TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Vladislav Zakharenkov, 176478 IAPM

MULTI-OBJECTIVE FEATURE SELECTION FOR ANOMALY-BASED INTRUSION DETECTION BY MEANS OF GENETIC ALGORITHMS

Master's Thesis

Supervisor: Margarita Spitšakova, PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Vladislav Zakharenkov, 176478 IAPM

MULTIKRITERIAALNE TUNNUSTE VALIMINE ANOMAALIA-PÕHISE SISSETUNGI TUVASTAMISEKS GENEETILISTE ALGORITMIDEGA

Magistritöö

Juhendaja: Margarita Spitšakova, PhD

Tallinn 2019

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vladislav Zakharenkov

07.05.2019

Abstract

The main goal of this thesis is to develop a multi-objective Genetic Algorithm-based search method that employs a combined filter-wrapper approach to feature selection and produces well-performing solutions in the domain of anomaly-based intrusion detection.

Anomaly-based intrusion detection is a binary classification task, which splits all observations from computer logs into nominals and anomalies. Classification model performance and complexity depends greatly on the number of features used. Feature selection is a multi-objective optimization task that focuses on finding the minimum number of features that provide maximum performance, as described by domain-specific criteria.

The results of the present thesis combine the robustness of filter approaches with the thoroughness and versatility of Genetic Algorithms in a multi-phase search method, which is benchmarked against traditional techniques on a classical intrusion detection dataset. The proposed algorithm outperforms compared approaches and delivers efficient solutions and exceptional explanatory power.

This thesis is written in English and is 61 pages long, including 7 chapters, 23 figures and 11 tables.

Kokkuvõte

Multikriteriaalne tunnuste valimine anomaalia-põhise sissetungi tuvastamiseks geneetiliste algoritmidega

Antud töö põhieesmärk on välja arendada genetiilisel algoritmil põhinev otsingumeetod, mis kasutab tunnuste valimisel kombineeritud filter-wrapper lähenemist ning toodab edukalt toimivaid lahendusi anomaalia-põhise sissetungi tuvastamiseks.

Anomaalia-põhine sissetungi tuvastamine on binaarne klassifitseerimisülesanne, mis jagab kõik arvuti logide kirjed nominaalideks ja anomaaliateks. Klassifitseerimismudeli jõudlus ja keerukus tugevalt sõltuvad kasutatud tunnuste arvust. Tunnuste valimine on multikriteriaalne optimeerimisülesanne, mis fokusseerub maksimaalse mudeli jõudluseks vajalikku minimaalse tunnuste arvu leidmisele domeenspetsifiliste kriteeriumite järgi.

Antud töö tulemuseks on multifaasiline otsingumeetod, mis kombineerib filterlähenemise robustsust geneetiliste algoritmide põhjalikkuse ja mitmekülgsusega. Otsigumeetodit võrreldatakse tranditsiooniliste lähenemistega klassikalise sissetungi tuvastamise andmestiku põhjal. Väljapakutud algoritm ületab võrreldavate lähenemisviiside jõudluse ning omab erakordset seletusvõimet.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 61 leheküljel, 7 peatükki, 23 joonist, 11 tabelit.

Contents

List of Figures

List of Tables

1. Introduction

There is a great demand for anomaly detection systems in monitoring, analysis and security. Practical solutions are needed where traditional rule-based approaches fail or get too costly to implement. In such cases, machine learning is the go-to approach. Performance of machine learning applications depends greatly on the efficiency of feature engineering and feature selection processes. While the former deals with extracting all sorts of useful information from the data, the latter makes it possible to improve the predictive power of the model, bring down its computational cost and help simplify the system in general. This is achieved by focusing on a select subset of features that are relevant only to the specific task at hand. The automation and improvement of these processes is a subject of active research.

1.1 Anomaly-based intrusion detection

Anomalies (*outliers, deviants, discordants, abnormalities*) are data points which deviate so much from the remaining data that one can suspect them to have been generated by a different mechanism [1]. As such, they carry important information about the underlying generative process and provide useful application-specific insights [2].

Anomaly detection is the task of recognizing observations in any given dataset as either nominals or anomalies. This can be done by scoring each data point according to some metric that quantifies the point's level of abnormality or by performing straightforward binary classification [2]. The former is easily converted into the latter by setting a confidence threshold [3].

Anomaly detection is of particular interest for security systems, where malicious activities can be picked up due to their unusual behavior patterns, instead of having to be checked against pre-defined signatures [4], [5]. Computer logs, being a rich source of information about the system state, provide the best data for learning to successfully detect anomalies in real-time [5], [6]. In this context, nominals correspond to patterns and behaviors that are expected to be present in the data,

e.g. normal system operations, whereas anomalies serve as a catch-all for various unexpected events, such as software and hardware failures, exploits, attacks and malfunctions [7]. In security, malicious activities are dealt with by intrusion detection systems [4], [5]. Anomaly-based intrusion detection systems are often deployed alongside other security tools [4], [5], [8], in concordance with the goals of redundancy and diversity [9].

1.2 Feature selection

Features (*predictors, independent variables, input variables*) are separate measurable characteristics of observed data points [10]. Feature selection is a combinatorial optimization task with two primary objectives: maximizing model performance and minimizing the number of features used [11]. With computer logs, one may need to consider additional optimization criteria. For any given task, processing irrelevant features may induce unnecessary load on the underlying systems – high throughput combined with the cost of aggregate calculations, database queries and interactions with external systems can cause significant slowdowns. Relying on as few features as possible helps not only improve model performance, but potentially reduce coupling between systems.

Since most applications share similarities in architecture and process flow, there are quite a few widely used features (e.g. event counters, state flags, IP addresses, network protocol specifics) that have universal explanations regardless of domain, whereas some features can be challenging to make sense of even for human experts. Domain specifics and feature characteristics may need to be accounted for when performing feature selection. Additional criteria directly affect the nature of obtained solutions.

Depending on the data being labeled fully, partially or not at all, feature selection algorithms can be supervised, semi-supervised or unsupervised, respectively [12]. Unsupervised feature selection is the most challenging problem, since no ground truth is present. Supervised feature selection, while being the easiest possible setting, is still NP-hard [13]. This thesis focuses on the supervised setting. Beyond data concerns, approaches to feature selection can be split into three main categories: filter, wrapper and embedded methods [11], [12], [13]. Their differences and specifics are addressed in section 2.1 of this document.

1.3 Problem statement

Feature selection is inherently a multi-objective task [14] – besides conflicting primary objectives of maximum performance and minimum number of features, many other criteria can be introduced, such as feature explainability, acquisition cost, etc. Multi-objective problems are characterized by a set of optimal non-dominated solutions, none of which can excel each other in one aspect without falling behind in another [15]. Having access to such a set of solutions allows human experts to adequately evaluate trade-off possibilities when making the final decision regarding the most suitable selection of features [16].

However, the overwhelming majority of feature selection methods are only capable of solving single-objective tasks [14], [17]. In most traditional feature selection algorithms, multiple objectives are reduced to a single objective via a weighted sum or other means of scalarization, which can effectively render optimal solutions unreachable through heuristic search [15], [18]. Moreover, single-objective algorithms usually yield a single solution, leaving no room for decision making and greatly limiting further capabilities for analysis. While the applicability and efficiency of existing single-objective algorithms is not under question, the bigger picture suggest that a shift towards multi-objective methods is the preferred way of tackling the problem of feature selection [19].

When the desired output is a set of solutions, it follows naturally that the expected problem solving method should operate on sets of solutions as well. Population-based approaches leverage the advantages of assessing many candidate solutions at once in order to build towards better performance [16]. Evolutionary Computation is a set of population-based global optimization techniques that draw heavily from biology and are notably well-suited for multi-objective problems [15], [16]. Genetic Algorithms in particular offer a paradigm which befits the binary task of feature selection exceptionally well [20]. A number of Evolutionary Algorithms are designed for multi-objective optimization [21], [22], [23], [24] and have been successfully applied to feature selection in the domain of intrusion detection [25], [26], [27].

A recent survey on Evolutionary Computation approaches to feature selection [14] has indicated a clear lack of combined multi-objective approaches; at the same time, pure filter and wrapper methods in evolution continue to produce positive results. This thesis aims to contribute towards research of combined evolution-based approaches within the domain of anomaly-based intrusion detection.

1.4 Objectives

The main goal of this thesis is to develop a multi-objective Genetic Algorithm-based search method that employs a combined filter-wrapper approach to feature selection and produces well-performing solutions in the domain of anomaly-based intrusion detection. This involves data processing and handling, evaluation of separate methods and their combinations, design and implementation of a multi-phase algorithm, experimental parameter tuning, result validation and subsequent analysis using relevant baselines and benchmarks.

1.5 Related work

A variety of multi-objective Evolutionary Computation techniques have been successfully used for feature selection [14], most notable being Particle Swarm Optimization [25], [28], Differential Evolution [22], Genetic Algorithms [21], [29] and more general Evolutionary Algorithms [21], [26], [30]. All of these fall into the population-based paradigm, but differ by the nature of the phenomena that they mimic and the way solutions are represented and operated upon [15], [16], [31]. Given that these algorithms are probabilistic and their results are highly sensitive to search parameters, all of these works differ in the way of handling the process of evolution. Apart from the obvious minimization of feature count, the metrics used for fitness functions which evaluate solutions also differ greatly. These include: accuracy [29], error rate [28], true positive rate, true negative rate [26], false positive rate [25], area under the receiver operating characteristic curve [30], entropy and mutual information based metrics [21], as well as multi-stage combinations [28], [30], weighted sums [25] and method-specific fitness metrics. Approaches to training and validation tend to diverge, and actual selected features are not always reported. This makes direct comparison of evolutionary methods quite difficult. However, the ultimate performance metrics always belong to the confusion matrix and reflect the predictive power of the classifiers trained on the selected subset of features.

The two works that have the most relevance to this thesis deal with feature selection for intrusion detection using KDD'99 and NSL-KDD datasets [32], [33], [34] (classical benchmarks in this field, described in section 5.1 of this document) and employ Particle Swarm Optimization and an Evolutionary Algorithm, respectively [25], [26]. Both report excellent results, with EFSA-CP algorithm claiming accuracy over 90% with as few as 4 features [25] and MOEA-LS over 98% accuracy with an average of 9 features [26].

1.6 Outline

This thesis is organized in the following way. Chapter 2 introduces the core concepts behind the topics of feature selection, Evolutionary and Genetic Algorithms, as well as provides some details of techniques that this thesis relies on. Chapter 3 focuses on the research method, covering the general strategy, main concerns, approach to validation and selection of tools. Chapter 4 provides a detailed explanation of the proposed search algorithm and rationale behind its design. Chapter 5 is dedicated to the experiment, followed by thorough analysis of obtained results. Chapter 6 assesses the performance, general properties and the future of the proposed method, leading to the conclusion of this thesis.

2. Theoretical background

The core techniques that are principal to this thesis are drawn from mature and well established scientific fields with a wide and deep body of knowledge. This chapter provides a brief overview of feature selection methods and the basic theory behind Evolutionary and Genetic Algorithms, focusing on the explanation of methods that are used in this thesis in more detail.

2.1 Feature selection methods

Feature selection methods can be categorized into three groups: filter, wrapper and embedded methods. Combined or hybrid approaches aim to take advantage of both filter and wrapper methods [35], [36]. Usually these are filter-wrapper combinations.

2.1.1 Filters

Filter methods select features with the highest utility based on some quantifiable metric. This allows to reduce the number of features by casting aside the ones that score below a certain threshold. This approach is the most computationally lightweight, since no model training is involved. The various criteria in use are entropy, information gain, mutual information [37], [38], [39], [40], Gini index, Fisher score [41], correlation [42], consistency [43], [44], permutation tests [45] and Relief-based algorithms [46], to name a few.

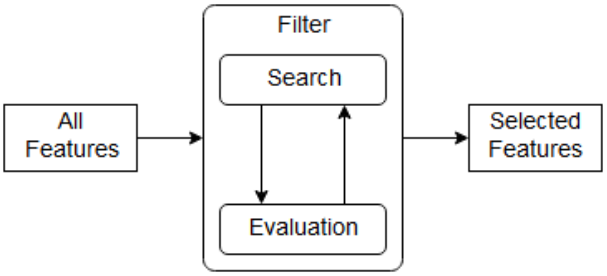

Figure 2.1: Filter feature selection

Since filter methods are not dependent on any learning algorithm, they produce more general feature subsets and are highly versatile. It is important to note that some statistics cannot capture feature interactions unless specifically adapted for multivariate analysis [38], [47].

The three filter methods used in this thesis are mutual information, Fisher score, and Pearson's χ^2 test. These techniques are described in more detail below.

Mutual Information

In information theory, entropy is the self-information of a random variable. Mutual information is a measure of the amount of information one random variable contains about another. It is a special case of a more general relative entropy, which is a measure of the distance between two probability distributions [48].

Fisher score

The Fisher score of a feature is calculated as the ratio of the explained variance between different classes to the unexplained variance within these classes [41]:

$$F_i = \frac{\sum p_j(\mu_i - \mu_{ij})^2}{\sum p_j \cdot \sigma_{ij}^2} \tag{2.1}$$

where:

- p_j is the fraction of observations belonging to the j-th class,
- μ_i is the overall mean of the i-th feature,
- μ_{ij} is the mean of the i-th feature in the j-th class,
- σ_{ij} is the variance of the i-th feature in the j-th class.

Pearson's χ^2 test

Pearson's χ^2 test is a criterion which measures the likelihood of a given set of observations to have been generated by a purely random process [49]. This is achieved by comparing the frequency distributions of the tested variables to a theoretical random distribution. The more the joint frequency distribution differs from the one expected to occur by chance, the more likely it is that the variables in question are statistically dependent.

2.1.2 Embedded

Embedded methods incorporate feature selection into the process of learning, where feature utility is evaluated based on its effect on the objective function of the model [50]. This means that the actual feature selection method itself cannot be separated from model training.

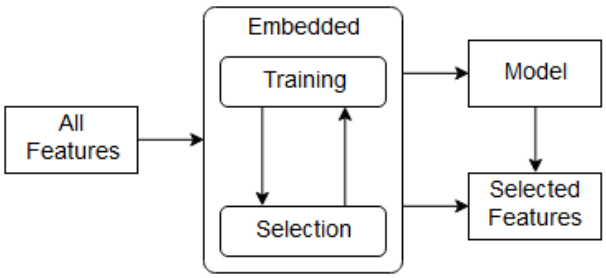

Figure 2.2: Embedded feature selection

Embedded techniques include decision trees, neural networks, lasso methods and others [51], [52], [53], [54]. The computational complexity of embedded methods is entirely dependent on the complexity of the trained models. Since the qualities of selected features are very model-specific, embedded methods are less robust than the filter approach, but offer an advantage by capturing feature interactions.

Decision tree classifier

Decision tree classifiers are prediction models constructed by recursively partitioning a data set and fitting a simple model to each partition [55]. While building the tree, the features that are the best at partitioning the data set are selected. The resulting model can be easily visualized and explained, so the decision tree can be seen as a white box example of an embedded feature selection method. It is computationally cheap and versatile.

2.1.3 Wrappers

Wrapper methods use concrete learning algorithms to evaluate performance on candidate feature subsets, meaning that at each search step a model is trained, evaluated and discarded. This is the most computationally expensive approach, since it always involves training multiple models. However, wrapper methods lead to higher performance scores in practice [35], [56]. Wrapper results are tuned to the specific model type and therefore inherit the properties of the chosen learning algorithm. As a rule, they excel at capturing feature interactions and in general perform a more exhaustive search than other approaches.

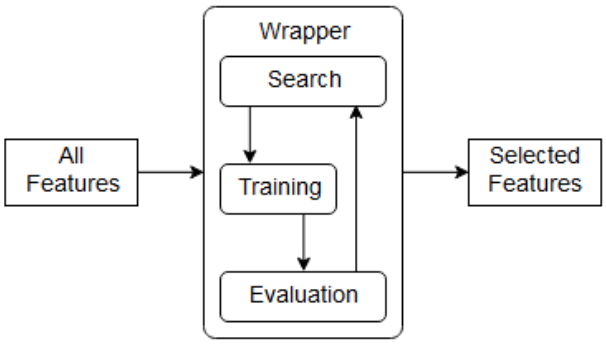

Figure 2.3: Wrapper feature selection

The efficiency of the search algorithm becomes critical in wrapper methods [57], since the number of candidate feature subsets is $2^N - 1$, where N is the number of features. Naturally, classical search algorithms like Best-First Search [58], Branch and Bound [59], [60], [61] and others have been used extensively [35]. Among the most widely used wrapper methods are numerous variations of sequential selection [62] and floating search [63] algorithms with top-down and bottom-up approaches. To select relevant features, some wrapper methods use contrast variables – randomly constructed features with similar properties, but without any relation to the ground truth [64], [65]. Metaheuristics like Simulated Annealing [35], GRASP [66], tabu search [67] have also been successfully applied to feature selection.

Recursive feature elimination

Recursive feature elimination is a greedy top down approach that begins with a full set of features and then proceeds to iteratively drop features using internal explanations of embedded models until the desired number of features remains. This method has the advantage of assessing the entire set of available features and with each step eliminating the one feature that has the least effect on model training errors [53].

Sequential forward floating selection

Sequential forward floating selection is a semi-greedy bottom up approach that starts off with an empty subset and then proceeds to add features with the highest significance, assess their performance, then conditionally drop the least relevant ones until no further improvements can be made. It continues to do so (float back and forth, hence the name of the algorithm) until it arrives to the desired subset size [63].

2.2 Evolutionary Algorithms

When dealing with optimization problems like feature selection, there is no known orderly way to arrive to an optimal solution within a reasonable time limit. Such problems call for the use of metaheuristics [16], [68]. Evolutionary Algorithms are population-based metaheuristics that use the theory of evolution as their guiding principle. In this approach, survival of the fittest becomes the driving force towards better solutions [68]. This makes evolutionary techniques particularly robust and suited for optimization tasks, as they deal well with large search spaces[15], [16], [69].

An Evolutionary Algorithm starts off with the initial population of data structures that represent candidate solutions. Each generation, individual solutions from the population are scored using a fitness function. Better performing solutions have a higher chance of being selected for reproduction and making it through to the next generation, while the remaining solutions with poor performance are more likely to be discarded. The offspring are generated by tweaking operations, which manipulate and alter the parent data structures to create new solutions [16]. The cycle continues for any number of generations until the stopping criteria is met. Although such iterative processes tend to converge on local optima [15], evolution is capable of jumping between local optima thanks to the variations introduced to the offspring solutions by tweaking operators [68].

2.2.1 Genetic Algorithms

Genetic Algorithms are the best-known and most utilized subset of Evolutionary Algorithms. These search methods are based on the mechanics of natural selection and natural genetics [68]. Here, the underlying data structures that represent individual solutions are strings of genes. Classically these are bit strings, but it is possible to use more complex elements if gene expressions are not binary. This representation allows a variety of assisting methods and heuristics to be used in order to ensure the diversity of the population and better solution quality [20], [23].

2.2.2 Genetic operators

In order to simulate the mechanics of natural processes, Genetic Algorithms use the following genetic operators: selection, crossover and mutation. The former serves as a guide to the evolution process, while the latter two facilitate genetic changes in the population.

Selection

The selection operator imitates the reproductive selection process and dictates which of the individuals get chosen for reproduction. There are many ways to achieve this. The simplest selection operator is truncation, where only the top n individuals out of the whole population are able to reproduce. Tournament selection divides the population into groups of n individuals and selects the best out of each group. This can be a recursive process in order to arrive to the desired number of selected individuals. Fitness-proportionate and ranked selection are probabilistic operators that tip the odds in favor of better-performing individuals [70].

Crossover

The crossover operator generates an offspring solution by mimicking the sexual reproduction of two parents. This involves an exchange of spliced genetic material between parent individuals. The level of splicing is usually determined in advance. This can be single-point crossover, where parents contribute large continuous chunks of genes to their offspring, multi-point for smaller chunks, or uniform crossover for a random distribution of individual parent genes. [71].

Mutation

Mutation is a straightforward operator that imitates DNA copying errors found in nature [20]. As a result, a random gene of an individual is modified. This process is highly dependent on gene expression possibilities. In case of bit strings, this usually means simple gene activation or deactivation.

2.2.3 NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a multi-objective algorithm most notable for its selection operator, which is capable of fast and efficient selection of individuals with two important properties: good spread of solutions and better convergence near the true Pareto-optimal front compared to other methods [23]. For the task of feature selection, both of these properties are very important. NSGA-II achieves great results by combining non-dominated sorting of individuals by fronts with crowding tournament selection if a certain front is underrepresented, which leads to the preservation of the spread in the final selection of individuals.

It is important to note that NSGA-II successor, NSGA-III [24], has been shown to perform better as the number of objectives grows higher [72], but at the time of writing no official implementation has been released.

2.2.4 $\mu + \lambda$ Evolution Strategy

$\mu + \lambda$ Evolution Strategy is an Evolutionary Algorithm that operates in the following way. At the beginning of each generation, μ parents generate λ offspring. The population then consists of $\mu + \lambda$ individuals whose fitnesses are assessed and μ survivors with the highest fitness scores are chosen to proceed into the next generation. This strategy is characterized by a fixed bracket for parent and offspring portions of the population and forces the parents to compete with their offspring for as long as it takes to develop better solutions [16]. This is an aggressive approach with associated risks of premature convergence, but it ensures that the resulting solutions are strictly better than the initial population or at least as good in case evolution completely fails to find any solutions that outperform the initial parent population.

3. Methodology

Given the NP-hard nature of the problem tackled in this thesis and the wide variety of techniques and tools that are available, it is important to make use of best practices and avoid pitfalls in order for the method to produce consistent results. This chapter describes the development strategy with which the solution to the problem is approached, including the main concerns, the validation procedure and the selection of tools.

3.1 Strategy

Genetic Algorithms are characterized by sensitivity to search parameters and their performance is greatly affected by the choice of genetic operators, so the taken approach is highly experimental and involves a lot of fine-tuning [15], [16], [20]. The chosen strategy is therefore an iteration between phases of empirical and experimental research: first, results are acquired and validated, upon analysis further improvements are made, experimental setup is adjusted and the process is repeated until the thesis objectives are met in full.

A combined filter-wrapper approach brings about the advantages and shortcomings of both techniques, so these need to be carefully considered and addressed. The robustness offered by filter methods needs to be balanced with model-dependent wrapper results to utilize the best of both worlds. The common combined approach is to use filter and wrapper methods in sequence so that filter results inform the following search performed by the wrapper [56]. Ensemble filter approaches have been shown to achieve good performance [73]. Genetic Algorithms, being stochastic methods, excel as wrapper phases [14].

Multi-objective feature selection puts an emphasis on producing a set of possible solutions that allow human experts to make informed decisions by evaluating trade-offs in various objectives. This means that the arrived-to solutions are not just about absolute improvements in performance, but provide some leeway for potential trade-offs, as well as help gain more information about the nature of the data in question.

Given that the produced set of solutions is non-dominated, it follows that some degradation in certain objectives is to be expected for the benefit of flexibility and informedness. Naturally, an acceptability threshold can be set to any desired level by human experts in charge of assessing the results.

3.2 Concerns

Provided the specifics of the problem and available methods, it is important not only to leverage their advantages, but to avoid the common pitfalls that accompany individual techniques or arise from their combinations. This section describes concrete concerns regarding chosen methods that have to be addressed in order for the algorithm to be viable and applicable in practice.

3.2.1 Computational cost

Wrapper methods are very computationally expensive, so bringing down training and evaluation time is a topmost priority. The use of computational resources directly translates into financial cost of operation, so strict control over computation time is needed. This can be achieved by choosing a cheap and efficient learning algorithm, namely the decision tree classifier, as well as adjusting the complexity of trained models and imposing limitations on the total number of models trained during search. These adjustments also help avoid overfitting the classifiers.

3.2.2 Scalability

Scalability lies at the core of feature selection, especially in wrapper and embedded approaches: increase in the number of features and observations leads to nonlinear growth of model complexity and training time. Parallelization and distributed processing are essential parts of any viable modern approach that involves heavy computation. Evolutionary Algorithms offer many ways of hierarchical chaining and parallelization of search processes, including multiple parallel runs on a distributed network and sharing of solutions between different instances [16].

3.2.3 Handling of mixed and missing data

Data obtained from computer logs comes in all forms: numeric (both continuous and discrete) and categorical (both ordinal and nominal). Data preprocessing and transformation is often a necessary step, but the desired algorithm must attempt to weaken these requirements and aim be as data type-agnostic as possible.

In the experimental setting, one can assume that there is no invalid or missing data. Naturally, this is different from actual live environments encountered in practice, so this needs to be accounted for. The choice of decision tree classifier as the learning algorithm makes it possible to handle all types of data (including missing values) without issues or significant loss of predictive power [55].

3.3 Validation

In machine learning, even the slightest discrepancies in the validation process can lead to wildly inaccurate results. Therefore, strict adherence to best validation practices is called for. In order to perform any kind of analysis and draw conclusions that are based in reality, it is necessary to apply the same validation process to all compared methods.

For all results presented in this thesis, feature selection is performed using the training part of the dataset, then the selected features subsets are evaluated separately using 5-fold cross-validation on the test part of the dataset. The same classifier is used for validation. For wrapper methods, the classifiers used for feature selection in training are matched as well. This is done to eliminate inconsistencies in the experiment that may affect the reported results.

3.3.1 Metrics

The two metrics selected for performance evaluation are detection rate and false alarm rate. The choice of metrics follows the trends in ranking metrics used for evaluation of intrusion detection systems [74]. Together, these provide a clear overview of model performance. Detection rate (DR), also known as true positive rate, is defined as follows:

$$DR = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.1}$$

False alarm rate (FAR), also known as false positive rate, is defined as follows:

$$FAR = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{3.2}$$

Best-regarded performance is a combination of high detection rate with low false alarm rate. High detection rate ensures the desired level of provided security, which is a must-have for intrusion detection systems, while a low false alarm rate minimizes the waste of human expert effort that is associated with investigating reported anomalies. The balance between the two accurately reflects the real-life practical challenges of intrusion detection.

Besides being good indicators of model performance, optimization of these two metrics is a conflicting task. Maximizing detection rate is as easy as classifying all observations as anomalies, but that would maximize false alarm rate as well. Analogously, ignoring anomalies would minimize false alarm rate, but the detection rate would then plummet. The inherent conflict makes these two a good choice of objectives for optimization to test the capabilities of the proposed method.

3.3.2 Baseline

The baseline chosen for solution viability is model performance on all features with 1% loss adjustment for each metric, which is a very conservative threshold. This minuscule loss is accepted to entertain the possibilities of trade-offs, where a slight decrease in one objective can bring about a disproportionate increase in another.

3.3.3 Evaluation of component contribution

Since the developed algorithm uses a combined approach, it is important to evaluate individual phases and components of the algorithm separately and assess their usefulness. This is done to check whether the combination of these methods offers any improvement in performance. It is essential to identify the contribution of each technique to the final performance and ensure that this contribution is quantifiably positive.

3.3.4 Comparison with other methods

Algorithm performance is compared with four filter methods: entropy, mutual information, Fisher score and Pearson's χ^2 test (last three are used in the filter phase of the algorithm) and two wrapper methods: recursive feature elimination and sequential forward floating selection. These wrappers are guided by the preferences of the decision tree, which helps distinguish between the natural behavior of the classifier and the efficiency of evolution-based search that is based on that classifier's performance.

Results are also compared with the ones reported in existing literature. Comparison based solely on performance cannot be considered conclusive, since the means of training (if such is performed) and validation may differ between the compared experiments. Of course, general performance levels are expected to be the same. On the other hand, a comparison of selected feature subsets between different studies will provide a higher degree of objectivity and help verify and validate the obtained results.

3.4 Tools

The programming language of choice is Python due to its popularity for machine learning applications and the author's experience and familiarity with it. There are a number of established and well-tested libraries that use the C language under the hood with minimal overhead. For scientific calculations, *NumPy* [75] and *SciPy* [76] libraries are used. Quick data manipulation is done using *pandas* [77]. Standard machine learning algorithm implementations are available through *scikit-learn* [78] and *MLxtend* [79]. For Evolutionary Algorithms, *DEAP* [80] is chosen for its efficiency, maturity, extensibility, parallelization and multiprocessing capabilities.

4. Proposed method

This thesis proposes a multi-objective Genetic Algorithm for feature selection that uses a combined approach of filter-wrapper implemented in a sequence. This chapter provides a detailed explanation of the proposed search algorithm, documents its structure, capabilities, limitations and the rationale behind the implementation.

4.1 Algorithm overview

The proposed algorithm consists of two phases: filter and wrapper. It takes a full set of features as input and outputs a set of non-dominated feature subsets that fall within a specified size range. The algorithm is designed to benefit from the speed and versatility of bivariate filter methods, while applying reinforcement learning techniques to capture multivariate interactions at a preset computational cost.

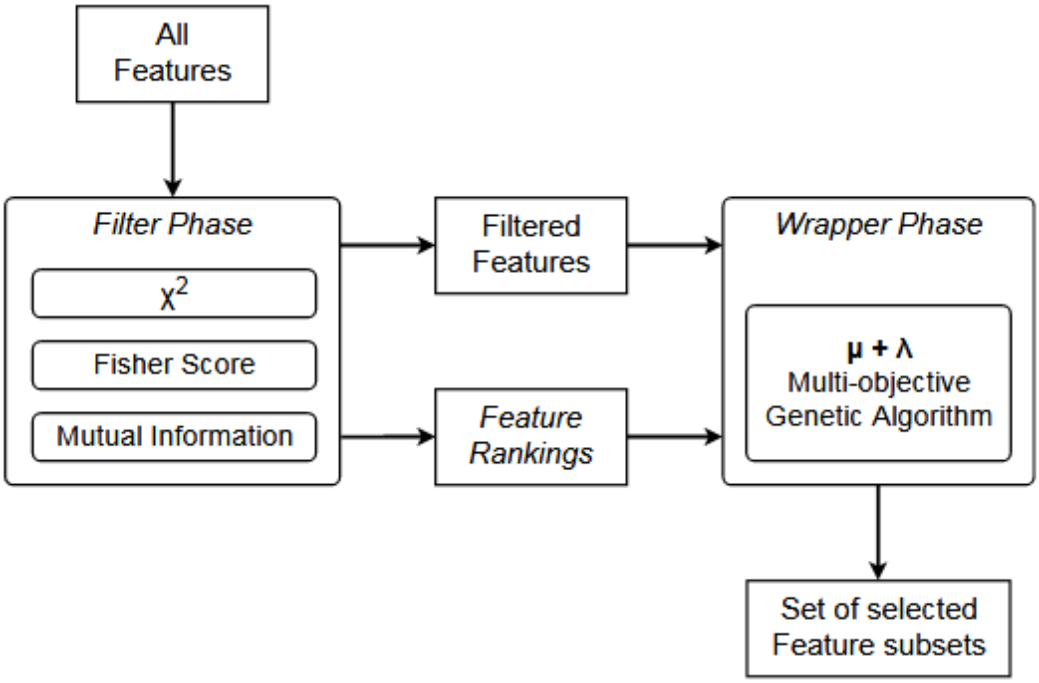

Figure 4.1: Proposed search algorithm

4.2 Filter phase

The inspiration for the filter phase was drawn from filter ensemble rankings presented in the uEFS (univariate ensemble feature selection) method [73]. The core idea behind the taken approach is the following. Since filters base their selection of features on one specific criterion, the top N features chosen by the filter are all bound to have similar properties to some extent. When this is coupled with bivariate filters' inability to capture complex feature interactions, relying on a single filter output when the desired feature properties are unknown can cause potentially important features to be ranked so low as to be excluded altogether. Combining filter outputs and basing feature selection on ensemble ranking helps overcome these issues and end up with a more diverse range of features to work with. However, significant disagreements between ensemble members can lead to much lower scores for otherwise relevant and useful features. The proposed approach deals with this issue by using a ranking table instead of a single harmonized ensemble ranking list.

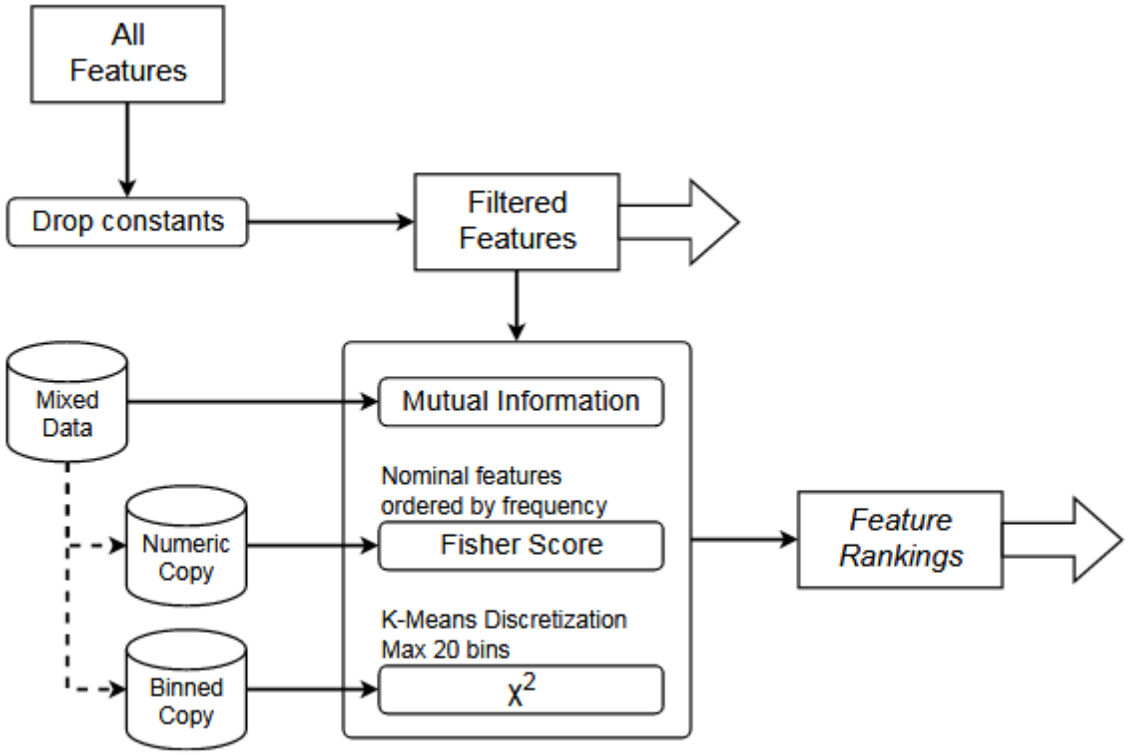

Figure 4.2: Filter phase of the proposed algorithm

During the filter phase, features are filtered and ranked. Since every filter method makes specific assumptions about the type of data, automatic transformation is performed to satisfy the filter requirements. Mutual information can handle both continuous and discrete data, while Fisher score cannot handle categorical variables and the χ^2 test works with discrete data only.

4.2.1 Removing features

The filter phase is careful not to remove any features besides the ones that carry zero information. Thus, only constant features are dropped at this stage. Effective discarding of low-ranking features is delegated to further stages of the algorithm and depends on the supplied search parameters. The mechanism is elaborated upon in section 4.3.1.

4.2.2 Ranking features

The primary purpose of the filter phase is to produce feature rankings that will inform the search performed in the wrapper phase. An example of produced rankings is presented below. The disagreement between different filter methods is apparent.

Pearson's χ^2 test	**Fisher score**	**Mutual information**
service	same_srv_rate	src_bytes
src_bytes	dst_host_same_srv_rate	service
flag	logged_in	flag
dst_bytes	dst_host_serror_rate	diff_srv_rate
diff_srv_rate	serror_rate	dst_host_diff_srv_rate
same_srv_rate	dst_host_srv_serror_rate	same_srv_rate
dst_host_srv_count	srv_serror_rate	count
dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_srv_count
logged_in	diff_srv_rate	dst_host_same_srv_rate
dst_host_diff_srv_rate	dst_host_rerror_rate	dst_bytes
dst_host_serror_rate	urgent	dst_host_serror_rate
count	rerror_rate	serror_rate
serror_rate	protocol_type	dst_host_srv_serror_rate
dst_host_srv_serror_rate	dst_host_srv_rerror_rate	srv_serror_rate
srv_serror_rate	srv_rerror_rate	dst_host_same_src_port_rate
dst_host_count	num_shells	logged_in
dst_host_srv_diff_host_rate	srv_diff_host_rate	dst_host_srv_diff_host_rate
srv_diff_host_rate	dst_host_srv_diff_host_rate	dst_host_count
dst_host_srv_rerror_rate	root_shell	srv_count
dst_host_same_src_port_rate	flag	protocol_type
srv_count	su_attempted	dst_host_rerror_rate
protocol_type	is_guest_login	srv_diff_host_rate
rerror_rate	dst_host_same_src_port_rate	dst_host_srv_rerror_rate
dst_host_rerror_rate	num_access_files	rerror_rate
srv_rerror_rate	wrong_fragment	srv_rerror_rate
duration	land	duration
wrong_fragment	num_file_creations	hot
hot	dst_host_srv_count	wrong_fragment
num_root	count	num_compromised
num_compromised	hot	is_guest_login
num_access_files	dst_host_count	num_root
is_guest_login	service	num_failed_logins
num_file_creations	num_compromised	num_file_creations
su_attempted	num_root	num_access_files
root_shell	num_failed_logins	root_shell
num_failed_logins	srv_count	su_attempted
num_shells	duration	num_shells
land	dst_bytes	land
urgent	src_bytes	urgent

Table 4.1: Filtered feature rankings for NSL-KDD when targeting all attacks

4.3 Wrapper phase

The wrapper phase is a multi-objective $\mu + \lambda$ Genetic Algorithm wrapped around a decision tree classifier. The results of filter phase are effectively embedded into the initial state of the wrapper and this information transfer allows the algorithm to pick up the search where the filters left off. The benefit here is three-fold. First, the robustness of filter-based solutions is carried over to the wrapper phase without the need to perform costly model training. Second, this process has straightforward parametrization and makes the search more flexible and focused. Third, the solutions found in the wrapper phase are guaranteed to be better or at least as good as the ones that are compiled based on feature rankings alone.

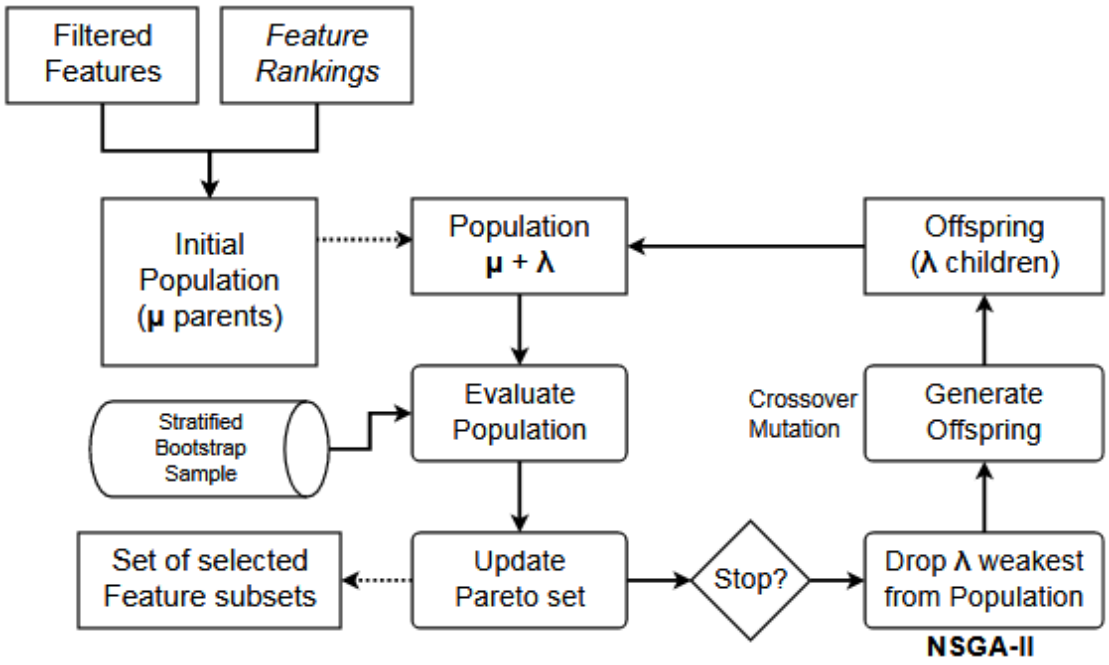

Figure 4.3: Wrapper phase of the proposed algorithm

4.3.1 Initial population

The wrapper phase begins with the initial population being seeded by subsets of top N_{min} to N_{max} features based on each filter's ranking without duplicates. The practice of initial population seeding serves the goal of fast-tracking the evolution process, thus allowing a more thorough exploration of the search space that surrounds the initially seeded solutions [16], [68]. The main associated risk with this technique is premature convergence on local optima if the population is not diverse enough. This can be counteracted by introducing a significant degree of uniform randomness into the genetic material. The mechanism selected for that purpose are the mutation and crossover operators, which are described in section 4.3.3.

4.3.2 Fitness function

Each generation, two comparatively small stratified bootstrapped samples are produced to evaluate the population. The entire population is then evaluated on the same pair of samples in order to facilitate fair competition between individuals. For each individual feature subset in the population, a shallow decision tree classifier is trained on the first sample and tested on the second. The tuple containing the resulting performance (detection rate and false alarm rate) together with the size of the subset are assigned to the individual as its fitness score. If the solution performs well enough to live through multiple generations, it is re-evaluated every time and its fitness is averaged across all runs.

The evolution process does not seem to favor cached cross-validated scores as individual fitnesses, since this causes the classifiers to overfit and perform consistently worse than filter-based solutions. Although caching saves a lot of computational resources, it does not produce positive results. Stratified bootstrapped samples, however, always present a fresh view of the data and can be controlled in terms of size and fraction of anomalies. This makes the underlying learning process potentially infinite, but adjustable in terms of computation time and complexity.

4.3.3 Genetic operators

During each generation, μ parents create λ offspring. The offspring is produced by either mutating one parent individual or performing crossover of two parents. The probability of mutation versus crossover is parameterizable.

Selection

The proposed algorithm utilizes the advantages of NSGA-II selection operator to select the top μ Pareto-optimal individuals for reproduction, turning that fraction of the population into the reproductive elite that directs the search towards more optimal solutions.

Mutation

The proposed algorithm uses a highly active probabilistic mutation operator that randomly gains or loses a feature. The operator is parameterizable, so it can be configured to exhibit bias towards gaining or losing features. Since natural selection favors smaller feature subsets and this can quickly lead to premature convergence, the mutation operator is set up to add random features more often than to remove them.

Crossover

The crossover operator follows tertiary logic based on the similarity of parent individuals as expressed by Jaccard index (for two sets, size of their intersection over the size their of union). If parents' Jaccard index is high, meaning that they have a lot in common, then the offspring inherits all the common genes of the parents. If parents' Jaccard index is low, meaning that the two parents are very different, the offspring inherits their symmetrical difference. The rationale behind this operation is as follows. In a multi-objective landscape, some individuals may survive thanks to their specialization on a single objective. Naturally, different objectives may favor different sets of genes. When the operator encounters drastically different solutions that are both eligible for reproduction, it fuses their differences together in hopes of producing a new individual that is well-adapted to both objectives. For intermediate Jaccard index values, the offspring inherits a random subset of features from both parents. Threshold levels for operator behavior are parameterizable as well.

4.3.4 Search parameters

Genetic Algorithms are known for having many parameters that are used to direct the search procedure. The following table provides a summary of parameters for the proposed search algorithm.

Parameter	**Value range**	**Description**
N_{min}	$[1 \,..\, N]$	Minimum number of features to select
N_{max}	$[1 \,..\, N]$	Maximum number of features to select
μ	$\mathbb{N}^+$	Size of the parent fraction of the population
λ	$\mathbb{N}^+$	Size of the offspring fraction of the population
$P_{crossover\ v\ mutation}$	$[0 \,..\, 1]$	Probability of crossover versus mutation
$P_{mutation\ growth}$	$[0 \,..\, 1]$	Mutation bias towards growth versus loss
J_{high}	$[0 \,..\, 1]$	High threshold for Jaccard index in crossover
J_{low}	$[0 \,..\, 1]$	Low threshold for Jaccard index in crossover
S	$\mathbb{N}^+$	Size of the samples used for fitness evaluation
ν	$(0 \,..\, 1)$	Fraction of anomalies in evaluation samples
M	$\mathbb{N}^+$	Number of fitness evaluations to perform

Table 4.2: Search parameters of the proposed algorithm

Reaching M evaluations serves as the stopping criterion for the evolution process.

5. Experiment and analysis

The proposed method is put through a proper validation process using relevant benchmarks. This chapter documents the conducted experiment and its results. Initial observations are followed by a detailed analysis of the method's performance and the solutions it produced.

5.1 Setup

All calculations were performed using Python version 3.6.8 on consumer-grade hardware equipped with a quad-core processor clocked at 3.60 GHz and 8 GB of memory. Decision trees of depth 3 and 6 are used to train and test, respectively.

5.1.1 Benchmark dataset

For anomaly detection, it is especially important that the data is drawn from real-world generating processes or modeled to reflect them as close as possible [7]. It is also extremely important that no data leakage occurs [81]. For these reasons, the dataset chosen for performance evaluation is NSL-KDD [33], [34], an improved version of the classical KDD'99 dataset [32], a known benchmark for intrusion detection with countless references in existing literature on the subject.

NSL-KDD contains information about network connection traffic described by 41 features (34 numerical, 7 categorical). 39 different attack classes are presented in the dataset, collectively describing four major attack groups: denial of service (DoS, 10 classes), port scanning and probing (Probe, 6 classes), remote to local access (R2L, 16 classes), and user to root privilege escalation (U2R, 7 classes).

The dataset comes pre-split into training and test subsets. These are made up of 125973 and 22544 non-redundant records, respectively. There are 22 attack classes in training and 37 in test, so over one third of attacks are concealed until validation. The feature selection methods evaluated in this thesis use a reduced training subset which consists of 25192 records and contains 21 attack classes. Reported results are cross-validated feature subset performances on the previously unseen test subset.

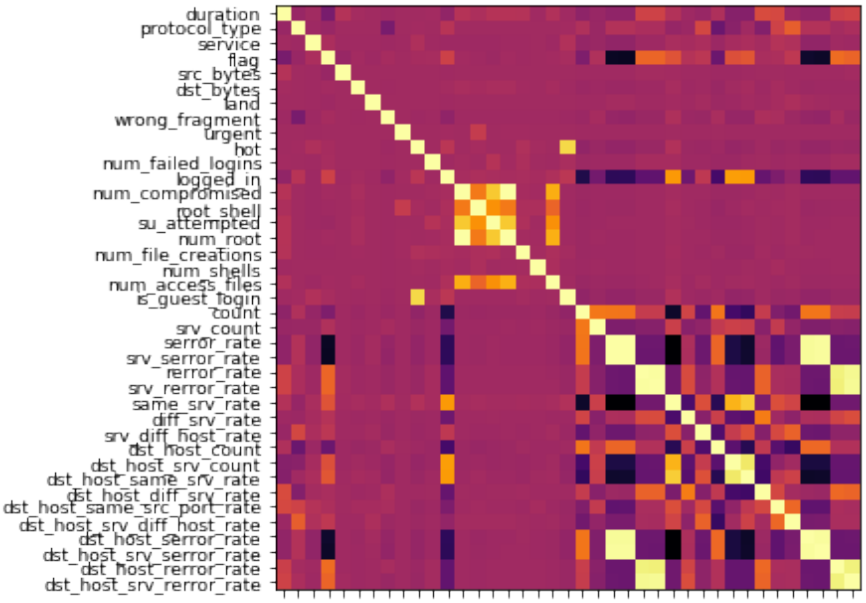

Figure 5.1: NSL-KDD informative feature correlation matrix

5.1.2 Search parameters

The following search parameters were used to obtain the results reported in this thesis. These values were arrived to by a combination of best practices and rough approximations, as well as ad hoc measures and experience-based tuning. A brief summary of rationales for the choices of specific values is provided in the table below.

Parameter	**Value**	**Rationale**
N_{min}	2	Smallest possible subset size to warrant heuristic search
N_{max}	19	Permissive upper bound, roughly halves the search space
μ	Auto	$3 \cdot (N_{max} - N_{min} + 1)$, as described in section 4.3.1
λ	$4 \cdot \mu$	Very modest population size, proportional exploration
$P_{crossover\ v\ mutation}$	0.20	Soften the disruptiveness of crossover, focus on mutation
$P_{mutation\ growth}$	0.75	Bias towards feature gain, as explained in section 4.3.3
J_{high}	0.80	Tuned based on performance
J_{low}	0.20	Tuned based on performance
S	4000	Sample approx. 95% of training data over 10 generations
ν	0.5	Perfectly balanced training set
M	5000	Results in max. 1 minute wall time for wrapper phase (on given parameter and classifier combination)

Table 5.1: Values of search parameters used to obtain reported results

5.2 Results

The presented results were obtained by taking the best of 10 runs on 5 separate target groups: all attacks (56.9% of the dataset), DoS (33.1%), Probe (10.7%), R2L (12.2%) and U2R (0.9%). The best run was decided based on the highest achieved scores, number and nature of solutions, as well as overall quality of the population. No solution aggregation took place, even though it would have improved the achieved results. This was done in order to demonstrate the efficiency of the method when run as a single process, which is the typical setting for benchmarking.

Since the number of obtained solutions is too high to process by hand, they are automatically sorted into four categories of interest:

- Superior (performing better than all features on all objectives)
- Acceptable (above established baseline, 1% loss allowed on each metric)
- High DR (detection rate is higher than the baseline)
- Low FAR (false alarm rate is lower than the baseline)

The last two groups provide important general information about features that perform well on individual objectives. They are useful in analysis of feature roles and characteristics and can help guide further exploration of the data at hand.

In order to better assess the efficiency of the method, it is important to track and distinguish between seeded solutions that were created during the filter phase and evolved solutions at the end of the wrapper phase. Ideally, these sets should not have any solutions in common. A significant overlap between the two can be interpreted as a failure of the proposed algorithm's wrapper phase to improve on the output of filters, especially if initial solutions end up being superior. Overlaps with solutions reported by other studies are favorable, as such feature subsets usually possess properties of local optima based on the heuristics of the compared method.

5.2.1 All attacks

The main validation test is an evaluation of general performance when all attacks are treated as anomalies. Under this setting, class proportions are more or less balanced.

Out of the initial 43 seeded solutions, 142 non-dominated ones evolved, with 2 survivors remaining from the initial population.

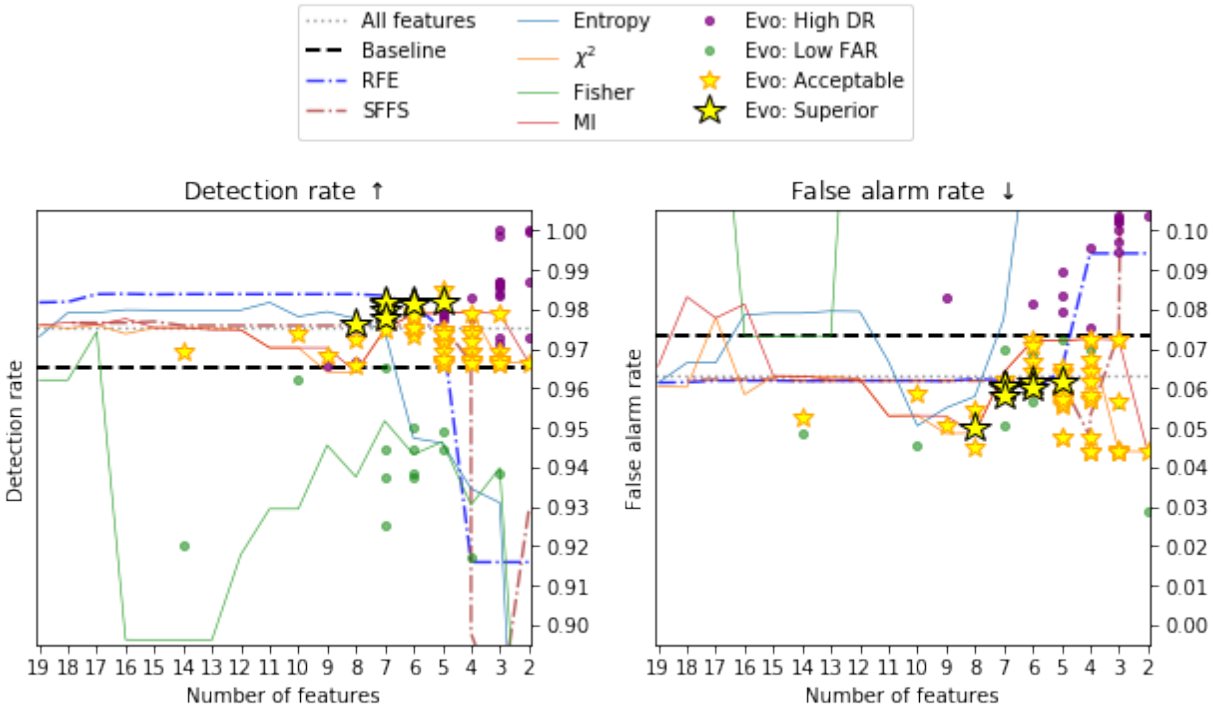

Figure 5.2: Performance comparison on all attacks

Two small solutions from the filter phase have survived the evolution process and landed in the acceptable range. The proposed method succeeded in producing a large number of superior and acceptable solutions of various sizes. The best ones are between 5 and 8 features.

	Initial	**Evolved**
Superior	0	8
Acceptable	2	54

Table 5.2: Grouping of obtained solutions on all attacks

DR	**FAR**	**Number of features**	**Selection method**
97.5%	6.3%	41	All features
98.2%	**6.2%**	5	Proposed method
97.6%	**5.0%**	8	Proposed method
96.6%	**4.4%**	2	χ^2, MI

Table 5.3: Key performance scores on all attacks

When targeting all attacks and taking both objectives into consideration, the evolutionary approach performs better than all other presented methods. Mutual information and χ^2, being core components of the proposed method, come very close. They obtain similar scores on separate metrics and produce a 2-feature solution that boasts the lowest false alarm rate. Recursive feature elimination can produce results that perform better than the proposed method in terms of detection rate for large feature subsets, but not as well as evolution when false alarms are concerned. The proposed method has a clear advantage over compared wrappers when looking at the smallest possible subsets, with the mean size of acceptable solutions being 4 features, the exact subset size at which performances of other wrappers begin to deteriorate sharply.

5.2.2 DoS

This target group focuses only on denial of service attacks, which constitute one third of the dataset, with the remaining observations treated as nominals. Out of 53 initially seeded solutions, 120 non-dominated ones evolved, with 1 survivor. No superior solutions have emerged, but a single acceptable subset of 4 features was found during the wrapper phase.

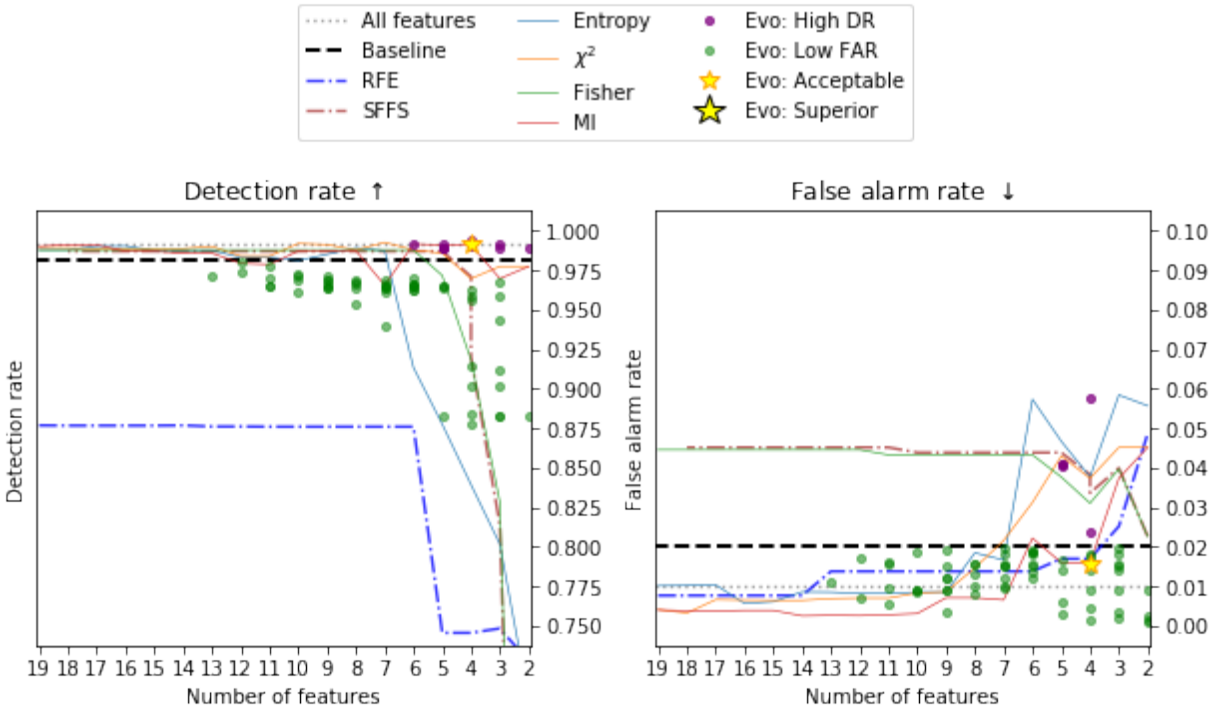

Figure 5.3: Performance comparison on DoS attacks

The performance on all features is already excellent, so it is extremely difficult to improve upon, but still possible to reduce the number of features used.

As far as trade-offs go, a 9-feature solution with a considerably high detection rate and a three-fold decrease in false alarms was produced.

DR	FAR	Number of features	Selection method
99.1%	1.0%	41	All features
99.2%	1.5%	4	Proposed method
97.1%	**0.3%**	9	Proposed method

Table 5.4: Key performance scores on DoS attacks

Under these circumstances, the proposed method performs only marginally better than filters on smaller subsets, but is considerably more successful at both objectives when compared to standard wrapper approaches.

5.2.3 Probe

This target group is aimed at probing and surveillance attacks, which constitute roughly one tenth of the dataset. Out of the initial population of 51, 108 non-dominated solutions evolved, with 1 survivor. 5 acceptable solutions were found, all with a relatively high feature count: 12, 13, 14. As with the DoS attack group, no superior solutions emerged.

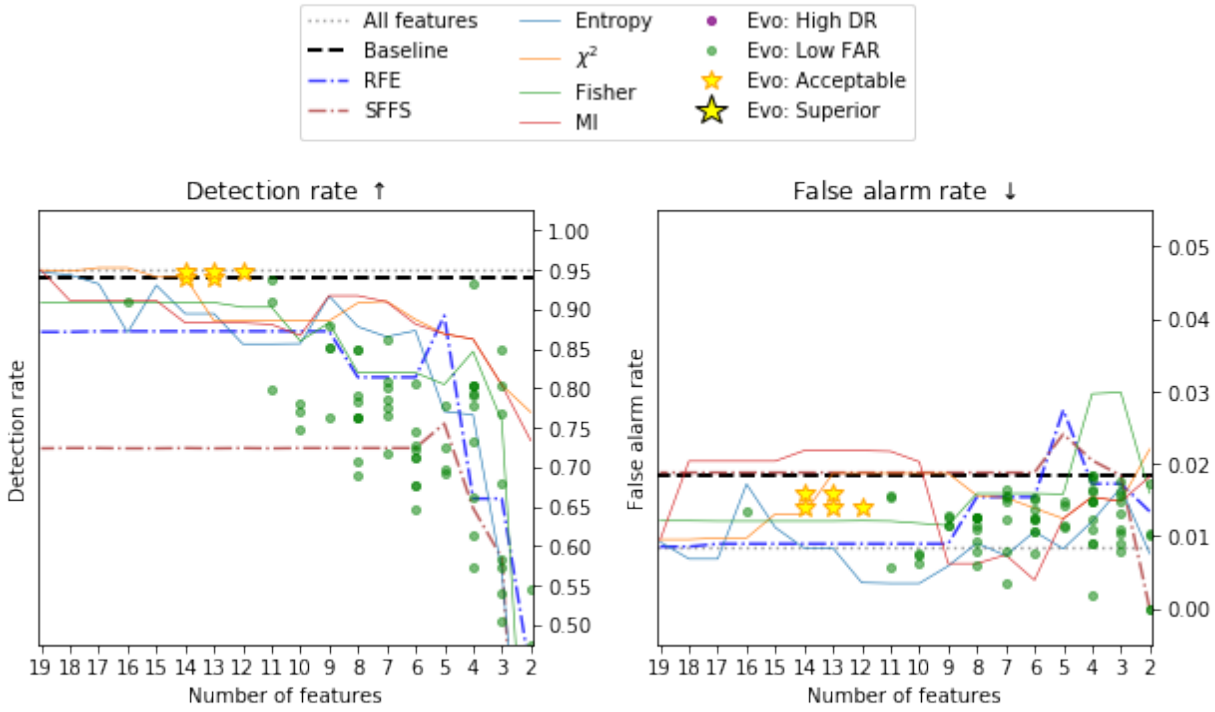

Figure 5.4: Performance comparison on Probe attacks

The decline in performance of all compared methods as the number of features grows smaller suggests objective difficulty of this target group. However, the proposed algorithm has produced a 4-feature solution that shows a clear advantage over the achievements of all other techniques on that subset size, including the filters that are used in the first stage of the proposed algorithm.

DR	FAR	Number of features	Selection method
94.9%	**0.9%**	41	All features
94.6%	1.4%	12	Proposed method
93.2%	1.2%	4	Proposed method

Table 5.5: Key performance scores on Probe attacks

5.2.4 R2L

This target group isolates remote to local access attacks, roughly one eighth of the dataset. Out of the initial 54 seeded solutions, 105 evolved, with 1 survivor. 5 are superior, showing a clear improvement in detection rates and marginally lower false alarm rates. 10 fall into the acceptable range with improved detection but more proneness to cause false alarms.

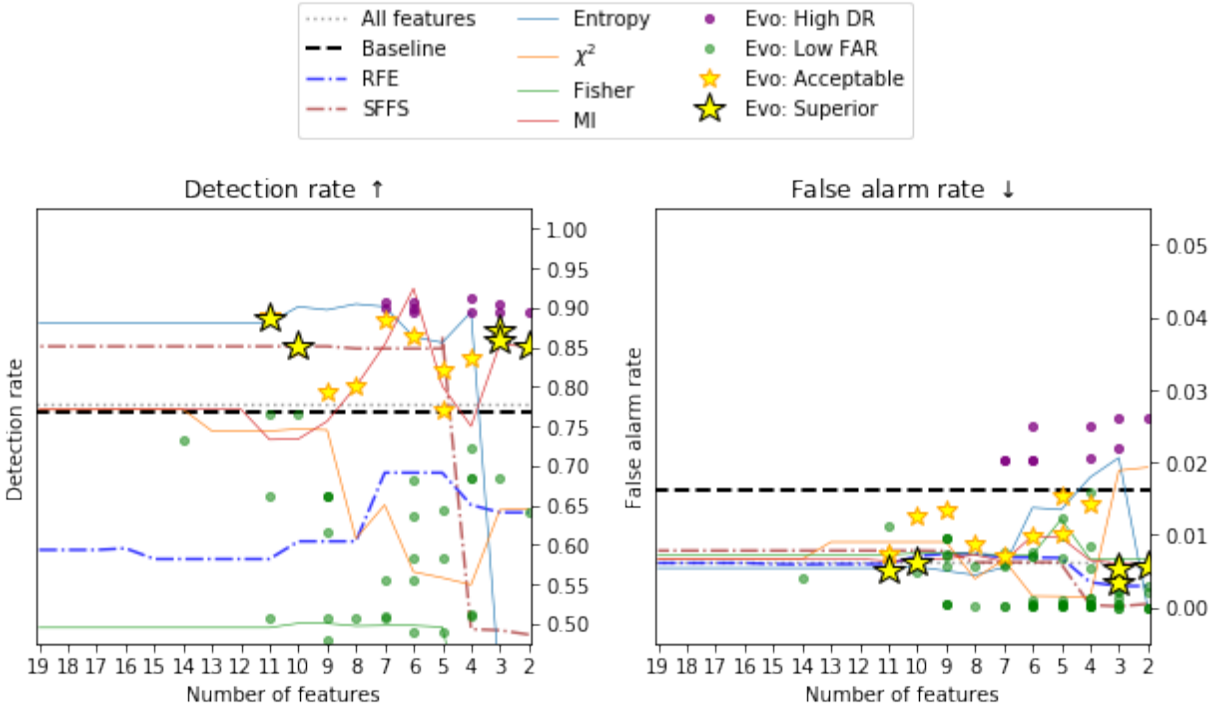

Figure 5.5: Performance comparison on R2L attacks

On this target group, mutual information filter performs very well, yielding a superior 2-feature solution. The proposed method builds on that foundation by finding 4 more superior feature subsets and a number of acceptable candidates.

	Initial	**Evolved**
Superior	1	4
Acceptable	0	10

Table 5.6: Grouping of obtained solutions on R2L attacks

It is apparent that filters contribute a great deal to the success of the proposed approach. Even though the Fisher score performs notably poorly on this particular target group and it can be seen to affect the overall results, significant improvements in performance are achieved during the wrapper phase. The proposed method still yields higher detection rates on the smallest feature subsets, which contrasts with pure wrapper approaches - recursive feature elimination is holding a steady score at first, but fails as the number of features falls below 5. The entropy-based filter displays consistently good scores all the way down to 7 features and in this case rivals the proposed algorithm in terms of performance.

DR	**FAR**	**Number of features**	**Selection method**
77.7%	0.6%	41	All features
88.8%	**0.5%**	11	Proposed method
87.3%	**0.3%**	3	Proposed method
85.0%	0.6%	2	MI
91.2%	2.1%	4	Proposed method

Table 5.7: Key performance scores on R2L attacks

5.2.5 U2R

This target group deals with the least represented type of attacks - user to root privilege escalations. These make up a mere 0.9% of the dataset. This setting is particularly difficult due to the heavy class imbalance.

Out of the initial 54 solutions inherited from the filter phase, 46 new ones evolved with no survivors. The proposed method produced 5 superior solutions.

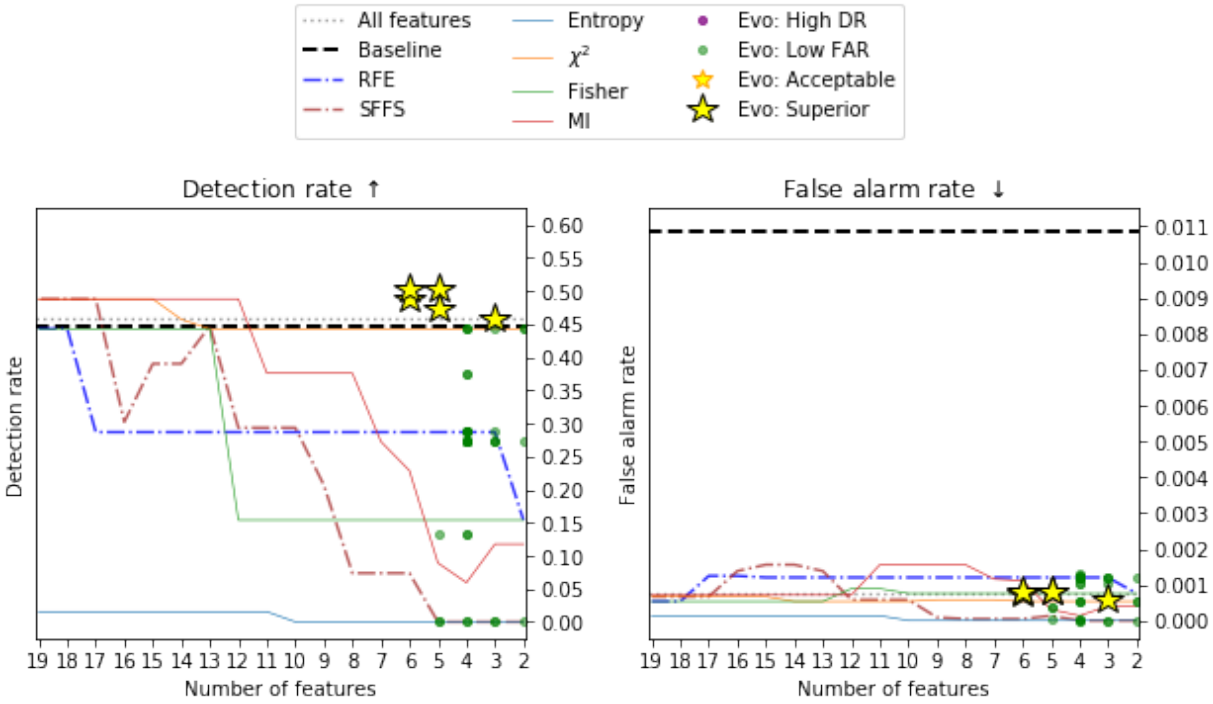

Figure 5.6: Performance comparison on U2R attacks

The overall performance on this target group is very poor. Since the false alarm rate is virtually null for all methods, detection rate serves as the deciding metric. All compared approaches, both filter and wrapper, fail rapidly in that respect as the number of features goes down. The exception is the χ^2 filter, which scores consistently near the baseline. Informed by this filter's initial solutions, the proposed algorithm finds feature subsets that display even better performance in the given setting, boosting detection rates without causing more false alarms.

DR	**FAR**	**Number of features**	**Selection method**
45.8%	0.08%	41	All features
50.3%	0.08%	5	Proposed method
45.8%	**0.05%**	3	Proposed method

Table 5.8: Key performance scores on U2R attacks

In general, performance results for all groups are in line with those found in relevant literature [33], [34], [82], [83], [84] with the poor performance on U2R explained by the nature of the chosen classifier, since similar setups report a comparable performance drop as well. The results can be therefore considered valid.

5.3 Selected features

One of the advantages of the proposed approach is the output of multiple good solutions at once. This greatly helps with analysis and subsequent settlement on the final feature subset, since much more informed decisions can be made.

Target group	**Selected features**	**Selection method**
All attacks	count, dst_bytes, dst_host_srv_count, service, src_bytes (5)	Proposed method
	count, diff_srv_rate, dst_bytes, dst_host_rerror_rate, dst_host_srv_count, hot, service, src_bytes (8)	Proposed method
	service, src_bytes (2)	χ^2, MI
DoS	diff_srv_rate, rerror_rate, service, src_bytes (4)	Proposed method
	count, diff_srv_rate, dst_host_diff_srv_rate, dst_host_same_srv_rate, dst_host_serror_rate, dst_host_srv_count, same_srv_rate, src_bytes, wrong_fragment (9)	Proposed method
Probe	count, diff_srv_rate, dst_bytes, dst_host_rerror_rate, dst_host_same_srv_rate, dst_host_serror_rate, dst_host_srv_count, dst_host_srv_serror_rate, flag, serror_rate, service, srv_diff_host_rate (12)	Proposed method
	diff_srv_rate, dst_bytes, service, src_bytes (4)	Proposed method
R2L	count, diff_srv_rate, dst_host_same_srv_rate, dst_host_srv_count, logged_in, num_access_files, root_shell, same_srv_rate, service, src_bytes, su_attempted (11)	Proposed method
	count, service, src_bytes (3)	Proposed method
	service, src_bytes (2)	MI
	count, hot, protocol_type, service (4)	Proposed method
U2R	dst_bytes, num_file_creations, serror_rate, service, src_bytes (5)	Proposed method
	num_compromised, num_file_creations, src_bytes (3)	Proposed method

Table 5.9: Selected feature subsets for key performance scores

It is apparent that the most important features are *src_bytes*, *dst_bytes*, *service*, *count*, *diff_srv_rate*, and *dst_host_srv_count*. Note the presence of group-indicative features like *dst_host_rerror_rate* (Probe) and *hot* (R2L) in the 8-feature subset that was selected for all attacks by the proposed method. For U2R, *num_file_creations* is an important feature. On the whole, selected features are in agreement with the results reported in existing literature [25], [34], [82], [83], [84], [85], [86].

5.4 Analysis of selected features

Since the selected feature subsets are quite small, visual analysis can be performed. The most frequently occurring pair of features is (*src_bytes*, *service*).

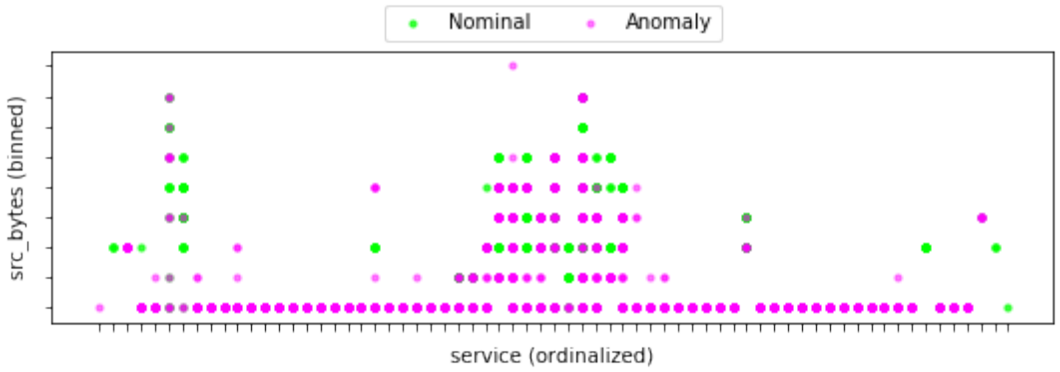

Figure 5.7: Two most frequently selected features

There is no linear separability between nominals and anomalies, but a large isolated cluster is present. *dst_bytes* allows to separate some of the border regions.

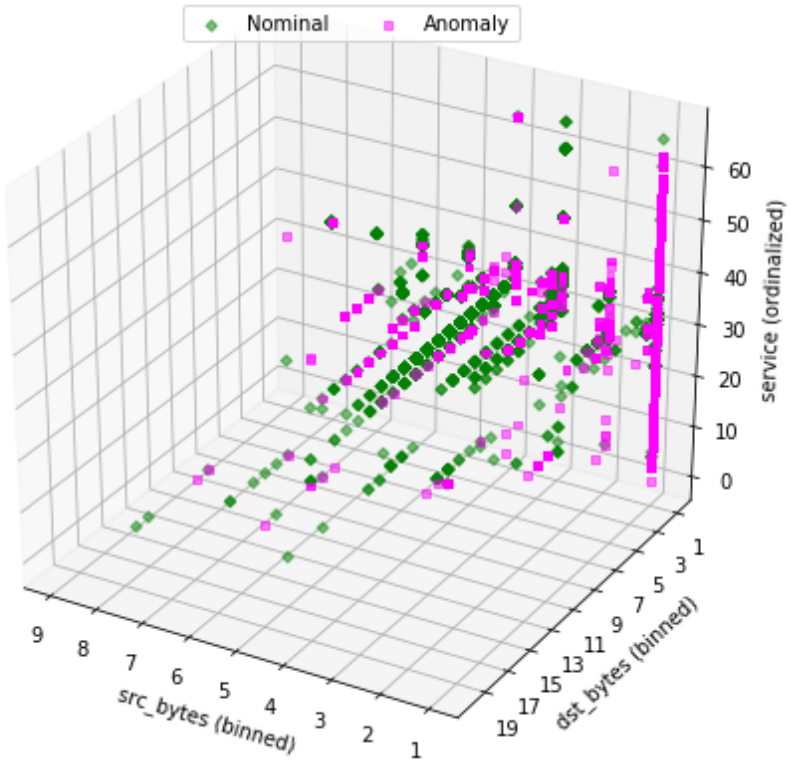

Figure 5.8: Three fundamental network traffic features selected

Further segregation can be achieved by projecting onto *dst_host_rerror_rate*.

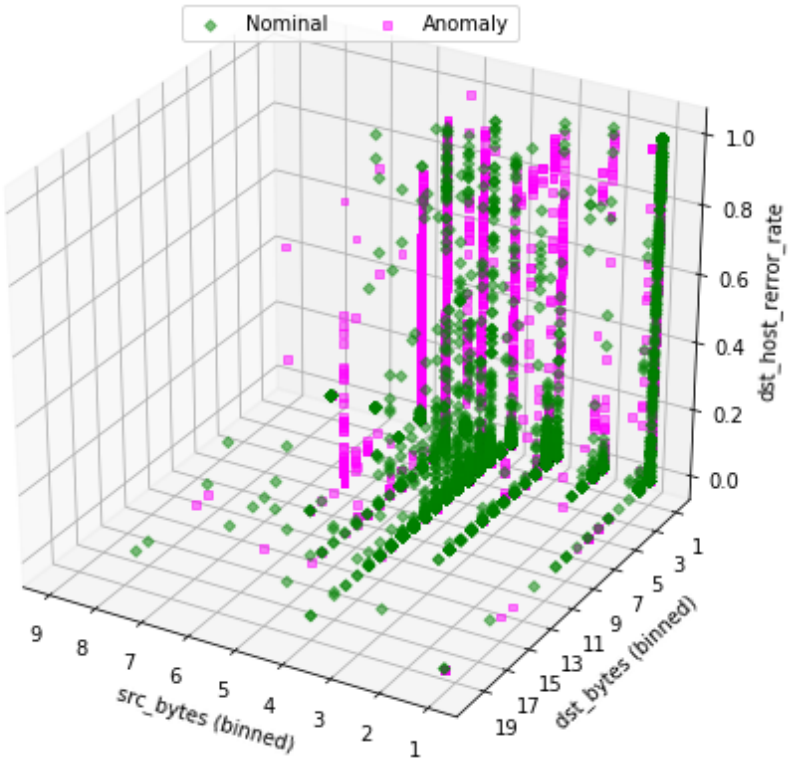

Figure 5.9: Increased separation in problem area using selected error rate feature

Two selected features, *count* and *dst_host_srv_count*, form a large cluster, but there is a large amount of mixing around the border areas.

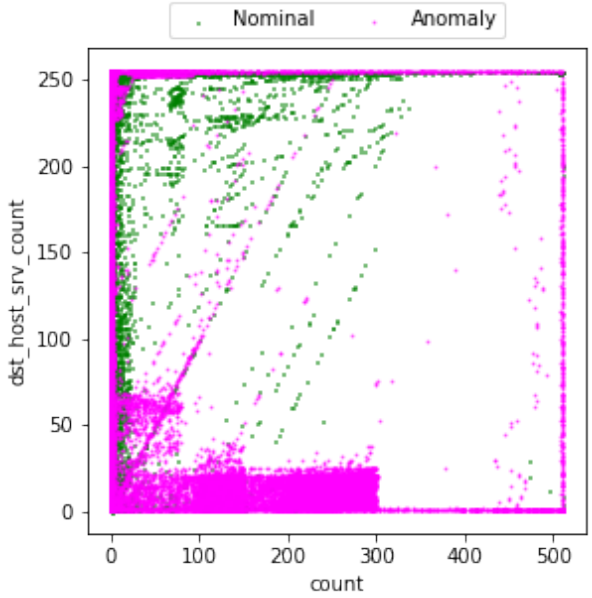

Figure 5.10: Clustering of selected counter features

dst_host_rerror_rate is able to achieve separation in the problematic border areas.

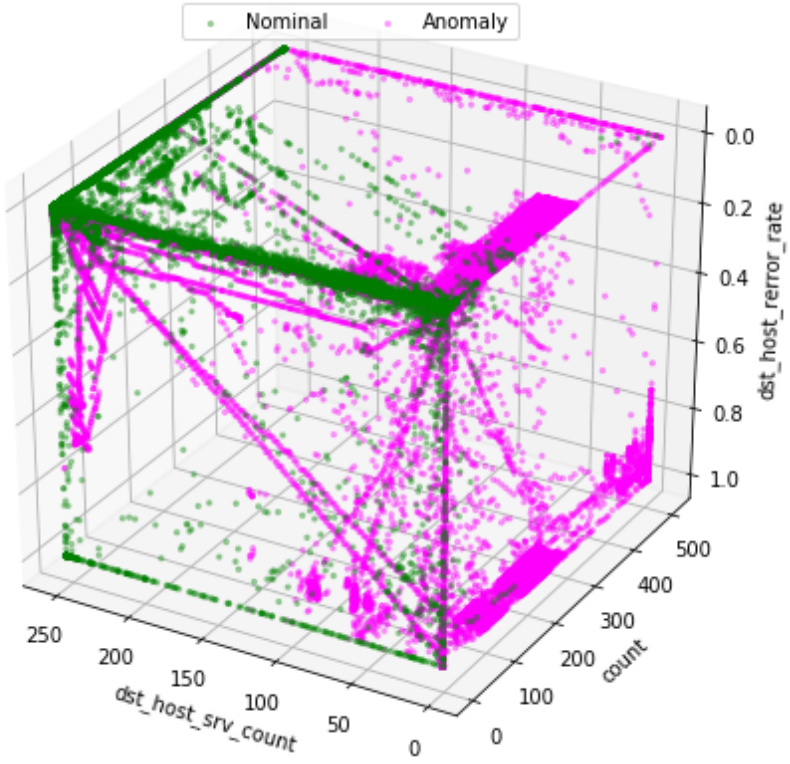

Figure 5.11: Increased separation in counter feature clustering

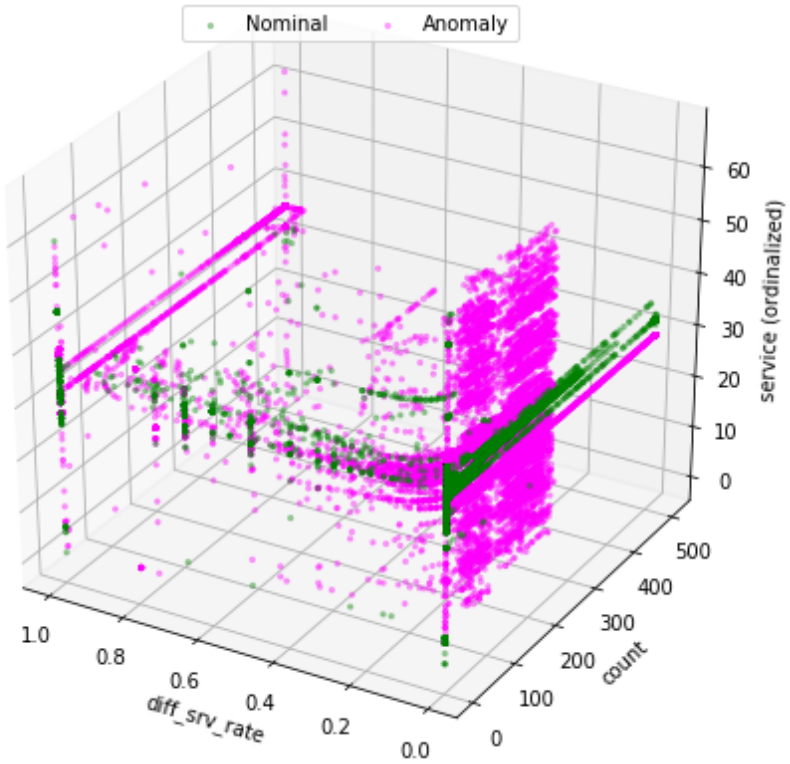

Figure 5.12: General tendency towards clustering using selected features

After visual analysis, it is clear that the proposed method selected useful features.

5.5 Convergence on selected features

As previously mentioned, it is important to assess the overlap between the feature subsets produced during the filter phase with the evolved solutions. This helps measure the level of inter-phase contribution, identify the source and importance of selected features and get a deeper understanding of the performed search process and the associated challenges. For these purposes, genetic drift, i.e. relative feature frequencies between the initial and evolved feature subsets, are analyzed.

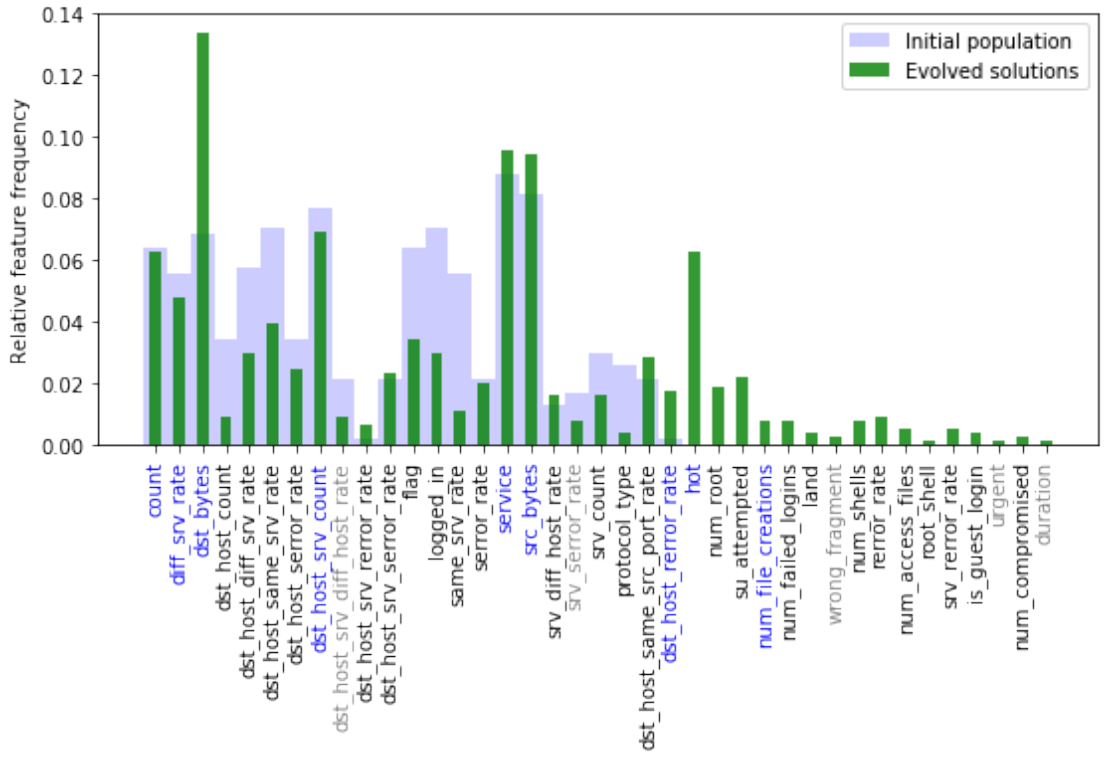

Figure 5.13: Genetic drift in the proposed method on all attacks

On all attacks, out of the 9 key features, 6 had a significant presence in the initial population and 1 was barely represented. The evolution has considerably amplified *dst_bytes* and *dst_host_rerror_rate* and generated a strong presence for previously nonexistent group-specific features *hot* and *num_file_creations*. It has also considerably downplayed correlated features that occurred in the initial population, thus resolving the issue that accompanies the chosen filter methods. All non-zero features are represented in the evolved solution set to some extent, so the proposed method's output is rich with information. This helps to better understand the problem landscape and perform more rigorous analysis using outputs from multiple runs.

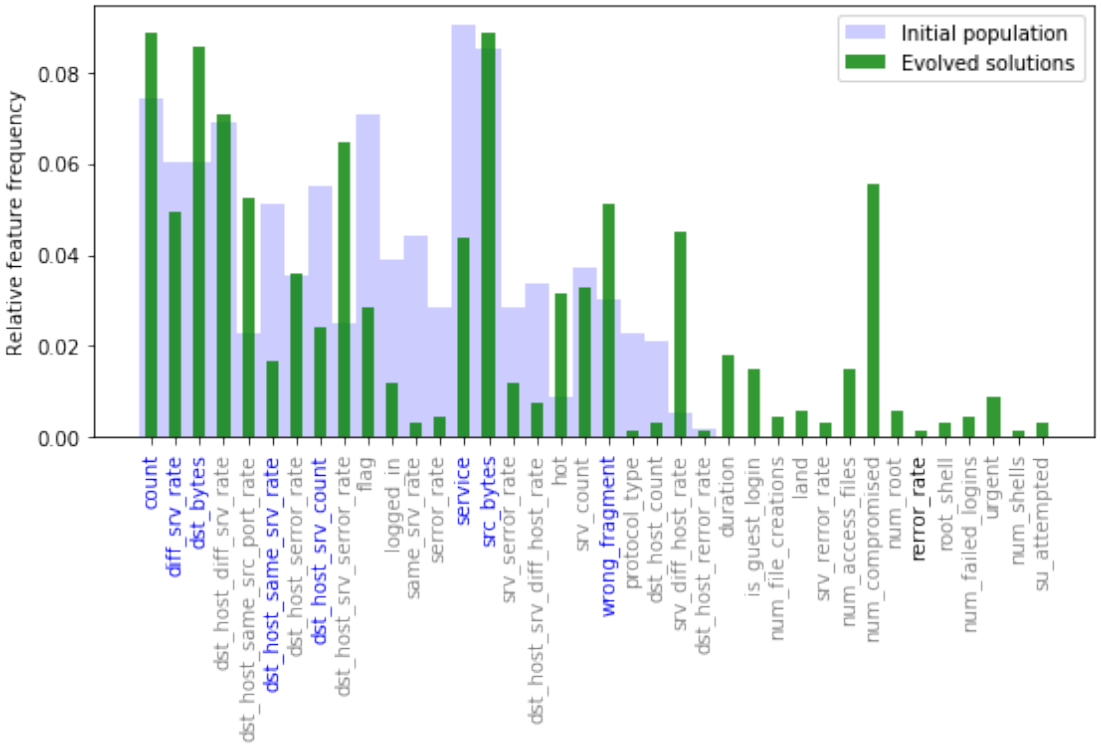

Figure 5.14: Genetic drift in the proposed method on DoS attacks

For the DoS attack group, all of the important features are already present in the initial population. The wrapper dampens a number of irrelevant features and boosts signature denial of service features, such as *count* and *wrong_fragment*.

The evolution on DoS attacks also leads to some questionable features being encouraged, most likely picked up in an attempt to improve on already very good performance. A closer look at the results reveals that a number of these features are stepping stones towards a solution with a marginally better detection rate. However, a highly active feature *num_compromised* ended up not contributing to the accepted solution, yet it has a very strong presence in the gene pool.

The explanation for this becomes apparent when the evolved population is grouped by features and compared in detail. Early on in the evolutionary process, this feature has been randomly attached to a well-performing solution via mutation, adding a slight edge to its performance. Then, via uniform crossover, it was combined with very successful features into a new unique breed of solution, managing to secure a place among the reproductive elite for long enough to increase its concentration in the gene pool. It was later rendered obsolete by better solutions, but at that point, speciation has already occurred. These kinds of evolutionary mechanisms may lead to improvement of outcomes, and in this case *num_compromised* has played its role.

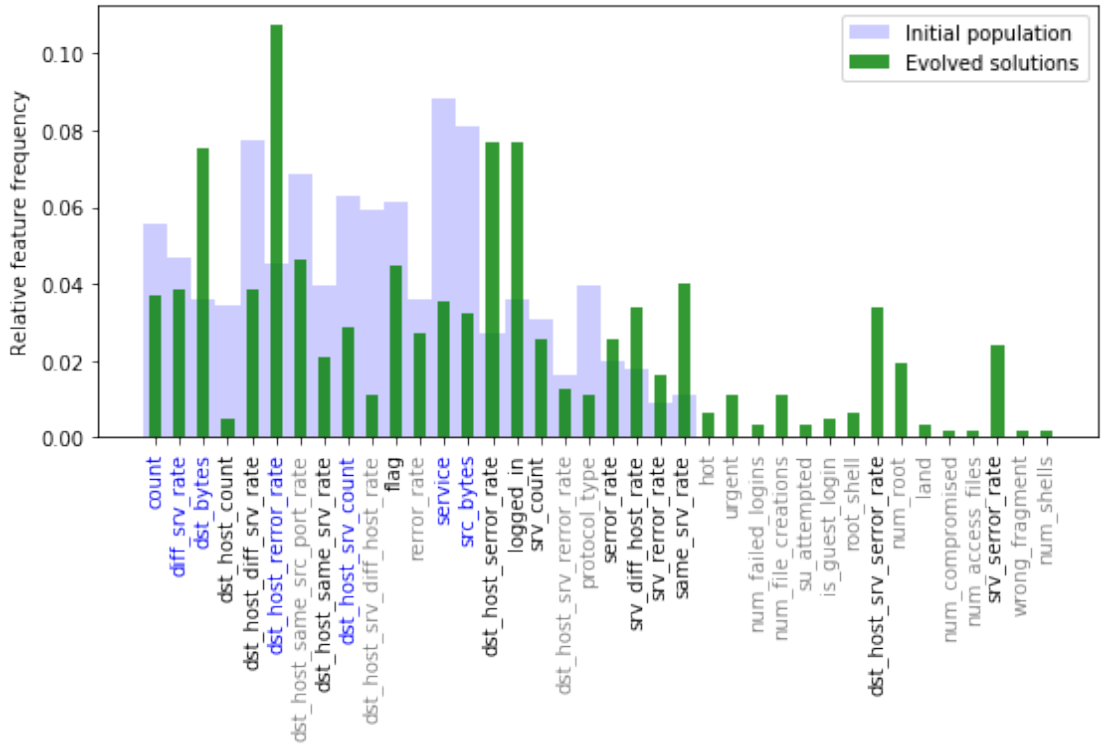

Figure 5.15: Genetic drift in the proposed method on Probe attacks

The genetic drift on Probe attacks reflects significant changes in the population, with *dst_bytes*, *dst_host_rerror_rate* and some others being boosted by evolution, and the frequency of relevant *src_bytes*, *service*, *dst_host_srv_count* declining twofold, suggesting reorganization of importances and active search underway.

The same is true for R2L attacks, but to a lesser degree – the frequencies of most important features are more or less in balance. Semantically related features such as *root_shell*, *num_shells*, *num_access_files* and *su_attempted* naturally emerge during the wrapper phase, contributing to the superior solution. *See next page for graph.*

For U2R attacks, 8 features become extinct, with 5 more on the way, indicating that the evolution took a different direction from what was initially seeded. A disproportionate increase in *root_shell* and *num_compromised* can be observed. An examination of the evolved solutions indicates that these two features occur in the majority of the population, predominantly in tandem. Since the pair contributes to a superior 5-feature solution, but is not needed in another 5-feature solution that performs even better, it can be concluded that an early domination of the gene pool has occurred, similarly to the case with DoS attacks. Here, this was caused by the lack of diversity in samples presented to the classifier, which also explains the low overall performance scores. *See next page for graph.*

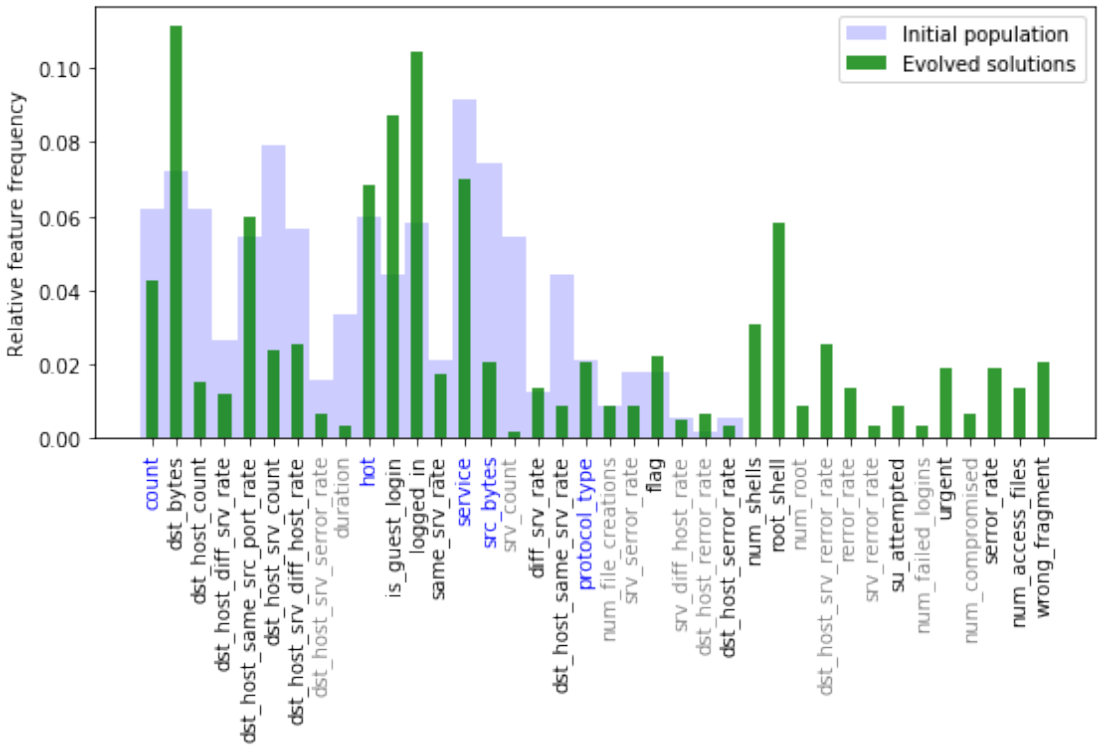

Figure 5.16: Genetic drift in the proposed method on R2L attacks

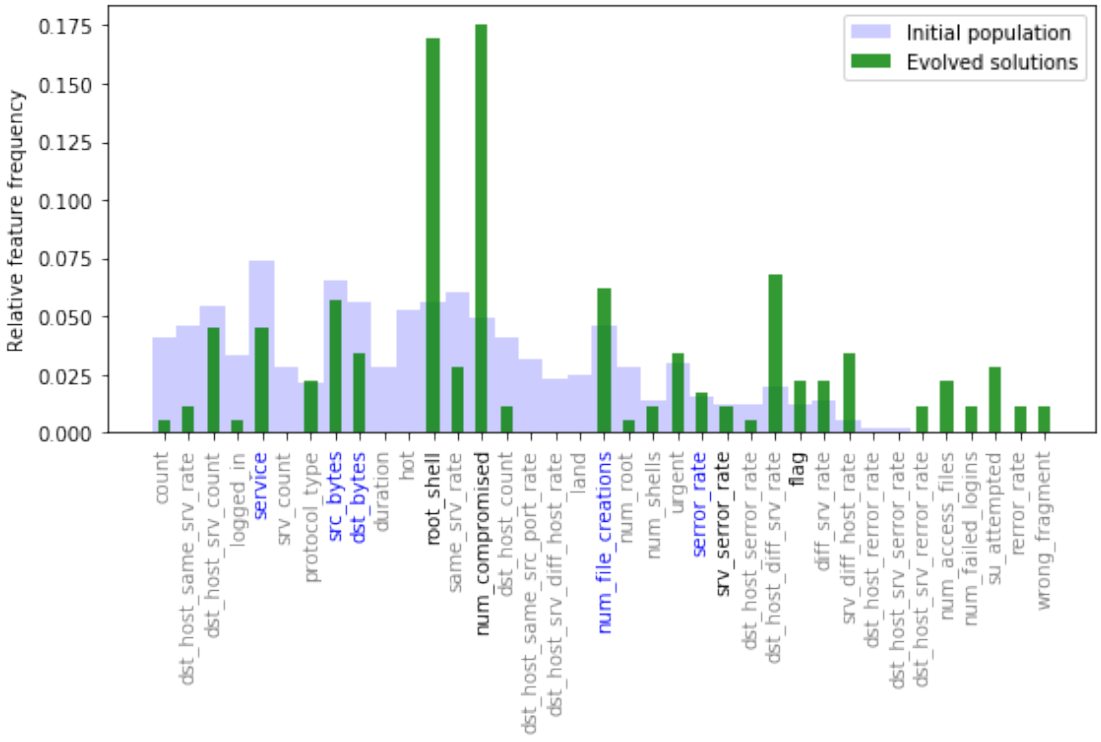

Figure 5.17: Genetic drift in the proposed method on U2R attacks

6. Discussion

Based on the results of the conducted experiments, and the comparison of the obtained results not only with other techniques but with individual components of the proposed algorithm, it can be concluded that the approach taken in this thesis has been successful. The various properties of the proposed method, both positive and negative, need to be taken into consideration in order to map out the best use cases and identify ares for further improvement.

The proposed method can be seen to outperform pure filter and wrapper approaches, especially when selecting smaller feature subsets. This is thanks to three components: the filter-based information embedded into the initial genetic material, the stochastic search procedure that follows, and its ability to perform multi-objective optimization. The algorithm produces a wide selection of solutions in a relatively short time, during which it manages to investigate a larger portion of the search space compared to traditional wrapper methods by using small bootstrapped samples for evaluation. This allows it to abandon clearly poor candidates (which constitute the majority of the search space) without having to train on full data, with the added benefit of producing fresh datasets ad infinitum. Ever-changing evaluation samples serve as a safeguard for the feature selection process against spontaneous overfitting that may occur in the chosen classifier.

Using bootstrapped samples to re-evaluate the entire population on each generation is definitely a rewarding technique that the proposed method cannot do without, but it becomes progressively wasteful for long runs if there is a population with a high proportion of veterans that have seen most of the data already. Even though any desired level of caching can be implemented, small bootstrapped samples will cumulatively cost more in terms of evaluation efforts than using all of the data at once. This is due to the nature of sampling with replacement, where one needs to draw roughly 3 times more samples than the size of the original data to ensure that each observation has been sampled with approx. 95% probability. However, seeing how this technique helps avoid the dangers associated with the chosen classifier and evaluates unfit solutions at a lesser cost, this trade-off is more than welcome.

The contribution of filters to the efficiency of this method cannot be understated: randomly seeded populations with the same search parameters fail to perform as well as the proposed multi-phase algorithm. The ensemble-like approach to ranking-based seeding makes it possible to use any number of filters or even manual rankings to initialize wrapper search without forcing consensus. In the case of filters chosen for this thesis, Fisher score seems to perform with the least consistency out of the three, but during the course of experiments it has proved to be an important member of the ensemble, since it introduces a vital portion of informed diversity into the initial frontier of explored solutions, helping to avoid homogeneity and premature convergence.

The instability of reported filter performances has three main causes: their failure to account for feature interactions, the embedded feature selection method of the decision tree classifier used for training and evaluation, and the data transformation that takes place in order to satisfy various filter requirements. Filter-based feature ranking is not an accurate depiction of optimal feature subsets, but a simple assessment of individual features. The forced introduction of ordinality to nominal features can create some confusion in data-sensitive filters. When the internal mechanism of the classifier is in discord with the given criterion, poor performance follows. These are expected and known drawbacks of the corresponding methods.

Although the proposed algorithm improves on the results of the filter phase by means of evolution, a poor initial population can be disruptive to the entire method, serving as a setback instead of boosting its progress, especially when the algorithm has limited time to produce results. This is typical of all population-based approaches [16]. Naturally, as uniform randomness is introduced to the genetic material via mutation, the process will recover and advance towards better solutions, but the initial damage will have been done. It is questionable as to which combination of filters would produce a backward population like that, but in order to circumvent this unlikely scenario, preliminary verification of filter-based solutions can be performed to gauge their potential impact.

With evolutionary approaches naturally favoring parallelization, the proposed algorithm is designed for maximal control over the explored search space and the complexity of performed computations, which is critical when dealing with NP-hard problems due to their enormous size. This degree of reconfigurability makes it possible to address the time factor and tailor the algorithm to run efficiently on a distributed network of machines with different computational capabilities. The tools selected for implementation enable this with very little additional code.

The output of the proposed algorithm not only provides one with the means of performing fast and efficient feature selection, but allows a detailed and thorough analysis of the problem domain, feature characteristics and their interactions. Even at modest population sizes and basic objectives, such as the ones used in this thesis, the explanatory power and the quality of solutions contained within non-dominated sets of solutions is unmatched by traditional single-objective feature selection methods.

There are many ways to improve on the proposed method. Three prospects for future work spring to mind. The first one is making the algorithm adaptive in a variety of aspects. Allowing dynamic reconfiguration during the search procedure based on intermediate results, backtracking to previous states, automatically estimating convergence and using more intricate fitness evaluation techniques can all be found in many successfully applied Evolutionary Algorithms and will only enhance the speed and thoroughness of performed search, minimizing some of the weaknesses associated with stochastic approaches. The second undertaking would be to devise and employ more powerful genetic operators in order to fully leverage domain knowledge and available heuristics by incorporating them into the process of evolution. As demonstrated in this thesis, the choice of genetic operators serves as the driving force behind the wrapper phase and plays a key role in the way search is performed. The third avenue for future work would be to focus on the development of analytical tools and metrics to accompany the method, making it possible to explore, map and explain the problem domain with speed and efficiency that is sought after in practice when tackling challenging problems.

7. Conclusion

The aim of this thesis was to develop a Genetic Algorithm-based search method for multi-objective feature selection, applicable in the domain of anomaly-based intrusion detection. The main motivation was to realize the high potential of multi-objective methods using combined filter-wrapper feature selection techniques, a combination that was shown in recent literature surveys to be poorly explored. The proposed method was designed, its results were validated and benchmarked on a classical intrusion detection dataset against traditional feature selection methods. The obtained results demonstrated the method's effectiveness and aligned with the ones reported in relevant literature.

The main concerns were computational cost and scalability of the method, as well as its ability to handle mixed data. For these purposes, the algorithm was designed to be easily configurable to match any desired depth, breadth and duration of search. The proposed approach combines the robustness of filter selection methods with the thoroughness and versatility of Genetic Algorithms. It is guided by multiple objectives and finds sets of non-dominated Pareto-optimal solutions, which bring great flexibility and informedness to the final decision made by the end user.

In conclusion, all of the objectives set forth in this thesis have been successfully accomplished. The proposed approach manages to address all of the main concerns that were raised. It augments the strengths of used components' while softening their weaknesses and isolates flaws to less likely scenarios that can be circumvented with proper planning and procedures. Naturally, it cannot completely eliminate drawbacks that are characteristic of the methods themselves, but provides high performance, efficient solutions and exceptional explanatory power.

Acknowledgments

I would like to thank my supervisor Margarita Spitšakova for her guidance and strategical advice, without which the core ideas of this thesis would not have been generated; Sven Nõmm, who has provided useful feedback which addressed important aspects of this work; Ivan Senilov, for sharing his experience and practical knowledge of machine learning. Last, but not least, I would like to thank my friends and family, especially my mother and father – for their undying support and encouragement.

Bibliography

[1] Hawkins, D. M. *Identification of Outliers*. Monographs on applied probability and statistics. 1980.

[2] Aggarwal, C. C. *Outlier Analysis*. 2013.

[3] Ishida, T., Niu, G., Sugiyama, M. Binary Classification from Positive-Confidence Data. In *Advances in Neural Information Processing Systems 31*, pages 5917–5928. 2018.

[4] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18 – 28, 2009.

[5] Jyothsna, V., Rama Prasad, V. V., Munivara Prasad, K. A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, 28:26–35, 2011.

[6] Du, M., Li, F., Zheng, G., Srikumar, V. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[7] Emmott, A. F., Das, S., Dietterich, T., Fern, A., Wong, W.-K. Systematic Construction of Anomaly Detection Benchmarks from Real Data. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, ODD '13, pages 16–21, 2013.

[8] Maza, S., Touahria, M. Feature Selection Algorithms in Intrusion Detection System: A Survey. *KSII Transactions on Internet and Information Systems*, 12:5079–5099, 2018.

[9] Littlewood, B., Strigini, L. Redundancy and Diversity in Security. In *Computer Security – ESORICS 2004*, pages 423–438, 2004.

[10] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[11] Guyon, I., Elisseeff, A. An Introduction to Variable and Feature Selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[12] Miao, J. and Niu, L. A Survey on Feature Selection. *Procedia Computer Science*, 91:919–926, 2016.

[13] Tang, J., Alelyani, S., Liu, H. *Feature selection for classification: A review*, pages 37–64. 2014.

[14] Xue, B., Zhang, M., Browne, W. N., Yao, X. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[15] Weise, T. *Global Optimization Algorithm: Theory and Application*. 2009.

[16] Luke, S. *Essentials of Metaheuristics* . Second edition, 2013.

[17] Kudo, M., Sklansky, J. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25 – 41, 2000.

[18] Das, I., Dennis, J. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural Optimization*, 14:63–69, 1997.

[19] Alaa, T., Anbar, M., Al-Ani, A., Al-tamimi, B., Faleh, N. Review on Feature Selection Algorithms for Anomaly-Based Intrusion Detection System. pages 605–619, 2019.

[20] Mitchell, M. *An Introduction to Genetic Algorithms*. 1998.

[21] Xue, B., Cervante, L., Shang, L., Browne, W., Zhang, M. Multi-objective evolutionary algorithms for filter based feature selection in classification. *International Journal on Artificial Intelligence Tools*, 22, 2013.

[22] Xue, B., Fu, W., Zhang, M. Differential evolution (DE) for multi-objective feature selection in classification. *GECCO 2014 - Companion Publication of the 2014 Genetic and Evolutionary Computation Conference*, 2014.

[23] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[24] Deb, K., Jain, H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[25] Zhou, L., Liu, Y., Chen, G. A Feature Selection Algorithm to Intrusion Detection Based on Cloud Model and Multi-Objective Particle Swarm Optimization. In *2011 Fourth International Symposium on Computational Intelligence and Design*, volume 2, pages 182–185, 2011.

[26] Hameed, S., Mahmood, D. A Multi-Objective Evolutionary Algorithm based Feature Selection for Intrusion Detection. *Iraqi Journal of Science*, 58(1C): 536–549, 2017.

[27] Balasaraswathi, V. R., Sugumaran, M., Hamid, Y. Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms. *Journal of Communications and Information Networks*, 2(4): 107–119, 2017.

[28] Xue, B., Zhang, M., Browne, W. Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach. *IEEE transactions on cybernetics*, 43:1656–1671, 2013.

[29] Khan, A., Baig, A. R. Multi-Objective Feature Subset Selection using Non-dominated Sorting Genetic Algorithm. *Journal of Applied Research and Technology*, 13(1):145 – 159, 2015.

[30] Zhou, Z., Li, S., Qin, G., Folkert, M., Jiang, S., Wang, J. Automatic multi-objective based feature selection for classification. *CoRR*, abs/1807.03236, 2018.

[31] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. 1996.

[32] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, 2009.

[33] Revathi, S., Malathi, A. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. *International Journal of Engineering Research & Technology (IJERT)*, 2:1848–1853, 2013.

[34] Dhanabal, L., Shantharajah, S. P. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. 2015.

[35] Jovic, A., Brkic, K., Bogunovic, N. A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205, 2015.

[36] Thaseen, I. S., Kumar, C. A. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University - Computer and Information Sciences*, 29(4):462 – 472, 2017.

[37] Peng, H., Long, F., Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[38] Bonev, B. *Feature Selection based on Information Theory*. PhD thesis, University of Alicante, 2010.

[39] Vergara, J. R., Estévez, P. A. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014.

[40] Bennasar, M., Hicks, Y., Setchi, R. Feature selection using Joint Mutual Information Maximisation. *Expert Systems with Applications*, 42(22):8520 – 8532, 2015.

[41] Aggarwal, C. C. *Data Classification: Algorithms and Applications*. 2014.

[42] Hall, M. A. Correlation-based Feature Selection for Machine Learning. Technical report, 1999.

[43] Dash, M., Liu, H. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1):155 – 176, 2003.

[44] Shin, K., Fernandes, D., Miyazaki, S. Consistency Measures for Feature Selection: A Formal Definition, Relative Sensitivity Comparison and a Fast Algorithm. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'11*, pages 1491–1497, 2011.

[45] Radivojac, P., Obradovic, Z., Dunker, A. K., Vucetic, S. Feature Selection Filters Based on the Permutation Test. In *Proceedings of the 15th European Conference on Machine Learning*, ECML'04, pages 334–346, 2004.

[46] Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., Moore, J. H. Relief-Based Feature Selection: Introduction and Review. *CoRR*, abs/1711.08421, 2017.

[47] Gu, Q., Li, Z., Han, J. Generalized Fisher Score for Feature Selection. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pages 266–273, 2011.

[48] Cover, T. M., Thomas, J. A. *Elements of Information Theory*. 1991.

[49] Pearson, K. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302): 157–175, 1900.

[50] Aggarwal, C. C. *Data Mining: The Textbook*. 2015.

[51] Tibshirani, R. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[52] Lal, T. N., Chapelle, O., Weston, J., Elisseeff, A., Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. Embedded Methods. *Feature Extraction, Foundations and Applications*, pages 137–165, 2006.

[53] Chen, X., Jeong, J. C. Enhanced recursive feature elimination. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 429–435, 2007.

[54] Liu, C., Wang, W., Zhao, Q., Shen, X., Konan, M. A New Feature Selection Method Based on a Validity Index of Feature Subset. *Pattern Recogn. Lett.*, 92 (C):1–8, 2017.

[55] Loh, W.-Y. Classification and Regression Trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:14 – 23, 2011.

[56] Mwadulo, M. A Review on Feature Selection Methods For Classification Tasks. *International Journal of Computer Applications Technology and Research*, 5: 395–402, 2016.

[57] Kohavi, R., John, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273 – 324, 1997.

[58] Xu, L., Yan, P., Chang, T. Best first strategy for feature selection. volume 2, pages 706 – 708, 1988.

[59] Narendra, P. M., Fukunaga, K. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computers*, 26(9):917–922, 1977.

[60] Ris, M., Barrera, J., Martins Jr., D. C. A branch-and-bound feature selection algorithm for U-shaped cost functions. *CoRR*, abs/0810.5573, 2008.

[61] Frank, A., Geiger, D., Yakhini, Z. A Distance-Based Branch and Bound Feature Selection Algorithm. *CoRR*, abs/1212.2488, 2012.

[62] Aha, D. W., Bankert, R. L. *A Comparative Evaluation of Sequential Feature Selection Algorithms*, pages 199–206. 1996.

[63] Pudil, P., Novovičová, J., Kittler, J. Floating Search Methods in Feature Selection. *Pattern Recogn. Lett.*, 15(11):1119–1125, 1994.

[64] Rudnicki, W., Wrzesien, M., Paja, W. All Relevant Feature Selection Methods and Applications. *Studies in Computational Intelligence*, 584:11–28, 2015.

[65] Altmann, A., Tolosi, L., Sander, O., Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[66] Arpita, N., Gaur, D. Hybrid Feature Selection Approach Based on GRASP for Cancer Microarray Data. *Journal of computing and information technology*, 25 (2):133–148, 2017.

[67] Zhang, H., Sun, G. Feature selection using tabu search method. *Pattern Recognition*, 35(3):701 – 711, 2002.

[68] Ashlock, D. *Evolutionary Computation for Modeling and Optimization*. 2006.

[69] Neri, F., Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[70] Jebari, K. Selection Methods for Genetic Algorithms. *International Journal of Emerging Sciences*, 3:333–344, 2013.

[71] Umbarkar, A., Sheth, P. D. Crossover operators in Genetic Algorithms: a review. *ICTACT Journal on Soft Computing*, 6(1), 2015.

[72] Ciro, G. C., Dugardin, F., Yalaoui, F., Kelly, R. A NSGA-II and NSGA-III comparison for solving an open shop scheduling problem with resource constraints. *IFAC-PapersOnLine*, 49(12):1272 – 1277, 2016.

[73] Ali, M., Ali, S. I., Kim, D., Hur, T., Bang, J., Lee, S., Kang, B. H., Hussain, M. uEFS: An efficient and comprehensive ensemble-based feature selection methodology to select informative features. *PLOS ONE*, 13(8):1–28, 2018.

[74] Kumar, G. Evaluation Metrics for Intrusion Detection Systems-A Study. *International Journal of Computer Science and Mobile Applications*, 11, 2015.

[75] Oliphant, T. E. *Guide to NumPy*. 2nd edition, 2015.

[76] Jones, E., Oliphant, T. E., Peterson, P. et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`. [Online] Accessed: 07.05.2019.

[77] McKinney, W. pandas: a Foundational Python Library for Data Analysis and Statistics. *Python High Performance Science Computer*, 2011.

[78] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[79] Raschka, S. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, 3(24), 2018.

[80] Fortin, F-A., De Rainville, F-M., Gardner, M-A., Parizeau, M., Gagné, C. DEAP: Evolutionary Algorithms Made Easy . *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[81] Kaufman, S., Rosset, S., Perlich, C. Leakage in Data Mining: Formulation, Detection, and Avoidance. volume 6, pages 556–563, 2011.

[82] Hosseinzadeh Aghdam, M., Kabiri, P. Feature Selection for Intrusion Detection System Using Ant Colony Optimization. *International Journal of Network Security*, 18:420–432, 2016.

[83] Gaikwad, D. P., Thool, R. C. Intrusion Detection System using Ripple Down Rule learner and Genetic Algorithm. volume 5, pages 6976–6980, 2014.

[84] Mukherjee, S., Neelam, S. Intrusion Detection using Naive Bayes Classifier with Feature Reduction. *Procedia Technology*, 4:119 – 128, 2012.

[85] Popoola, E., Adewumi, A. Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision tree. *International Journal of Network Security*, 19:660–669, 2017.

[86] Eid, H. F., Hassanien, A. E., Kim, T-H., Banerjee, S. Linear Correlation-Based Feature Selection for Network Intrusion Detection Model. In *Advances in Security of Information and Communication Networks*, pages 240–248, 2013.