Sakarya University Journal of Computer and Information Sciences
http://saucis.sakarya.edu.tr/

DOI: 10.35377/saucis...1223054 RESEARCH ARTICLE

Preprocessing Impact Analysis for Machine Learning-Based Network Intrusion Detection

Hüseyin Güney

Bahçeşehir Cyprus University, Department of Computer Engineering, Nicosia, Northern Cyprus, Türkiye

Corresponding author:
Hüseyin Güney,
Bahçeşehir Cyprus University,
Department of Computer Engineering,
Nicosia, Northern Cyprus
E-mail address:
huseyin.guney@baucyprus.edu.tr

Received: 22 December 2022
Revised: 14 March 2023
Accepted: 03 April 2023
Published Online: 30 April 2023

Citation: Hüseyin Güney (2023). Preprocessing Impact Analysis for Machine Learning-Based Network Intrusion Detection. *Sakarya University Journal of Computer and Information Sciences*. 6 (1) https://doi.org/10.35377/saucis...1223054

ABSTRACT

Machine learning (ML) has been frequently studied to build intelligent systems in many problem domains. For example, one of the application areas of ML in cybersecurity is to develop intelligent intrusion detection systems (IDSs) for malicious network activity detection. However, intelligent IDS development is challenging due to many available methods in the current literature, including different types of classification algorithms and preprocessing techniques. Therefore, revealing the best-fitting methods for intrusion detection would help practitioners develop efficient detection systems. For this purpose, this study has conducted extensive experiments using the support vector machines (SVM) classifier and feature selection (FS) technique, several data normalisation techniques, and a classifier optimisation algorithm to analyse the impact of preprocessing techniques on classification. These methods were tested on three open network intrusion datasets, NSL-KDD, UNSW-NB15, and CICIDS2017. Finally, the results were analysed to investigate each method's impact on model performance and extract insights for building intelligent IDS. The optimised model achieved an accuracy of 81.51% with two features, 85.27% with 32 features, and 99.43% with 16 features for the NSLKDD, UNSW-NB15, and CICIDS2107 testing datasets, respectively. Furthermore, the results exhibited that data preprocessing has improved classification performance, and the log-scaling normalisation technique outperformed the z-score and min-max. Additionally, the results suggested that SVM-based FS improved classification performance and significantly reduced model complexity. In addition, the conclusion was drawn that classifier optimisation could enhance the performance of the classifier-dependent FS technique, such as SVM FS. However, it was observed that an inadequate feature set in the classifier optimisation process could result in worse performance; therefore, this problem must be addressed during the optimisation process for accurate optimisation. In conclusion, this study provided insights into data preprocessing in ML applications and showed the significance of data preprocessing for building accurate and efficient IDSs.

Keywords: Data Preprocessing, Classifier Optimisation, Feature Selection, Network Intrusion Detection System, Support Vector Machines.

1. Introduction

Modern computer applications are essential to daily life, providing a wide range of services. Developments of modern computer applications have led to the exchange of high-volume data over the Internet, including users' sensitive data [1]. Intrusion detection systems (IDSs) are promising for protecting user data due to their ability to monitor computer systems to determine malicious activities [2]. However, an intelligent system that can learn and recognise attacks autonomously is needed because of high-volume network traffic and various attack types. From the point of view of machine learning (ML), this is a classification and dimensionality reduction problem that can be developed to classify malicious activities.

The basic steps for the development of classification models include data preprocessing (encoding, data normalisation, and dimensionality reduction), classifier optimisation, model training, and model evaluation [3]. Encoding is applied if a feature needs to be converted to another type [3]. For example, if a classifier (e.g., support vector machine (SVM)) cannot process a categorical feature, the features must be converted to a nominal feature before model development. Data normalisation aims to transform feature values into a new range to create a better distribution of features and smaller feature space with the aim of improving classification performance and reducing model complexity [4]. Feature selection (FS) also aims at the same

goal as data normalisation by selecting the most relevant features and removing noisy and irrelevant features [5]. Finally, to develop an accurate model, it is necessary to apply preprocessing techniques prior to model development [4,5].

The network intrusion detection benchmark datasets were created in the articles [6], [7], and [8], NSL-KDD, UNSW-NB15, and CICIDS2017, respectively. First, the authors created NSLKDD to overcome the shortcomings of the KDDCUP99 dataset. Next, to tackle the shortcomings of NSLKDD, UNSW-NB15 was created. Finally, CICIDS2017 was created as an up-to-date and modern dataset.

In the study [9], the authors conducted several experiments to evaluate the performance of normalisation techniques, including decimal scaling, min-max normalisation, and z-score normalisation. In addition, Yin et al. proposed a deep learning (DL)-based IDS model using recurrent neural networks (RNN-IDS), which were normalised using the min-max method [10]. In [11], an ensemble model was developed for intrusion detection using min-max normalisation and feature selection techniques. In work [12], the authors designed and implemented several ML settings with different FS techniques using the C4.5 classifier to build intrusion detection methods. Among four FS techniques, Information Gain, Correlation-based FS, ReliefF, and Symmetrical Uncertainty, the InfoGain FS technique achieved the best performance when applied before the C4.5 classifier as a preprocessing technique for dimensionality reduction. The results revealed that the InfoGain-C4.5 pair achieved the best accuracy with 17 features. This study emphasised the importance of feature selection as a preprocessing step. In Article [13], several experiments were conducted that combined three FS techniques with various classifiers to analyse the impact of the filter and embedded FS on intrusion detection. Chi-square, information gain, and SVM recursive feature elimination (SVM-RFE) were used as the two filter and one embedded FS technique, respectively. When the filter methods were compared to SVM-RFE, SVM-RFE achieved the best performance due to the ability of SVM-RFE to select features with respect to their usefulness rather than their relevancy. As a result, this study showed that embedded FS techniques might be more promising for intrusion detection than filter methods. In addition, the authors stated that the best-performing classifier was SVM with all FS techniques. In conclusion, FS helps improve the accuracy of attack detection. However, it should be noted that embedded FS techniques are computationally more complex than filter FS techniques.

Malik et al. [14] proposed a NIDS based on the particle swarm optimisation (PSO) algorithm. In the study, extensive experiments were conducted to show that selecting the relevant features helped improve the classification performance of the proposed method. In [15], another PSO-based study was conducted, showing that the PSO-enabled SVM outperformed the default SVM configuration. Finally, Khammassi and Krichen [16] combined a wrapper FS technique using genetic algorithms with a logistic regression classifier for the detection of network intrusions. They achieved the maximum performance with 18 features for the KDD dataset and 20 for the UNSW-NB15 dataset. In the study, the effectiveness of FS was discussed based on the obtained experimental results. However, wrapper FS techniques have a relatively high computational complexity compared to embedded and filter methods.

In [17], packet preprocessing techniques were used to improve convolutional neural network (CNN) performance for intrusion detection. For this purpose, three preprocessing techniques, direct, weighted, and compressed, were developed and applied to CNN. In addition, CNN with the direct preprocessing technique was evaluated on the NSLKDD dataset. As a result, the authors stated that packet preprocessing had improved the model's performance.

The authors used deep neural networks to compare data preprocessing techniques [18]. In the study, a trial and error approach was conducted to find classifier settings, such as the number of layers, the number of neurons in each layer, and the optimiser algorithm, to obtain the optimised model. UNSW-NB15 was used in the study, whereas two different techniques were employed in this dataset. The first technique was Log transformation and MinMaxScaling, and the second was Z-score encoding and dummy encoding. Finally, it was mentioned that this particular study aimed to prove the concept that data preprocessing improves the performance of ML algorithms.

T. Ahmet and M. N. Aziz conducted an experimental study to classify intrusions in computer networks [19]. First, preprocessing and feature selection were applied, and the obtained data were classified using k-NN, SVM and Naïve Bayes classifiers. The experimental results obtained using the KDDCup99 benchmarking dataset showed that SVM with preprocessing and feature selection achieved the best performance, where Min-max normalisation and Correlation-based FS and Particle Swarm Optimisation were applied as the preprocessing and feature selection techniques, respectively.

P. Nimbalkar and D. Kshirsagar proposed a feature selection method for intrusion detection [20]. The study focused on DoS and DDoS attacks and proposed the FS method based on Information Gain and Gain Ratio FS techniques. The proposed method was evaluated on IoT-Bot and KDDCup99 datasets using the JRip classifier. The proposed system selected 16 and 19 features for the IoT-Bot and KDDCup99 datasets, respectively, and it was concluded that it performed better than the method with the complete feature set.

In [21], Naïve Bayes and KNN classifiers were used to develop a two-tier classifier. In addition, for feature selection, the Discriminant Analysis method was used. The proposed method was evaluated on the NSLKDD dataset. In [22], the authors proposed an intrusion detection framework for botnet using feature selection. According to the obtained results on the CICIDS2017 dataset, the Correlation Attribute Eval FS technique with JRip classifier achieved the best performance for botnet detection.

In order to reduce the time complexity of ML algorithms, fast kNN was proposed [23], and it was shown that fast kNN maintains the model's accuracy and reduces time complexity. Several filter FS techniques were combined for DoS attack detection, including Information Gain Ratio (IGR), Correlation (CR), and ReliefF (ReF). The results showed the significance of feature selection. A deep learning approach was used in the study [24] for intrusion detection to improve detection performance, and the proposed method was evaluated on the NSLKDD dataset [25].

In the present study, several experiments were conducted to measure the impact of preprocessing on ML applications, such as developing intelligent IDSs, using data normalisation, feature selection, and classifier optimisation algorithms. For this purpose, several network intrusion detection benchmark datasets were used, namely NSLKDD, UNSW-NB15, and CICIDS2017. To evaluate data normalisation techniques, three different and frequently used techniques (min-max, z-score, and logarithmic scaling) were applied to the original dataset before model development. Additionally, the SVM classifier was optimised using an exhaustive search algorithm, grid-search, where the optimal kernel function, cost parameter and gamma parameter were selected for all benchmarking datasets.

As a result, this study aimed to measure the impact of several data normalisation techniques to find the most effective technique for ML classification applications. Additionally, a classifier-dependent feature selection method was used to investigate feature selection performance on intrusion classification. Moreover, classifier optimisation was applied to classification and FS methods for analysing the impact of its classification performance with a reduced and complete feature set. Furthermore, the classifiers developed with default settings and optimised with different settings were compared to reveal the impact of feature selection in classifier optimisation. Finally, the ML model settings and chosen preprocessing techniques were presented to provide important insights for practitioners in the field. In summary, the contributions of this study are as follows.

(1) Detailed analysis of normalisation techniques and their impact on classifier performance
(2) Impact of classifier parameter optimisation on feature selection and classification
(3) Important insights into the development of intelligent intrusion detection systems.

The remainder of this paper is organised as follows. Section 2 explains the materials and methods used in this study. Section 3 provides details about the experimental setup. Section 4 presents and discusses the experimental results. Finally, the conclusion of this study is stated in Section 5.

2. Materials and Methods

In this section, the materials and methods used for this study are explained, including data preprocessing techniques and SVM. Furthermore, the used network intrusion benchmark datasets were mentioned in detail.

2.1 Benchmark datasets

NSLKDD dataset [6]: The NSLKDD dataset was extracted from the KDDCup'99 dataset as its enhanced version to overcome the shortcomings of the KDDcup'99 dataset. Although NSLKDD may not be the best representation of real networks, containing some synthetic data, it is a valuable benchmark dataset for evaluating the intrusion detection model. The NSLKDD dataset comprises two partitions: $KDDTrain^{+}$ (training dataset) and $KDDTest^{+}$ (test dataset). The $KDDTest^{-21}$, as a test dataset, is a subset of $KDDTest^{+}$ where the samples correctly classified by all classifiers in [26] have been removed. Thus, $KDDTest^{-21}$ is a more challenging dataset than $KDDTest^{+}$ for classification algorithms. In addition, the binary NSLKDD dataset contains two types of network activities: normal and attack. Table 1 presents the statistics of this dataset. Furthermore, the features in this dataset belong to three categories, namely *basic features* (Feature Nos. 1–10), *content features* (Feature Nos. 11–20), and *traffic features* (Feature Nos. 23–41) [10]. Finally, there are 41 features and 1 class label in this dataset.

Table 1 Record Distribution of the NSLKDD Dataset

Dataset	Normal	Attacks	Total
$KDDTrain^{+}$	67343	58630	125973
$KDDTest^{+}$	9711	12833	22544
$KDDTest^{-21}$	2152	9698	11850

UNSW-NB15 Dataset [7]: Moustafa and Slay published the UNSW-NB15 benchmark dataset for network intrusion detection in 2015 as a comprehensive dataset. They stated that this dataset was developed to overcome the shortcomings of the NSLKDD dataset. This dataset includes 47 features, where categories of features are as follows: *flow features* (1–5), *basic features* (6–18), *content features* (19–26), *time features* (27–35), and *general-purpose features* (36–47). The UNSW-NB15

training dataset and the UNSW-NB15 test dataset are the two predefined splits of this dataset. Statistics for the dataset are listed in Table 2.

Table 2 Record Distribution of the UNSW-NB15 Dataset

Dataset	Total Sample Size	Normal	Attacks
Training Set	175,341	56,000	119,341
Testing Set	82,332	37,000	45,332

CICIDS2017 Dataset [8]: The CICIDS2017 dataset was created by the Canadian Institute of Cybersecurity in 2017 as a network evaluation dataset. The CICIDS2017 dataset contains realistic network activity records extracted from an actual network setup. In addition, it comprises the most up-to-date and widely used attack types. In the dataset, network flows are based on the time stamp, source and destination IPs, source and destination ports, and protocols. The dataset consists of 78 features and a label column where the records were collected over a week. The CICIDS2017 Wednesday dataset used in this study contains normal activity records and DoS attacks. For computational reasons, the dataset was under-sampled by randomly selecting 20% of its records. Then, it was split into two partitions as the training and testing datasets. Statistics are shown in Table 3.

Table 3 Record Distribution of the CICIDS2017 Wednesday Dataset

Dataset	Total Sample Size	Normal	Attacks
Training Set	138,480	88,191	50,289
Testing Set	137,806	87,817	49,989

2.2 Data preprocessing

Data preprocessing in ML can be described as the process of investigating and transforming the dataset in terms of sample distribution (i.e., data normalisation) and dimensionality (i.e., feature selection or extraction) to improve classification performance as the primary objective. The secondary objective of preprocessing is to reduce model complexity in terms of features (dimensionality) and feature value ranges (. In addition, encoding of feature values is only applied when the classifier needs conversion of feature type. For example, SVM does not accept categorical features; thus, a categorical feature must be converted to a nominal one.

The encoding process comprises three main steps: (1) identification of categories in a categorical feature, that is, finding distinct values, (2) creation of new features for each distinct value; encoding will generate new features as many as the number of unique values in the categorical feature, (3) and finally, for each feature, the records are set to 1, and the rest is set to 0 [3].

Data normalisation can be considered as a scaling process that creates a new scale of feature values to transform the distribution of the dataset into a more balanced form. Additionally, this process helps to create a smaller hyperspace for the classifier and can lead to higher classification performance and lower computational complexity [4]. Z-score, min-max, and logarithmic-scaling (log-scaling) are the frequently used techniques in the field of ML. Min-max and z-score normalisation are linear techniques in which these functions perform a linear mapping of the feature space [4]. However, log-scaling is a nonlinear technique that transforms values in a nonlinear space. On the other hand, Min-max normalisation maps feature values into the range of [0, 1] using the minimum and maximum values. Furthermore, z-score normalisation (standardisation) transforms a feature into a new range using mean and standard deviation values, where the standard deviation of the transformed feature is always one. Additionally, log-scaling uses a log function to scale down the feature values non-linearly. Equations are listed in the following equations (eq. 1, eq. 2, and eq. 3), respectively. Note that x_{min} and x_{max} are the minimum and maximum values of the feature x in Equation 1. In equation 2, x_μ and x_σ are the mean and standard deviation values of the feature x. In addition, x_i represents the i^{th} value in feature x for all equations.

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (1)$$

$$x'_i = \frac{x_i - x_\mu}{x_\sigma} \quad (2)$$

$$x'_i = \log(x_i + 1) \quad (3)$$

Finally, dimensionality reduction reduces feature space by removing irrelevant/noisy features (feature selection) or transforming the feature set (feature extraction) into a new space. Feature selection retains the values of the features, but feature extraction changes them while transforming the space [5]. Feature selection and feature extraction are the main categories of dimensionality reduction.

2.3 Support Vector Machine Classifier

Classification is the process of building an ML model using supervision with a dataset that contains input and output data. As a supervised ML algorithm, SVM has several significant characteristics that make it a successful classifier. These are (1) the ability to find an optimal hyperplane between two linearly separable classes by maximising the class boundaries, (2) the ability to construct a high-dimensional hyperplane, (3) embedded in FS techniques as a robust FS technique, and (4) its kernel trick that allows one to create nonlinear boundaries [26, 27]. Furthermore, the kernel functions provide access to the higher space dimensions without explicitly defining the mapping function. Therefore, the performance of the SVM classification is highly dependent on its kernel function and parameters, which implies that classifier optimisation is crucial for SVM [28]. Classifier optimisation aims to find the optimal values for the classifier kernel and parameters. However, finding the best-fitted values is a dataset-specific problem and heuristic algorithms are used to solve this problem, which is time-consuming.

2.4 Feature selection

Feature selection techniques aim to select the most relevant features by removing redundant and irrelevant features, which has been widely applied in ML applications before model construction to improve model accuracy and reduce computational complexity. Embedded/Wrapper FS techniques use a classifier to obtain feature weights for ordering features from the most relevant to the least relevant, aiming to select the optimal subset of features. On the contrary, filter FS techniques check feature relevance for the same purpose. Since embedded/wrapper techniques are classifier-dependent (e.g., SVM, Random Forest), they are more accurate, whereas filter FS techniques are classifier-independent and thus faster than embedded/wrapper FS techniques.

In this study, the SVM feature selection technique (SVM-FS) [5] uses the feature weights vector created by the SVM training process to rank features with the aim of measuring the impact of classifier-dependent FS on intrusion detection. Algorithm 1 represents the algorithm for SVM-FS. Unlike SVM-RFE [5], the used SVM-FS technique runs SVM train function ones to obtain feature weights and rank feature weights accordingly. As it is shown in line 2, SVM training runs to obtain support vectors (SV) and coefficients (coefs) from constructed SVM model, and then this information is used to calculate the weight vector, which is ordered to obtain feature ranks (shown in lines 3 and 4). Note that the generated ranked feature list is in descending order, where the most significant weight represents the best feature; therefore, the first feature in the list is the most relevant, and the last is the least relevant feature.

Algorithm 1. SVM Feature Selection Technique (SVM-FS)

Input: Training Dataset
Output: Ranked Feature List

1. Initialise the training dataset, TrainingDataset.
2. SVM_Model = SVM_Training(TrainingDataset, Kernel, CostValue, Kernel_Parameter_Values)
3. Feature_Weight_List = CreateWeightList(SVM_Model$coefs, SVM_Model$SV)
4. Ranked_Feature_List = argmax(Feature_Weight_List)

3. Experimental Setup

The experiments were carried out using the software implemented with R programming (v4.0.2) [29] and RStudio [30] on a PC, Intel® i7 Core(TM) i7-4790 CPU @ 3.60GHz, 32 GB DDR3 Ram, 256 GB SSD, and Microsoft® Windows 10 Pro x64. In addition, the R programming package, e1071 [31], was used to implement the SVM classifier. This experimental study involves (1) data normalisation, (2) feature selection, (3) classifier optimisation, and (4) model training and evaluation.

First, the model has never seen the test dataset to avoid normalisation, FS, or classification bias. As mentioned in the benchmark datasets section, each dataset was split into two sets: training and testing. The model was constructed using the training dataset, and evaluation was performed on the testing dataset; hold-out validation was used. Furthermore, during the data normalisation process, the required values were obtained from the training dataset and applied to the test dataset to avoid bias can be caused by normalisation. Additionally, FS was performed on the training dataset to avoid FS bias. The model's performance was measured with the top k features after the feature ranking where k = 1, 2, 3, 4, 8, 16, 32, 64, and feature set size. Since the NSLKDD and UNSW-NB15 datasets contain categorical features, encoding was applied. After encoding, the feature size of the datasets was 122 and 194, respectively. The SVM classifier with default settings (c=1, gamma = 1 / |

feature set |, degree = 3) was implemented. Finally, grid search was used for SVM optimisation, where a kernel function, cost parameter, and the value search of the chosen kernel parameter were performed. Thus, the kernel function was selected among linear, polynomial, sigmoid, and radial basis functions (RBF) in the classifier optimisation process. Furthermore, the cost and gamma value spaces were searched where $c > 0$ and $\gamma > 0$. Performance evaluation is the final and critical stage of ML model development since it shows the model's accuracy. For this purpose, accuracy and f-score performance evaluation metrics were used [3]. The equations of these metrics are given in (4) and (7), where (5) and (6) were used to calculate F-score.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$\text{F} - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{Precision + Recall} \tag{7}$$

Here, TP, TN, FP, and FN are explained below.

- True Positive (TP): Truly classified attack class (positive) sample.
- True Negative (TN): Truly classified normal class (negative) sample.
- False Positive (FP): Falsely classified normal class sample, classified as an attack.
- False Negative (FN): Falsely classified attack class sample, classified as normal activity.

4. Experiment Results

The experiment results in this study are examined in four sections, including SVM kernel function selection, comparison of normalisation techniques, the impact of FS, and the impact of classifier optimisation.

4.1. SVM kernel function performance comparison

The kernel function comparison test showed that the RBF kernel function outperformed the others for all datasets. SVM-RBF with default parameter values fits well with the training datasets and outperforms the other kernels for the KDDTest^{+}, UNSW-NB15, and CICIDS2017 Wednesday testing datasets. The linear kernel achieved the best for the KDDTest^{-21} dataset but did not fit the training dataset as RBF did. The results are shown in Table 4.

Table 4 Performance of the Kernel Functions of the SVM Classifier in terms of Accuracy

Kernel Function	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
Linear	59.3%	64.3%	**69.1%**	78.13%	69.73%	42.44%	42.28%
Polynomial	15.0%	22.5%	38.6%	30.89%	43.38%	63.88%	63.87%
Sigmoid	40.9%	53.2%	55.4%	68.06%	55.06%	63.68%	63.72%
Radial	**99.9%**	**72.4%**	47.5%	**99.24%**	**75.69%**	**99.99%**	**72.45%**

4.2. Normalisation technique selection

This subsection discusses the impact of normalisation techniques by applying min-max, z-score, and log-scaling. However, since CICIDS2017 contains negative values, the log-scaling was not applied to this dataset. In addition, normalisation technique evaluation tests were performed with the entire feature set; that is, no FS prior to classification was applied for this experiment. Moreover, the SVM-RBF classifier with default settings was used for classification. The accuracies obtained for the original and normalised datasets are listed in Table 5.

The results showed that the normalisation of the datasets significantly improved the classification performance. The achieved accuracies were as follows: for KDDTrain^{+}, the accuracy was 99.5%, which was very close to 100%; it retained the model's performance on the original dataset. For the NSLKDD testing datasets, KDDTest^{+} and KDDTest^{-21}, the accuracies were 80.8% and 63.4%, respectively; normalisation increased the model performance by 7.6% on KDDTest^{+} and 15.9% on KDDTrain^{-21}. Furthermore, z-score and log-scaling normalisation methods have achieved almost the same performance for

the UNSW-NB15 training and testing datasets, where the model's performance was improved by approximately 6%. When the min-max and z-score normalisation methods were compared for the CICIDS2017 datasets, it was shown that the z-score method outperformed the min-max. With the z-score normalisation, the accuracy was improved by approximately 26% for the Wednesday testing dataset. In conclusion, the results showed that the normalisation of the data led to a considerable improvement in the performance of SVM-RBF. It was also observed that log-scaling and z-score achieved similar performance and outperformed the min-max technique for all datasets.

In addition, statistical tests were performed on the NSLKDD dataset to further understand the normalisation techniques' impact. First, some features with high variation were randomly selected, and the minimum, maximum, mean, and standard deviation values were calculated before and after min-max, z-score, and log-scaling. In addition, the skewness and kurtosis values were calculated.

Table 5 Impact of Normalisation Techniques on the Performance of SVM-RBF in terms of Accuracy

Normalisation Technique	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
-	**99.9%**	72.4%	47.5%	**99.24%**	75.69%	**99.99%**	72.45%
Min-max	97.4%	74.7%	52.1%	92.92%	78.76%	97.03%	96.85%
Z-score	98.5%	76.7%	55.9%	93.63%	81.47%	98.58%	**98.48%**
Log-scaling	99.5%	**80.8%**	**63.4%**	93.64%	**81.55%**	-	-

Table 6 Statistics of Normalisation Techniques Using KDDTest+ Dataset

		Original	**Min-Max**	**Z-Score**	**Log-Scaling**
Duration - Feature No **1**	Mean	287.14	0.030	-5.890e-18	0.322
	Std. Dev.	2604.51	0.136	1	1.451
	Min. Val.	0	0	-0.110	0
	Max. Val.	42908	1	16.364	10.667
	Skewness	11.88	5.34	11.88	5.34
	Kurtosis	156.07	28.95	156.07	28.95
Source Bytes - Feature No **86**	Mean	45566.74	0.153	-2.023e-20	3.230
	Std. Dev.	5870331	0.142	1	2.982
	Min. Val.	0	0	-0.008	0
	Max. Val.	1379963888	1	235.067	21.045
	Skewness	190.66	0.31	190.66	0.31
	Kurtosis	39351.93	-0.67	39351.93	-0.67
Destination Host Count - Feature No **113**	Mean	128.15	0.858	3.637e-17	4.757
	Std. Dev.	99.21	0.241	1	1.334
	Min. Val.	0	0	-1.836	0
	Max. Val.	255	1	0.734	5.545
	Skewness	-0.83	-1.73	-0.83	-1.73
	Kurtosis	-1.07	1.88	-1.07	1.88

When the min and max values were observed for all normalisation techniques, it was shown that min-max obtained the smallest range, whereas the log-scaling created slightly larger ranges, but the ranges generated by the z-score varied, depending on the level of variation in the features. The z-score transformed the datasets into a distribution where the standard deviation is one, and log-scaling transformation obtained a dataset similar standard deviation to z-score. On the contrary, since min-max creates values within the range [0,1], standard deviation and mean values are also in this range. Additionally, it was observed that the z-score scaled the feature space down; however, it did not change the skewness and kurtosis values. The results showed that min-max and log-scaling alter the distribution of the dataset, and both have transformed the datasets with the same skewness and kurtosis values. Finally, the results in Table 6 suggest that log-scaling has led to a better representation of the dataset for ML applications.

4.3. Impact of feature selection

In this section, the impact of the SVM-FS technique is investigated. For this reason, it was applied with the default parameters before model construction. Then SVM-RBF with the default parameters was trained and evaluated using the ranked feature

list obtained by SVM-FS. This experiment was conducted on the normalised NSLKDD, UNSW-NB15, and CICIDS2017 Wednesday datasets. The results of the experiments are presented in Table 7. The best-achieved accuracy for the normalised NSLKDD test datasets was 85.8% and 74.1% using the top eight features. That is, SVM-FS removed 114 features and improved accuracy by 5% for $KDDTest^{+}$ and 10.7% for $KDDTest^{-21}$. Furthermore, for the UNSW-NB15 dataset, 162 features were removed, and the accuracy achieved was maintained. Finally, for the Wednesday CICIDS2017 Wednesday dataset, SVM-FS removed 14 features. In summary, the SVM-FS technique improved the performance of the SVM classifier by selecting the relevant features, and it reduced the number of features used for the model (reduced model complexity).

4.4. Impact of classifier optimisation on SVM's performance

In this section, the impact of classifier optimisation on the performance of the classifier is investigated. Each classification method has some parameters that need to be optimised for the target dataset to build the best-fitted model. Optimised parameters of a classifier are different for each input dataset, showing that this is a dataset-specific process and time-consuming since the parameters have a wide range of possible values for finding the one that fits well. Therefore, searching parameter spaces for the optimal values is an important aspect of classifier optimisation. This study uses a grid search algorithm for SVM classifier parameter optimisation. Since the radial basis function was found to be the best kernel function for this problem domain, the gamma parameter was optimised along with the cost parameter. For computational reasons, a limited search space was created for cost and gamma values, where 10^{6}, 10^{5}, 10^{4}, 10^{3}, 10^{2}, 10^{1}, 10^{0}, 10^{-1}, 10^{-2}, 10^{-3}, and 10^{3}, 10^{2}, 10^{1}, 10^{0}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6} were the members of cost and gamma list, respectively. After that, SVM with predefined cost and gamma values was run, and the accuracies were recorded. Additionally, to observe the impact of FS on classifier optimisation, two different feature sets were used: a reduced set (top eight features) and a complete feature set of the input dataset. The best classifier was found, and the optimised classifier was used to build the classification model. SVM-FS was used for feature ranking. After ranking, the best k features were selected for model construction, where k is 1, 2, 3, 4, 8, 16, 32, 64, and the complete feature set.

Table 7 Performance Evaluation of the SVM-RBF with Reduced Feature Sets in terms of Accuracy

Feature Subset	$KDDTrain^{+}$ Training Set	$KDDTest^{+}$ Testing Set	$KDDTest^{-21}$ Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set/Testing Set	
Top **1** Feature	89.2%	76.3%	55.7%	90.15%	75.13%	88.22%	88.34%
Top **2** Features	89.3%	75.4%	54.1%	91.70%	76.30%	89.41%	89.48%
Top **3** Features	89.3%	75.4%	54.1%	92.11%	76.40%	91.42%	91.42%
Top **4** Features	89.1%	75.4%	54.1%	92.28%	76.48%	91.63%	91.61%
Top **8** Features	94.6%	**85.8%**	**74.1%**	93.45%	81.01%	87.99%	87.79%
Top **16** Features	98.1%	81.8%	65.9%	93.48%	81.01%	96.19%	95.87%
Top **32** Features	99.0%	78.2%	58.7%	93.60%	81.50%	99.83%	99.74%
Top **64** Features	99.5%	80.7%	63.4%	93.62%	81.53%	**99.88%**	**99.75%**
Top **78** Features	-	-	-	-	-	99.88%	99.74%
Top **122** Features	99.5%	80.8%	63.4%	-	-	-	-
Top **128** Features	**99.9%**	72.4%	47.5%	**93.64%**	**81.55%**	-	-
Top **194** Features	-	-	-	93.64%	81.55%	-	-

As mentioned, default and optimised SVM models were implemented to compare the performance of the SVM classifier and the significance of classifier optimisation with and without feature selection. The first experiment was the SVM-RBF with default parameter settings as the FS technique and classifier. The obtained results are presented in Table 7. Table 8 presents the performance results of the SVM optimised with a reduced feature set. Additionally, the results for SVM optimised using the complete feature set are shown in Table 9. Two different settings of SVM classifier optimisation were developed to accomplish a further analysis for understanding the impact of feature selection in the classifier optimisation process. Accuracy

was used to measure the performance of all the models. However, another performance evaluation metric was used to evaluate the performance of the optimised SVM. Thus, Table 10 represents the performance of optimised SVM in terms of F-score. As shown in Equation 7, the F-score performance evaluation metric uses precision and recall values to calculate the model's performance, which better represents a model's performance where the benchmarking dataset is not balanced in class distribution. The formulation for precision and recall is given in Equations 5 and 6.

The results exhibited that feature selection helps improve classification performance or reduce feature set size. For example, the classification performance of SVM was improved by around 5% for the NSLKDD dataset, where 120 features were removed. On the other hand, 162 features were removed for the UNSW-NB15 dataset, and the classification performance was maintained. On the other hand, classifier optimisation also improved classification performance or reduced feature set size. For instance, classification performance was improved for the UNSW-NB15 dataset by ~4%. Furthermore, classification performance was improved for the CICIDS2017 dataset.

Table 8 Performance Evaluation of the SVM Optimisation with Reduced Feature Set in terms of Accuracy

Feature Subset	$KDDTrain^{+}$ Training Set	$KDDTest^{+}$ Testing Set	$KDDTest^{-21}$ Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.02%	78.02%	60.15%	68.06%	55.06%	88.22%	88.34%
Top **2** Features	95.26%	**81.51%**	**65.68%**	90.59%	75.43%	89.41%	89.48%
Top **3** Features	96.26%	77.79%	58.13%	90.62%	76.08%	91.42%	91.42%
Top **4** Features	98.57%	79.45%	61.33%	90.63%	76.10%	91.63%	91.61%
Top **8** Features	99.67%	76.01%	54.36%	93.45%	80.98%	87.99%	87.79%
Top **16** Features	99.88%	79.20%	60.46%	94.14%	83.01%	96.19%	95.87%
Top **32** Features	99.94%	79.42%	60.96%	94.78%	84.52%	99.83%	99.74%
Top **64** Features	**99.96%**	78.52%	59.32%	**94.98%**	85.25%	**99.88%**	**99.75%**
Top **78** Features	-	-	-	-	-	99.88%	99.74%
Top **122** Features	99.96%	76.99%	56.59%	-	-	-	-
Top **128** Features	-	-	-	**94.98%**	**85.30%**	-	-
Top **194** Features	-	-	-	94.98%	85.30%	-	-

The results showed that classifier optimisation has a notable impact on FS and classifier performance when applied to both FS and classifier. However, optimisation achieved worse performance for the NSLKDD dataset than the default settings, showing the importance of selecting the correct set of parameter values. On the other hand, when the results of classifier optimisation with a complete feature set were compared with the reduced feature set, it was observed that both achieved similar performance. However, in some cases, the use of reduced feature sets negatively affected performance, which is plausible because the selected feature subset is not the best representation of the informative features. It should be noted that, as shown in this study, neither default classifier nor classifier optimisation guarantees the best performance. For example, when the default SVM used for both FS and classification (SVM-RBF, cost = 1, and gamma = 1/feature size), it outperformed the optimised SVM (optimised using Grid-search, SVM-RBF, cost = 10^2 and γ: 10^{-1}).

Table 9 Performance Evaluation of SVM Optimisation with All Features in terms of Accuracy

Feature Subset	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.02%	78.02%	60.15%	90.57%	75.41%	63.65%	63.67%
Top **2** Features	95.26%	**81.51%**	**65.68%**	92.11%	76.57%	80.78%	80.69%
Top **3** Features	96.26%	77.79%	58.13%	92.46%	78.18%	81.94%	81.83%
Top **4** Features	98.57%	79.45%	61.33%	92.68%	78.83%	84.11%	83.92%
Top **8** Features	99.67%	76.01%	54.36%	92.88%	79.12%	96.30%	96.27%
Top **16** Features	99.88%	79.20%	60.46%	94.31%	83.98%	99.43%	99.36%
Top **32** Features	99.94%	79.42%	60.96%	95.03%	85.27%	99.74%	99.66%
Top **64** Features	**99.96%**	78.52%	59.32%	**95.10%**	**85.31%**	**99.79%**	**99.72%**
Top **78** Features	-	-	-	-	-	99.79%	99.72%
Top **122** Features	99.96%	76.99%	56.59%	-	-	-	-
Top **128** Features	-	-	-	95.10%	85.31%	-	-
Top **194** Features	-	-	-	95.10%	85.31%	-	-

Table 10 Performance Evaluation of SVM Optimisation with All Features in terms of F-score

Feature Subset	KDDTrain^{+} Training Set	KDDTest^{+} Testing Set	KDDTest^{-21} Testing Set	UNSW-NB15 Training Set	UNSW-NB15 Testing Set	CICIDS2017 (Wed) Training Set	CICIDS2017 (Wed) Testing Set
Top **1** Feature	90.07%	77.68%	69.93%	89.97%	77.91%	84.89%	85.00%
Top **2** Features	94.40%	**81.54%**	**74.84%**	93.55%	81.79%	86.53%	86.58%
Top **3** Features	96.63%	79.80%	72.14%	94.16%	82.96%	88.87%	88.84%
Top **4** Features	98.20%	78.04%	69.27%	94.88%	83.84%	89.16%	89.11%
Top **8** Features	99.46%	74.59%	63.71%	95.30%	84.96%	80.29%	79.89%
Top **16** Features	99.72%	73.74%	62.42%	95.84%	86.92%	94.96%	94.51%
Top **32** Features	99.87%	76.31%	66.39%	96.38%	87.94%	99.77%	99.64%
Top **64** Features	**99.92%**	73.54%	62.32%	96.45%	87.99%	**99.83%**	**99.66%**
Top **78** Features	-	-	-	-	-	99.83%	99.64%
Top **122** Features	99.92%	74.18%	63.32%	-	-	-	-
Top **128** Features	-	-	-	**96.46%**	**88.01%**	-	-
Top **194** Features	-	-	-	96.45%	88.01%	-	-

Table 11 Summary of the Default and Optimised Model on Testing Datasets

Dataset	Preprocessing			Classifier	Performance
	Data Normalisation	Classifier Optimisation	Feature Selection	Optimised SVM Configuration	Accuracy (Selected Feature Size)
NSLKDD	Log-scaling	-	SVM-FS	RBF Kernel C: 1 and γ: 0.008	85.80% (8/122)
UNSW-NB15	Log-scaling	-	SVM-FS	RBF Kernel C: 1 and γ: 0.005	81.50% (32/194)
CICIDS2017 Wednesday	Z-Score	-	SVM-FS	RBF Kernel C: 1 and γ: 0.013	98.48% (64/78)
NSLKDD	Log-scaling	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-1}	81.51% (2/122)
UNSW-NB15	Log-scaling	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-2}	85.30% (64/194)
CICIDS2017 Wednesday	Z-Score	Applied with Reduced Feature Set	Optimised SVM-FS	RBF Kernel C: 10^5 and γ: 10^0	99.74% (32/78)
NSLKDD	Log-scaling	Applied with Full Feature Set	Optimised SVM-FS	RBF Kernel C: 10^2 and γ: 10^{-1}	81.51% (2/122)
UNSW-NB15	Log-scaling	Applied with Full Feature Set	Optimised SVM-FS	RBF Kernel C: 10^0 and γ: 10^{-1}	85.27% (32/194)
CICIDS2017 Wednesday	Z-Score	Applied with Full Feature Set	Optimised SVM-FS	C: 10^4 and γ: 10^{-1}	99.43% (16/122)

However, the optimised configuration removed more features than the default and obtained worse performance, yet a more accurate selection of features. On the other hand, there is a significant improvement in the performance of the default SVM in the CICIDS2017 dataset after optimisation; that is, a 1% higher accuracy was achieved while removing 48 features. Another critical point to emphasise is the FS problem in classifier optimisation. The results exhibited that using a predefined set of features may not result in a better model since the selected feature subset may contain irrelevant features or may not contain all the relevant features, as seen from the results of the NSLKDD dataset.

In conclusion, the results revealed that a detailed FS process must be performed before classifier optimisation. However, it is known that filter FS techniques are not as accurate as embedded/wrapper techniques, and embedded/wrapper techniques suffer from the mentioned FS problem in classifier algorithms since they are classifier-dependent FS techniques. As a result, a potential solution would be the ensemble of filter methods prior to classifier optimisation to remove irrelevant features or a classifier optimisation algorithm incorporating FS. Finally, a summary of the SVM configuration for all setups are given in Table 11.

5. Conclusion

Cybersecurity is an emerging issue in information technologies, and it has become more critical with modern networking and applications. Since Internet users continuously share their sensitive data on the Internet, it is crucial to protect these data. ML-enabled IDSs are promising for this purpose. However, it is challenging for developers to select the appropriate techniques for this problem domain among many possible candidates. Furthermore, developing such an ML system requires a deep understanding of the problem and the input data. To this end, data preprocessing plays a crucial role in developing accurate and efficient ML models. Therefore, this study aimed to analyse the impact of preprocessing techniques on the ML algorithm to find the best-fitting techniques for intrusion detection datasets.

In this study, the SVM was used as a classifier for preprocessing impact analysis due to its outstanding performance and simple implementation. Additionally, SVM was also used as a feature selection technique. For the evaluation of the model, three binary intrusion detection datasets were used, namely NSL-KDD, UNSW-NB15 and CICIDS2017. First, the most

widely used kernel functions (linear, polynomial, sigmoid, and radial basis function) were tested to find the best-fitting kernel function of SVM for intrusion detection. This setting was then used to select the normalisation technique, where the techniques were applied to all datasets. Next, SVM-RBF was performed on each normalised dataset, and the best-performing normalisation technique was found. Finally, FS and classifier optimisation was performed to observe the impact of each on intrusion detection. In summary, the contributions of this study are (1) a detailed analysis of normalisation techniques and their impact on classifier performance, (2) the impact of classifier parameter optimisation on feature selection and classification, and (3) important insights for developing intelligent intrusion detection systems.

It was observed that log-scaling is the best technique for normalising intrusion detection datasets due to their high variance in features. However, when a dataset contains negative values, log-scaling is not applicable. However, the z-score can be an alternative since it achieved a similar performance and outperformed the min-max technique. As a result, data normalisation improves the classification performance in this problem domain. In addition, removing irrelevant features helps to build efficient methods. In conclusion, FS and classifier optimisation improved classification performance and reduced model complexity. However, it was observed that irrelevant features could affect the performance of optimisation algorithms that FS in classifier optimisation must be addressed for building intelligent and efficient IDSs.

Finally, this study can be repeated with more datasets and various ML algorithms, which can be one future direction of this study. Another future direction of this study would be the development of an algorithm that selects features within the optimisation process. A potential solution to this problem would be the ensemble of filter methods prior to classifier optimisation to remove irrelevant features or a classifier optimisation algorithm incorporating feature selection.

References

[1] Ham, Jeroen Van Der. “Toward a Better Understanding of “Cybersecurity”.” Digital Threats: Research and Practice 2.3 (2021): 1-3.

[2] Khraisat, Ansam, et al. “Survey of intrusion detection systems: techniques, datasets and challenges.” Cybersecurity 2.1 (2019): 1-22.

[3] Ahmad, Zeeshan, et al. “Network intrusion detection system: A systematic study of machine learning and deep learning approaches.” Transactions on Emerging Telecommunications Technologies 32.1 (2021): e4150.

[4] Singh, Dalwinder, and Birmohan Singh. “Investigating the impact of data normalisation on classification performance.” Applied Soft Computing 97 (2020): 105524.

[5] Guyon, Isabelle, et al. “Gene selection for cancer classification using support vector machines.” Machine learning 46.1 (2002): 389-422.

[6] Tavallaee, Mahbod, et al. “A detailed analysis of the KDD CUP 99 data set.” 2009 IEEE symposium on computational intelligence for security and defense applications. Ieee, 2009.

[7] Moustafa, Nour, and Jill Slay. “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set.” Information Security Journal: A Global Perspective 25.1-3 (2016): 18-31.

[8] Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. “Toward generating a new intrusion detection dataset and intrusion traffic characterisation.” ICISSp 1 (2018): 108-116.

[9] Zhang, Xiaoyuan, Daoyin Qiu, and Fuan Chen. “Support vector machine with parameter optimisation by a novel hybrid method and its application to fault diagnosis.” Neurocomputing 149 (2015): 641-651.

[10] Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017;5:21954-21961. doi:10.1109/access.2017.2762418.

[11] Tang, Chaofei, Nurbol Luktarhan, and Yuxin Zhao. “SAAE-DNN: Deep learning method on intrusion detection.” Symmetry 12.10 (2020): 1695.

[12] Pervez, Muhammad Shakil, and Dewan Md Farid. “Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs.” The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014). IEEE, 2014.

[13] Janarthanan, Tharmini, and Shahrzad Zargari. “Feature selection in UNSW-NB15 and KDDCUP’99 datasets.” 2017 IEEE 26th international symposium on industrial electronics (ISIE). IEEE, 2017.

[14] Malik, Arif Jamal, Waseem Shahzad, and Farrukh Aslam Khan. “Network intrusion detection using hybrid binary PSO and random forests algorithm.” Security and Communication Networks 8.16 (2015): 2646-2660.

[15] Kanakarajan, Navaneeth Kumar, and Kandasamy Muniasamy. “Improving the accuracy of intrusion detection using gar-forest with feature selection.” Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015. Springer, New Delhi, 2016.

[16] Khammassi, Chaouki, and Saoussen Krichen. “A GA-LR wrapper approach for feature selection in network intrusion detection.” computers & security 70 (2017): 255-277.

[17] Jo, Wooyeon, et al. "Packet preprocessing in CNN-based network intrusion detection system." Electronics 9.7 (2020): 1151. https://doi.org/10.3390/electronics9071151.
[18] Kumar, VD Ambeth. "An Effective Comparative Analysis of Data Preprocessing Techniques in Network Intrusion Detection System Using Deep Neural Networks." Smart Intelligent Computing and Communication Technology 38 (2021): 14.
[19] Ahmad, Tohari, and Mohammad Nasrul Aziz. "Data preprocessing and feature selection for machine learning intrusion detection systems." ICIC Express Lett 13.2 (2019): 93-101.
[20] Nimbalkar, Pushparaj, and Deepak Kshirsagar. "Feature selection for intrusion detection system in Internet-of-Things (IoT)." ICT Express 7.2 (2021): 177-181. https://doi.org/10.1016/j.icte.2021.04.012.
[21] Pajouh HH, Dastghaibyfard GH, Hashemi S. Two-tier network anomaly detection model: A machine learning approach. Journal of Intelligent Information Systems. 2015;48(1):61-74. doi:10.1007/s10844-015-0388-x.
[22] Jabbar AF, Mohammed IJ. Development of an optimised botnet detection framework based on filters of features and machine learning classifiers using CICIDS2017 dataset. IOP Conference Series: Materials Science and Engineering. 2020;928(3):032027. doi:10.1088/1757-899x/928/3/032027.
[23] Krishna KV, Swathi K, Rao BB. A novel framework for nids through fast knn classifier on CICIDS 2017 dataset. International Journal of Recent Technology and Engineering (IJRTE). 2020;8(5):3669-3675. doi:10.35940/ijrte.e6580.018520.
[24] Kshirsagar D, Kumar S. An efficient feature reduction method for the detection of Dos Attack. ICT Express. 2021;7(3):371-375. doi:10.1016/j.icte.2020.12.006.
[25] Azzaoui H, Boukhamla AZ, Arroyo D, Bensayah A. Developing new deep-learning model to enhance network intrusion classification. Evolving Systems. 2021;13(1):17-25. doi:10.1007/s12530-020-09364-z.
[26] Prajapati, Gend Lal, and Arti Patle. “On performing classification using SVM with radial basis and polynomial kernel functions.” 2010 3rd International Conference on Emerging Trends in Engineering and Technology. IEEE, 2010.
[27] Zhang, Xiaoyuan, Daoyin Qiu, and Fuan Chen. “Support vector machine with parameter optimisation by a novel hybrid method and its application to fault diagnosis.” Neurocomputing 149 (2015): 641-651.
[28] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. “A practical guide to support vector classification.” (2003): 1396-1400.
[29] R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
[30] RStudio Team (2019). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.
[31] Meyer, David, et al. “Package ‘e1071’.” The R Journal (2019).

Conflict of interest

The author declares that there are no potential conflicts of interest.

Funding

This research did not receive a specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

The datasets generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

Ethical Approval and Informed Consent

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Plagiarism Statement

This article has been scanned by iThenticate ™.