ransomware may be considered a zero-day due method of delivery or how it is obfuscated to evade anti-virus detection; however, once it begins executing, its behaviour determines whether it is an evolved variant or not. A ransomware variant may be delivered via a zero-day attack that exploits a new vulnerability, but its behavioural patterns during execution may not deviate much or at all from the patterns of previous ransomware. Ransomware that displays behavioural patterns during execution that differ from what the machine learning classifier expects shows true evolutionary characteristics, as the dynamic behaviour of the malware has changed. The transcend system [14] acknowledges that malware can evolve in ways that make it difficult for even

machine learning detection systems to detect; this evolution is described as concept drift in a machine learning system. If malware's behavioural patterns and statistical properties change beyond the scope of what a machine learning system defines as malicious behaviour, detection rates will start to decrease. Changes in statistical properties and dynamic behaviour during execution is what we would classify as true malware evolution, as even machine learning systems would struggle to detect them. An example of the differences between evolution and zero-day is the WannaCry ransomware attack, considered a zero-day threat. The attack was carried out using the eternal blue and double pulsar exploits. These two exploits are windows SMB and privilege based and allow the ransomware to execute, the zero-day aspect of this attack. If WannaCry was loaded into a system using eternalBlue; however, it behaved the same as previous ransomware, its characteristics would not be considered evolutionary, only that it had been propagated using zero-day exploits, eternalBlue and doublePulsar. WannaCry could be considered evolved because of the way it encrypted files and propagated itself through networks; these aspects of the ransomware were behavioural evolutions and would display dynamic behavioural characteristics not previously associated with ransomware.

1.3. Contributions of this paper

· Behavioural analysis of ransomware characteristics that change or "evolve" over time.

· Proposal of a feature selection architecture, which provides an optimal feature set showing promising performance when exposed to concept drift. FeSA's feature set remains robust over time; the main element is maintaining a slower degradation rate in detection rate.

1.4. Paper Organisation

The remainder of this paper is as follows: Section 2 covers work related to our research. Section 3 covers our proposal and the background information that accompanies our work. Section 4 describes our experiments, and section 5 discusses the results of the experiments shown in section 4. Section 6 concludes and expands our work.

2. Related work

This section explores related work which has influenced our research. Section 2.1 investigates studies that apply concept drift with pros and cons. This study also investigates evolutionary algorithms and how they can be used in concept drift detection and adaption. This study also investigates the use of machine learning detection for ransomware and how these systems tackle zero-day threats. The related studies identify the gaps in ransomware detection and concept drift in ransomware detection systems, and the genetic algorithms in section 2.2 point us towards possible solutions.

2.1. Concept Drift

Ransomware variants that display evolutionary qualities that are different from their predecessors are always emerging, which may be difficult for ransomware detection systems to identify. Good examples of evolving ransomware are the MedusaLocker and WannaCry ransomware families. MedusaLocker is a ransomware variant that targets antivirus and ransomware detection modules to turn them off and disable them from running in safe mode [40]; this variant of ransomware is extremely evasive and effective in disabling endpoint protection and preventing ransomware detection modules from working. The WannaCry ransomware variant was propagated through a Windows SMB vulnerability that the public had not seen before the infection, although it was known to the NSA at the time. The two variants mentioned behaved vastly differently from the ransomware before them and made it clear that a zero-day that showed characteristics far beyond the current behavioural profile can cause detection systems to fail.

The Transcend System proposed in [14] is a framework that can work with any machine learning algorithm to output confidence values for predictions. Predictions can be modelled differently; confidence values can be extracted from a random forest depending on how many trees vote for the chosen prediction. Confidence

values can be extracted from a Support Vector Machine by measuring a prediction's distance from the hyperplane. Obtaining confidence from a clustering approach would involve measuring the distance of a prediction from a centroid. The transcend system aims to identify how similar the classified instance is to the rest of the instances in its class and how similar the instance is to samples in the other classes. The transcend system measures the confidence of a prediction and combines the value with the confidence the predictor has in other classes to determine how credible the prediction is. Predictions that fall below the credibility threshold will have to be investigated manually by an IT team or some administrative presence. Transcend does not use any evolutionary feature selection algorithms to train algorithms; however, the framework proposed by them uses a similar structure and approach to an evolutionary algorithm.

The system proposed in [17] combines human intervention with underlying machine algorithms to address concept drift in an adversarial machine learning scenario; this system attempts to classify adversarial learning as an evolutionary family of the training dataset. The system proposed in [17] stresses the need for retraining and human interaction to handle concept drift effectively. The type of concept drift addressed in [17] is a type of drift introduced by adversarial techniques that are not addressed in any other related studies referenced in this study. The system proposed in [18] uses anomaly detection to distinguish between genuine changes in a web application and malicious changes; however, this system also relies on retraining to adapt to concept drift and reduce false-positive rates. The system proposed in [18] is unique because it looks specifically for malign and benign changes; despite this, the retraining of parts of the model is necessary to adapt to the detected changes. The systems that address concept drift seem to rely on retraining and human intervention instead of having a specifically constructed mechanism to counteract the effects of concept drift.

The system used in [38] uses the Heterogeneous Euclidean Overlap Metric (HEOM) to detect concept drift in detecting malicious web URLs. The system combines Gradient Boosted Trees to detect malicious URLs and the GTB algorithm with the HEOM measurement. The concept drifts detection component of the system in [38] attempts to identify the differences between the data distribution between the old training data and the new incoming data. The distance between the training set and the newer data is calculated using the HEOM. The research presented in [39] attempts to detect concept drift in malicious URL detection systems and uses the Wilcoxon Rank-Sum. The Wilcoxon Rank-Sum test is a non-parametric test that allows the user to determine whether two samples are from the same population. In the context of malicious web URLs, the Wilcoxon Rank-Sum test allows the system to determine whether the incoming URL matches its allocated classification. Thus, if a concept drift is detected, the system will be immediately retrained.

2.2. Genetic Algorithms

A genetic algorithm is a search heuristic that takes Charles Darwin's natural evolution theory [29]. This algorithm mimics the process of natural selection, which will select the strongest to survive and produce offspring. A genetic algorithm will apply this logic to a dataset and can be used to produce an optimal feature set. The system proposed in [28] uses a genetic algorithm to produce an optimal feature set for malware detection. A typical genetic algorithm will repeat its evaluation and crossover phase, creating numerous features to obtain optimal features. This research uses a genetic engineering approach to reduce the number of generations and features needed to produce the optimal feature set, otherwise known as a feature set. The structure of a genetic algorithm is shown below.

· **Fitness Function:** The fitness function determines the ability each individual has to compete, in the context of a detection system, this would be determined by how accurate a feature set is.

· **Population Generation:** The initial population of individuals is generated randomly from the pool of available chromosomes; in most cases, chromosomes represent features that will create a feature set.

· **Selection** The selection phase is designed to take the fittest individuals and allow them to pass their genes onto the next generation. In the context of a detection system, these would be feature sets that achieve the highest accuracy.

· **Crossover:** Crossover is the process of two selected individuals being mated to produce a child, which will be a combination of both parents. This phase can be repeated with the offspring and so forth but can be limited to a select number of generations.

· **Mutation:** Genes of the offspring can be subject to mutation with a low random probability, in the context of a feature selection algorithm, this can mean inheriting a random feature that does not exist in either parent.

The StreamGP algorithm [27] constructs an ensemble Genetic Programming and a boosting algorithm. The StreamGP system generates decision trees that are trained on different parts of a data stream. StreamGP has a concept drift detection system inbuilt, which, once triggered, will build a new classifier using CGPC, the cellular genetic programming method described in [27]. The populations of data in this algorithm are sets of individual data blocks which are initially drawn randomly. The newly created classifier is added to the ensemble, and the weights of each classifier are then updated; this system creates a new classifier when concept drift is detected rather than constantly adapted to the newest block of data like the EACD proposed in [19].

The EACD system [19] proposes a genetic algorithm approach to combatting concept drift. This evolutionary algorithm is multi-layered with a base and a genetic layer; both layers act as a natural selection mechanism to find the strongest feature set. The base layer will select a set number of features and save them as feature sets. These feature sets are saved and evaluated. The highest performing feature sets are passed into a secondary genetic layer that will "breed" feature sets by randomly crossing strong feature sets to create strong offspring. This breeding step is carried out until the overall system's accuracy is higher on the newest data. The number of repetitions of the breeding step is defined by the maximum number of generations the system will allow. This genetic approach produces promising results, finding optimal feature sets for systems that model scenarios that present concept drift.

The Online Genetic Algorithm (OGA) [27] is a rule-based learner that updates its ruleset based on the data stream's evolution. Like the base layer in the EACD system, the initial rulesets are chosen randomly, and the genetic algorithm is applied when a new block of data is encountered to update the rulesets. This process is repeated until the end of the data stream. Each block of data is a different iteration, which leads to a large number of iterations. OGA does not limit the number of iterations the algorithm can go through, which means it can become very expensive.

2.3. Ransomware Detection

Ransomware detection research that integrates concept drift is a rarity in the research space. The Elderan system described in [33] considers zero-day attacks and tests on samples that the model has not been trained on. The Elderan system's accuracy drops from 96% to 93% when exposed to zero-day threats; however, it is unclear if the zero-day threats are more than a couple of months ahead of the training set. The explicit testing on zero-day threats is explored in [32] and [34], similar to the Elderan system, which can be considered testing under concept drift; however, the zero-day samples are not guaranteed to display concept drift in regards to the training samples.

The RansHunt system described in [31] attempts to predict future ransomware trends by training on "Ransomwall", a ransomware hybrid that authors predicted to be a future ransomware type. According to the creators of RansHunt, a worm component would be used to spread ransomware through a compromised network. This prediction approach and preparation for future trends could prevent models from degrading under concept drift. The system explored in [35] explores using a generative adversarial system to produce variations of ransomware that might deceive ransomware detection systems; this approach is designed to highlight the need for ransomware detection systems to be reinforced.

3. FeSA- Feature Selection Architecture

The previous sections in this study discussed malware evolution and ransomware. This section introduces our proposal to combat the concept drift in ransomware detection systems. The FESA system proposes using

an architecture, which generates feature sets for ransomware detection systems through information gain and a genetic algorithm. Our approach relies on the user's underlying machine learning algorithm but is compatible with any machine learning approach. The underlying machine learning algorithm will be the classifier trained on the feature set produced by the FeSA architecture. Genetic algorithms are proven effective for concept drift scenarios when used by the systems described in [26], [27] and [28]; the obtained results lead us to FeSA, which does not entirely rely on the natural selection mechanism to produce an optimal feature set.

3.1. Preliminaries

This section contains necessary background information on concept drift and genetic algorithms. This section also presents table 1, which gives the notation of the symbols used throughout the paper.

Table 1: Notations

Symbol	**Explanation**
x_i	A feature in a feature set.
$\overline{x}$	The feature x does not appear
$IG(x_i)$	Information Gain for a feature x_i
c_i	The classification of an instance into category i
$p(c_i\|x)$	Conditional probability of the i^{th} category given the feature x appears.
$p(c_i\|\overline{x})$	The conditional probability of the i^{th} category given the feature x does not appear.
$\|Z\|$	The size of the set of important features. The important feature set is added to every feature set produced by FeSA.
a	The proportion of features from the feature set which meet the requirements for being important features.
$T(f)$	Total features in the initial feature set.
$\|N\|$	The size of a feature set generated by the base layer, this feature set is part of the first generation of feature sets produced.
r	A proportion of the original feature pool.
H	High performance feature sets.
m	Maximum feature set limit.
d_r	Average detection rate of feature sets in the base layer.
a_r	Average accuracy of feature sets in the base layer.
Y_i	A feature set produced in the base layer.
H_{rand1}	· A selected parent feature set in the genetic layer.
H_{rand2}	A selected parent feature set in the genetic layer.
O_i	An offspring feature set in the genetic layer.
T	A set of offspring feature sets.

3.1.1. Concept Drift

Concept drift is defined as the change in relationships between inputs and output data in the underlying problem over time [15]. Concept drift will make classifiers degrade over time leading to more incorrect classifications. Incorrect classifications in the context of malware detection can cause problems. A malware analysis team would have high standards for abandoning an ageing classification model [14]. In the context of a ransomware classification, a model would have to be constantly monitored for signs of concept drift due to the damage one ransomware infection can cause. Concept drift can occur gradually over time or artificially to cause classifiers' errors, as stated in [17].

Concept drift can fall into the following three categories;

· **Gradual Concept Drift:** A gradual change over time.

· **Cyclical Concept Drift:** A recurring or cyclical change.

· **Abrupt Concept Drift:** A sudden or abrupt change.

The relationship between a classifier and its predictions is defined as $p(y|x)$ and concept drift can be defined as changes in $p(x, y)$ [12]. The changes in this joint probability can be identified through its components, suggesting that different detection aspects can cause concept drift.

The FeSA system is built to adapt to sudden concept drift and gradual concept drift. Sudden concept drift is the type of concept drift that poses the biggest threat to a malware detection system. The sudden appearance of new ransomware which does not conform to a model's current configuration is a problem that cannot be solved by retraining unless the retraining is done before the system is exposed to the new ransomware. FeSA is effective when dealing with gradual concept drift because the system is built using features from different distributions. Using different distributions to build the FeSA feature set allows the system to capture the best possible feature set, which applies to ransomware from different eras. Capturing common features from many different types of ransomware from different periods gives FeSA the best chance of having features that will remain relevant in the future.

3.2. FeSA Architecture

We propose FeSA, a feature selection architecture for ransomware detection under concept-drift. The FeSA architecture is shown in figure 1 and is comprised of three main components. FeSA architecture is built following the structure of a genetic algorithm. The FeSA architecture needs to be provided with an initial feature pool to create feature sets with. The number of features in this initial feature pool is user-defined. The larger the number of features in the initial feature pool, the larger the number of unique and diverse feature sets the base layer can create. The feature ranker selects a set of "important" features from the feature pool to pass onto the feature base layer. The base layer generates a set of random feature sets from the feature pool, ensuring these feature sets include the important features. The feature sets in the base layer are evaluated, and their detection rate and overall accuracy are calculated. The feature sets that achieve accuracy and detection rates above the average accuracy and detection rates of all of the feature sets in the base layer are defined as high-performance and passed onto the genetic layer. The genetic layer performs a breeding crossover procedure involving selecting two high-performance feature sets from the base layer and combining them to produce a new feature set; the user defines the number of times the crossover process is repeated. In theory, the combination of high-performance feature sets from the base layer should produce new feature sets which can achieve higher accuracy and detection rates than feature sets combined to create them.

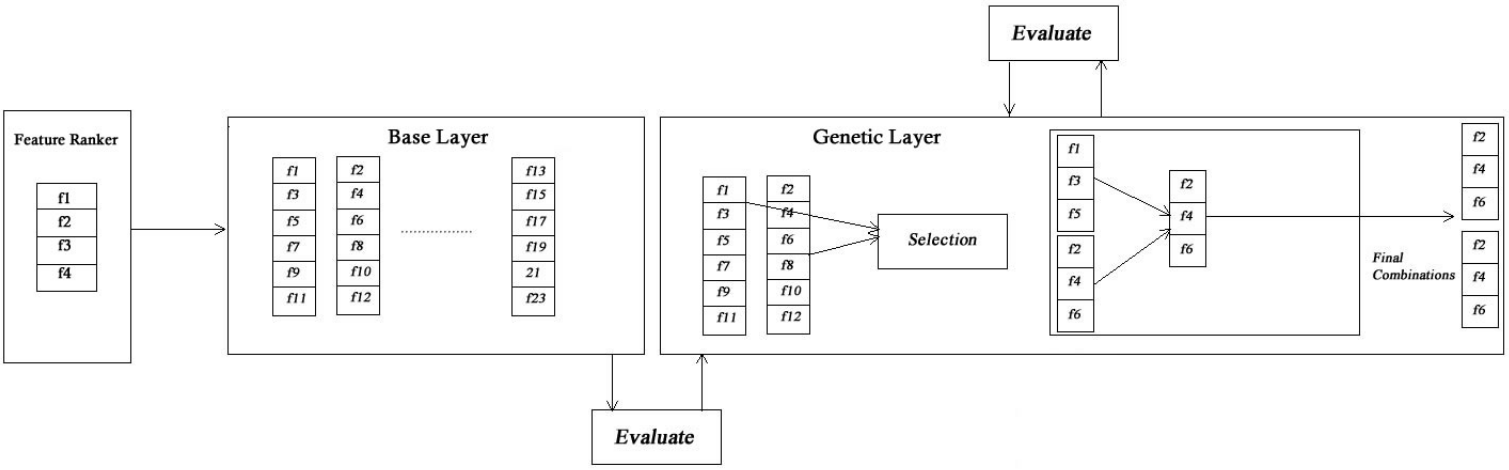

Figure 1: FeSA: Feature Selection Architecture

3.2.1. FeSA Feature Ranker Algorithm

The initial population of feature sets is randomly generated; however, the FeSA architecture uses a feature ranker to identify the highest information gain features. Information gain reduces the complexity of the generated important features because random feature selection requires multiple selections to find the optimal set. FeSA controls the base each feature set is built upon, ensuring strong feature sets. Before generating the initial population, a feature ranking algorithm is proposed to decide which features are most important. The feature ranker is the base component of our system because it ranks features in order of their importance and attaches a numerical value to this ranking. Information gain is calculated according to Eq.1. The feature importance step is designed to provide the initial "building blocks" for each feature set. The ranker algorithm uses information gain to isolate the most important features; it determines information gain and then ranks features to gain information. Information gain is the reduction in entropy after a dataset is split on an attribute. Entropy is defined as a measure of randomness in information; therefore, the higher the entropy, the harder it is to draw any conclusions from the data [36].

Information gain (IG) is a reduction in entropy when splitting on an attribute and is calculated in eq.1, c_i represents the ith class category i.e. ransomware or benign, and $p(c_i)$ is the probability of ith category. $p(c_i|x)$ is the conditional probability of the ith category given the feature x appears. and $p(c_i|\overline{x})$ is the conditional probability of the ith category given the feature x does not appear.

$$IG(x_i) = -\sum_{i=1}^{m} p(c_i) + p(x)\sum_{i=1}^{m} p(c_i|x) \cdot log\, p(c_i|x) + p(\overline{x})\sum_{i=1}^{m} p(c_i|\overline{x}) \cdot log\, p(c_i|\overline{x}) \quad (1)$$

The feature set, taken from the feature ranker, is defined in eq.2. The variable a is dependent on the features defined by the feature ranker as essential. $T(f)$ represents the total features in the original feature pool. Our FeSA implementation chooses features with an information gain equal to or greater than 0.5 as "important" features. The decision to set 0.5 as the threshold value was based on the fact that information gain is a reduction in entropy, a measure of randomness; therefore, features were chosen, which took away at least half of the data's randomness. Based on experimental observations, very few features exceeded or matched this value during our experiments which meant a threshold of 0.5 would mean only some features are selected as "important". Z is defined as the set of important features, which every feature set must contain, $|Z|$ is the cardinality of this set. An algorithmic representation of the feature ranker is shown in Algorithm 1. In addition to defining the key features included in each feature set, the ranker eliminates

features deemed to provide 0 information gain. The FeSA system uses the ranker to identify features that present zero information gain and excludes them from the base layer and subsequent genetic layer.

$$|Z| = \frac{a}{T(f)} \tag{2}$$

Algorithm 1 FeSA Feature Ranker

Input: Initial features $x_0, ..., x_i$
Output: Important feature set Z

1: **for** Initial Features x_0 to x_i **do**
2: Calculate Information Gain for each feature using:
$IG(x_i) = -\sum_{i=1}^{m} p(c_i) + p(x) \sum_{i=1}^{m} p(c_i|x) \cdot log\, p(c_i|x) + p(\overline{x}) \sum_{i=1}^{m} p(c_i|\overline{x}) \cdot log\, p(c_i|\overline{x})$
3: **if** $IG(x_i) \geqslant 0.5$ **then**
4: Add x_i to important feature set Z
5: **end if**
6: Return important feature set Z
7: **end for**

Algorithm 1 shows the operation of the feature ranker. The feature ranker takes an initial set of features x_0 to x_i and calculates each feature's information gain IG. If the feature x_i has an information gain value above or equal to 0.5, it is added to the important feature set Z. Each feature in the important feature set is denoted as z_i.

3.2.2. FeSA Fitness Function

The fitness function used by FeSA calculates the average detection rate and accuracy amongst all feature sets in the current generation. The highest performing feature sets which display above average detection rates and accuracy are passed onto the next generation by the fitness function. The fitness function is used in the base layer and the subsequent genetic layer. Our fitness function uses the ranker's values, but indirectly as opposed to directly in its calculations. The ranker will enforce features with the highest information gain and eliminate features with no information gain, thus ensuring that the feature sets produced in the base and genetic layers will provide as much information as possible while removing excess features that provide no information.

3.2.3. FeSA Base Layer

The FeSA base layer acts as the initial population generation required by a genetic algorithm. The base layer randomly generates feature sets from a pool of initial features. The initial population is a requirement of a genetic algorithm and is needed in order to generate strong feature sets in the genetic layer; the main difference between the base layer and a regular population layer is that the ranker has already defined a set of features that are enforced in each generated feature set. The ranker enforcing important features in the base layer feature sets means the base layer feature sets will already have higher accuracy than if the feature sets were randomly generated. The base layer follows on from the ranker.

$$|N| = (\frac{r}{100} \cdot T(f)) + |Z| \tag{3}$$

The number of features selected per feature set is shown in eq.3, where r is the proportion of the initial feature pool in each generated feature set and N is a feature set generated by the base layer. N is calculated as r divided by 100, which obtains a proportion of $T(f)$, $T(f)$ being all the of the features in the initial feature pool plus $|Z|$ which is the important feature set chosen by the ranker.

This process of generating feature sets is repeated as often as the user defines and will define the population size for each generation. The number of repetitions will be balanced with a defined size for each feature set. There are many features in the feature pool; therefore, possible combinations should be heavily regulated to avoid massive computational costs. FeSA evaluates the feature sets that have been generated using the initial population with a random forest classifier. The user's use of the underlying algorithm is flexible and determined based on their features and data. The random forest performed best with our features; therefore, it is chosen as our underlying algorithm. The feature sets will be evaluated on overall accuracy and ransomware detection rate; therefore, only the most accurate feature sets with the highest detection rates are passed onto the next phase. The highest performing feature sets are determined by calculating the average accuracy and ransomware detection rates of all feature sets and passing on the feature sets with accuracy and detection rate above the average. The structure of the feature set generation layer is shown in algorithm 2. The abbreviations used in algorithm 2 are as follows, True Positive (TP), False Positives (FP), False Negatives (FN), True Negatives(TN).

Algorithm 2 shows the operation of the FeSA architecture base layer. The initial features are taken, and new feature sets are generated, including the important feature sets Z, denoted as Y_i. The important features from the feature ranker are added to each generated feature set. The average detection rate and accuracy of every feature set generated, Y_i, is calculated, and if it shows above-average performance, it is placed in the H, the set of high-performance features.

Algorithm 2 FeSA Base Layer

Input: Initial features $x_0, ..., x_i$, Important feature set Z
Output: High performance feature sets H

$m \rightarrow$maximum feature sets
Average detection rate $d_r = 0$
Average accuracy $a_r = 0$
Total detection rate $t_d = 0$
while feature set count$< m$ **do**
    Generate feature set Y_i
    Add important features Z to Y_i
end while
Calculate detection rate of Y_i using:
True Positive Rate (TPR) = $\frac{TP}{TP+FN}$
Calculate accuracy of Y_i using:
Accuracy (ACC) = $\frac{TPR}{TPR+TN+FP+FN}$
Calculate average detection rate d_r using:
$d_r = \frac{\sum_{i=0}^{m} TPR(Y_i)}{m}$
Calculate average accuracy a_r using:
$a_r = \frac{\sum_{i=0}^{m} ACC(Y_i)}{m}$
for $Y_0 ..., Y_i$ **do**
    if detection rate & accuracy of $Y_i > d_r$ & a_r **then**
        Add Y_i to high performance feature sets H
    end if
end for
    return H

3.2.4. FeSA Genetic Layer Algorithm

The FeSA genetic layer acts as the crossover phase in a genetic algorithm. The genetic layer is needed to produce strong feature sets. The feature sets are expected to reach optimal performance after the crossover phase has been completed multiple times. The feature sets produced in the genetic layer will be candidates for the optimal feature set. The genetic layer contains the high-performance feature sets from the base layer

and will combine these high-performance feature sets using the uniform crossover method to yield more accurate feature sets. The genetic layer has the advantage of enforcing important features in each feature set; therefore, the iterations needed for the feature sets to reach optimal performance is reduced in theory.

The genetic selection layer is a breeding mechanism for the highest performing feature sets taken from the initial feature selection layer. The genetic layer is made up of "parent" and "offspring" feature sets. The "parent" feature sets are the high-performance feature sets from the base layer. The "offspring" feature sets are produced by choosing two-parent feature sets and combining them with a crossover function. High performing "parent" feature sets will be combined using uniform crossover, generating "offspring" feature sets. In theory, the offspring feature sets will display a higher performance level than the preceding generation. Uniform crossover takes two-parent feature sets and combines them. For each corresponding feature in each parent feature set, the feature the offspring feature set receives is determined by a coin-flip; this is a probability of 0.5. The crossover function used by FeSA is user-defined; however, for our purpose and need to enforce particular features into feature sets, the uniform crossover function proved to be the most efficient. The resulting offspring feature sets are evaluated, and the feature set with the highest average detection rate and overall accuracy is chosen as the optimal feature set. An important factor in this phase is that only one generation generates the optimal feature set. The structure of the genetic layer is shown in algorithm 3.

Algorithm 3 FeSA Genetic Layer

Input: High performance feature sets H
Output: Optimal feature set O_i

$m \rightarrow$Max feature set count
$n \rightarrow$Current feature set count
Offspring feature set T
while $n < m$ **do**
    Select random base feature set 1 H_{rand1} from set H
    Select random base feature set 2 H_{rand2} from set H
    Perform uniform crossover using H_{rand1} and H_{rand2} & Generate mixed feature set O_i
    if Duplicate features are detected **then**
      Replace duplicate feature with a random feature x_i from feature pool
    end if
    Add O_i to offspring set T
end while
return Optimal $O_i \in T$ which has the highest average detection rate & accuracy.

Algorithm 3 shows the genetic layer of the FeSA architecture. The High-performance feature sets from the base layer. Two random high-performance feature sets are selected H_rand1 and H_rand2 , and uniform crossover is carried out to mix the two high-performance feature sets to create a new feature set O_i. The process of mixing the high-performance feature sets is repeated m times until completion. The best performing of these newly generated feature sets is stored in set T. The best performing feature set out of the newly generated feature sets in T is selected as optimal.

3.3. Mutation

The mutation function in a genetic algorithm is the introduction of diversity. A mutation would mean an offspring feature set inheriting a feature not present in either parent feature set in a feature selection context. During the crossover phase, duplicate features are prohibited from being in a feature set. If there is a feature set with duplicate features, duplicates will be replaced with a random feature from the feature pool, leading to a 0.01% mutation rate. The low mutation rate is used to eliminate unnecessary randomness from the FeSA architecture.

4. Experimental Setup

Our experiments' main aim was to test the FeSA architecture's effectiveness and compare it to a genetic and an evolutionary-based feature selection algorithm. The architecture is evaluated in scenarios where the test samples display concept drift and do not behave according to what the classifier expects and compare it with other similar feature selection algorithms. FeSA has also compared a greedy stepwise algorithm, genetic search, evolutionary search, best-first search and harmony search. Table 2 shows the formulas used to calculate the performance metrics. We refer to the TPR of ransomware as detection rate throughout this paper.

Table 2: Performance Metrics

Metric	Calculation	Value
TPR (True Positive Rate) / Recall	$\frac{TP}{(TP+FN)}$	Correct classification of Ransomware.
False Positive Rate (FPR)	$\frac{FP}{FP+TN}$	Benign software classed as Ransomware.
False Negative Rate (FNR)	1-TPR	Ransomware classed as benign.
Precision	$\frac{TP}{TP+FP}$	Proportion of ransomware classifications, that are actually ransomware.

4.1. Testbed

4.1.1. Environment

Our test environment consists of a Cuckoo sandbox analysis environment that generates the data used for our datasets. Each ransomware and benign executable in our dataset was executed in a virtual machine running Windows 7 in VirtualBox. The virtual machines were cloaked and hardened by VMCloak and Paranoid Fish. The virtual machines were hardened to make them look and behave as close to a physical machine as possible. The hardening process was undertaken due to modern malware using anti-sandbox technology to prevent proper execution in a sandbox environment. Each execution was limited to two minutes; this was the default for Cuckoo Sandbox. The machine learning platform used for the is WEKA(Waikato Environment for Knowledge Analysis), a collection of machine learning algorithms for data analysis.

4.1.2. Data

The ransomware samples used are from 2013 to 2019; the samples from 2013 to 2015 were gathered using the Elderan dataset[6], which contained a list of hashes for each ransomware sample they used. These samples are used as there was a wide range of ransomware from 2013 to 2015. The samples from 2016 to 2019 were gathered based on popularity and how much each ransomware family has made in ransoms. The dataset consists of 639 ransomware files and 531 benign files; the benign files are a mix of windows executables that includes legitimate software that behaves similarly to ransomware, such as AxCrypt, Bitlocker 7zip and VerCrypt. Our experiments are carried out using a random forest and 10-fold cross-validation. The random forest algorithm is used because it is the algorithm that performs the best with our API features. Our base data set contains 400 benign and 531 ransomware samples. New ransomware files and new benign files were added for each round of experiments in concept drift conditions. Each round of experiments uses the most prominent ransomware samples from 2017, 2018 and 2019. API (Application Programming Interface) data is extracted for each sample. The API calls dictate how an executable interacts with the OS and what functions an executable invokes.

This research uses a nonconformity measure to prove concept drift exists in the datasets used. The credibility p-value of each prediction is to measure the credibility of each prediction a classifier makes. The p-value measures the proportion of instances, which are as different or more different from the rest of the instances in the dataset as the new instance z. A high credibility value means that z is very similar to the objects in the class chosen by the classifier, and low credibility would imply the opposite. The experiments were carried out using the random forest classifier; therefore, the prediction probabilities extracted from the random forest are used to calculate the p-values needed to prove drift. For example, it can be observed that, when trained on data from 2013-15, the average credibility of predictions on ransomware from 2015 was 0.9. When the 2015 model is tested on ransomware from 2016-17, predictions' credibility drops to an average of 0.74. The drop in credibility is present in every scenario in the series of experiments carried out. There is an average drop in the credibility of predictions of 0.21. The drop in credibility shows that the classifier becomes more uncertain of its predictions, which indicates the data is behaving in a way it is not prepared for; this indicates concept drift.

4.1.3. Features

The feature pool of 320 features consisted of API calls used by Windows programs during execution. The 320 features fall into API 16 call categories, which the feature set sizes are based on. We aim to capture two features per category on average; however, this does not always prove the case due to the natural selection mechanism. We choose to capture two features from each category to limit the feature size and complexity of the crossover phase.

4.2. Experiments

Our experiments are set up to test the strength of the feature sets produced by FeSA in concept drift scenarios. The experiments compare the performance of the FeSA feature sets with other nature-inspired feature selection algorithms. The datasets used in these experiments are structured to display real-life concept drift scenarios, and the validity of the concept drift in these datasets are tested by p-values, as mentioned in section 4. The concept drift effect is achieved by having ransomware and benign software from different periods. Each classifier is tested on data produced after the data the classifier is trained on. The process runs on our base dataset that contains ransomware and benign samples from 2013 to 2015. The optimal feature set is produced, and FeSA trains a random forest using 10-fold cross-validation and observes the results. The benchmark algorithms are tuned and run by us. The settings used for the benchmark algorithms are tuned to compare them to FeSA as fairly as possible. The closest algorithms to FeSA, the genetic search and the evolutionary search, are given an advantage over FeSA, in which they use more generations to generate their feature sets. The underlying classification algorithm used with FeSA and the benchmark feature selection algorithms is the random forest classifier. FeSA's results are compared to the results obtained from the greedy stepwise algorithm, genetic search, evolutionary search, best-first search and harmony search. After observing results on data the classifiers would see as up to date data, FeSA tests how the classifiers perform on ransomware and benign data from 2016 and 2017. Ransomware and benign samples from 2016 and 2017 would represent concept drift as their behavioural patterns are different, as per our observations. The process of observing detection rates under concept drift is repeated by training on data from 2013-2017 and testing on data from 2018 and again, repeated for data up to 2019. It is observed how the feature sets produced by the FeSA architecture perform compared to the feature sets produced by the greedy stepwise algorithm, genetic search, evolutionary search, best-first search and harmony search. The results of our experiments are shown in section 5.

4.2.1. Detection Phase

The detection phase of the framework is tested in the experimental test phase. The optimal feature set chosen by FeSA is tested on a dataset made up of ransomware and benign files from a future time to the training data. Our detection phase simulates the system coming into contact with ransomware which displays concept drift and is from a different distribution and may behave differently to what the classifier expects of ransomware. The breakdown of the datasets is described in section 4.2. The detection phase is repeated for the algorithms FeSA is compared with.

4.2.2. Genetic Search Algorithm

The genetic algorithm used as a benchmark for our experiments was a basic genetic algorithm that followed the structure described in section 3.1. The experiments used the configuration suggested by WEKA [37], a generational and population limit of 20, a crossover probability of 0.6, and a mutation probability of 0.033. The default settings in both the genetic search and evolutionary meant these algorithms had a significant advantage over FeSA in population generation and feature set size. The machine learning algorithm used with the Genetic Search feature selection algorithm is the random forest to maintain consistency and fairness compared with the FeSA algorithm. The genetic algorithm used in the experiments used both overall accuracy and overall information gain as fitness functions as provided by WEKA and found minimal difference between the two; therefore, the overall accuracy was chosen as it was closest to the fitness function used by FeSA.

4.2.3. Evolutionary Search Algorithm

As presented in WEKA, the evolutionary algorithm also followed a similar structure to the genetic algorithm described in section 3.1; however, it uses a different configuration. The evolutionary algorithm used a tournament selection method with a mutation probability of 0.1. The tournament selection approach ensures that the fittest feature sets are passed onto the next generation. The generation and population limit were set to 20, respectively, like the genetic algorithm. The machine learning algorithm used with the evolutionary algorithm feature selection is the random forest to maintain consistency and fairness compared with the FeSA algorithm. The fitness function used for the evolutionary algorithm is the overall accuracy to maintain consistency with the FeSA algorithm.

5. Experimental Results and Discussion

5.1. Experimental Results

This section explores and elucidates our results. In our experiments, the FeSA architecture used the random forest classifier with 10-fold cross-validation. Our implementation used the WEKA API and an in-house feature extraction program to create our dataset. The experimental findings are presented in Table 3, Table 4 and figure 2. Table 3 shows the key statistics of the detection algorithms tested, including the FeSA architecture when concept drift is not applied. The experiments' first aim was to ensure that FeSA was a viable feature extraction without considering concept drift. FeSA must function as a normal feature selection algorithm before it is tested on an evolving concept. Based on the results in Table 3, FeSA architecture produces features robust in time. Table 4 shows the performance of FeSA and the algorithms FeSA is compared with when under concept drift. The experiments aimed to demonstrate that the FeSA architecture will be superior when exposed to concept drift, and we conclude this has been achieved. Our initial observations are as expected, that the effect of concept drift degrades a ransomware classifier, as it would degrade any other classifier which works in a rapidly changing environment. Our second observation is that using nature-based feature selection algorithms helps slow the degradation of detection rate and accuracy caused by concept drift. Figure 2 provides a visual representation of the classifiers' performance degradation trained and tested under concept drift. Figure 2 does not consider the testing on 2018 data, as the results showed an increase in the detection rate. The detection rate is specifically the rate of correctly identified ransomware samples. The false-positive rate, precision, and recall are based on the systems' overall performance, including the benign samples.

Table 4 shows the average reduction in detection rate under concept drift; it is observed that a feature selection algorithm that pinpoints distinguishing features can help significantly reduce the effects of concept drift on a classifier. Table 3 shows that FeSA architecture maintains a detection rate above 96% and a false positive rate close to the greedy stepwise algorithm, genetic search, evolutionary search, best-first search and harmony search. The first set of experiments simulates a scenario where the samples adhere to the current concept, and the test samples do not stray from the statistical rules the model has created for differentiating ransomware and benign software. Our results presented in Table 3 show that FeSA is a viable feature selection algorithm for training a system and is not necessarily viable for systems prone

to concept drift. The FeSA architecture achieved stronger detection rates across the periods it has been deployed in and consistently outperforms the feature selection algorithms it has been compared to. Most importantly, it can be observed that the FeSA architecture outperforms the evolutionary search and the genetic search while using significantly fewer features and generations of feature sets. There is an emphasis on detection because it is the most important statistic; however, our false-positive rates are competitive with other feature selection algorithms. It can be observed that the FeSA architecture can generate a strong feature set for a random forest classifier while using only two generations, compared to the 20 generations used by the genetic and evolutionary search algorithms.

The experimental results for a detection scenario that introduces concept drift is presented in Table 4. A concept drift scenario is when the test samples do not adhere to the statistical rules and properties the classifier has learned to differentiate ransomware and benign software, in Table 4. It is observed that the most consistent algorithms are the genetic search, evolutionary search and our feature selection architecture, FeSA. Our experiments use the same data and concept drift scenario for each algorithm while observing each classifier's performance changes. The classifier's accuracy and detection rate fell in each scenario except being trained on data from 2013 to 2017 and tested on data from 2018. The feature set generated by FeSA maintains a detection rate above 93% in all our concept drift scenarios, which is higher than all the other approaches it is compared to. The results presented in table 4 gives us a promising base to further build on for dealing with concept drift in ransomware detection systems. The discrepancy in the year 2018 may explain the behavioural changes in ransomware inadvertently benefiting our API based feature pool. The key observation made from Table 4 is that the average reduction in detection rate in concept drift is the lowest in our feature selection algorithm. The other feature sets suffer a higher average reduction in detection rate, which is the key statistic. Figure 2 shows the average reduction in detection rate by each approach; this does not consider the anomalous behaviour in the 2018 dataset. In terms of false positives, our approach appears to struggle marginally more than the other approaches, which would require further research. Our approach appears highly effective when generating a feature set that can identify ransomware. The maintenance of the high detection rate is the key statistic in malware, especially ransomware. Besides ours, the best performing algorithm is the evolutionary search; however, it required 20 generations to reach its optimal solution with a population size of 20 per generation. Our solution uses an initial population of 32 feature sets and one generation of offspring with a population size of 64 feature sets. Our feature sets are also significantly smaller than the genetic and evolutionary search algorithms' optimal feature sets.

Our initial population's average accuracy and detection rate are 77%, and the first generation's average accuracy and detection rate rise to an average of 94%. The accuracy and detection are not expected to increase similarly with increased feature set generations; however, a marginal increase is expected if the number of generations created is higher than one generation. Our experiments use one initial population and one generation to demonstrate the potential of this scenario.

Table 3: Experimental Results without Concept Drift

	FeSA	**Best First**	**Evolutionary Search**	**Genetic Search**	**Greedy Stepwise**	**Harmony Search**
Time Complexity	$O(n) + O(gnm))$	$O(n \cdot Log(n))$	$O(gnm))$	$O(gnm))$	$O(n)^2$	$O(n)$
Feature Count	**32**	**20**	**110**	**106**	**20**	**33**
Trained and tested on 13-15	· Detection: 96.3% · FPR: 5.8% · Precision: 0.942 · Recall: 0.941	· Detection: 92.0% · FPR: 6.7% · Precision: 0.940 · Recall: 0.940	· Detection: 95.7% · FPR: 4.4% · Precision: 0.955 · Recall: 0.955	· Detection: 93.2% · FPR: 6.2% · Precision: 0.942 · Recall: 0.942	· Detection : 90.4% · False Positive Rate: 4.4% · Precision: 0.963 · Recall: 0.936	· Detection: 93.0% · FPR: 5.9% · Precision: 0.944 · Recall: 0.944
Trained and tested on 13-17	· Detection: 96.7% · FPR: 5.8% · Precision: 0.942 · Recall: 0.941	· Detection : 90.1% · FPR: 8.0% · Precision: 0.931 · Recall: 0.932	· Detection: 95.3% · FPR: 5.2% · Precision: 0.948 · Recall: 0.948	· Detection: 95.3% · FPR: 5.2% · Precision: 0.948 · Recall: 0.948	· Detection: 90.1% · FPR: 8.0% · Precision: 0.931 · Recall: 0.932	· Detection: 93.0% · FPR: 5.9% · Precision: 0.944 · Recall: 0.944
Trained and tested on 13-18	· Detection: 96.3% · FPR: 5.9% · Precision: 0.941 · Recall: 0.940	· Detection: 91.5% · FPR: 7.0% · Precision: 0.935 · Recall: 0.936	· Detection: 94.7% · FPR: 5.4% · Precision: 0.946 · Recall: 0.946	· Detection: 91.5% · FPR: 6.5% · Precision: 0.927 · Recall: 0.927	· Detection: 91.5% · FPR: 7.7% · Precision: 0.927 · Recall: 0.927	· Detection: 95.0% · FPR: 5.5% · Precision: 0.945 · Recall: 0.945

Table 4: Experimental Results under Concept Drift

	FeSA	**Best First**	**Evolutionary Search**	**Genetic Search**	**Greedy Stepwise**	**Harmony Search**
Feature Count	**32**	**20**	**110**	**106**	**20**	**33**
13-15 Tested on 2016/17	· Detection: 93.2% · FPR: 7.4% · Precision: 0.913 · Recall: 0.846	· Detection: 77.4% · FPR: 2.7% · Precision: 0.940 · Recall: 0.942	· Detection: 86.7% · FPR: 1.4% · Precision: 0.968 · Recall: 0.968	· Detection: 73.5% · FPR: 4.4% · Precision: 0.961 · Recall: 0.919	· Detection: 79.2% · FPR: 2.4% · Precision: 0.946 · Recall: 0.948	· Detection: 83.0% · FPR: 1.4% · Precision: 0.961 · Recall: 0.962
13-17 Tested on 2018 Data	· Detection: 100% · FPR: 3.4% · Precision: 0.917 · Recall: 0.786	· Detection: 97.8% · FPR: 2.22% · Precision: 0.979 · Recall: 0.976	· Detection: 100% · FPR: 5.5% · Precision: 0.943 · Recall: 0.936	· Detection: 100% · FPR: 1.4% · Precision: 0.989 · Recall: 0.988	· Detection: 100% · FPR: 2.4% · Precision: 0.982 · Recall: 0.979	· Detection: 54.0% · FPR: 1.4% · Precision: 0.922 · Recall: 0.926
13-18 Tested on 2019 Data	· Detection: 93.5% · FPR: 3.4% · Precision: 0.917 · Recall: 0.944	· Detection: 87.1% · FPR: 11.9% · Precision: 0.968 · Recall: 0.966	· Detection : 90.3% · FPR: 1.4% · Precision: 0.979 · Recall: 0.978	· Detection : 87.1% · FPR: 1.4% · Precision: 0.975 · Recall: 0.975	· Detection: 87.1% · FPR: 3.1% · Precision: 0.963 · Recall: 0.960	· Detection: 83.9% · FPR: 1.7% · Precision: 0.969 · Recall: 0.969

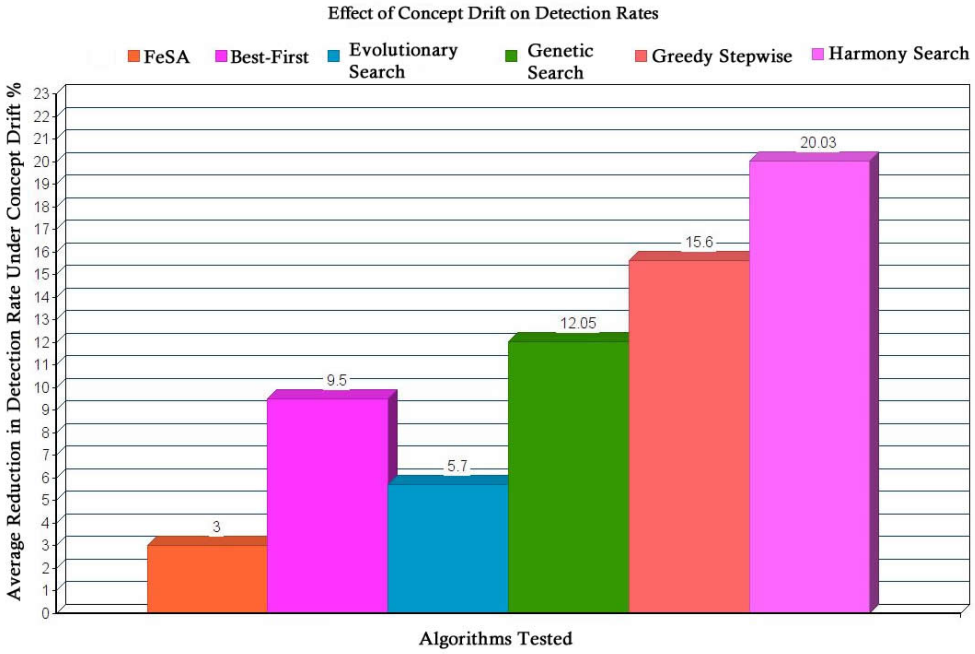

Figure 2: Detection Rates Drop-off

5.2. Discussion

Our results show promise that the FeSA architecture can provide effective and accurate machine-learning detection for evolving ransomware. Regarding situations that do not involve concept drift, FeSA proves to be an effective feature selection algorithm. Table 3 demonstrates that the feature set provided by FeSA yields competitive figures in terms of false positives, recall and, precision. The statistic to be improved is the false-positive rate. FeSA maintains a false positive rate between 5.8% and 5.9%, whereas evolutionary search achieves a false positive rate as low as 4.4% for the 2013-2015 data. FeSA has a more competitive false-positive rate in 2013-2017 and 2013-2018 data sets, but it does not achieve the lowest in either test, as evolutionary search achieves the lowest false-positive rate in each test; however, in all tests demonstrated in Table 3, FeSA yields the highest detection rate maintaining a detection rate above 96% consistently, this is a positive sign as the detection rate is what is viewed as most important for ransomware detection. Once again, the algorithms compared to the evolutionary search achieves a consistent detection rate close to FeSA. Figure 2 and Table 4 show that the performance drop off is reduced compared to other popular feature selection algorithms when encountering concept drift. Table 4 shows that the FeSA architecture produces a feature set that achieves the highest or joint-highest detection rate during the three concept drift scenarios tested. In the concept drift scenarios, FeSA is consistent and maintains a minimal and consistent drop off in detection rate, whereas the algorithms FeSA is compared to behave erratically, showing a much steeper drop-off in detection rate. An example of erratic behaviour would be the harmony-search dataset, which displays an initial detection rate of 93% for 13-17 data, as seen in Table 3. However, when exposed to data from 2018, the detection rate plummets to 54%. Table 4 shows that the FeSA false positive rate remains competitive, yet it is greater than the false-positive rates of genetic search and evolutionary search. We believe the false-positive rate increases due to the limits on feature set sizes and the generation of feature sets produced. Evolutionary and genetic search that FeSA competes with do not have constraints on feature set size and have a higher limit on generations of feature sets it can produce, allowing them to produce feature sets with higher accuracy and reduced false-positive rates. The constraints on FeSA might limit its ability to capture a full picture of benign behaviour; however, this can be improved in the future. Figure 2 shows the drop-off in

the detection rate under concept drift for a random forest trained on the feature sets suggested by different feature selection algorithms. It is observed that the random forest trained on the FeSA features experiences an average reduction in the detection rate of 3%. The x-axis of figure 2 shows which algorithms have been tested with and against, and the y-axis shows the average reduction in the detection rate under concept drift. The high detection rates that FeSA shows in Table 3 and the maintenance of a high detection rate shown in Table 4 are because FeSA enforces fundamental features in distinguishing ransomware from benign software while being combined with the already proven process of natural selection via a genetic algorithm. The detection rate's importance is stressed because of the damaging effects ransomware can have on any system it infects. False positives, in the case of ransomware, are significantly more damaging than false negatives. The experiments use the random forest classifier because this algorithm, in particular, works well with the API-based feature set; it can distinguish the difference between benign files and ransomware effectively and consistently. Our research has explored the use of different algorithms, similar to the approaches used in [5-12], but the random forest proved the most effective to use as an underlying algorithm. Our use of API calls can be expanded to optimise the capabilities of a genetic approach by incorporating the use of static and network features. Regardless of how expansive the feature set is, the main shortcoming of this approach is that it cannot actively react to concept drift and will need further work to incorporate a mechanism that allows the system to react to concept drift. In the system's current state, it is proactive to combat concept drift and shown its effectiveness; however, a mechanism that allows the system to be reactive to concept drift is necessary. Overall, FeSA achieved what it sets out to do; the use of FeSA will provide a machine learning detection system with a robust feature set that will show consistent performance under concept drift. FeSA reduces the need for constant re-training and can increase the time intervals between re-training an intrusion detection machine-learning system.

5.3. Alternative Datasets

We have carried out experiments with our framework on two alternative datasets produced by the researchers in [6] and [42]. This dataset produced by Sgandurra et al. contains ransomware files found between 2012 to 2015 and has a feature count of over 30,000. The feature set produced by Sgandurra et al. contained static strings and directory specific features, which meant some features were exclusive to the test machines used by the researchers. The feature set also contained API calls and drops, which we retained to carry out experiments, as API calls and drops were not exclusive to the machine the ransomware had run on. The experimental results of the imperial dataset are presented in Tables 5 and 6. The experimental results demonstrate that the binary nature of the feature set is not optimal, and the rate of false positives is high for all the feature selection algorithms used in the experiments. FeSA performs the best overall, crucially maintaining performance from year to year instead of the majority of the other feature selection algorithms. The dataset produced by Berrueta et al. contains data from 70 ransomware strains, with features constructed from network data. The ransomware strains are taken from 2015 to 2019, and the dataset is structured to allow testing to be done on zero-day ransomware strains. The experiments on this data are shown in table 7, and we observe that FeSA performs well compared to all of the alternative feature selection algorithms besides the Greedy-Stepwise approach. We observe FeSA achieves strong detection results on zero-day ransomware in this dataset with significantly fewer features and performs consistently well across the three datasets that we have evaluated.

Table 5: Experimental Results with Imperial Dataset

	FeSA	**Best First**	**Evolutionary Search**	**Genetic Search**	**Greedy Stepwise**	**Harmony Search**
Feature Count	**16**	**258**	**160**	**106**	**14**	**36**
Trained and tested on 12-13	· Detection: 99.3% · FPR: 5.1% · Precision: 0.931 · Recall: 0.918	· Detection: 98.0% · FPR: 6.9% · Precision: 0.921 · Recall: 0.910	· Detection: 95.5% · FPR: 9.6% · Precision: 0.920 · Recall: 0.920	· Detection: 92.2% · FPR: 5.6% · Precision: 0.955 · Recall: 0.955	· Detection : 98.0% · False Positive Rate: 6.9% · Precision: 0.921 · Recall: 0.909	· Detection: 81.0% · FPR: 12.7% · Precision: 0.904 · Recall: 0.901
Trained and tested on 2014	· Detection: 97.5% · FPR: 8.4% · Precision: 0.918 · Recall: 0.911	· Detection : 83.9% · FPR: 13.2% · Precision: 0.871 · Recall: 0.870	· Detection: 92.4% · FPR: 6.2% · Precision: 0.939 · Recall: 0.939	· Detection: 89.0% · FPR: 9.5% · Precision: 0.905 · Recall: 0.907	· Detection: 84.7% · FPR: 13.5% · Precision: 0.866 · Recall: 0.866	· Detection: 95.8% · FPR: 11.1% · Precision: 0.893 · Recall: 0.883
Trained and tested on 2015	· Detection: 97.1% · FPR: 12.8% · Precision: 0.912 · Recall: 0.909	· Detection: 94.7% · FPR: 16.1% · Precision: 0.881 · Recall: 0.879	· Detection: 96.2% · FPR: 9.5% · Precision: 0.927 · Recall: 0.926	· Detection: 95.1% · FPR: 13.6% · Precision: 0.898 · Recall: 0.897	· Detection: 94.7% · FPR: 16.1% · Precision: 0.881 · Recall: 0.879	· Detection: 94.3% · FPR: 14.4% · Precision: 0.889 · Recall: 0.888

Table 6: Imperial Dataset with Concept Drift

	FeSA	**Best First**	**Evolutionary Search**	**Genetic Search**	**Greedy Stepwise**	**Harmony Search**
Feature Count	**16**	**258**	**160**	**106**	**14**	**36**
Trained 2013 and tested on 2014	· Detection: 96.6% · FPR: 11.0% · Precision: 0.895 · Recall: 0.883	· Detection: 95.8% · FPR: 8.5% · Precision: 0.151 · Recall: 0.911	· Detection: 87.1% · FPR: 7.0% · Precision: 0.941 · Recall: 0.953	· Detection: 89.8% · FPR: 5.3% · Precision: 0.956 · Recall: 0.951	· Detection : 95.8% · False Positive Rate: 8.5% · Precision: 0.911 · Recall: 0.915	· Detection: 94.1% · FPR: 9.8% · Precision: 0.902 · Recall: 0.899
Trained on 2014 and tested on 2015	· Detection: 91.4% · FPR: 16.0% · Precision: 0.868 · Recall: 0.867	· Detection : 57.4% · FPR: 21.1% · Precision: 0.789 · Recall: 0.709	· Detection: 86.6% · FPR: 9.4% · Precision: 0.899 · Recall: 0.891	· Detection: 81.3% · FPR: 10.9% · Precision: 0.88 · Recall: 0.862	· Detection: 59.3% · FPR: 19.4% · Precision: 0.805 · Recall: 0.726	· Detection: 89.5% · FPR: 15.8% · Precision: 0.861 · Recall: 0.861

Table 7: Navarra University Dataset with Concept Drift

	FeSA	Best First	Evolutionary Search	Genetic Search	Greedy Stepwise	Harmony Search
Feature Count	**32**	**97**	**114**	**109**	**106**	**111**
Tested on Training Distribution	· Detection: 99.9% · FPR: 0.4% · Precision: 0.998 · Recall: 0.998	· Detection: 99.4% · FPR: 0.4% · Precision: 0.998 · Recall: 0.998	· Detection: 99.8% · FPR: 0.4% · Precision: 0.998 · Recall: 0.998	· Detection: 99.9% · FPR: 0.4% · Precision: 0.998 · Recall: 0.999	· Detection : 99.6% · FPR: 0.4% · Precision: 0.996 · Recall: 0.996	· Detection: 99.8% · FPR: 0.4% · Precision: 0.998 · Recall: 0.998
Tested on Zero-Day Distribution	· Detection: 85.1% · FPR: 14.9% · Precision: 0.999 · Recall: 0.999	· Detection: 78.9% · FPR: 21.1% · Precision: 0.989 · Recall: 0.989	· Detection: 78.1% · FPR: 21.9% · Precision: 0.989 · Recall: 0.989	· Detection: 78.8% · FPR: 21.2% · Precision: 0.956 · Recall: 0.951	· Detection : 87.7% · FPR: 12.3% · Precision: 0.998 · Recall: 0.998	· Detection: 78.1% · FPR: 21.9% · Precision: 0.998 · Recall: 0.998

6. Conclusion and Future Work

To conclude, Our research has demonstrated that using a feature selection algorithm can combat the effects of concept drift in a classification system. Our research has also demonstrated that FeSA is an effective feature selection algorithm for ransomware detection under concept drift. Our research uses a wide array of benign and ransomware files to simulate concept drift, showing its existence in the ransomware detection space and how to remediate its effects. Our system is evaluated realistically, and the results produced are promising, with the FeSA system outperforming the genetic search and evolutionary search algorithms. The FeSA system can maintain a high-performance level with fewer offspring feature sets and smaller set sizes. We acknowledge that the system proposed would be a part of a system that would work as a complete ransomware detection system. The feature set generation is a proactive measure for concept drift in a detection system; however, FeSA requires a mechanism to react to concept drift as no preparation can fully prepare for every way ransomware may evolve. Our future work on this system will have to take the results and data from this feature engineering approach and incorporate them into a concept drift adaption system. In the future, the aim is to combine our feature selection algorithm with a mechanism that can classify unknown and drifting samples using the measured concept drift of a sample. A secondary objective is to use non-conformity and similarity measures to aid classifiers when the classifier is uncertain in its predictions. Our goal is to maintain the same detection rates under concept drift as expected under normal conditions with low false-positive rates when building on our system.

7. Acknowledgements

We would like to acknowledge the following contributors:
Funding: This work was supported by Department of Computer Science, School of Mathematics, Computer Science and Engineering, City, University of London.
Reviewers: We would like to thank the anonymous reviewers at Elsevier for providing feedback on the initial submission of this research paper.

References

[1] De Groot. J: A History of Ransomware Attack: The Biggest and Worst Ransomware Attack of All Time, 2017 [2018] [Accessed 22/11/2018] Available at: https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time

[2] Saxena. P, Breed of MBR Infecting Ransomware – an analysis by Quick Heal Security Labs, 2018 [2018] [Accessed 02/10/2020] Available at: https://blogs.quickheal.com/breed-mbr-infecting-ransomware-analysis-quick-heal-security-labs/

[3] Goodchile. J, This Is Not Your Father's Ransomware, 2020 [2020] [Accessed 03/10/2020] Available at: https://www.darkreading.com/edge/theedge/this-is-not-your-fathers-ransomware/b/d-id/1337484

[4] Cook. S, 2018-2020 Ransomware statistics and facts, 2020 [2020] [Accessed 05/10/2020] Available at: https://www.comparitech.com/antivirus/ransomware-statistics/

[5] Clement. J, Ransomware - Statistics & Facts, 2019, [2019] [Accessed 05/10/2020] Available at: https://www.statista.com/topics/4136/ransomware/

[6] Sgandurra.D, Munoz-Gonzalez.L, Mohsen.R, Lupu.E. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection, 2016, CoRR abs/1609.03020

[7] Shaukat. S, Ribeiro. V: RansomWall: A Layered Defence System against Cryptographic Ransomware Attacks using Machine Learning, Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2018,pp. 356–363.

[8] Khan.F, McNube.C, Lakshmana.R, Kadry.S, Nam.Y, A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning, in IEEE Access, 2020,vol. 8,pp. 119710-119719.

[9] Chen.L, Yang.C-Y, Paul.A, Sahita.R, Towards resilient machine learning for ransomware detection, KDD 2019, Aug 04–08, 2019, Alaska

[10] Seong.B, Gyu.L & Eul Gyu.I. Ransomware detection using machine learning algorithms. Concurrency and Computation: Practice and Experience (2019) Version September 28, 2020 submitted to IoT 54 of 54.

[11] Hwang. J, Kim. J, Lee. S, Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques, Wireless Pers Commun 112, 2597–2609 (2020)

[12] Zuhair. H, Selamat. A, Krejcar. O, A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware, IEEE Access, vol. 7, pp. 47053-47067, 2019

[13] Gao.J, Fan.W, Han.J, Yu.P.S. A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions, Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA, 2007.

[14] Jordaney.R, Sharad.K, Dash.S.K, Wang.Z, Papini.D, Cavallaro.L, Transcend: Detecting Concept Drift in Malware Classification Models, Proceedings of the 26th USENIX Security SymposiumAugust 16–18, 2017, Vancouver, 2018

[15] Brownlee.J, A Gentle Introduction to Concept Drift in Machine Learning, 2017 [2017] [Accessed 10/10/2020] Available at: https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning/

[16] Thomas.K, Grier.C, Ma.J, Paxon.V, Song.D, Design and evaluation of a real-time URL spam filtering service. In 32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA (2011), pp. 447–462.

[17] Kantchelian.A, Afroz.A, Huang.S, Islam.A, Miller.B, Tschantz.M, Greenstadt.R, Joseph.A.D, Tygar.J.D Approaches to adversarial drift. In AISec'13, Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, Co-located with CCS 2013, Berlin, Germany, November 4, 2013 (2013), pp. 99–110.

[18] Maggi. F, Robertson. W. K, Krugel. C, and Vigna. G, Protecting a moving target: Addressing web application concept drift. In Recent Advances in Intrusion Detection, 12th International Symposium, RAID 2009, Saint-Malo, France, September 23-25, 2009. Proceedings (2009), pp. 21–40.

[19] Ghomeshi.H, Gaber.M.M., and Kovalchuk.Y, EACD: evolutionary adaptation to concept drifts in data streams, Data Mining and Knowledge Discovery (2019) 33:663-694, 2019.

[20] Sanders.A, 15 (CRAZY) Malware and Virus Statistics, Trends & Facts, 2020, [2020], [Accessed 10/10/20], Available at: https://www.safetydetectives.com/blog/malware-statistics/

[21] Jovanovic.B, Malware statistics – You'd better get your computer vaccinated, 2019, [2019], [Accessed 11/10/20], Available at: https://dataprot.net/statistics/malware-statistics/

[22] Singh.A, Walentstein.A, Lakhotia.A, Tracking Concept Drift in Malware Families, AISec '12: Proceedings of the 5th ACM workshop on Security and artificial intelligenceOctober 2012 Pages 81–92, 2012

[23] Hayes.M, Walenstein.A, and Lakhotia.A. Evaluation of malware phylogeny modelling systems using automated variant generation. Journal in Computer Virology, 5(4):335–343, 2009

[24] Cook.S, Malware statistics and facts for 2020, 2020, [2020], [Accessed 10/10/20], Available at: https://www.comparitech.com/antivirus/malware-statistics-facts/

[25] Sophos Whitepaper, The Rise of Enterprise Ransomware, 2019.

[26] Folino.G, Pizzuti.C, Spezzano.G, GP ensembles for large-scale data classification. IEEE Trans Evol Comput 10(5):604–616, 2006

[27] Folino.G, Pizzuti.C, Spezzano.G, An adaptive distributed ensemble approach to mine concept-drifting data streams. In: 19th IEEE international conference on tools with artificial intelligence, ICTAI 2007, vol 2, pp 183–188. IEEE, 2007

[28] Vivekanandan.P, Nedunchezhian.R, Mining data streams with concept drifts using genetic algorithm. Artif Intell Rev 36(3):163–178, 2011

[29] Fatima.A, Maurya.R, Dutta. M.K, Burget.R, Masek.J , Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning, 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019.

[30] Mallawaarachchi.V, Introduction to Genetic Algorithms, 2017, [2017] [Accessed 20/10/2020], Available at: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code
[31] Hasan. M, Rahman. M: RansHunt: A Support Vector Machines Based Ransomware Analysis Framework with Integrated Feature Set, 2017
[32] Takeuchi. Y, Sakai. K, and Fukumoto. S. 2018. Detecting Ransomware using Support Vector Machines.In ICPP '18 Comp: 47th International Conference on Parallel Processing Companion, August 13–16, 2018, Eugene, OR, USA. ACM, New York, NY, USA, 6 pages.
[33] Sgandurra. D & Munoz-Gonzalez. L & Mohsen. R & Lupu. E. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection, 2016, CoRR abs/1609.03020
[34] VinayKumar. R, Soman. KP, Senthil Velan. K.K, Ganorkan. S, Evaluating Shallow and Deep Networks for Ransomware Detection and Classification, 2017, 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Manipal, Karnataka.
[35] Chen.L, Yang.C-Y, Paul.A, Sahita.R, Towards resilient machine learning for ransomware detection, KDD 2019, Aug 04–08, 2019, Alaska
[36] Zhou.V, A Simple Explanation of Information Gain and Entropy, 2019, [2019] [Accessed 10/11/2020], Available at: https://victorzhou.com/blog/information-gain/
[37] WEKA Genetic Search Algorithm, 2020, [2020] [Accessed 20/10/2020], Available at: https://weka.sourceforge.io/doc.stable/weka/attributeSelection/GeneticSearch.html
[38] Singhal. S, Chawla. U, Shorey. R, Machine Learning & Concept Drift based Approach for Malicious Website Detection, 2020, 2020 12th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India.
[39] Tan.G, Zhang.P, Liu.Q, Liu.X, Zhu.C, Dou.F, Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts, 2018, 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering, New York, NY, USA.
[40] Collins.T, MedusaLocker Ransomware Will Bypass Most Antivirus Software, 2019 [2019] [Accessed 18/06/2021] Available at: https://www.secplicity.org/2020/05/19/medusalocker-ransomware-will-bypass-most-antivirus-software/
[41] Kaspersky, What is a Zero-day Attack? - Definition and Explanation, 2021 [2021] [Accessed 08/10/2021] Available at: https://www.kaspersky.co.uk/resource-center/definitions/zero-day-exploit
[42] Carson. H, Cyberhawk, 2007 [2007] [Accessed 08/11/2021] Available at: http://www.kickstartnews.com/reviews/utilities/cyberhawkzeroday
[43] E. Berrueta, D. Morato, E. Magaña and M. Izal, "Open Repository for the Evaluation of Ransomware Detection Tools," in IEEE Access, vol. 8, pp. 65658-65669, 2020, doi: 10.1109/ACCESS.2020.2984187.