9-30-2022

Anomaly detection in cybersecurity datasets via cooperative co-evolution-based feature selection

Bazlur A. N. M. Rashid
Edith Cowan University, anm.bazlur.rashid@gmail.com

Mohiuddin Ahmed
Edith Cowan University, mohiuddin.ahmed@ecu.edu.au

Leslie F. Sikos
Edith Cowan University, l.sikos@ecu.edu.au

Paul Haskell-Dowland
Edith Cowan University, p.haskelldowland@ecu.edu.au

10.1145/3495165

This is an Author's Accepted Manuscript of: Rashid, A. N. M. B., Ahmed, M., Sikos, L. F., & Haskell-Dowland, P. (2022). Anomaly detection in cybersecurity datasets via cooperative co-evolution-based feature selection. *ACM Transactions on Management Information Systems, 13*(3), article 29.

https://doi.org/10.1145/3495165

Anomaly Detection in Cybersecurity Datasets via Cooperative Co-Evolution-Based Feature Selection

A. N. M. BAZLUR RASHID*, Edith Cowan University, Australia
MOHIUDDIN AHMED, Edith Cowan University, Australia
LESLIE F. SIKOS, Edith Cowan University, Australia
PAUL HASKELL-DOWLAND*, Edith Cowan University, Australia

Anomaly detection from Big Cybersecurity Datasets is very important; however, this is a very challenging and computationally expensive task. Feature selection (FS) is an approach to remove irrelevant and redundant features and select a subset of features, which can improve the machine learning algorithms' performance. In fact, FS is an effective preprocessing step of anomaly detection techniques. This paper's main objective is to improve and quantify the accuracy and scalability of both supervised and unsupervised anomaly detection techniques. In this effort, a novel anomaly detection approach using FS, called Anomaly Detection Using Feature Selection (ADUFS), has been introduced. Experimental analysis was performed on five different benchmark cybersecurity datasets with and without feature selection and the performance of both supervised and unsupervised anomaly detection techniques were investigated. The experimental results indicate that instead of using the original dataset, a dataset with a reduced number of features yields better performance in terms of true positive rate (TPR) and false positive rate (FPR) than the existing techniques for anomaly detection. For example, with FS, a supervised anomaly detection technique, multilayer perception increased the TPR by over 200%, and decreased the FPR by about 97% for the KDD99 dataset. Similarly, with FS, an unsupervised anomaly detection technique, local outlier factor increased the TPR by more than 40%, and decreased the FPR by 15% and 36% for Windows 7 and NSL-KDD datasets, respectively. In addition, all anomaly detection techniques require less computational time when using datasets with a suitable subset of features rather than entire datasets. Furthermore, the performance results have been compared with six other state-of-the-art techniques based on a decision tree (J48).

CCS Concepts: • **Security and privacy** → **Anomaly detection**; • **Computing methodologies** → *Feature selection*; • **Mathematics of computing** → Evolutionary algorithms.

Additional Key Words and Phrases: anomaly detection, feature selection, cybersecurity, Big Data, cooperative co-evolution, machine learning

ACM Reference Format:
A. N. M. Bazlur Rashid, Mohiuddin Ahmed, Leslie F. Sikos, and Paul Haskell-Dowland. 2021. Anomaly Detection in Cybersecurity Datasets via Cooperative Co-Evolution-Based Feature Selection. In *Special Issue on Pattern-Driven Mining, Analytics and Prediction for Decision Making: ACM Trans. Manag. Inform. Syst.*. ACM, New York, NY, USA, 40 pages. https://doi.org/10.1145/XXX

*Corresponding Author: a.rashid@ecu.edu.au. Authors' address: A. N. M. B. Rashid, M. Ahmed, L. F. Sikos, and P. Haskell-Dowland, School of Science, Edith Cowan University, Australia; emails: a.rashid@ecu.edu.au, mohiuddin.ahmed@ecu.edu.au, l.sikos@ecu.edu.au, p.haskelldowland@ecu.edu.au.

Manuscript submitted to ACM

1 INTRODUCTION

Generating massive volumes of data is commonplace in the Big Data era, as seen in areas such as cybersecurity [37, 39–41]. While Big Data opens the door to the research community to explore new insights, analyzing the Big Cybersecurity Data produced by different network applications in (near-)real time is often computationally expensive [2, 42, 45]. Anomaly detection is related to identifying data patterns, which deviate unusually from expected behaviour. Anomaly detection is important in many domains, including cybersecurity, health, fault prevention, and Internet-of-Things (IoT)/sensor networks [5, 55, 56]. Therefore, anomaly detection involves a pattern-driving data mining process in Big Data Analytics. It can predict anomalous data that can later be used for decision-making in the decision support or management information systems (MIS) [9]. Both supervised and unsupervised machine learning-based anomaly detection techniques are used to learn, predict, detect, and classify data in this context [39]. Examples of some anonymity algorithms include [5, 25]. Cybersecurity data consist of several features (attributes in dataset terminology). However, not all of these are important, because some are redundant or irrelevant [48], and as such, may degrade the performance of ML-based anomaly detection algorithms. Feature selection (FS) is a process that selects the relevant features and removes the irrelevant ones to improve anomaly detection performance. Hence, removing unnecessary features from the cybersecurity data can help a lot to the cybersecurity personnel before the anomaly detection techniques are applied, which can reduce computational time and improve anomaly detection performance. Furthermore, removing irrelevant features from the datasets before the anomaly detection process can also reduce the storage requirement [1, 39, 40].

When a dataset consists of n features, it has 2^k possible solutions of selected feature subset, which makes a feature selection process computationally expensive. A number of search techniques, for example, evolutionary search, best search, or greedy search, can be applied to the large search space to select a suitable subset of features. *Evolutionary algorithms (EA)* are widely-used search techniques in this context. *Cooperative co-evolution (CC)*, a meta-heuristic algorithm, which follows a divide-and-conquer strategy, has been effectively applied in different domains, including a limited number of feature selection applications [39, 40]. We previously proposed a *Cooperative Co-Evolutionary Algorithm-Based Feature Selection (CCEAFS)* with a penalty-based wrapper objective function for Big Data. The experiments performed on six UCI ML repository datasets using six supervised ML classifiers perform better than the existing techniques [39]. We also proposed an improvement of CCEAS, called *Cooperative Co-Evolutionary Algorithm-Based Feature Selection with Random Feature Grouping (CCFSRFG)* [38], which further improved classification accuracy. In this paper, we apply the CCFSRFG technique for anomaly detection in cybersecurity datasets.

This paper introduces a novel anomaly detection approach by feature selection using a cooperative co-evolution technique, called *Anomaly Detection Using Feature Selection (ADUFS)*. This technique has been evaluated using both supervised and unsupervised ML-based anomaly detection algorithms. Five different cybersecurity datasets have been collected from the UNSW Canberra Cyber Centre repository,[1][28] the Canadian Institute for Cybersecurity,[2] and the UCI ML repository.[3] 10 supervised and 10 unsupervised anomaly detection techniques have been used to detect anomalies from the selected datasets [1, 20]. Based on the analysis of the comparative results, it has been observed that in terms of true positive rate (TPR) and execution time (ET), ADUFS, in most cases, outperforms the anomaly detection performance of the state of the art when using feature selection.

The feature selection process selects a subset of features, which is a representative subset of the original data. As a result, apart from anomaly detection techniques, other types of detections, such as network traffic analysis applications,

[1] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/
[2] https://www.unb.ca/cic/
[3] https://archive.ics.uci.edu/ml/

can also be applied on the datasets with feature selection to improve the detection performance with less computing time. This paper aims to investigate the following research questions:

- How can a feature selection process be applied to cybersecurity datasets that can select a suitable subset of features, which can improve the performance of the supervised and unsupervised anomaly detection techniques?
  - How can the supervised and unsupervised anomaly detection techniques be applied to the original datasets and the datasets with fewer features?
  - Can anomaly detection techniques perform as well on datasets with feature selection as on original datasets?
  - Can the datasets with feature selection reduce the execution time of anomaly detection?

1.1 Contribution to the Society

In the era of the digital world, we are now almost entirely dependent on electronic devices and communication via the internet. While we are getting advantages of this environment, so many safety and security issues are also involved. According to the Australian Cyber Security Centre (ACSC)[4], over 67,500 cybercrime reports were made through ReportCyber[5] during the 2020–21 financial year. This is an increase of about 13 per cent from the previous financial year. It indicates one cybercrime report is made roughly every eight minutes in Australia. Researchers are contributing from each corner to analyze a particular issue and provide an effective solution to tackle all the relevant issues. Accordingly, our proposed solution on anomaly detection for data analysis will help identify the cyber world's malicious activities. It can also help to identify important biomarkers for effective medical diagnosis. We proved throughout the experiments that our proposed solution is an effective solution for analyzing large-scale data, such as Big Data. Big Data consists of many attributes, most of which are irrelevant and result in a time-consuming learning process and identifying malicious activities or biomarkers. Therefore, we believe our proposed solution is an effective solution for society in dealing with the problems in several domains, such as cybersecurity or health care.

The rest of the paper is organized as follows. Section 2 presents a literature review on anomaly detection in cybersecurity datasets. Section 3 illustrates our novel anomaly detection approach by feature selection using cooperative co-evolution. Section 4 contains experimental results and analysis based on benchmark cybersecurity datasets. The conclusion and future work directions are included in Section 5.

2 LITERATURE REVIEW

Anomaly detection is an important data analysis task to detect the anomalous or abnormal data from a dataset [50]. The synonyms of anomaly detection are outlier detection, deviation detection, novelty detection, and exception mining, which are widely used and studied in machine learning and statistics. The wide range of application domains of anomaly detection includes cyber-situational awareness [47], network path estimation [32], intrusion detection, fraud detection, sensor networks, medical, public health, and image processing [1]. An example of anomaly detection is cyberattack detection [12]. A cyberattack can be a malicious attack, which may damage a computing system via unauthorized network access, code, or data injection [43].

An anomaly can be of three different types [3]: 1) *point/rare anomaly* (when a specific data instance deviates from the normal pattern in a dataset), 2) *contextual anomaly* (when a data instance behaves anomalously in a specific context), and 3) *collective anomaly* (when a collection of similar data instances behave anomalously in an entire dataset). Different anomaly detection techniques represent anomalies, either as a score or a binary (true/false) label. The anomaly detection

[4] https://www.cyber.gov.au/acsc/view-all-content/reports-and-statistics/acsc-annual-cyber-threat-report-2020-21

[5] https://www.cyber.gov.au/acsc/report

techniques based on score assign an anomaly score to each data instance in a dataset, after which the analyst can select anomalies based on a ranked score threshold. In the case of binary, the anomaly outputs can be labeled either as normal or anomaly [1, 4]. There are many types of cyberattacks, such as 1) *denial of service (DoS)* (when the normal computing environment is being interrupted, and the services become unavailable), 2) *probe* (when a targeted host or network is attacked to collect information for reconnaissance purposes), 3) *user to root (U2R)* (when an attacker tries to get illegal access to an administrative account for manipulating or abusing important resources), and 4) *remote to user (R2U)* (when an attacker tries to get local access of a targeted computing system to send packets over its network). In the literature, U2R and R2L are grouped into point/rare anomalies, DoS is grouped into collective anomalies, and probe is grouped into contextual anomalies [2–4].

A number of anomaly detection techniques are used to handle various types of attacks. In the literature, cybersecurity attacks have been handled by three dominant approaches: supervised, semi-supervised, and unsupervised [3, 4, 21]. A taxonomy of anomaly detection techniques is shown in Fig. 1.

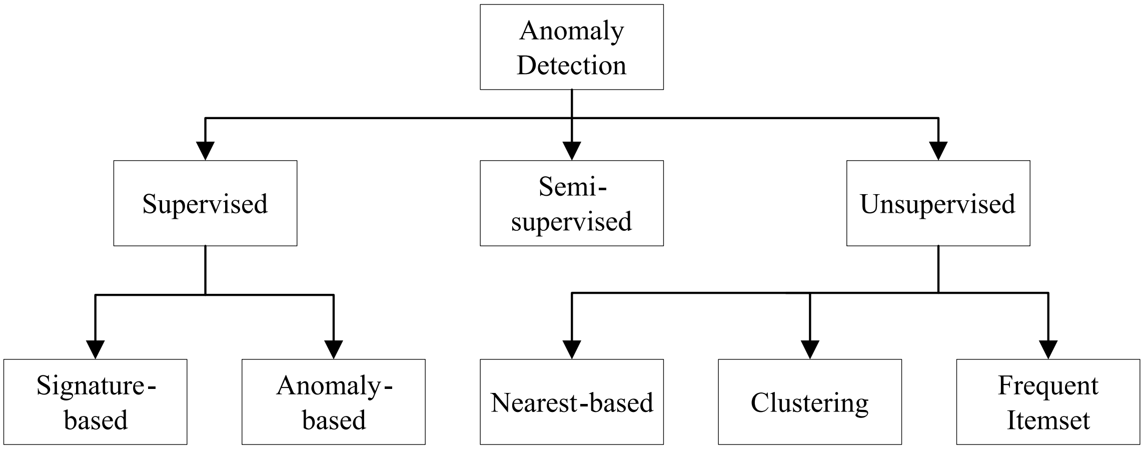


Fig. 1. A taxonomy of anomaly detection techniques [3, 4].

The anomaly detection techniques that rely on labeled training are supervised. Supervised techniques require training data, which is usually expensive to generate, and these techniques face difficulties when it comes to detecting new types of attacks. Semi-supervised methods require a small amount of labeled data for building a model to detect anomalies. Unsupervised techniques do not need any training data, and can detect previously unseen attacks [3].

Supervised anomaly detection techniques are further classified into signature-based and anomaly-based techniques. The former requires extensive knowledge of cyberattacks' characteristics, i.e., the attack's signature is the primary means for detecting anomalies. Failure to provide the attack's signature, these techniques are unable to detect new attacks. In contrast, anomaly-based methods depend on the normal traffic activity profile for building a knowledge base and take account of the activities that deviate from this base as anomalous. Although anomaly-based supervised detections perform better than signature-based detections in detecting zero-day attacks, anomaly-based methods require training to build a normal activity profile. In practice, training to build a normal activity profile is computationally expensive and depends on the availability of new normal traffic instances, which are rare and expensive [3]. Semi-supervised anomaly detection techniques are a subclass of supervised anomaly detection techniques and can train a model using a small amount of labeled data with a large amount of unlabeled data. When the characteristics of cyberattacks evolve over time, semi-supervised techniques will result in poor performance [1]. Unsupervised anomaly detection techniques use unlabeled data to draw inferences from datasets. Clustering, support vector machine (SVM), nearest neighbor, and frequent itemset data mining methods are core techniques in unsupervised anomaly detection approaches [3, 15, 19, 22].

Any type of information, especially if current, can be crucial for business continuity. Therefore, security models driven by cybersecurity datasets are important in both proactive and reactive defense mechanisms. Nowadays, operations based on government, commercial, military, and civilians are linked to network security and computer system availability. Therefore, cybersecurity data analysis is very important to indicate vulnerabilities and unveil security breaches, such as via detecting network inconsistencies [46]. The term *cybersecurity* can be defined as a set of measures to protect computers, programs, data, and networks from security risks and malicious actions in cyberspace. Cybersecurity tries to determine, prevent, and minimize the impacts of cyberattacks. This can be done by implementing a set of countermeasures and eliminating vulnerabilities of computer systems [24]. Feature selection is an essential preprocessing step in anomaly detection for cybersecurity data. In the literature, examples of feature selection as a preprocessing step in anomaly detection include generic feature selection (GeFS) for intrusion detection (IDS) [29], a least square support vector machine-based intrusion detection system (LSSVM-IDS) using a filter-based FS [7], feature extraction and FS to classify cyber traffic threats [27], a combined FS for cyber IDS [26], hierarchical FS for DDoS mitigation [23], an ensemble FS for cyber IDS [10], clustering and correlation-based FS to identify rare cyberattacks[8], and an FS-based automated assessment of open source cyberthreat intelligence sources[52]. While the feature selection process is applied to cybersecurity datasets to detect anomalies, evolutionary algorithms are the core search techniques used in feature selection. The mutual information-based binary gravitational search algorithmic wrapper FS for IDS [11] and the hybrid grasshopper-based FS for securing wireless network against cyberthreats [17] can serve as examples. Furthermore, evolutionary computations are widely used for anomaly detection [6, 18, 24]. Cooperative co-evolution (CC), a meta-heuristic algorithm, applies a divide-and-conquer strategy to decompose a large problems into several subproblems, optimizes each subproblem individually, and collaborates different subproblems to build a complete solution to the problem [34, 44]. Recently, the CC technique has been proposed in a wrapper-based FS process in Big Data (CCEAFS) by the authors of this paper and evaluated on a range of six datasets using six supervised ML classifiers. The performance results evaluated in terms of accuracy, precision, recall, and F1-score have performed better than the existing techniques [39]. In addition, the authors have also proposed a CC-based FS approach with a random feature grouping, called CCFSRFG. CCFSRFG outperforms CCEAFS on a mixed characteristics of datasets including lower samples higher features and higher samples with lower features [38]. To utilize its beneficial properties, in this paper, we apply CCFSRFG in cybersecurity datasets for anomaly detection.

A comparative study performed with ADUFS and existing recent literature based on feature selection approaches for cybersecurity data analysis is listed in Table 1. The Table clearly indicates that the proposed ADUFS approach have been investigated the application of feature selection in detecting anomalies in a large number of Big Cybersecurity Datasets and evaluated using a total of 20 supervised and unsupervised anomaly detection techniques.

In the next Section, the novel anomaly detection approach by feature selection using cooperative co-evolution in cyberesecurity data has been described in detail.

3 A NOVEL ANOMALY DETECTION APPROACH

In this paper, a novel anomaly detection approach, ADUFS, is introduced by feature selection (FS) using cooperative co-evolution (CC). The motivation to apply feature selection in anomaly detection comes from the success of the FS

Table 1. **A comparative study on feature selection-based approaches for cybersecurity data analysis.**

Reference	Techniques	Classifiers/Outliers	Datasets	Purpose	Outcomes	Limitations
[18]	DE, Flexible neural tree-based FS [16]	Rules-based	10% KDD99	Intrusion Detection (ID)	Higher detection rate and lower FPR on *if-then* security policies	Failed to validate with multi class traffic data
[7]	Filter-based FS on MI	1 supervised (SVM)	KDD99, NSL-kDD, and Kyoto 2006+	ID	Handles both linear and nonlinear features	Unbalanced sample distribution has not been considered
[27]	ANN-SNR based FS	ANN	CDX	Cyber traffic threats	Can detect no-threats and threats severity	ANN model development is computationally expensive
[11]	MI-BGSA-based wrapper FS	1 supervised (SVM)	NSL-KDD	IDS	higher accuracy and detection rate compared to standard method	Did not evaluated and compared with recent datasets and algorithms
[26]	FGLCC, CFA	1 supervised (decision tree)	KDD99	IDS	Higher accuracy and detection rate with lower false positive rates	Performance has not been evaluated with recent datasets
[8]	*k*-Means and correlation-based FS	two supervised (NB and J48)	UNSW_NB15	Rare cyber-attacks detection	Best accuracy with worm attacks	Feature selection could not optimal subset of features resulting in no impact on J48 classifier
[36]	SciKit Learn-based FS	4 supervised (LR, *k*-NN), DT, and RF	ISCX-URL-2016, NSL-KDD, and CICIDS-2017	IDS	Cross-comparison of baselined and feature selection algorithms	Validation results have not been performed
[23]	ANN-based FS	1 unsupervised (*k*-Means)	Collected data from ISP	DDoS mitigation	DDoS mitigation through feature selection	Validation requires using benchmark datasets
[17]	GOA and SA-based FS	1 supervised (SVM)	NSL-KDD and UNSW_NB15	Cyberthreats	Higher detection rate and lower false alarm rate	Validation requires with more recent datasets apart from UNSW_NB15
ADUFS (proposed)	CC-based FS [38]	10 supervised and 10 unsupervised ML algorithms	Windows 10, Windows 7, UNSW_NB15, KDD99, and NSL-KDD	Anomaly detection	Wide range of evaluation using benchmark datasets; both supervised and unsupervised detection	CCFSRFG performance can be improved with other suitable optimizers and collaboration techniques for higher anomaly detection rate

Note: DE–Differential Evolution, ANN-SNR–Artificial Neural Network-Signal-to-Noise Ratio, MI–Mutual Information, BGSA–Binary Gravitational Search Algorithm, FGLCC–Feature Grouping Based on Linear Correlation Coefficient, CFA–Cuttlefish Algorithm, GOA–Grasshopper Optimization Algorithm, SA–Simulated Annealing, CC–Cooperative Co-Evolution.

strategy in Big Data using CC (CCEAFS) [39]. After CCEAFS, we proposed another FS technique in Big Data using CC (CCFSRFG), which performs even better than CCEAFS [38]. In ADUFS, at first, the original datasets have been used to detect anomalies. Then the datasets with a reduced number of features have been used to detect anomalies. Both supervised and unsupervised anomaly detection techniques have been explored with original and datasets with a subset of features for detecting anomalies. In addition, all datasets have also been used with an equal number of instances to investigate the feature selection's effectiveness in anomaly detection for all datasets. The comparative results have been analyzed via different performance measures, including accuracy (ACC), true positive rate (TPR), false positive rate (FPR), and execution time (ET). In the following subsections, feature selection, cooperative co-evolution, and supervised and unsupervised anomaly detection techniques are described. Following the discussion on the related techniques, the proposed methodology has been discussed in detail.

3.1 Feature Selection

Feature selection (FS) is a technique that selects the relevant features to reduce the dimension of data in order to improve ML performance [40]. Formally speaking, feature selection is a process to select a subset of s features from a full set of n features ($s < n$) in a dataset by removing irrelevant and unimportant features, thereby representing it with a fewer number of features [14]. A search technique initiates the feature selection process to discover feature subsets. Then, feature subsets are evaluated by different performance measures, for example, classification accuracy. For instance, a terminating criterion is the maximum number of generations used to terminate the FS process. A validation method at the end of the FS process can test the validity of the selected subset of features [39, 40]. A taxonomy of FS approaches is illustrated in Fig. 2.

3.2 Cooperative Co-Evolution

In 1994, Potter and De Jong introduced the cooperative co-evolution (CC) technique to solve optimization problems, which are large-scale and complex [34]. This technique uses a divide-and-conquer approach to divide a large problem into multiple subproblems, and iteratively evolves the interacting co-adapted subproblems to build a complete solution.

The CC technique consists of decomposing an n-dimensional problem of search information $S = \{1, 2, \ldots, n\}$ into m subproblems $\{S_1, S_2, \ldots, S_m\}$ [34]. Each subproblem of the n dimensions represents a new search space $SP^{(i)}$ for a particular problem, where the rest of the dimensions n_j, with $j \neq S_i$, are kept fixed. This way, the entire search space is decomposed into m subproblems with lower dimensions, which can be handled using any population-based evolutionary computational approach. These subproblems can be optimized individually, and the communication between them is required only to evaluate the objective (fitness) function f. This implies that a candidate solution in search space $SP^{(i)}$ contains a few elements (comprising an individual I) of the n-dimensional problem ($I \in SP$). Therefore, in CC, a common n-dimensional context vector v is required to build using a collaborative individual (e.g., the current best individual) from each subproblem. A candidate solution to the problem is then formed by a cooperative algorithm to evaluate an individual in a subproblem; a candidate solution is built by joining representative collaborators from the context vector. Potter and De Jong decomposed an n-dimensional problem into n 1-dimensional subproblems in their original CC idea. In general, the n-dimensional problem can be decomposed into m subproblems with the same dimension, i.e., $n_m = n/m$ [53]. Following this, a CC technique can be formally defined as follows[51]:

An n-dimensional problem is decomposed as

$$S_i = \Big\{ (i-1) \times n_m + 1, \ldots, i \times n_m \Big\}$$

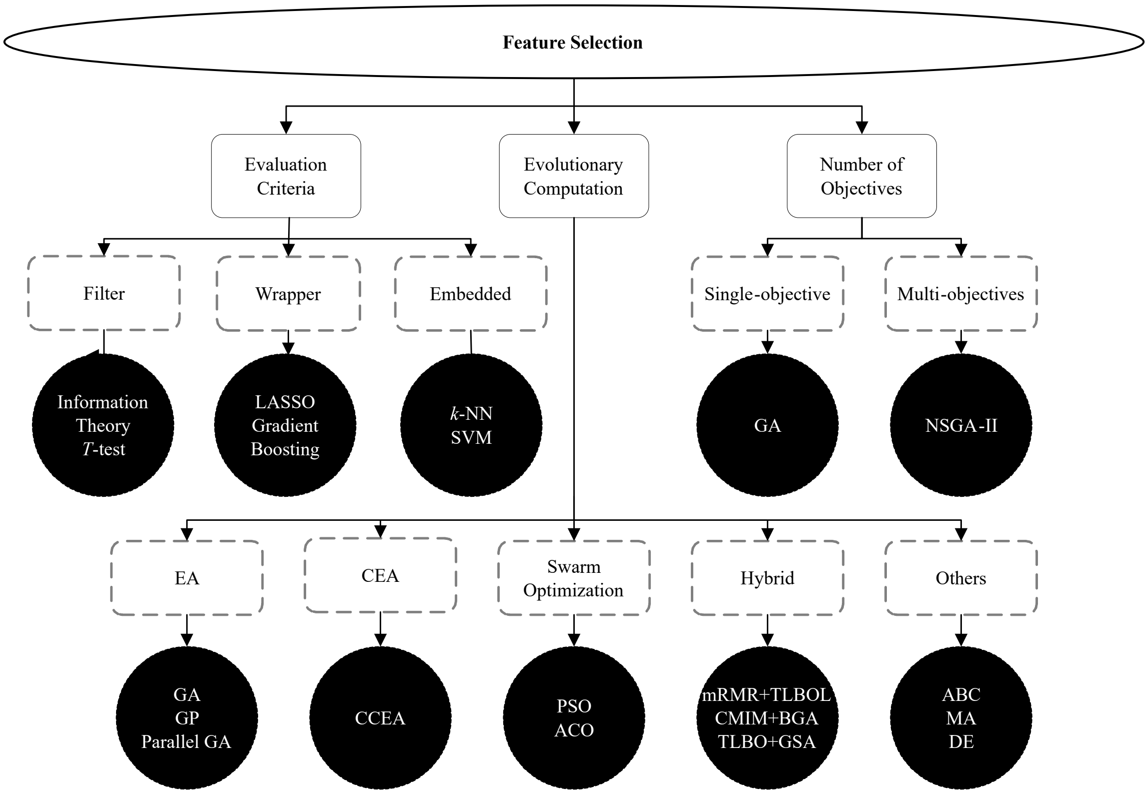


Fig. 2. A taxonomy of feature selection approaches [39].

and the context vector is built as

$$v = \underbrace{(v_1^{(1)}, \dots, v_{n_m}^{(1)}}_{v^{(1)}}, \underbrace{(v_1^{(2)}, \dots, v_{n_m}^{(2)}}_{v^{(2)}}, \dots, \underbrace{(v_1^{(m)}, \dots, v_{n_m}^{(m)})}_{v^{(m)}}{}^T$$

where $v^{(i)}$ is the n_m-dimensional problem, which represents the collaborative individual from the ith subproblem (e.g., the current best individual in $SP^{(i)}$):

$$v^{(i)} = \left(v_1^{(1)}, v_1^{(2)}, \dots, v_{n_m}^{(1)}\right)^T$$

Given the jth individual $I^{(i,j)} \in SP^{(i)}$ of the ith subproblem:

$$I^{(i,j)} = \left(I_1^{(i,j)}, I_2^{(i,j)}, \dots, I_{n_m}^{(i,j)}\right)^T$$

The fitness function of this problem is given by $f\left(v^{(i,j)}\right)$, where $v^{(i,j)}$ is defined as

$$v^{(i,j)} = \underbrace{(v_1^{(1)}, \dots, v_{n_m}^{(1)}}_{v^{(1)}}, \underbrace{(I_1^{(2)}, \dots, I_{n_m}^{(i,j)}}_{I^{(i,j)}}, \dots, \underbrace{(v_1^{(m)}, \dots, v_{n_m}^{(m)})}_{v^{(m)}}{}^T$$

In general, the fitness of $v^{(i,j)}$ is evaluated on context vector v by replacing the elements of the individual from the ith subproblem having the representative elements of individual $I^{(i,j)}$.

This confirms that CC is comprised of three main phases: 1) problem decomposition, 2) subproblem evolution, and 3) collaboration and evaluation.

3.2.1 *Problem Decomposition.*

The first phase of CC, i.e., how to decompose a large problem into subproblems, generally depends on the problem structure [44]. Problem decomposition techniques can be static or dynamic. If the problem is decomposed statically, it will have one element in each subproblem. In contrast, if the problem is decomposed dynamically, the grouping of elements into subproblems will be different. In the case of static decomposition, the problem is decomposed into subproblems before the evolutionary process starts, and all subproblems are fixed [13]. In contrast, in the case of dynamic decomposition, a problem is decomposed at the beginning; however, at the time of the evolutionary process, subproblems can self-adaptively tune to appropriate collaboration levels [30]. A few examples of decomposition techniques are presented in [30, 31, 57].

3.2.2 *Subproblem Evolution.*

After problem decomposition, each subproblem is allocated to a subpopulation. Each subpopulation is optimized individually either through a homogeneous or a heterogeneous evolutionary optimizer [44]. Optimizations of subpopulations can be carried out either sequentially or in parallel. In the sequential case, only one subpopulation evolves in each generation [33]. In contrast, in the parallel case, all subpopulations evolve in each generation simultaneously [54]. The most widely used evolutionary optimizer in this area are genetic algorithms (GAs), whereas differential evolution (DE) [49] is the most effective optimizer for CC.

3.2.3 *Collaboration and Evaluation.*

Once subproblems are optimized, subpopulations cooperate to build a complete solution to the problem. The fitness (objective function) of an individual is evaluated by selecting a collaborator individual from each subpopulation. The performance of this collaboration is specified as the fitness value to an individual being evaluated. Individuals having the best collaborating performance are joined together to discover the final solution to the problem at the end of a CC process [44]. 1+1 collaboration [35], the 1+N collaboration model [13], and reference sharing (RS) [44] are examples of collaboration models used in CC.

3.3 Supervised and Unsupervised Anomaly Detection Techniques

In this paper, both supervised and unsupervised anomaly techniques have been used to detect anomalies from cybersecurity datasets. In the case of supervised anomaly detection techniques, 10 supervised ML classifiers are used, which fall into 5 different categories: Bayesian network, functions, instance-based learning, meta-classifiers, and decision trees. The ML classifiers are naïve Bayes (NB), support vector machine (SVM), logistic regression (LR), simple logistic regression (SLR), multilayer perception (MLP), k-Nearest Neighbor (k-NN), multiclass classifier (MCC), J48, random forest (RF), and random tree (RT) [20]. In the case of unsupervised anomaly detection techniques, 10 unsupervised techniques are used, which fall into 3 different categories: nearest neighbor, clustering, and statistical. The unsupervised anomaly detection techniques are k-NN global anomaly score, local outlier factor (LOF), connectivity-based outlier factor (COF), approximate local correlation integral (aLOCI), local outlier probability (LoOP), influenced outlierness (INFLO), cluster-based local outlier factor (CBLOF), local density cluster-based outlier factor (LDCOF), clustering-based multivariate Gaussian outlier score (CMGOS), and histogram-based outlier score (HBOS) [1]. The supervised and unsupervised anomaly detection techniques are illustrated in Fig. 3 and Fig. 4.

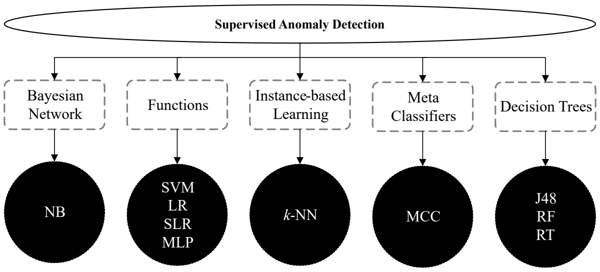


Fig. 3. Supervised anomaly detection techniques

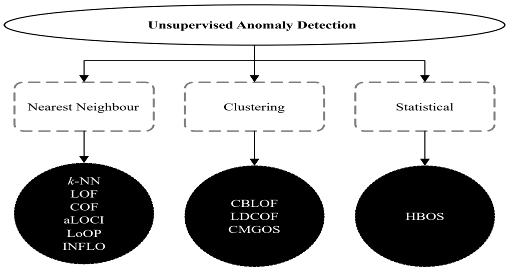


Fig. 4. Unsupervised anomaly detection techniques [3]

3.4 Methodology

The proposed anomaly detection approach, ADUFS, is illustrated in Fig. 5. The preprocessing stage mainly deals with the dataset conversion from CSV to ARFF to make it compatible with the experimental environment. Microsoft Excel and *WEKA*[6] have been used to preprocess the datasets. These datasets were then evaluated using both supervised and unsupervised anomaly detection approaches.

In the supervised anomaly detection approach, the datasets were split into training and test datasets. The split ratio between the training and test datasets were 60% and 40%, respectively. The training datasets were used to train the model using and evaluated using the test datasets. Normalization and missing value replacement operators were applied to preprocess the datasets for unsupervised anomaly detection approach when required. The anomaly detection performance was calculated in terms of TPR and ET using all the 10 supervised ML classifiers mentioned in Section 3.3. In the case of the unsupervised anomaly detection approach, the entire datasets were used to compute the outliers. TPR and ET were then calculated to evaluate the performance of the unsupervised anomaly detection approach using the 10 unsupervised anomaly detection techniques mentioned in Section 3.3.

After the original datasets were used to detect anomalies, the FS approach inspired by our previous work, called CCFSRFG [38], was used to reduce the dimension of the datasets. In CCFSRFG, a random feature grouping (RFG) based dynamic decomposition was applied to decompose the datasets into multiple subdatasets. Formally, the CCFSRFG used here can be described as follows:

A dataset D consisting of n features, i.e., $D = \{f_1, f_2, f_3, \ldots, f_n\}$. D is decomposed randomly into m subdatasets with s $(s < n)$ features in each subdataset:

$$D_1 = \left\{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\right\}$$

$$D_2 = \left\{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\right\}$$

$$\ldots$$

$$D_m = \left\{f_{i_1}, f_{i_2}, \ldots, f_{i_s}\right\}$$

When the correlations are associated with a dataset's records linearly, a linear correlation coefficient is applicable for measuring the linear dependency between two random attributes in a network traffic dataset. In practice, the correlation between the attributes (features) can be nonlinear for many real-world problems. Therefore, the nonlinear dependency between the two features cannot be measured by a correlation study. Hence, selecting a subset of features from the dataset that maximize the classification accuracy is more appropriate irrespective of whether the dependency

[6] https://www.cs.waikato.ac.nz/ml/weka/

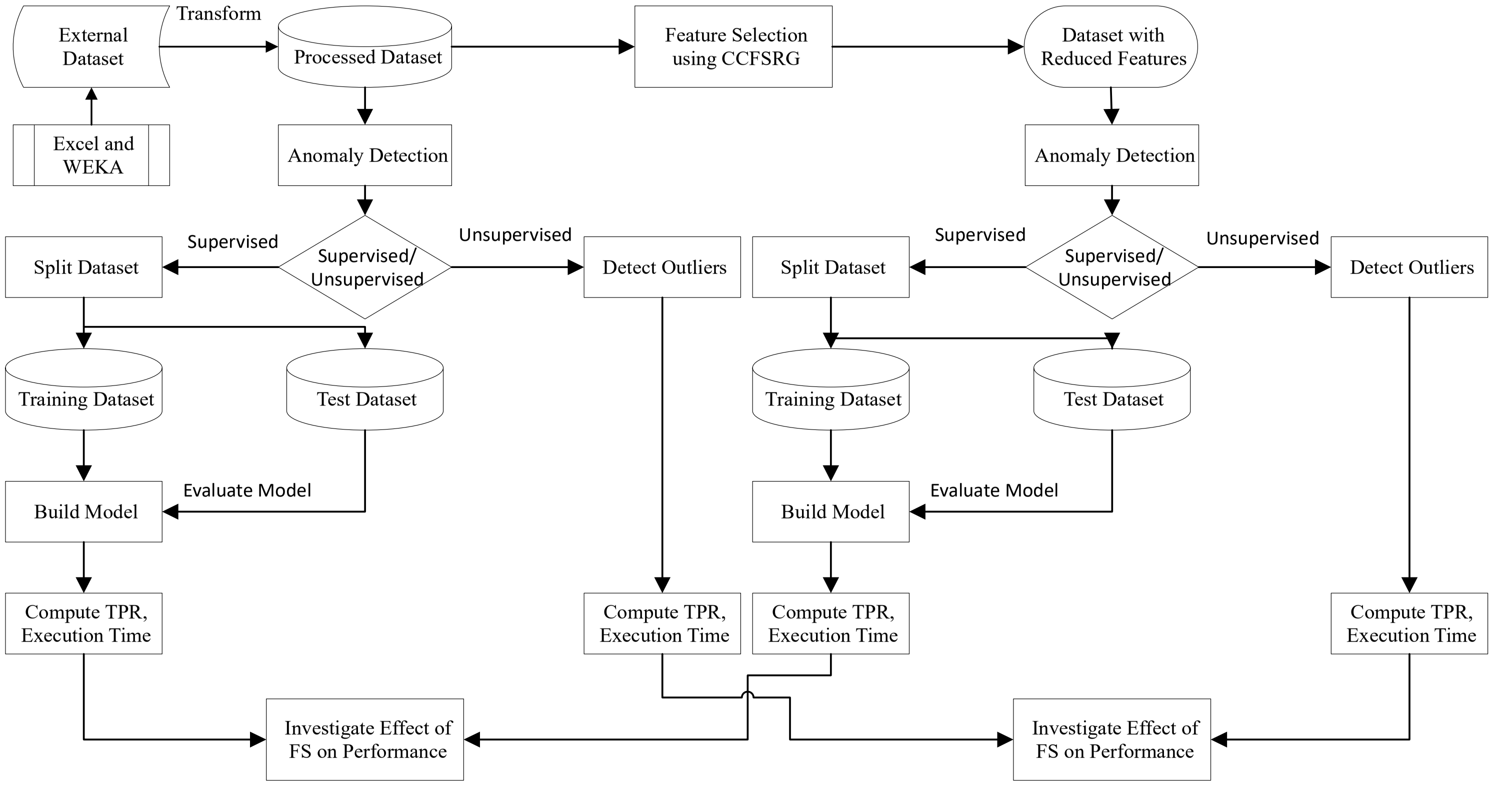


Fig. 5. Proposed feature selection-based anomaly detection approach (ADUFS)

between two features is linear or nonlinear [7]. Accordingly, the feature selection framework, CCFSRFG, with FRG as a decomposer, selects a suitable subset of features without considering correlation.

Each subdataset is represented using a subpopulation in CCFSRFG. Here, s is the number of features in each individual (i.e., s features of a subdataset). Consider the size of each subpopulation (sp) is sz. An example of subpopulation sp_1 consisting sz individual can be the following:

$ind_1 = \{0, 1, 1, 0, \ldots, 1\}$

$ind_2 = \{1, 1, 1, 0, \ldots, 0\}$

$\ldots$

$ind_{sz} = \{0, 1, 1, 1, \ldots, 1\}$

1 in an individual indicates that the feature in the corresponding is selected for the feature subset selection; 0 indicates that the feature is not selected for the feature subset selection. An individual in any subpopulation is evaluated by joining collaborators (i.e., individuals) from other subpopulations. For example, to evaluate individual ind_1 in subpopulation sp_1, s.t. collaborator ind_3 from subpopulation sp_2 and collaborator ind_2 from subpopulation sp_3, these three individuals are joined together to build a complete solution for the dataset with a reduced number features. Consider a random decomposition of 9 features into three subpopulations ($s = 4$), is assumed with sp_1 $\{ind_1\}$=$\{f_3, f_9, f_7, f_2\}$, sp_2 $\{ind_2\}$=$\{f_6, f_1, f_5, f_8\}$, and sp_3 $\{ind_4\}$=$\{f_4\}$. If features $\{f_7, f_2\}$ from sp_1 $\{ind_1\}$, $\{f_1, f_5\}$ from sp_2 $\{ind_2\}$, and $\{f_4\}$ from sp_3 $\{ind_4\}$ are selected because of a binary (0 or 1) representation of features, the complete solution with sorted feature indices can be defined as follows:

$solution = \{f_1, f_2, f_4, f_5, f_7\}$

The solution with this reduced number of features is then evaluated by the ML classifiers to measure accuracy, sensitivity, and specificity performance. The best individual with a reduced number of features and the highest classification accuracy is achieved by a penalty-based wrapper objective function introduced in the CCEAFS approach [39]. The objective function for this is defined as

$$f = w_1 * f_1 - w_2 * f_2 \tag{1}$$

where $f_1 = T_c/T$ and $f_2 = S/N$.

f is the overall objective function;

w_1 and w_2 are two control parameters for the objective functions f_1 and f_2, which are used to adjust the penalty term for f_1 and f_2, with $w_1 + w_2 = 1$;

T is the total number of test or train samples in the dataset (the test or train samples depend on the classification mode of using cross-validation or the supplied test set);

T_c is the number of correctly classified instances in the test or train samples;

S is the number of features selected in the subset; and

N is the total number of features in the dataset.

In the first generation of CCFSRFG, when there was no previous information available, random collaborators (i.e., individuals) from other subpopulations were used to build a complete solution that evaluated an individual in a subpopulation. The best individuals from other subpopulations are used as collaborators from Generation 1 onwards. The process continues until it reaches a fixed number of generations or until no better fitness is achieved over the generations.

Once the FS, i.e., CCFSRFG, was applied to the datasets, the datasets with a reduced number of features were used to detect anomalies using both supervised and unsupervised anomaly detection approaches. Similar to the supervised and unsupervised anomaly detection approaches with original datasets, the same steps were used with the datasets with a reduced number of features obtained through the FS using CCFSRFG. The performance of FS on datasets for anomaly detection was investigated and compared between the original datasets and datasets with a subset of features.

The five cybersecurity datasets used in this paper have different complexities that range from a different number of features and instances. To validate the effectiveness of the application of FS approach in cybersecurity datasets in detecting anomalies efficiently, all datasets were preprocessed with a reduced number of randomized instances (1,000 instances for each dataset), and the number of features remained the same as the original datasets. These datasets were then used to detect anomalies using supervised and unsupervised anomaly detection approaches similar to the process using original datasets. After that, CCFSRFG was applied to reduce the number of features from the datasets with a reduced number of instances. These datasets with feature selection were used to detect anomalies using both supervised and unsupervised anomaly detection approaches. Similar to the aforementioned performance comparison, the performance of FS on datasets for anomaly detection was investigated and compared between the datasets with a reduced number of instances and the datasets with a reduced number of instances and a reduced number of features. Algorithms 1 and 2 are the pseudocodes of the proposed ADUFS approach using supervised anomaly detection techniques and unsupervised anomaly detection techniques, respectively. A JAVA-based implementation of ADUFS is available at GitHub.[7]

4 RESULTS AND DISCUSSIONS

Experimental results are included in this Section and analyzed with and without feature selection approaches.

4.1 Dataset Details

The datasets used in the experiments is listed in Table 2 with normal and anomaly data distribution.

Table 2. **Distribution of normal and anomalous data. The Windows 10 and Windows 7 datasets belong to the TON_IoT dataset.**

Dataset	Normal (%)	Anomalous (%)	No. of instances	No. of features
Windows 10[8]	69.13	30.87	35,975	125
Windows 7[9]	78.92	21.08	28,367	133
UNSW_NB15[10]	44.94	55.06	82,332	42
NSL-KDD[11]	51.88	48.12	148,517	41
KDD99[12]	19.69	80.31	494,020	41

[7] https://github.com/bazlurrashid/cooperative-coevolution/tree/ADUFS/

Algorithm 1 ADUFS-Supervised

```
Input: *.ARFF formatted dataset files;
Output: TPR, FPR, and ET;
Read the training dataset and set the class index;
Count the number of classes in the training data into $numClasses$;
for $x = 1$ to $numClasses$ do
    Store the class value into $classValue[x]$;
end for
Define the classifier;
Build the classifier with the training dataset;
Read the unlabeled test dataset and set the class index;
Compute the number of unlabeled test instances into $numTestInstances$;
Read the ground truth dataset having only an anomaly class;
Count the number of anomaly instances in the ground truth data and store into $anomalyInstances$;
Read the labeled test dataset and set the class index;
for $x = 1$ to $numTestInstances$ do
    Read and store the class value of labeled test instance into $clsLabel$;
    Read and store the class label of labeled test instance into $actual[x]$;
    Classify the unlabeled instances and store the value into $preClassification$;
    Label the instances for the value stored in $preClassification$;
    Store the string value of labeled instances into $predString[x]$;
    if $actual[x] == predString[x]$ then
        Increase the value of $correct$ by 1;
    else
        Increase the value of $incorrect$ by 1;
    end if
end for
for $x = 1$ to $numClasses$ do
    Store the $classValue[x]$ into $testClass$;
    for $y = 1$ to $numTestInstances$ do
        if $testClass == actual[y]$ AND $actual[y] == predString[y]$ then
            Increase the value of $classCorrect[i]$ by 1;
        else if $testClass == actual[y]$ AND $actual[y]! = predString[y]$ then
            Increase the value of $classIncorrect[i]$ by 1;
        else
            // Do nothing here.
        end if
    end for
end for
for $x = 1$ to $numClasses$ do
    Compute $pctClassCorrect[x] = classCorrect[x]/numTestInstances * 100$;
    Compute $pctClassIncorrect[x] = classIncorrect[x]/numTestInstances * 100$;
end for
Compute $pctCorrect = Correct/numTestInstances * 100$;
Compute $pctIncorrect = inCorrect/numTestInstances * 100$;
Compute $TP = classCorrect[0]; FN = classIncorrect[0]$;
Compute $TN = classCorrect[1]; FP = classIncorrect[1]$;
Compute $TPR = (TP/(TP + FN)); FPR = (FP/(FP + TN))$;
Compute $TNR = (TN/(FP + TN)); FNR = (FN/(TP + FN))$;
Compute $TPR\_Anomalies = TN/anomalyInstances$;
Display TPR, FPR, and ET;
```

[8] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-ton-iot-Datasets/

[9] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-ton-iot-Datasets/

[10] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Algorithm 2 ADUFS-Unsupervised

```
Input: *.CSV formatted dataset files;
Output: TPR, FPR, and ET;
Using RapidMiner to compute the anomaly scores;
Sort the anomaly scores in descending order;
Separate the top instances based on the actual anomalies in the ground truth and store in a CSV file;
while Read the CSV files of actual anomalies and anomalies obtained in previous step until end do
    Split the read lines into columns and store the first column's values into $gInstances[x]$ and $outInstances[x]$;
    Increase the value of $x$ by 1;
end while
Compute the number of anomaly instances from both CSV files and store into $gSize$ and $outSize$, respectively;
Store the value of $outSize - 1$ into $nums$ array;
for $x = 1$ to $gSize$ do
    for $y = 1$ to length of $nums$ do
        if $gInstances[x] == outInstances[nums[y]]$ then
            Increase the value of $correct$ by 1;
            Remove index $y$ from the $nums$ array;
            Jump the execution to the inner loop to continue checking with other index values;
        end if
    end for
end for
Assign the size of $outInstances$ into $anomalies$;
Compute $inCorrect = anomalies - correct$;
Compute $TPR = correct/anomalies$;
Display TPR, FPR, and ET;
```

The five cybersecurity datasets have been used with increasing complexities. The datasets have been selected with samples between 28,367 and 494,020 and a dimensionality between 41 and 133. The share of normal instances in the datasets ranges between 19.69% and 78.92%, while the share of anomalous instances ranges between 21.08% and 80.31%. Here the anomalous instances include the instances that are not normal. These are DDoS, DoS, Injection, MITM, Password, XSS, and Scanning in the Windows 10 dataset; DDoS, DoS, Backdoor, Injection, Password, XSS, Scanning, and Ransomware in Windows 7; Reconnaissance, Backdoor, DoS, Exploits, Analysis, Fuzzers, Worms, Shellcode, and Generic in UNSW_NB15; and Buffer_overflow, Loadmodule, Perl, Neptune, Smurf, Guess_passwd, Pd, Teardrop, Portsweep, IPsweep, Land, FTP_Write, Back, IMap, Satan, Phf, NMap, Multihop, Warezmaster, Warezclient, Spy, and Rootkit in the KDD99 dataset. In the NSL-KDD dataset, there are only two classes: normal and anomaly. As one of the objectives of the proposed anomaly detection approach is to validate the application of FS in detecting anomalies by a reduced number of instances (1,000 instances) for each dataset, Table 3 lists the same datasets with a reduced number of instances and corresponding ratios of normal and anomalous instances. These datasets, with a reduced number of instances, have been selected carefully to maintain a similar proportion of normal and anomalous instances in each dataset. Interested readers can find the dataset descriptions in the respective dataset repositories.

4.2 Unsupervised Anomaly Detection Parameters

The parameters used for different unsupervised anomaly detection techniques using *RapidMiner*[13] are described here. The maximum value of *k*, when required for different anomaly detection techniques, has been selected based

[11] https://www.unb.ca/cic/datasets/nsl.html
[12] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[13] https://rapidminer.com

Table 3. **Distribution of normal and anomalous data with a reduced number of instances.**

Dataset	Normal (%)	Anomalous (%)	No. of instances	No. of features
Windows 10	68.70	31.30	1,000	125
Windows 7	77.50	22.50	1,000	133
UNSW_NB15	43.00	57.00	1,000	42
NSL-KDD	52.40	47.60	1,000	41
KDD99	18.10	81.90	1,000	41

on the ceiling of the square root of the total number of instances in a dataset, while the minimum value was kept at 2. For example, if a dataset has 100 instances, the maximum value of k is 10. Mixed measures and mixed Euclidean Distance parameters were set for k-NN, LOF, COF, aLOCI, LoOp, INFLO, CBLOF, LDCOF, and CMGOS. For aLOCI, $difference\ of\ levels\ L = 4$, $tree\ depth(levels) = 10$, $number\ of\ grids = 20$, $n\ min = 20$; For CBLOF, $alpha = 90.0$, $beta = 5.0$; For LDCOF, $gamma = 0.1$; For CMGOS, $probability\ for\ normal\ class = 0.975$, $gamma = 0.1$, $covariance\ estimation = Reduction$, $times\ to\ remove\ outlier = 1$; For HBOS, $parameter\ mode = all$.

4.3 The CC Parameters

The common CC parameters used in the experiments are listed in Table 4.

Table 4. **The CC parameters details**

Phases	Options
Problem decomposition	Dynamic (RFG)
Subproblem evolution	GA
Collaboration and evaluation	1+N

The problem decomposition parameters used in the experiments are listed in Table 5.

Table 5. **The decomposition parameters used for the experiments for original datasets and datasets with a reduced number of instances.**

Name	No. of Subpopulation	Subpopulation size
Windows 10	3 [42, 42, 41]	30
Windows 7	3 [45, 45, 43]	30
UNSW_NB15	2[22, 20]	30
NSL-KDD	2[21, 20]	30
KDD99	2[21, 20]	30

The common GA parameters used in the experiments are listed in Table 6.

In the case of GA optimization, the binary representation of the population is used, in which a binary 1 indicates that a feature is selected, and a binary 0 indicates that a feature is not selected from the dataset. Subpopulations were initialized randomly at generation 0. For CCFSRFG, in generation 0, since there is no previous history, to evaluate an individual in a subpopulation, random collaboration has been performed to collaborate with individuals from other

Table 6. **The GA parameters details**

Parameters	**Value**
Individual representation	Binary (0 or 1)
Crossover rate	100%
Mutation rate	5%
Elitism	1
Selection strategy	Tournament selection

subpopulations. In the subsequent generations, the best individuals from the previous generation were used as the collaborators for evaluating an individual in a subpopulation. Collaboration performance, i.e., the fitness value was assigned to the individual being evaluated. The best individuals were combined from all subpopulations to obtain the best individual in a generation. To evaluate an individual in any subpopulation, the parameters for weightings w_1 and w_2 were set to 0.6 and 0.4, respectively. A maximum number of 10,000 generations were allowed as a terminating condition for the evolutionary process while verifying that there is no further improvement in the fitness value.

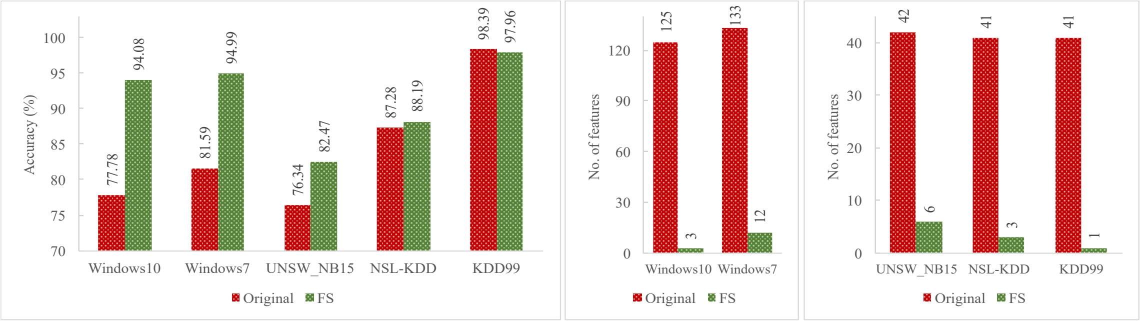
4.4 Experimental Results and Analysis

The first phase of our methodology is to apply the FS technique using CC in cybersecurity datasets to detect anomalies and compare the anomaly detection performances with and without FS. Hence, the FS method, CCFSRFG [38], proposed in our previous study using naïve Bayes classifier has been applied to the cybersecurity datasets listed in Table 2. The FS process performance was evaluated using cross-validation. Typically, multiple evaluation metrics are used to indicate the quality of a machine learning model. In this article, accuracy, true positive rate (TPR), false positive rate (FPR), and execution time (ET) have been used throughout the paper where required. A summary of the performance results of CCFSRFG is listed in Table 7 in terms of accuracy and number of features subset selection. The Table also lists the execution time of CCFSRFG of all datasets within an hour.

Table 7. **Summary of results for all datasets with and without FS using a naïve Bayes classifier.**

	Without FS		With FS		
Dataset	**ACC (%)**	**No. of features**	**ACC (%)**	**No. of features**	**Execution time (hour)**
Windows 10	77.78	125	94.08	3	18.96
Windows 7	81.59	133	94.99	12	13.69
UNSW_NB15	76.34	42	82.47	6	3.73
NSL-KDD	87.28	41	88.19	3	7.13
KDD99	98.39	41	97.96	1	22.93

From Table 7, it can be observed that CCFSRFG was able to select a suitable subset of features with a very low number of features compared to the original number of features in the dataset. With the exception of the KDD99 dataset, the classification accuracy was improved significantly for all other datasets. For the Windows 10, Windows 7, UNSW_NB15, and NSL-KDD datasets, the improvement in classification accuracy was 20.96%, 16.42%, 8.03%, and 1.04%, respectively. In the case of selecting a suitable feature subset that improves classification accuracy, the percentage of reduction observed for each dataset listed in order were 97.6%, 90.98%, 85.71%, 92.68%, and 97.56%, respectively. Although CCFSRFG was able to select a suitable subset of features with a reduced number of features for each dataset, this was at the cost of a slight reduction of 0.44% in classification accuracy compared to the original accuracy using all features in the KDD99 dataset. In terms of the execution time taken by each dataset for the FS process, the UNSW_NB15 dataset required the least amount of execution time, while KDD99 required the most. It can be seen that the execution time required for the feature selection process depends on the increasing complexities of the datasets used in the experiments. Here, the increasing complexities indicate either the increasing number of features or the increasing number of instances in the datasets. The performance results of FS on the original datasets are also displayed graphically in Fig. 6 to clearly show the improvements using feature selection.



(a) (b) (c)

Fig. 6. Performance evaluation of the NB classifier with and without FS on all datasets, (a) Accuracy (b) No. of features on Windows 10 and Windows 7 datasets (c) No. of features on UNSW_NB15, NSL-KDD, and KDD99 datasets.

The original datasets and the datasets with a reduced number of features are used to detect anomalies using supervised and unsupervised anomaly detection approaches. The experimental results of supervised anomaly detection techniques in terms of TPR (%) are summarized in Table 8. The values indicated with bold in Table 8 indicate the improvements in the application of FS in anomaly detection. Here, it can be observed that the highest TPRs (%) in detecting anomalies were 98.81, 99.14, 97.71, 79.51, and 99.65 for the original Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets by different supervised ML classifiers. The highest TPRs (%) in detecting anomalies by FS were 99.10, 99.01, 82.21, 94.30, and 97.86 for the Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets with the selected subset of features by different supervised ML classifiers.

The comparative performance results of anomaly detection using supervised anomaly detection techniques from the Table 8 is illustrated in Fig. 7 only where the application of FS in detecting anomalies improve the detection performance. It can be observed from Fig. 7 that for different datasets, different numbers of classifiers improve the detection performance. For the Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets, 8, 2, 2, 3, and 7 classifiers can improve the anomaly detection performance. In the case of Windows 10, the highest TPR was 99% (NB), and the lowest one was about 89% (MLP) by FS. For the next three datasets, the highest and the lowest TPR by FS were about 99% (J48) and 98% (RT), 66% (NB) and 58% (MLP), 94% (*k*-NN), and 60% (SVM), respectively. For KDD99, the highest and lowest TPR was 98% by RT and SVM, respectively. Furthermore, it can be observed that MLP was the most common classifier, which can improve the anomaly detection performance using FS in comparison to using original features except for the Windows 7 dataset. From the simulation results in Fig. 7, it can be concluded that the supervised

Table 8. **Summary of performance of the individual (supervised) anomaly detection techniques in terms of TPR (%) with and without FS for all datasets. "Ori" indicates TPR(%) without FS and "FS" indicates TPR(%) with FS.**

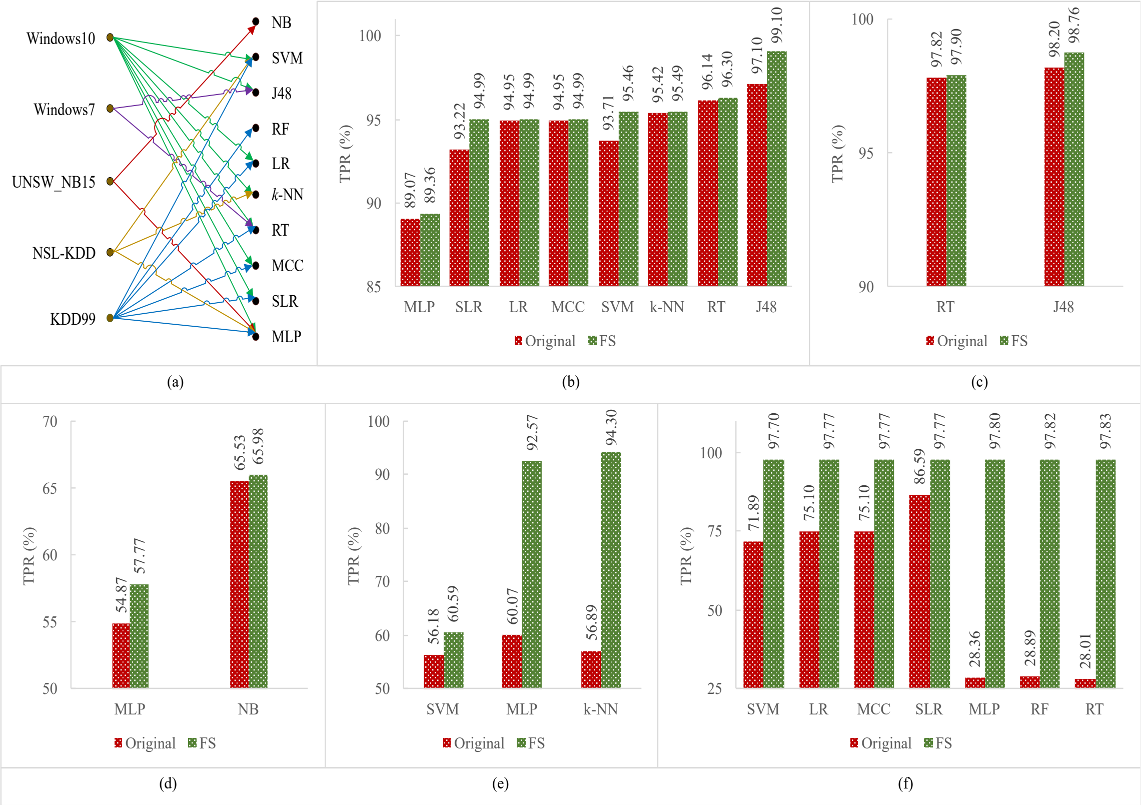
Classifier	**With or**	**Dataset**				
	Without FS	**Windows 10**	**Windows 7**	**UNSW_NB15**	**NSL-KDD**	**KDD99**
NB	Ori	96.81	86.21	65.53	65.18	98.02
	FS	94.63	81.46	**65.98**	60.80	97.75
SVM	Ori	93.71	85.22	67.14	56.18	71.89
	FS	**95.46**	72.68	57.98	**60.59**	**97.70**
J48	Ori	97.10	98.20	94.65	70.40	99.65
	FS	**99.10**	**98.76**	79.16	58.30	97.86
RF	Ori	98.81	99.14	97.71	65.26	28.89
	FS	97.93	99.01	56.12	58.31	**97.82**
LR	Ori	94.95	92.59	64.69	79.51	75.10
	FS	**94.99**	72.08	63.14	60.00	**97.77**
k-NN	Ori	95.42	94.43	72.93	56.89	98.99
	FS	**95.49**	58.51	63.29	**94.30**	97.83
RT	Ori	96.14	97.82	92.69	76.72	28.01
	FS	**96.30**	**97.90**	82.21	58.18	**97.83**
MCC	Ori	94.95	92.59	64.69	79.51	75.10
	FS	**94.99**	72.08	63.14	60.00	**97.77**
SLR	Ori	93.22	86.60	83.18	75.08	86.59
	FS	**94.99**	63.04	64.82	59.88	**97.77**
MLP	Ori	89.07	98.24	54.87	60.07	28.36
	FS	**89.36**	58.77	**57.77**	**92.57**	**97.80**

anomaly detection techniques perform equally good for the Windows 10 and KDD99 datasets for most of the classifiers when using feature selection.

The summary of timing performance of the individual supervised anomaly detection techniques in terms of execution time in second with and without FS for all original datasets is listed in Table 8.

The average of the execution time of the individual supervised anomaly detection techniques on all original datasets with and without FS is shown in Fig. 8. As it is expected to take reduced computing time to detect anomalies from all the original datasets with FS, the simulation results indicate the same. It can be observed that the highest amount of average time required for anomaly detection was by MLP, and the lowest amount of average time was required by RT when using original datasets. When FS was used, the highest time was required by MLP similar to the original datasets; however, the lowest one was required by the NB classifier. In summary, from the Fig. 8 (C), it can be clearly observed that a 68.29% decrease in average time was required for all datasets when using FS in comparison to the original datasets.

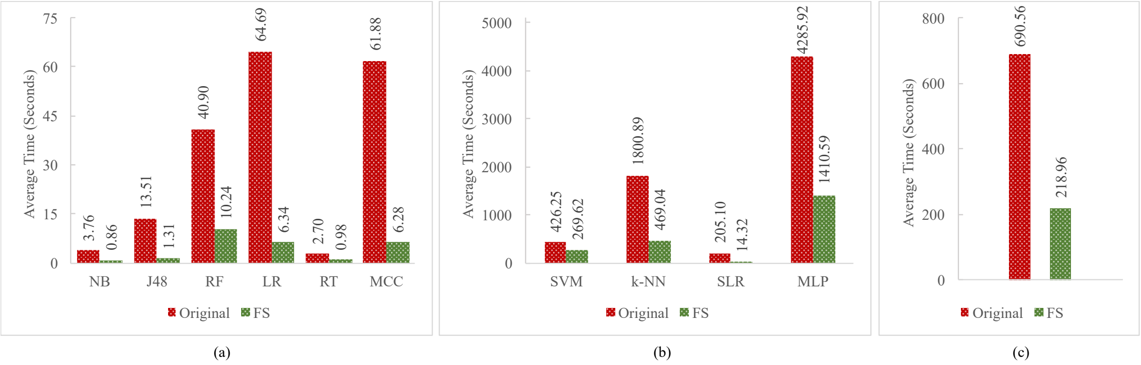
The experimental results of unsupervised anomaly detection techniques in terms of TPR (%) and ET (second) are summarized in Table 10 and 11. The values with bold text in Table 10 indicate the improvements in the application of FS in anomaly detection for all datasets. Here, it can be observed that the highest TPRs in detecting anomalies were



(a) (b) (c)

(d) (e) (f)

Fig. 7. Performance of the individual (supervised) anomaly detection techniques with and without FS on all datasets, (a) Performance improvement by FS for all dataset, (b) TPR for Windows 10, (c) TPR for Windows 7, (d) TPR for UNSW_NB15, (e) TPR for NSL-KDD, and (f) TPR for KDD99.



(a) (b) (c)

Fig. 8. Timing performance of the individual (supervised) anomaly detection techniques with and without FS on all datasets (a) NB, J48, RF, LR, RT, and MCC, (b) SVM, *k*-NN, SLR, and MLP, (c) Anomaly detection (supervised) timing performance comparison with and without FS on all datasets.

Table 9. **Summary of timing performance of the individual (supervised) anomaly detection techniques in terms of execution time (second) with and without FS for all datasets. "Ori" indicates execution time (second) without FS and "FS" indicates execution time (second) with FS.**

Classifier	With or Without FS	Dataset				
		Windows 10	Windows 7	UNSW_NB15	NSL-KDD	KDD99
NB	Ori	3.24	2.45	2.37	2.32	8.44
	FS	0.64	0.73	0.85	0.76	1.31
SVM	Ori	27.22	23.45	268.02	1,251.00	561.54
	FS	1.56	5.45	1,014.00	324.96	2.11
J48	Ori	7.78	4.82	5.28	19.78	29.88
	FS	0.83	1.03	1.62	1.34	1.71
RF	Ori	12.86	7.40	18.35	45.96	119.94
	FS	2.78	2.86	10.04	12.20	23.33
LR	Ori	29.63	8.12	60.66	71.16	153.90
	FS	0.84	1.04	24.01	2.14	3.66
k-NN	Ori	29.80	12.88	93.30	300.48	8,568.00
	FS	7.44	6.48	38.74	142.14	2,150.40
RT	Ori	2.38	1.61	1.72	2.16	5.64
	FS	0.62	0.73	0.94	0.92	1.67
MCC	Ori	27.75	8.06	60.72	72.42	140.46
	FS	0.88	1.03	23.56	2.10	3.83
SLR	Ori	45.41	46.65	236.04	128.70	568.68
	FS	2.25	2.39	28.01	11.07	27.90
MLP	Ori	985.20	810.00	4,644.00	4,503.60	10,486.80
	FS	4.90	15.49	2,481.60	26.47	47.34

about 50%(LDCOF), 60% (aLOCI), 70% (aLOCI), 89% (CBLOF, LDCOF), and 82% (INFLO) for all original datasets in order, respectively. On the other hand, the highest TPRs in detecting anomalies by FS were about 41% (CBLOF), 55% (*k*-NN), 60% (CBLOF, LDCOF), 78% (INFLO), and 85% (aLOCI) for all datasets in order with the selected subset of features, respectively.

The comparative performance results of anomaly detection using unsupervised anomaly detection techniques from Table 10 are illustrated in Fig. 9 one for the results when FS in detecting anomalies improve the detection performance compared to the original datasets. It can be observed that from Fig. 9 that a different number of anomaly detection techniques improve the detection performance for different datasets. For the Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets, 1, 9, 6, 4, and 5 anomaly detection techniques can improve the anomaly detection performance. In the case of the Windows 10 dataset, the TPR was above 30% (LOF). For the rest of the datasets, the highest and the lowest TPR by FS were about 55% (*k*-NN) and 22% (LoOP), 60% (LDCOF and CBLOF) and 45% (CMGOS), 78% (INFLO), and 35% (aLOCI), and 85% (aLOCI) and 75% (*k*-NN, COF, LoOP, and HBOS), respectively. Furthermore, it can be observed that LOF was the common anomaly detection technique, which can improve the detection performance when using feature selection for all datasets.

Table 10. **Summary of performance of the individual (unsupervised) anomaly detection techniques in terms of TPR (%) with and without FS for all datasets. "Ori" indicates TPR(%) without FS and "FS" indicates TPR(%) with FS.**

Technique	**With or**	**Dataset**				
	Without FS	**Windows 10**	**Windows 7**	**UNSW_NB15**	**NSL-KDD**	**KDD99**
k-NN	Ori	30.84	43.29	38.51	14.14	75.48
	FS	30.53	**55.25**	33.35	10.92	**75.48**
LOF	Ori	29.39	26.77	55.84	42.75	81.45
	FS	**30.55**	**37.98**	**56.43**	**63.32**	75.61
COF	Ori	33.17	23.81	54.21	45.25	75.48
	FS	26.26	**26.51**	**57.61**	15.87	**75.48**
aLOCI	Ori	32.62	59.73	69.99	8.97	75.48
	FS	28.69	31.82	18.38	**35.12**	**84.63**
LoOP	Ori	31.58	20.03	51.75	46.55	75.48
	FS	25.81	**22.46**	**57.60**	12.22	**75.48**
INFLO	Ori	30.94	25.94	59.64	45.72	81.66
	FS	28.90	**26.09**	56.15	**77.60**	75.61
CBLOF	Ori	47.16	5.23	41.14	89.15	77.34
	FS	41.16	**26.17**	**60.45**	38.70	75.49
LDCOF	Ori	50.20	9.06	40.70	89.15	77.34
	FS	10.83	**23.56**	**60.45**	38.01	75.74
CMGOS	Ori	20.79	11.59	36.84	27.35	75.68
	FS	15.03	**39.65**	**45.39**	**37.82**	75.57
HBOS	Ori	30.33	37.54	45.17	60.13	75.48
	FS	26.06	**43.96**	36.45	44.06	**75.48**

The summary of timing performance of the individual unsupervised anomaly detection techniques in terms of execution time is listed in Table 11.

The average execution times of the unsupervised anomaly detection techniques on all original datasets with and without FS is shown in Fig. 10. The simulation results confirm that the anomaly detection techniques take less time than the original datasets with feature selection. It can be observed that the highest amount of average time required for anomaly detection by COF and the lowest amount of average time was required by HBOS when using the original datasets. When FS was used, LOF requires the most time, and aLOCI the least. From Fig. 10 (c), it can be clearly observed that an 80.66% decrease in average time was required for all datasets when using FS in comparison to using the original datasets.

The second phase of the methodology is to apply the FS technique using CC in the cybersecurity datasets to detect anomalies in order to confirm the impact of FS with the same number of instances in each dataset. Hence, the FS method CCFSRFG using naïve Bayes classifier has been applied to the cybersecurity datasets listed in Table 3 with a reduced number of instances (1,000 instances). Similar to the previous phase, the FS process performance was evaluated using cross-validation. A summary of the performance results of CCFSRFG in the reduced number of instances datasets is

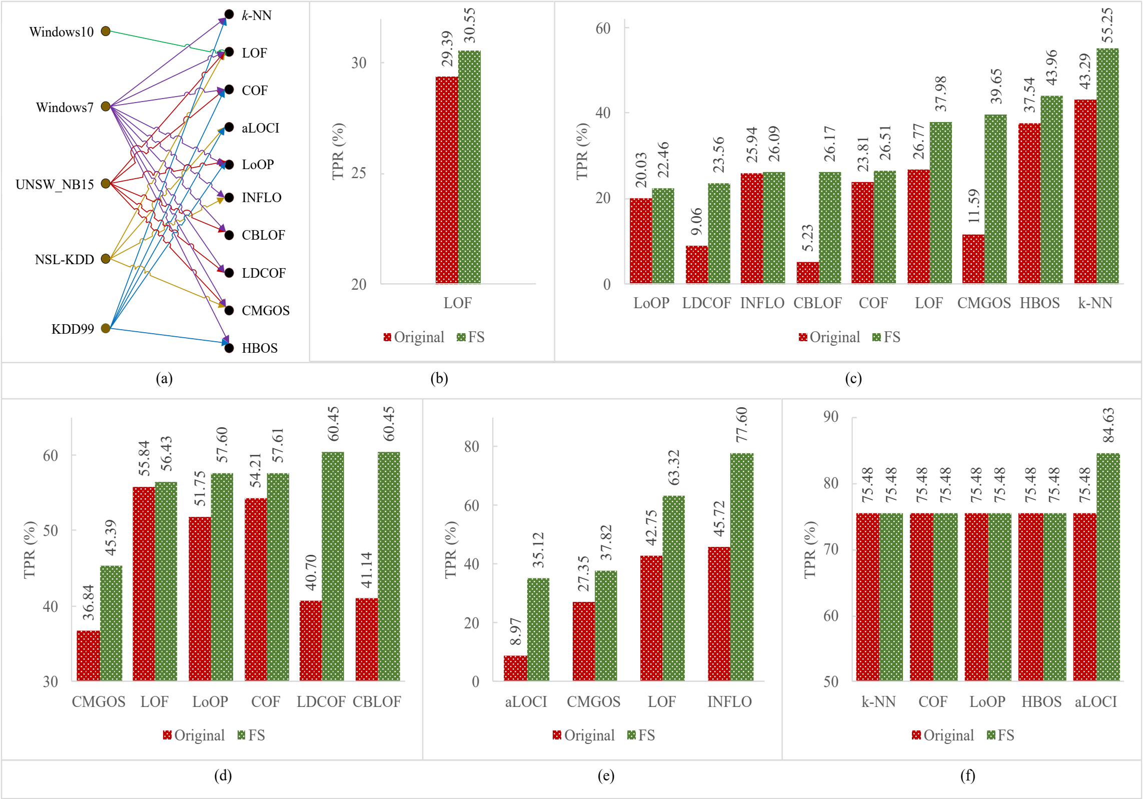


Fig. 9. Performance of the individual (unsupervised) anomaly detection techniques with and without FS on all datasets, (a) Performance improvement by FS for all dataset, (b) TPR for Windows 10, (c) TPR for Windows 7, (d) TPR for UNSW_NB15, (e) TPR for NSL-KDD, and (f) TPR for KDD99.

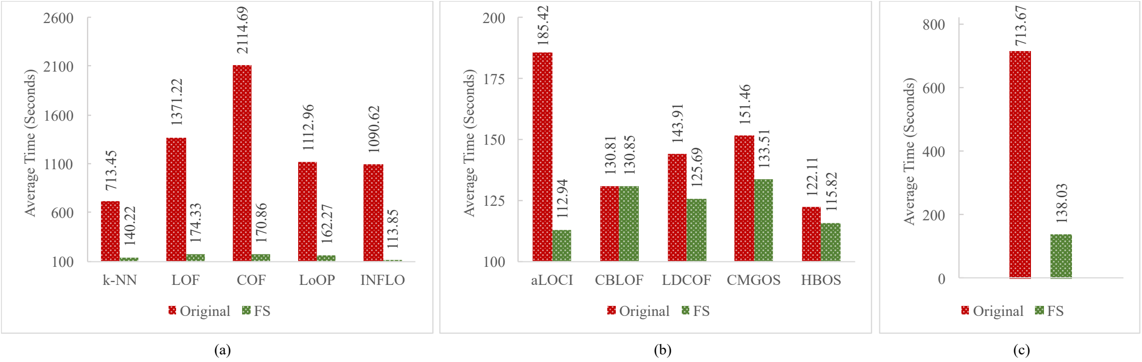


Fig. 10. Timing performance of the individual (unsupervised) anomaly detection techniques with and without FS on all datasets, (a) *k*-NN, LOF, COF, LoOP, and INFLO, (b) aLOCI, CBLOF, LDCOF, CMGOS, and HBOS, (c) Anomaly detection (unsupervised) timing performance comparison with and without FS on all datasets.

Table 11. **Summary of timing performance of the individual (unsupervised) anomaly detection techniques in terms of execution time (second) with and without FS for all datasets. "Ori" indicates execution time (second) without FS and "FS" indicates execution time (second) with FS.**

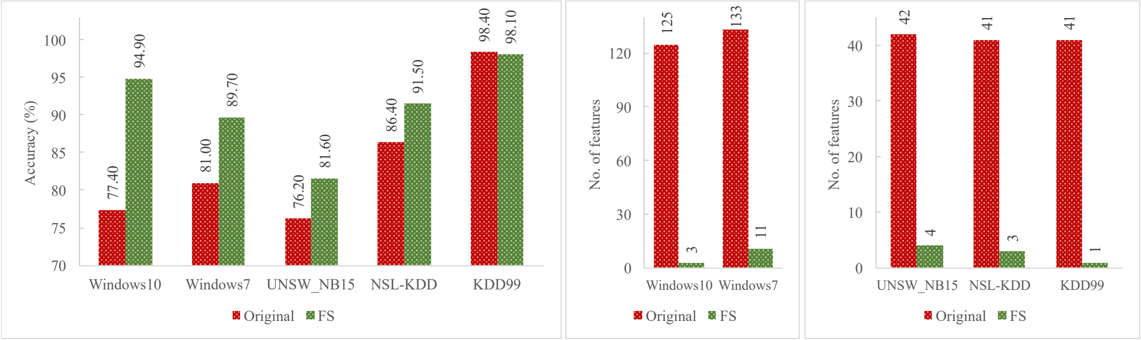
Technique	**With or Without FS**	**Dataset**				
		Windows 10	**Windows 7**	**UNSW_NB15**	**NSL-KDD**	**KDD99**
k-NN	Ori	215.91	153.43	317.32	2,35.55	2,645.06
	FS	79.99	29.39	112.45	27.84	451.44
LOF	Ori	225.89	122.44	298.88	2,430.32	3,778.58
	FS	83.76	32.41	147.20	17.52	590.76
COF	Ori	367.85	209.45	419.23	3,004.18	6,572.72
	FS	95.11	41.53	137.18	26.32	554.16
aLOCI	Ori	13.85	16.36	45.80	92.88	758.20
	FS	3.78	10.54	14.61	24.23	511.56
LoOP	Ori	241.86	137.44	260.47	1,969.43	2,955.62
	FS	77.80	29.95	103.33	26.32	573.96
INFLO	Ori	188.95	119.43	257.89	1,969.93	2,916.92
	FS	61.73	32.56	102.42	8.52	364.02
CBLOF	Ori	6.79	9.46	21.78	23.76	592.26
	FS	0.82	0.47	8.63	19.11	625.20
LDCOF	Ori	6.75	9.47	21.78	25.13	656.40
	FS	0.85	0.49	8.18	20.65	598.26
CMGOS	Ori	19.06	11.49	25.98	41.58	659.20
	FS	0.80	1.45	9.09	22.22	634.00
HBOS	Ori	1.88	1.43	8.36	19.68	579.22
	FS	0.78	0.46	10.43	20.93	546.48

listed in Table 12 in terms of accuracy and the number of features. The table also lists the execution time of CCFSRFG of all datasets.

Table 12. **Summary of results for all datasets having a reduced number of instances with and without FS using a naïve Bayes classifier.**

	Without FS		With FS		
Dataset	**ACC (%)**	**No. of features**	**ACC (%)**	**No. of features**	**Execution time (minute)**
Windows 10	77.40	125	94.90	3	21.75
Windows 7	81.00	133	89.70	11	19.82
UNSW_NB15	76.20	42	81.60	4	2.25
NSL-KDD	86.40	41	91.50	3	2.75
KDD99	98.40	41	98.10	1	1.97

From Table 12, it can be observed that CCFSRFG was able to select a suitable subset of features similar to the first phase with a very low number of features. Similar to the first phase—with the exception of the KDD99 dataset—the classification accuracy was improved significantly for all datasets. For the Windows 10, Windows 7, UNSW_NB15, and NSL-KDD datasets, the percentage of improvement in classification accuracy was 22.61%, 10.74%, 7.09%, and 5.89%, respectively. In selecting a suitable feature subset that improves classification accuracy, the reduction observed for each dataset listed in order were 97.60%, 91.73%, 90.48%, 92.68%, and 97.56%, respectively. Although CCFSRFG was able to select a suitable subset of features with a reduced number of features in each dataset, this came at the cost of a very slight reduction in classification accuracy (0.30%) compared to the original accuracy using all features in the KDD99 dataset. It was similar to the first phase of the methodology. The feature selection process is very time-consuming. This section's objective was to see the impacts of the FS process on each dataset with an equal number of instances. It can be observed that the execution time required for the FS process depends on the increasing complexities of each dataset, i.e., here it depends on the number of features in the original dataset. To clearly show the performance improvements on each dataset by a FS process on an equal number of instances, the results are also displayed in Fig. 11.



(a) (b) (c)

Fig. 11. Performance evaluation of the NB classifier with and without FS on all datasets with a reduced number of instances, (a) Accuracy, (b) No. of features on the Windows 10 and Windows 7 datasets, and (c) No. of features on the UNSW_NB15, NSL-KDD, and KDD99 datasets.

As this stage, the methodology aims to investigate the impacts of FS on an equal number of instances in each dataset, the datasets with the reduced number of instances (1,000 instances) were experimented to detect anomalies again using both supervised and unsupervised anomaly detection approaches. The experimental results of the supervised anomaly detection techniques in terms of TPR (%) are summarized in Table 13. The values with bold texts indicate the improvements in anomaly detection when using feature selection. Here, it can be observed that a larger number of supervised anomaly detection techniques were able to improve the detection performance using FS for four datasets. For Windows 7, only one algorithm (*k*-NN) could improve detection performance with FS compared two algorithms (J48 and RT) with FS using the original Windows 7 dataset. The highest TPRs (%) reported by any anomaly detection technique for each dataset in order when using FS was 97.35 (SVM, J48, LR, MCC, and SLR), 92.55 (*k*-NN), 96.40 (NB), 92.19 (LR, MCC), and 98.13 (LR and MCC), respectively.

The comparative performance results of supervised anomaly detection techniques on the datasets with a reduced number of instances are displayed in Fig. 12 only when FS can improve the detection performance for any dataset. It can be observed from the simulation that an increased number of supervised anomaly detection techniques can

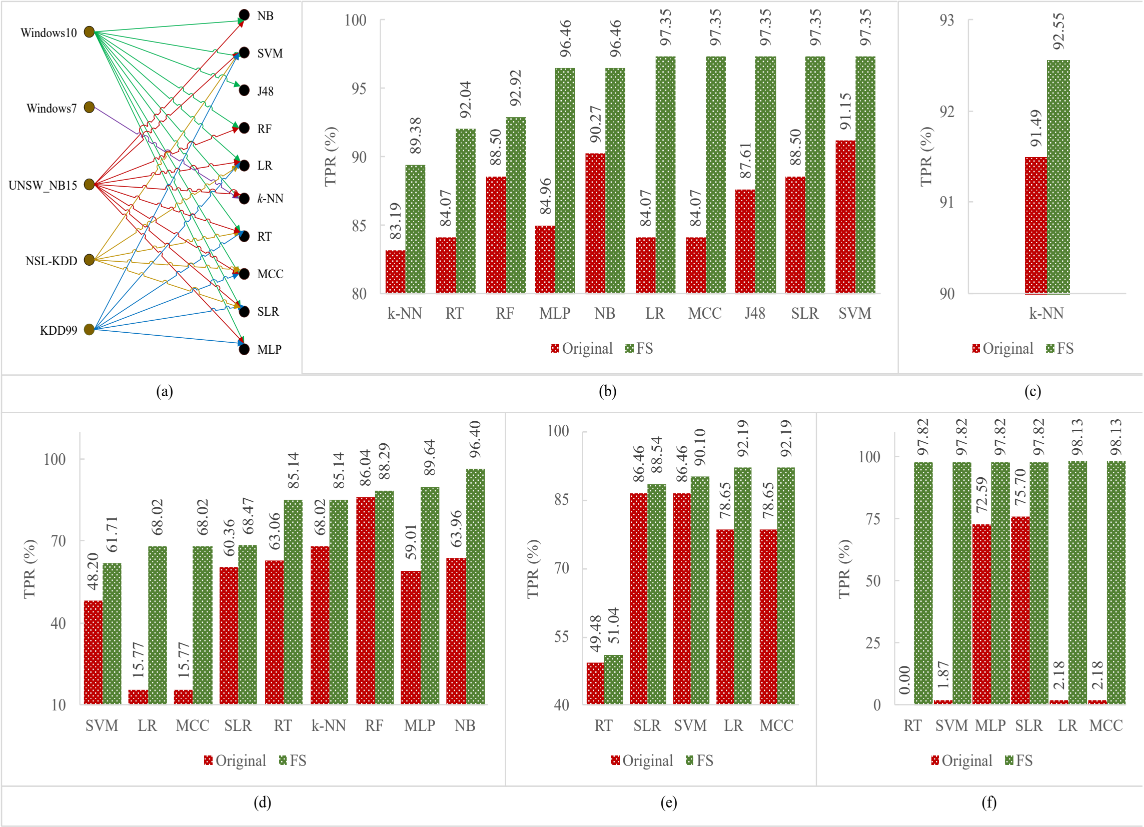
Table 13. **Summary of performance of the individual (supervised) anomaly detection techniques in terms of TPR (%) with and without FS for all datasets with a reduced number of instances. "Ori" indicates TPR(%) without FS and "FS" indicates TPR(%) with FS.**

Classifier	With or	Dataset				
	Without FS	Windows 10	Windows 7	UNSW_NB15	NSL-KDD	KDD99
NB	Ori	90.27	85.11	63.96	89.06	98.44
	FS	**96.46**	70.21	**96.40**	79.17	98.13
SVM	Ori	91.15	79.79	48.20	86.46	1.87
	FS	**97.35**	10.64	**61.71**	**90.10**	**97.82**
J48	Ori	87.61	96.81	94.14	94.79	99.07
	FS	**97.35**	84.04	87.84	81.77	97.82
RF	Ori	88.50	90.43	86.04	95.83	98.75
	FS	**92.92**	89.36	**88.29**	60.94	97.82
LR	Ori	84.07	82.98	15.77	78.65	2.18
	FS	**97.35**	60.64	**68.02**	**92.19**	**98.13**
k-NN	Ori	83.19	91.49	68.02	89.58	99.69
	FS	**89.38**	**92.55**	**85.14**	76.04	97.82
RT	Ori	84.07	86.17	63.06	49.48	0.00
	FS	**92.04**	85.11	**85.14**	**51.04**	**97.82**
MCC	Ori	84.07	82.98	15.77	78.65	2.18
	FS	**97.35**	60.64	**68.02**	**92.19**	**98.13**
SLR	Ori	88.50	84.04	60.36	86.46	75.70
	FS	**97.35**	57.45	**68.47**	**88.54**	**97.82**
MLP	Ori	84.96	91.49	59.01	84.90	72.59
	FS	**96.46**	84.04	**89.64**	81.25	**97.82**

improve detection performance, which can be seen from the cases of four datasets (except for Windows 7) as compared to the original datasets, as illustrated in Fig. 7. It can be noted that a total of 9 classifiers improved the detection performance on the UNSW_NB15 dataset with 1,000 instances. In contrast, in the same dataset with all instances, only two classifiers improved detection performance when using feature selection. Furthermore, in the Windows 10 dataset with 1,000 instances, all supervised classifiers can improve anomaly detection performance, increasing two more classifiers' performance compared to the original Windows 10 dataset.

The summary of timing performance of the individual supervised anomaly detection techniques in terms of execution time with and without FS for all datasets with the reduced number of instances is listed in Table 14.

The average execution time of the individual supervised anomaly detection techniques on all datasets with the reduced number of instances with and without FS is displayed in Fig. 13. Similar to the original datasets, the time taken by each supervised detection technique is significantly lower with than without feature selection. Datasets with the reduced number of instances take an 82.95% decrease average time by FS compared to not using feature selection (Fig. 13 (c)). It can be observed that with an equal number of instances in each dataset, the average execution time was



(a) (b) (c)

(d) (e) (f)

Fig. 12. Performance of the individual (supervised) anomaly detection techniques with and without FS on all datasets with a reduced number of instances, (a) Performance improvement by FS for all dataset, (b) TPR for Windows 10, (c) TPR for Windows 7, (d) TPR for UNSW_NB15, (e) TPR for NSL-KDD, and (f) TPR for KDD99.

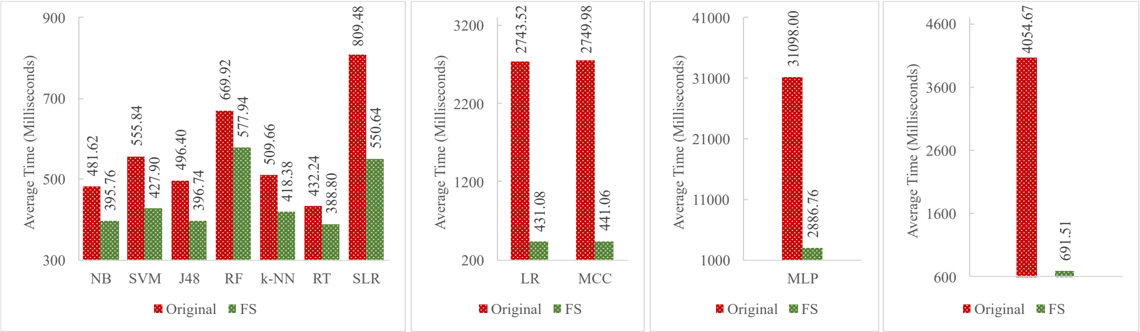
reduced by a 17.67% execution time (in the original dataset, the average time taken by FS approach was 68.29% decrease compared to not using feature selection from Fig. 8 (c)).

The experimental results of individual unsupervised anomaly detection techniques with and without FS in terms of TPR (%) are summarized in Table 15 for datasets with the reduced number of instances (1,000) each. The values with bold text indicate the improvements in detecting anomalies with FS for all datasets. It can be observed that there was no common algorithm that was able to improve anomaly detection performance with FS compared to not using FS for all datasets; however, different algorithms perform better for different datasets with FS. The highest TPRs (%) for anomaly detection by different datasets were 39.62 (CBLOF), 40.89 (LoOP), 65.26 (CBLOF, LDCOF), 74.58 (CMGOS), and 86.20 (aLOCI), respectively.

The comparative performance results of unsupervised anomaly detection techniques on the datasets with the reduced number of instances with and without FS process are illustrated in Fig. 14 only when the FS improved the detection performance over without FS. It can be observed that from Fig. 9 that a different number of anomaly detection techniques improve the detection performance for different datasets. For the Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets, 3, 8, 4, 6, and 5 anomaly detection techniques can improve the anomaly detection performance.

Table 14. **Summary of timing performance of the individual (supervised) anomaly detection techniques in terms of execution time (millisecond) with and without FS for all datasets with a reduced number of instances. "Ori" indicates execution time (millisecond) without FS and "FS" indicates execution time (millisecond) with FS.**

Classifier	**With or Without FS**	**Dataset**				
		Windows 10	**Windows 7**	**UNSW_NB15**	**NSL-KDD**	**KDD99**
NB	Ori	556.70	533.10	435.90	447.70	434.70
	FS	399.40	409.50	407.10	373.70	389.10
SVM	Ori	564.30	625.70	602.70	499.40	487.10
	FS	432.40	424.50	424.30	454.90	403.40
J48	Ori	546.60	603.70	461.20	444.20	426.30
	FS	403.00	424.30	396.40	375.30	384.70
RF	Ori	724.70	732.60	661.50	632.40	598.40
	FS	566.60	611.90	605.20	535.20	570.80
LR	Ori	912.40	2,221.00	4,107.00	5,961.00	516.20
	FS	415.40	435.70	418.90	458.20	427.20
k-NN	Ori	591.80	548.70	440.00	497.50	470.30
	FS	407.90	433.40	400.10	405.60	444.90
RT	Ori	476.20	486.00	412.00	384.10	402.90
	FS	387.10	398.50	393.60	381.40	383.40
MCC	Ori	949.00	2,292.00	4,085.00	5,890.00	533.90
	FS	424.90	446.30	437.80	482.70	413.60
SLR	Ori	752.60	858.30	887.00	894.30	655.20
	FS	531.30	605.00	507.00	587.90	522.00
MLP	Ori	22,470.00	35,680.00	54,630.00	22,250.00	20,460.00
	FS	556.50	832.70	605.60	11,930.00	509.00



(a) (b) (c) (d)

Fig. 13. Timing performance of the individual (supervised) anomaly detection techniques with and without FS on all datasets with a reduced number of instances, (a) NB, SVM, J48, RF, *k*-NN, RT, and SLR, (b) LR and MCC, (c) MLP, and (d) Anomaly detection (supervised) performance comparison with and without FS on all datasets with a reduced number of instances.

Table 15. **Summary of performance of the individual (unsupervised) anomaly detection techniques in terms of TPR (%) with and without FS for all datasets with a reduced number of instances. "Ori" indicates TPR(%) without FS and "FS" indicates TPR(%) with FS.**

Technique	With or	Dataset				
	Without FS	Windows 10	Windows 7	UNSW_NB15	NSL-KDD	KDD99
k-NN	Ori	29.71	43.56	41.40	15.97	77.90
	FS	23.96	28.89	**52.81**	**63.45**	**77.90**
LOF	Ori	28.12	33.78	61.58	44.96	82.05
	FS	26.84	**38.67**	48.77	**67.65**	**85.84**
COF	Ori	21.41	37.78	58.95	45.38	78.02
	FS	**23.00**	**38.67**	51.23	**64.29**	77.90
aLOCI	Ori	26.84	31.11	57.89	4.83	77.90
	FS	23.00	**39.11**	24.56	**72.90**	**86.20**
LoOP	Ori	24.60	30.67	55.79	47.48	77.90
	FS	**26.84**	**40.89**	47.19	**56.30**	**77.90**
INFLO	Ori	26.52	32.00	58.95	49.58	83.52
	FS	**28.75**	**36.00**	53.33	30.04	82.66
CBLOF	Ori	49.20	26.67	41.58	76.26	78.88
	FS	39.62	**30.22**	**65.26**	47.06	77.90
LDCOF	Ori	50.48	22.22	38.42	76.26	78.75
	FS	11.82	**23.11**	**65.26**	43.28	78.27
CMGOS	Ori	27.80	13.33	50.18	52.94	80.71
	FS	12.14	**18.22**	41.40	**74.58**	78.14
HBOS	Ori	30.35	38.67	57.72	59.24	77.90
	FS	12.46	29.78	**60.35**	21.85	**77.90**

Although more algorithms can improve the anomaly detection performance for the Windows 10 and NSL-KDD datasets, a drop in the number of algorithms was observed for Windows7 and UNSW_NB15 datasets. Furthermore, it can be observed that unlike the original datasets, here there was no common algorithm that could improve detection performance with FS for all datasets. However, it can be concluded that at least 3 anomaly detection techniques were able to improve detection performance with feature selection.

The summary of the timing performance of the individual unsupervised anomaly detection techniques in terms of execution time in a millisecond with and without FS for all datasets with the reduced number of instances is listed in Table 16.

The average timing performance of the individual unsupervised anomaly detection techniques using datasets with the reduced number of instances with and without FS is shown in Fig. 15. Likewise, with the original datasets, the unsupervised anomaly detection techniques within this case also consume a significantly lower computation, as expected. From Fig. 15 (b), it can be seen that a 9.03% decrease was observed in terms of average execution time with FS compared to without FS.

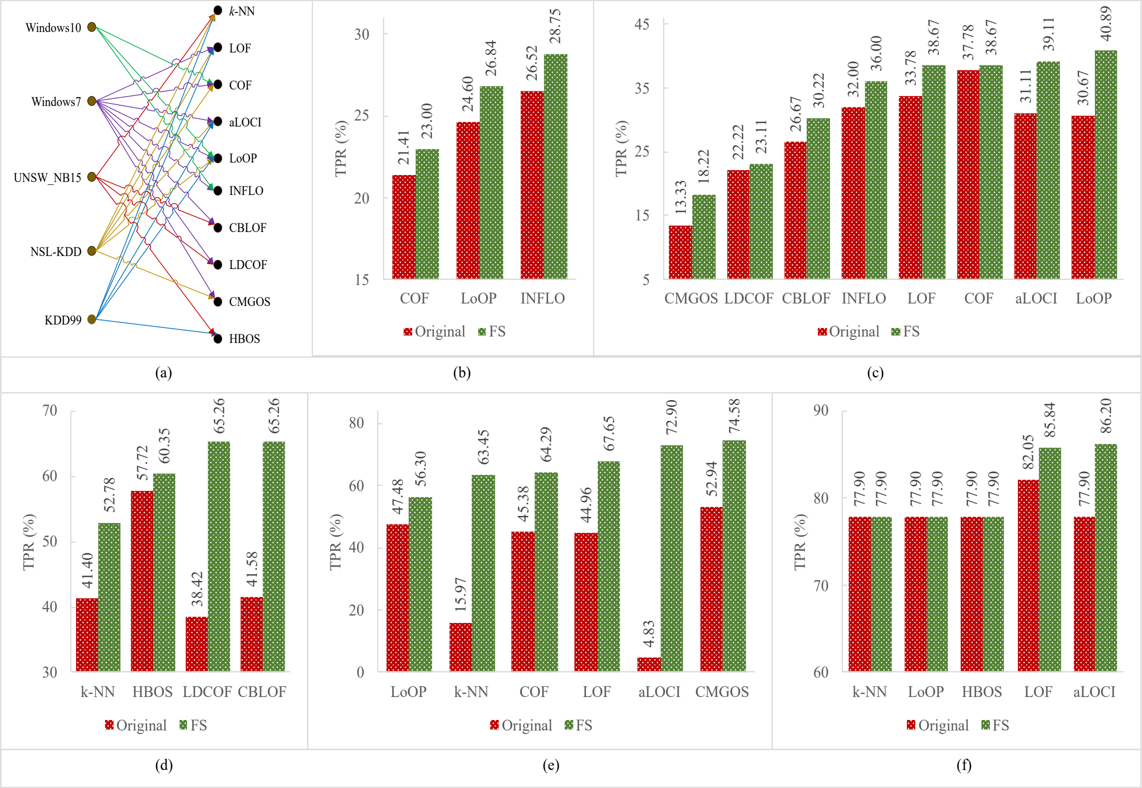
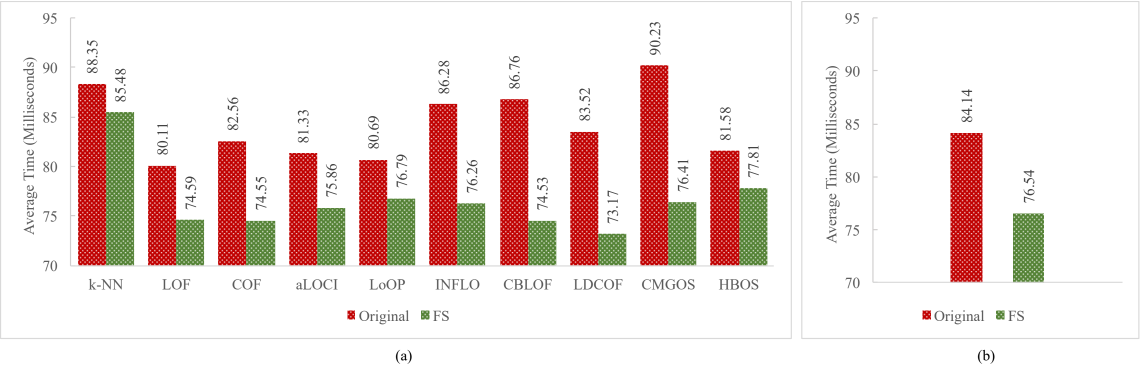


Fig. 14. Performance of the individual (unsupervised) anomaly detection techniques with and without FS on all datasets with a reduced number of instances, (a) Performance improvement by FS for all dataset, (b) TPR for Windows 10, (c) TPR for Windows 7, (d) TPR for UNSW_NB15, (e) TPR for NSL-KDD, and (f) TPR for KDD99.



(a)

(b)

Fig. 15. (a) Timing performance of the individual (unsupervised) anomaly detection techniques with and without FS on all datasets with a reduced number of instances, and (b) Anomaly detection (unsupervised) performance comparison with and without FS on all datasets with a reduced number of instances.

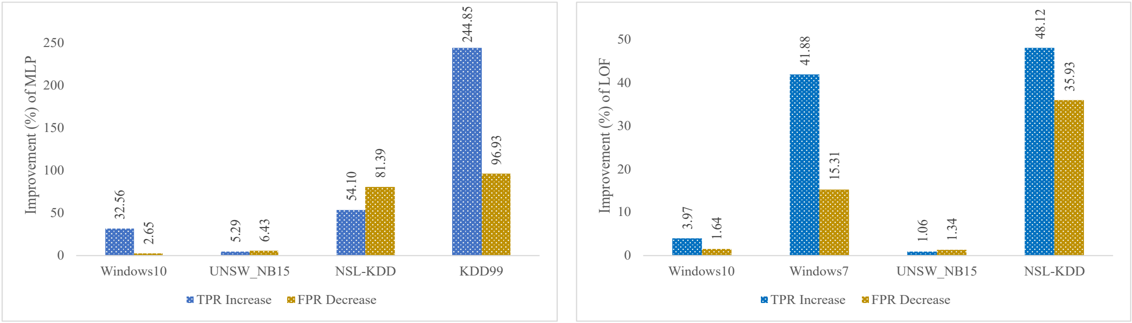
Table 16. **Summary of timing performance of the individual (unsupervised) anomaly detection techniques in terms of execution time (millisecond) with and without FS for all datasets with a reduced number of instances. "Ori" indicates execution time (millisecond) without FS and "FS" indicates execution time (millisecond) with FS.**

Technique	With or Without FS	Dataset				
		Windows 10	Windows 7	UNSW_NB15	NSL-KDD	KDD99
k-NN	Ori	88.11	76.27	97.78	82.96	96.65
	FS	87.25	82.73	98.18	71.21	88.01
LOF	Ori	75.41	66.82	82.81	81.54	93.99
	FS	75.89	65.75	87.27	65.87	78.17
COF	Ori	77.66	71.17	97.72	71.61	94.66
	FS	71.42	66.92	79.57	70.01	84.85
aLOCI	Ori	80.31	77.45	83.41	81.57	83.89
	FS	73.65	67.77	82.18	75.46	80.23
LoOP	Ori	80.76	68.08	81.87	82.10	90.66
	FS	68.14	75.08	81.69	75.44	83.61
INFLO	Ori	86.87	72.68	82.15	83.09	106.60
	FS	67.84	67.18	90.08	73.36	82.82
CBLOF	Ori	83.50	95.54	93.79	72.37	88.62
	FS	68.61	67.60	82.84	73.18	80.42
LDCOF	Ori	83.53	72.80	91.21	72.86	97.21
	FS	69.50	66.10	74.06	69.17	87.01
CMGOS	Ori	109.60	77.95	96.78	77.84	89.00
	FS	75.72	64.81	86.06	68.51	86.94
HBOS	Ori	80.96	70.91	90.51	76.10	89.44
	FS	71.61	78.59	75.83	76.77	86.27

4.5 Key Improvements

From the experimental results and analysis (Fig. 7 and 9), it can be observed that in both cases of supervised and unsupervised anomaly detection techniques, there was at least one technique that was able to improve the anomaly detection performance in terms of TPR and decreased FPR (false positive rate). These two techniques were MLP in the supervised detection approach and LOF in the unsupervised detection approach.

The key improvements with feature selection when evaluated by both supervised and unsupervised anomaly detection approaches for at least one technique each in four datasets are displayed in Fig. 16. The simulation shows that TPR was significantly increased by over 200% for KDD99, and FPR was decreased by about 97% compared to the original data by supervised anomaly detection (MLP) with FS. Apart from KDD99, for the Windows 10 and NSL-KDD datasets, TPR was increased at a higher rate; however, for UNSW_NB15, the TPR improvement was about 5% only. Although FPR was decreased significantly for the NSL-KDD and KDD99 datasets, the FPR decrease rate for Windows 10 and UNSW_NB15 was only above 2% and 6%, respectively. In the case of unsupervised anomaly detection approach, LOF improved TPR for Windows 7 and NSL-KDD both by more than 40%, and FPR was decreased by over 15% and about 36%, respectively.



(a)

(b)

Fig. 16. Performance improvement by supervised and unsupervised anomaly detection approaches by FS for at least one technique each in four datasets: (a) MLP, and (b) LOF.

Although an improvement in TPR and a decrease in FPR were observed for the Windows 10 and UNSW_NB15 datasets; however, the rate was between 1% and 4%.

4.6 Comparison with the State-of-the-Art Algorithms

The proposed anomaly detection approach, ADUFS, has been compared with six state-of-the-art techniques. The compared algorithms are Cyber intrusion detection by feature grouping based on linear correlation coefficient (FGLCC) [26], Wrappers for feature subset selection (WrapperSubsetEval), evaluate attribute subsets using a classifier to estimate the merit of the subset (ClassifierSubsetEval) and evaluate the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them (CfsSubsetEval). FGLCC is a recent feature selection approach used to detect cyber intrusion by Mohammadi et al. in 2019. In this paper, FGLCC has been implemented based on the algorithms in [26]. A JAVA-based implementation of ADUFS is available at GitHub.[14] FGLCC selected feature subsets in three different ways. First, FGLCC has been experimented on each dataset to obtain the same number of features as retrieved by CCFSRFG from each dataset. For example, CCFSRFG selected 6 features from the UNSW_NB15 dataset. Similarly, FGLCC also selected 6 features from the same dataset. However, the features selected by CCFSRFG and FGLCC are different as illustrated in Table 19. Second, FGLCC selected a fixed number of features: 10 and 15 from each dataset; likewise, the experiment results in [26] except for KDD99 because feature selection using FGLCC requires at least two clusters. The rest three feature subset selection approaches WrapperSubsetEval (WSE), ClassififerSubsetEval(CSE), and CfsSubsetEval(CFSE) have selected from WEKA [15]. These three methods are wrapper-based feature selection approaches using evolutionary computation similar to CCFSRFG. Experiments were performed using 10-fold cross-validation and testing with an independent dataset. All five datasets were split into 75% and 25%. The first 75% were used to cross-validate the model, and the independent 25% were used to test the model's performance in terms of true positive rate, false positive rate, and accuracy. For the feature selection by WEKA methods, the naïve Bayes classifier and best search strategy were used. Performance of all methods were evaluated based a decision tree (J48) classifier. The selected subset of features from each dataset by different feature selection techniques are listed in Tables 17-21.

[14] https://github.com/bazlurrashid/cooperative-coevolution/tree/ADUFS/

[15] (https://www.cs.waikato.ac.nz/ml/weka/)

Table 17. Selected features by different feature selection techniques on Windows 10 dataset.

Methods	No. of Features	Selected Features
CCFSRFG [38]	3	f_1, f_{76}, f_{106}
FGLCC$_3$ [26]	3	f_1, f_{65}, f_{124}
FGLCC$_{10}$ [26]	10	$f_1, f_{17}, f_{24}, f_{30}, f_{39}, f_{44}, f_{45}, f_{65}, f_{93}, f_{124}$
FGLCC$_{15}$ [26]	15	$f_1, f_{17}, f_{24}, f_{30}, f_{34}, f_{39}, f_{44}, f_{45}, f_{65}, f_{80}, f_{84}, f_{88}, f_{93}, f_{101}, f_{124}$
WSE	4	$f_1, f_{67}, f_{71}, f_{90}$
CSE	4	$f_1, f_{67}, f_{71}, f_{90}$
CFSE	3	f_1, f_{70}, f_{86}

Table 18. Selected features by different feature selection techniques on Windows 7 dataset.

Methods	No. of Features	Selected Features
CCFSRFG [38]	12	$f_5, f_{11}, f_{25}, f_{30}, f_{31}, f_{101}, f_{104}, f_{115}, f_{116}, f_{118}, f_{125}, f_{130}$
FGLCC$_{12}$ [26]	12	$f_1, f_3, f_8, f_{19}, f_{23}, f_{34}, f_{97}, f_{102}, f_{106}, f_{121}, f_{125}, f_{132}$
FGLCC$_{10}$ [26]	10	$f_1, f_{19}, f_{20}, f_{23}, f_{34}, f_{97}, f_{102}, f_{121}, f_{125}, f_{132}$
FGLCC$_{15}$ [26]	15	$f_1, f_8, f_{10}, f_{19}, f_{20}, f_{23}, f_{34}, f_{97}, f_{102}, f_{106}, f_{110}, f_{116}, f_{121}, f_{125}, f_{132}$
WSE	8	$f_1, f_{19}, f_{24}, f_{34}, f_{120}, f_{122}, f_{124}, f_{130}$
CSE	7	$f_1, f_{19}, f_{24}, f_{34}, f_{42}, f_{120}, f_{130}$
CFSE	1	f_1

Table 19. Selected features by different feature selection techniques on UNSW_15 dataset.

Methods	No. of Features	Selected Features
CCFSRFG [38]	6	$f_2, f_{13}, f_{14}, f_{25}, f_{32}, f_{41}$
FGLCC$_6$ [26]	6	$f_7, f_{13}, f_{23}, f_{26}, f_{35}, f_{38}$
FGLCC$_{10}$ [26]	10	$f_5, f_7, f_{12}, f_{13}, f_{23}, f_{26}, f_{34}, f_{35}, f_{37}, f_{38}$
FGLCC$_{15}$ [26]	15	$f_5, f_6, f_7, f_{12}, f_{13}, f_{16}, f_{23}, f_{24}, f_{25}, f_{26}, f_{34}, f_{35}, f_{37}, f_{38}, f_{39}$
WSE	6	$f_2, f_{12}, f_{25}, f_{32}, f_{39}, f_{42}$
CSE	5	$f_2, f_{12}, f_{25}, f_{32}, f_{39}$
CFSE	4	$f_7, f_{10}, f_{32}, f_{35}$

Tables 22-24 list compare the feature selection performance of all five datasets by all methods in terms of TPR, FPR, and ACC. It can be observed that the results obtained by different methods are quite similar and close to each other for Windows 10, Windows 7, and KDD99 datasets for all algorithms. It can be noted that the lowest FPR achieved by CCFSRFG for Windows 10 was 3.98%. Although TPR and FPR for the UNSW_NB dataset obtained by CCFSRFG are lower than FGLCC and CFSE, these are not less than by WSE and CSE methods. A close TPR obtained by CCFSRFG and FGLCC$_3$ for the NSL-KDD dataset. However, it can be noted that FGLCC$_3$ significantly increased FPR for both NSL-KDD and KDD99 datasets, resulting in an increased number of the wrong prediction by this algorithm from an inappropriate selection of features.

Table 20. Selected features by different feature selection techniques on NSL-KDD dataset.

Methods	No. of Features	Selected Features
CCFSRFG [38]	3	f_{26}, f_{27}, f_{37}
FGLCC$_3$ [26]	3	f_{20}, f_{32}, f_{37}
FGLCC$_{10}$ [26]	10	$f_7, f_{15}, f_{20}, f_{26}, f_{28}, f_{29}, f_{32}, f_{36}, f_{37}, f_{41}$
FGLCC$_{15}$ [26]	15	$f_2, f_7, f_{15}, f_{20}, f_{26}, f_{28}, f_{29}, f_{30}, f_{32}, f_{33}, f_{35}, f_{36}, f_{37}, f_{40}, f_{41}$
WSE	3	f_3, f_4, f_8
CSE	3	f_3, f_4, f_{11}, f_{13}
CFSE	8	$f_4, f_5, f_6, f_{12}, f_{26}, f_{29}, f_{30}, f_{37}$

Table 21. Selected features by different feature selection techniques on KDD99 dataset.

Methods	No. of Features	Selected Features
CCFSRFG [38]	1	f_{23}
FGLCC$_3$ [26]	3	f_{15}, f_{20}, f_{35}
FGLCC$_{10}$ [26]	10	$f_5, f_6, f_{15}, f_{20}, f_{27}, f_{29}, f_{34}, f_{35}, f_{39}, f_{40}$
FGLCC$_{15}$ [26]	15	$f_5, f_6, f_{15}, f_{20}, f_{26}, f_{27}, f_{28}, f_{29}, f_{32}, f_{34}, f_{35}, f_{38}, f_{39}, f_{40}, f_{41}$
WSE	12	$f_3, f_{10}, f_{23}, f_{27}, f_{28}, f_{31}, f_{32}, f_{34}, f_{36}, f_{39}, f_{40}, f_{41}$
CSE	13	$f_3, f_6, f_{10}, f_{23}, f_{27}, f_{28}, f_{31}, f_{32}, f_{34}, f_{36}, f_{39}, f_{40}, f_{41}$
CFSE	5	$f_6, f_{12}, f_{23}, f_{31}, f_{32}$

Table 22. Classification performance comparison of different feature selection techniques on Windows 10 and Windows 7 datasets.

Methods	Datasets					
	Windows 10			Windows 7		
	TPR (%)	FPR(%)	ACC(%)	TPR (%)	FPR(%)	ACC(%)
CCFSRFG [38]	99.51	**3.98**	98.38	99.59	2.16	99.24
FGLCC* [26]	99.83	4.24	98.50	99.86	2.53	99.35
FGLCC$_{10}$ [26]	99.54	4.23	98.31	99.73	2.61	99.24
FGLCC$_{15}$ [26]	99.52	4.24	98.30	99.79	2.60	99.28
WSE	99.55	4.56	98.21	99.79	2.15	99.38
CSE	99.55	4.56	98.21	99.79	2.15	99.38
CFSE	99.79	4.12	98.51	99.87	2.40	99.39

FGLCC* indicates FGLCC$_3$ and FGLCC$_{12}$ for Windows 10 and Windows 7 datsets, respectively.

Comparison of anomaly detection performance by different methods with feature selection techniques on all datasets is illustrated in Table 25.

The anomaly detection performance comparison in terms of TPR and FPR is illustrated in Fig. 17 and Fig. 18. It can be observed that the highest TPR achieved by different techniques are FGLCC$_3$, CFSE, FGLCC$_{15}$, FGLCC$_{15}$, and FGLCC$_{10}$ for Windows 10, Windows 7, UNSW_NB15, NSL-KDD, and KDD99 datasets, respectively. However, the lowest FPR

Table 23. Classification performance comparison of different feature selection techniques on UNSW_NB15 dataset.

Methods	TPR (%)	FPR(%)	ACC(%)
CCFSRFG [38]	73.84	12.73	80.09
FGLCC_6 [26]	88.01	6.46	90.90
FGLCC_{10} [26]	90.86	5.67	92.71
FGLCC_{15} [26]	91.46	5.08	93.30
WSE	73.27	10.66	80.42
CSE	73.27	10.65	80.42
CFSE	87.30	6.66	90.43

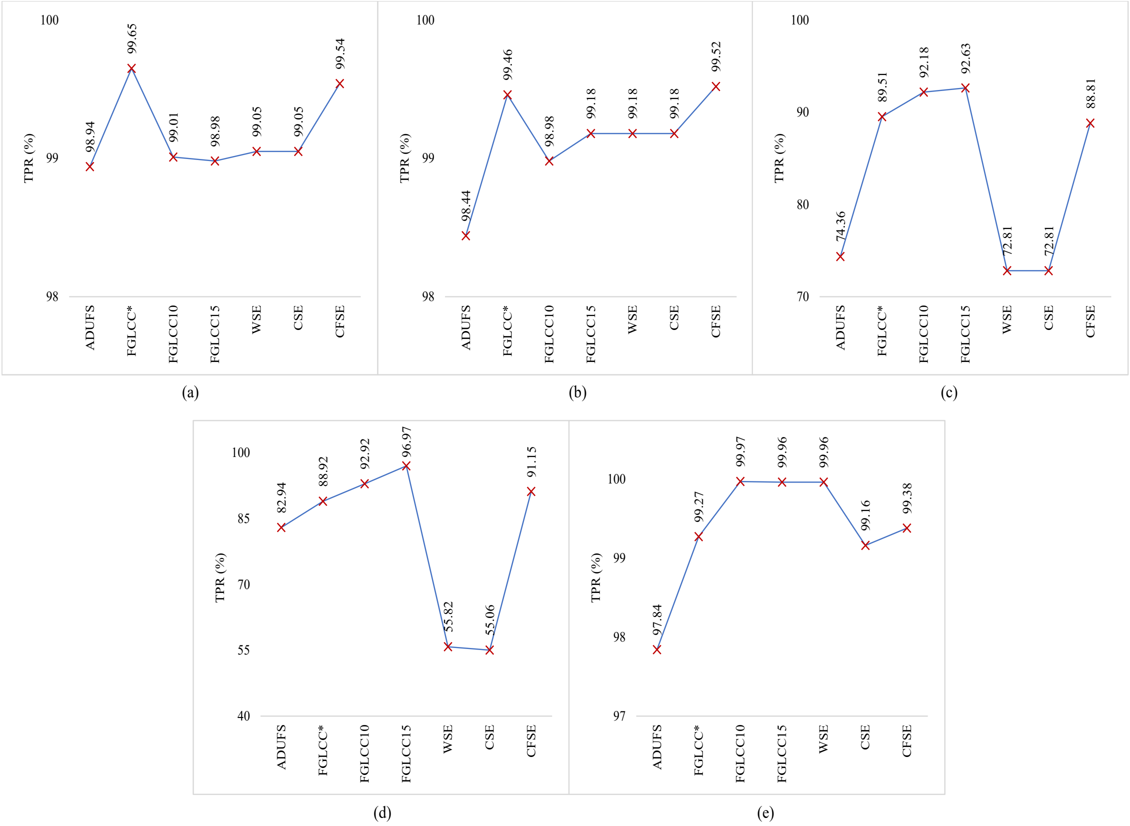
Table 24. Classification performance comparison of different feature selection techniques on NSL-KDD and KDD99 datasets.

Methods	Datasets					
	NSL-KDD			KDD99		
	TPR (%)	FPR(%)	ACC(%)	TPR (%)	FPR(%)	ACC(%)
CCFSRFG [38]	86.16	2.45	90.81	91.71	0.33	98.00
FGLCC_3 [26]	85.50	32.55	74.07	91.35	14.27	86.11
FGLCC_{10} [26]	93.58	5.10	94.20	99.86	0.20	99.56
FGLCC_{15} [26]	97.23	1.61	97.78	99.84	0.30	99.94
WSE	1.43	65.92	27.06	99.84	0.30	99.94
CSE	1.34	66.16	26.73	96.63	0.30	99.29
CFSE	92.27	2.92	94.44	97.48	0.15	99.38

Table 25. Anomaly detection comparison of different methods with feature selection techniques on all datasets.

Methods	Datasets									
	Windows 10		Windows 7		UNSW_NB15		NSL-KDD		KDD99	
	TPR (%)	FPR(%)	TPR (%)	FPR(%)	TPR (%)	FPR(%)	TPR (%)	FPR(%)	TPR (%)	FPR(%)
ADUFS (proposed)	98.94	**4.10**	98.44	**2.18**	74.36	10.85	82.94	2.08	97.84	0.33
FGLCC^* [26]	99.65	4.41	99.46	2.58	89.51	6.18	88.92	42.92	99.27	16.52
FGLCC_{10} [26]	99.01	4.38	98.98	2.65	92.18	5.55	92.92	4.99	99.97	0.20
FGLCC_{15} [26]	98.98	4.38	99.18	2.65	92.63	4.96	96.97	1.59	99.96	0.30
WSE	99.05	4.73	99.18	2.18	72.81	8.69	55.82	7.99	99.96	0.30
CSE	99.05	4.73	99.18	2.18	72.81	8.68	55.06	7.66	99.16	0.30
CFSE	99.54	4.27	99.52	2.45	88.81	6.34	91.15	2.74	99.38	0.15

FGLCC^* indicates FGLCC_3 for Windows 10, NSL-KDD, and KDD99 datasets, respectively. FGLCC^* indicates FGLCC_{12} for Windows 7 and FGLCC_6 for UNSW_NB15 datasets, respectively.



(a) (b) (c)

(d) (e)

Fig. 17. Performance comparison by different anomaly detection approaches with feature selection for all datasets in terms of TPR: (a) Windows 10, (b) Windows 7, (c) UNSW_NB15, (d) NSL-KDD, and (e) KDD99.

obtained by different methods are ADUFS, ADUFS, WSE, and CSE, FGLCC$_{15}$, FGLCC$_{15}$, and CFSE for the datasets in order.

5 CONCLUSION AND FUTURE WORK

This paper introduced the application of a feature selection process based on cooperative co-evolution on cybersecurity datasets for anomaly detection. It investigated both supervised and unsupervised anomaly detection techniques with and without feature selection on five cybersecurity datasets. Furthermore, the datasets with a reduced number of instances (1,000) followed the same procedure to compare feature selection effectiveness in detecting anomalies. Moreover, the performance of the proposed anomaly detection approach, ADUFS, has been compared with six state-of-the-art anomay/intrusion detection system via feature selection in terms of TPR, FPR, and ACC.

The experiments' twofold objectives using all datasets were to compare the individual anomaly detection performances in terms of TPR and ET both for supervised and unsupervised anomaly detection techniques investigated in this paper. At first, the original datasets were experimented with and without feature selection to detect anomalies using both supervised and unsupervised techniques in terms of true positive rates and execution time. For the FS process, CCFSRFG,

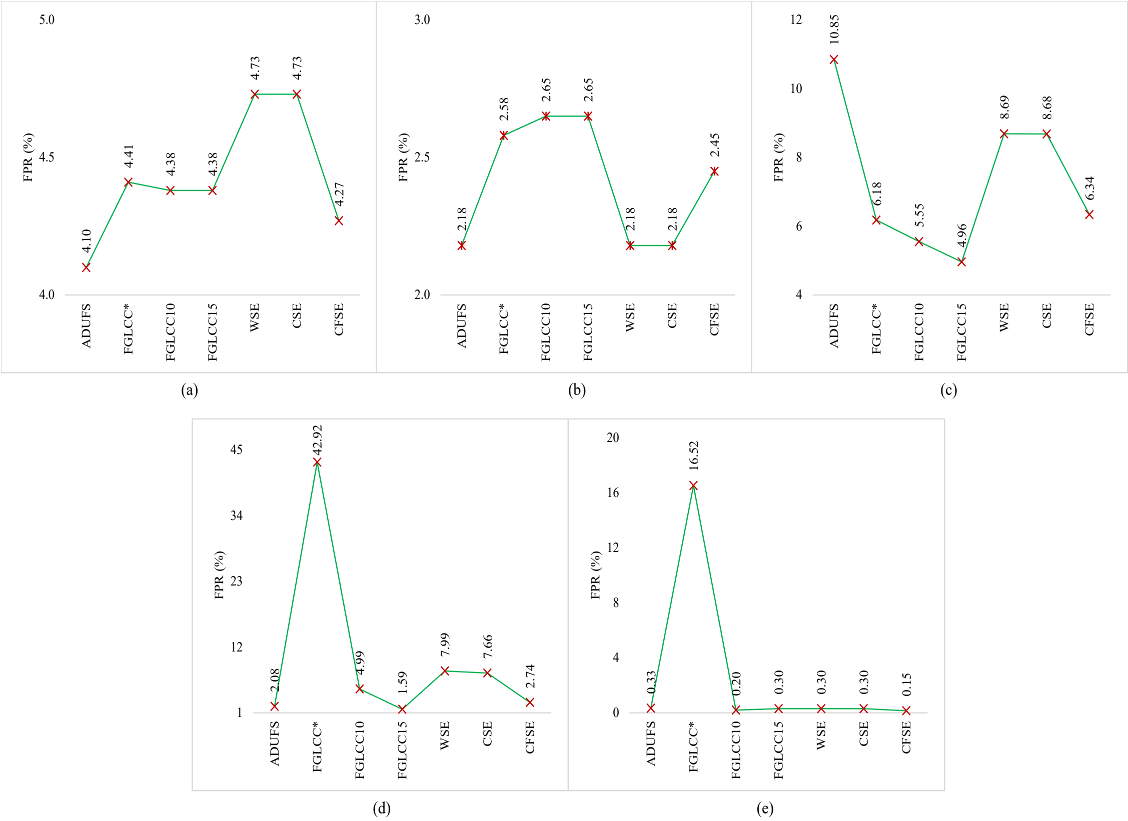


Fig. 18. Performance comparison by different anomaly detection approaches with feature selection for all datasets in terms of FPR: (a) Windows 10, (b) Windows 7, (c) UNSW_NB15, (d) NSL-KDD, and (e) KDD99.

a feature selection approach proposed in a previous study, was applied to select an appropriate subset of features that represent the dataset with a higher classification accuracy. Once the original datasets were used to detect the anomalies using both supervised and unsupervised approaches, the datasets were reduced to have an equal number of instances, thereby investigating the impacts of feature selection with a common characteristic for all datasets. Though FS using evolutionary computation is a time-consuming approach, it was investigated that if a suitable FS process can be applied before anomaly detection, the detection performance can be improved significantly.

It can be concluded from the performance results in terms of both true positive rates and execution time that FS can be recommended to apply before anomaly detection from the datasets. The actual timing performance with FS for anomaly detection will always depend on the datasets' complexities, including the number of features, the number of instances, and the data itself. From the investigation of feature selection in cybersecurity data for anomaly detection using the proposed approach, ADUFS, a number of recommendations can be made. For example, ADUFS can be widely used for forensic investigations in which detection performance is more important than execution time. Next, ADUFS can be applied to different security frameworks. In addition, ADUFS can be used to automate the security data analysis using feature selection and anomaly detection. Therefore, as future work, an automation of ADUFS will be investigated that can be applied in a wide range cybersecurity datasets, in different security frameworks, and in forensic investigation with accurate results.

ACKNOWLEDGMENTS

This research is supported by the Edith Cowan University (ECU) Higher Degree by Research Scholarship (HDRS) and the ECU School of Science Research Scholarship.

REFERENCES

[1] M. Ahmed. 2019. Intelligent Big Data summarization for rare anomaly detection. *IEEE Access* 7 (2019), 68669–68677. https://doi.org/10.1109/ACCESS.2019.2918364

[2] M. Ahmed, A. Anwar, A. N. Mahmood, Z. Shah, and M. J. Maher. 2015. An investigation of performance analysis of anomaly detection techniques for Big Data in scada systems. *EAI Endorsed Trans. Indust. Netw. & Intellig. Syst.* 2, 3 (2015), e5. https://doi.org/10.4108/inis.2.3.e5

[3] M. Ahmed, A. N. Mahmood, and J. Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60 (2016), 19–31. https://doi.org/10.1016/j.jnca.2015.11.016

[4] M. Ahmed, A. N. Mahmood, and M. R. Islam. 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems* 55 (2016), 278–288. https://doi.org/10.1016/j.future.2015.01.001

[5] U. Ahmed, J. C. W. Lin, G. Srivastava, and Y. Djenouri. 2021. A deep Q-learning sanitiszation approach for privacy preserving data mining. In *Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking* (Nara, Japan) *(ICDCN '21)*. Association for Computing Machinery, New York, NY, USA, 43—-48. https://doi.org/10.1145/3427477.3429990

[6] A. A. Alabdel Abass, M. Hajimirsadeghi, N. B. Mandayam, and Z. Gajic. 2016. Evolutionary game theoretic analysis of distributed denial of service attacks in a wireless network. In *2016 Annual Conference on Information Science and Systems*. 36–41. https://doi.org/10.1109/CISS.2016.7460473

[7] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan. 2016. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.* 65, 10 (2016), 2986–2998. https://doi.org/10.1109/TC.2016.2519914

[8] S. Bagui, E. Kalaimannan, S. Bagui, D. Nandi, and A. Pinto. 2019. Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *Security and Privacy* 2, 6 (2019), e91. https://doi.org/10.1002/spy2.91

[9] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, A. Cano, and J. C. W. Lin. 2020. A two-phase anomaly detection model for secure intelligent transportation ride-hailing trajectories. *IEEE Transactions on Intelligent Transportation Systems* (2020), 1–11. https://doi.org/10.1109/TITS.2020.3022612

[10] A. Binbusayyis and T. Vaiyapuri. 2019. Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach. *IEEE Access* 7 (2019), 106495–106513. https://doi.org/10.1109/ACCESS.2019.2929487

[11] H. Bostani and M. Sheikhan. 2017. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft computing* 21, 9 (2017), 2307–2324. https://doi.org/10.1007/s00500-015-1942-8

[12] A. Branitskiy and I. Kotenko. 2018. Applying artificial intelligence methods to network attack detection. In *AI in Cybersecurity*, L. F. Sikos (Ed.). Springer, Cham. https://doi.org/10.1007/978-3-319-98842-9_5

[13] A. Bucci and J. B. Pollack. 2005. On identifying global optima in cooperative coevolution. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York., 539–544. https://doi.org/10.1145/1068009.1068098

[14] B. Chakraborty and A. Kawamura. 2018. A new penalty-based wrapper fitness function for feature subset selection with evolutionary algorithms. *Journal of Information and Telecommunication* 2, 2 (2018), 163–180. https://doi.org/10.1080/24751839.2018.1423792

[15] C. H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh. 2019. Algorithms for frequent itemset mining: a literature review. *Artificial Intelligence Review* 52, 4 (2019), 2603–2621. https://doi.org/10.1007/s10462-018-9629-z

[16] Y. Chen, A. Abraham, and B. Yang. 2006. Feature selection and classification using flexible neural tree. *Neurocomputing* 70, 1 (2006), 305–313. https://doi.org/10.1016/j.neucom.2006.01.022

[17] S. Dwivedi, M. Vardhan, and S. Tripathi. 2020. Incorporating evolutionary computation for securing wireless network against cyberthreats. *The Journal of Supercomputing* (2020), 1–38. https://doi.org/10.1007/s11227-020-03161-w

[18] S. Elsayed, R. Sarker, and J. Slay. 2015. Evaluating the performance of a differential evolution algorithm in anomaly detection. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2490–2497. https://doi.org/10.1109/CEC.2015.7257194

[19] P. Fournier-Viger, J. C. W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le. 2017. A survey of itemset mining. *WIREs Data Mining and Knowledge Discovery* 7, 4 (2017), e1207. https://doi.org/10.1002/widm.1207

[20] M. Gómez Ravetti and P. Moscato. 2008. Identification of a 5-protein biomarker molecular signature for predicting alzheimer's disease. *PLOS ONE* 3, 9 (09 2008), 1–12. https://doi.org/10.1371/journal.pone.0003111

[21] N. M. Karie, N. M. Sahri, and P. Haskell-Dowland. 2020. IoT threat detection advances, challenges and future directions. In *2020 Workshop on Emerging Technologies for Security in IoT*. 22–29. https://doi.org/10.1109/ETSecIoT50046.2020.00009

[22] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2, 1 (2019), 1–22. https://doi.org/10.1186/s42400-019-0038-7

[23] I. Ko, D. Chambers, and E. Barrett. 2019. Unsupervised learning with hierarchical feature selection for DDoS mitigation within the ISP domain. *ETRI Journal* 41, 5 (2019), 574–584. https://doi.org/10.4218/etrij.2019-0109

[24] J. Kusyk, M. U. Uyar, and C. S. Sahin. 2018. Survey on evolutionary computation methods for cybersecurity of mobile ad hoc networks. *Evolutionary Intelligence* 10, 3-4 (2018), 95–117. https://doi.org/10.1007/s12065-018-0154-4

[25] Y. Li, J. Chen, Q. Li, and A. Liu. 2020. Differential privacy algorithm based on personalized anonymity. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. 260–267. https://doi.org/10.1109/ICBDA49040.2020.9101213

[26] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour. 2019. Cyber intrusion detection by combined feature selection algorithm. *Journal of Information Security and Applications* 44 (2019), 80–88. https://doi.org/10.1016/j.jisa.2018.11.007

[27] K. L. Moore, T. J. Bihl, K. W. Bauer Jr, and T. E. Dube. 2017. Feature extraction and feature selection for classifying cyber traffic threats. *The Journal of Defense Modeling and Simulation* 14, 3 (2017), 217–231. https://doi.org/10.1177/1548512916664032

[28] N. Moustafa and J. Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

[29] H. T. Nguyen, S. Petrović, and K. Franke. 2010. A comparison of feature-selection methods for intrusion detection. In *Computer Network Security*, I. Kotenko and V. Skormin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 242–255. https://doi.org/10.1007/978-3-642-14706-7_19

[30] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. 2013. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation* 18, 3 (2013), 378–393. https://doi.org/10.1109/TEVC.2013.2281543

[31] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao. 2017. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation* 21, 6 (2017), 929–942. https://doi.org/10.1109/TEVC.2017.2694221

[32] D. Philp, N. Chan, and L. F. Sikos. 2019. Decision support for network path estimation via automated reasoning. In *Intelligent Decision Technologies 2019*, I. Czarnowski, R. J. Howlett, and L. C. Jain (Eds.). Springer, Singapore, 335–344. https://doi.org/10.1007/978-981-13-8311-3_29

[33] M. A. Potter. 1997. *The design and analysis of a computational model of cooperative coevolution.* Ph.D. Dissertation. George Mason University, VA, United States.

[34] M. A. Potter and K. A. De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257. https://doi.org/10.1007/3-540-58484-6_269

[35] M. A. Potter and K. A. D. Jong. 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation* 8, 1 (2000), 1–29. https://doi.org/10.1162/106365600568086

[36] A. Powell, D. Bates, C. Van Wyk, and D. de Abreu. 2019. A cross-comparison of feature selection algorithms on multiple cyber security data-sets. In *FAIR*. 196–207. http://ceur-ws.org/Vol-2540/FAIR2019_paper_69.pdf

[37] A. N. M. B. Rashid. 2018. Access methods for Big Data: current status and future directions. *EAI Endorsed Transactions on Scalable Information Systems* 4, 15 (2018). https://doi.org/10.4108/eai.28-12-2017.153520

[38] A. N. M. B. Rashid, M. Ahmed, L. F. Sikos, and P. Haskell-Dowland. 2020. Cooperative co-evolution for feature selection in Big Data with random feature grouping. *Journal of Big Data* 7, 1 (2020), 1–42. https://doi.org/10.1186/s40537-020-00381-y

[39] A. N. M. B. Rashid, M. Ahmed, L. F. Sikos, and P. Haskell-Dowland. 2020. A novel penalty-based wrapper objective function for feature selection in Big Data using cooperative co-evolution. *IEEE Access* 8 (2020), 150113–150129. https://doi.org/10.1109/ACCESS.2020.3016679

[40] A. N. M. B. Rashid and T. Choudhury. 2019. Knowledge management overview of feature selection problem in high-dimensional financial data: Cooperative co-evolution and MapReduce perspectives. *Problems and Perspectives in Management* 17, 4 (2019), 340. https://doi.org/10.21511/ppm.17(4).2019.28

[41] A. N. M. B. Rashid and T. Choudhury. 2021. Cooperative co-evolution and MapReduce: A review and new insights for large-scale optimization. *International Journal of Information Technology Project Management (IJITPM)* 12, 1 (2021), 29–62. https://doi.org/10.4018/IJITPM.2021010102

[42] S. Sadik, M. Ahmed, L. F. Sikos, and A. K. M. N. Islam. 2020. Toward a sustainable cybersecurity ecosystem. *Computers* 9, 3 (2020), 74. https://doi.org/10.3390/computers9030074

[43] D. Schatz, R. Bashroush, and J. Wall. 2017. Towards a more representative definition of cyber security. *Journal of Digital Forensics, Security and Law* 12, 2 (2017), 53–74. https://doi.org/10.15394/jdfsl.2017.1476

[44] M. Shi and S. Gao. 2017. Reference sharing: a new collaboration model for cooperative coevolution. *Journal of Heuristics* 23, 1 (2017), 1–30. https://doi.org/10.1007/s10732-016-9322-9

[45] L. F. Sikos. 2020. Packet analysis for network forensics: A comprehensive survey. *Forensic science international: digital investigation* 32 (2020), 200892. https://doi.org/10.1016/j.fsidi.2019.200892

[46] L. F. Sikos, D. Philp, S. Voigt, C. Howard, M. Stumptner, and W. Mayer. 2018. Provenance-aware LOD datasets for detecting network inconsistencies. In *Joint Proceedings of the International Workshops on Contextualized Knowledge Graphs, and Semantic Statistics co-located with 17th International Semantic Web Conference*, S. Capadisli, F. Cotton, J. M. Giménez-García, A. Haller, E. Kalampokis, V. Nguyen, A. Sheth, and R. Troncy (Eds.). RWTH Aachen University, Aachen. http://ceur-ws.org/Vol-2317/article-03.pdf

[47] L. F. Sikos, M. Stumptner, W. Mayer, C. Howard, S. Voigt, and D. Philp. 2018. Automated reasoning over provenance-aware communication network knowledge in support of cyber-situational awareness. In *Knowledge Science, Engineering and Management*, W. Liu, F. Giunchiglia, and B. Yang (Eds.). Springer, Cham, 132–143. https://doi.org/10.1007/978-3-319-99247-1_12

[48] L. F. Sikos, M. Stumptner, W. Mayer, C. Howard, S. Voigt, and D. Philp. 2018. Representing network knowledge using provenance-aware formalisms for cyber-situational awareness. *Procedia Computer Science* 126 (2018), 29–38. https://doi.org/10.1016/j.procs.2018.07.206

[49] R. Storn and K. Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359. https://doi.org/10.1023/A:1008202821328

[50] S. Thudumu, P. Branch, J. Jin, and J. J. Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional Big Data. *Journal of Big Data* 7, 1 (2020), 1–30. https://doi.org/10.1186/s40537-020-00320-x

[51] G. A. Trunfio, P. Topa, and J. Wąs. 2016. A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution. *Information Sciences* 372 (2016), 773–795. https://doi.org/10.1016/j.ins.2016.08.080

[52] A. Tundis, S. Ruppert, and M. Mühlhäuser. 2020. On the automated assessment of open-source cyber threat intelligence sources. In *Computational Science – ICCS 2020*, V. V. Krzhizhanovskaya, Gábor Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, Sérgio Brissos, and João Teixeira (Eds.). Springer International Publishing, Cham, 453–467. https://doi.org/10.1007/978-3-030-50417-5_34

[53] F. van den Bergh and A. P. Engelbrecht. 2004. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 8, 3 (2004), 225–239. https://doi.org/10.1109/TEVC.2004.826069

[54] R. P. Wiegand. 2003. *An analysis of cooperative coevolutionary algorithms*. Ph.D. Dissertation. George Mason University, VA, United States.

[55] J. M. T. Wu, G. Srivastava, J. C. W. Lin, Y. Djenouri, M. Wei, R. M. Parizi, and M. S. Khan. 2021. Mining of high-utility patterns in Big IoT-based databases. *Mobile Networks and Applications* (2021), 1–18. https://doi.org/10.1007/s11036-020-01701-5

[56] J. M. T. Wu, G. Srivastava, M. Wei, U. Yun, and J. C. W. Lin. 2021. Fuzzy high-utility pattern mining in parallel and distributed Hadoop framework. *Information Sciences* 553 (2021), 31–48. https://doi.org/10.1016/j.ins.2020.12.004

[57] Z. Yang, K. Tang, and X. Yao. 2008. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences* 178, 15 (2008), 2985–2999. https://doi.org/10.1016/j.ins.2008.02.017