Leevy and Khoshgoftaar *J Big Data* (2020) 7:104
https://doi.org/10.1186/s40537-020-00382-x

Journal of Big Data

SURVEY PAPER Open Access

A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data

Joffrey L. Leevy* and Taghi M. Khoshgoftaar

*Correspondence:
jleevy2017@fau.edu
Florida Atlantic University,
777 Glades Road, Boca Raton,
FL 33431, USA

Abstract

The exponential growth in computer networks and network applications worldwide has been matched by a surge in cyberattacks. For this reason, datasets such as CSE-CIC-IDS2018 were created to train predictive models on network-based intrusion detection. These datasets are not meant to serve as repositories for signature-based detection systems, but rather to promote research on anomaly-based detection through various machine learning approaches. CSE-CIC-IDS2018 contains about 16,000,000 instances collected over the course of ten days. It is the most recent intrusion detection dataset that is big data, publicly available, and covers a wide range of attack types. This multi-class dataset has a class imbalance, with roughly 17% of the instances comprising attack (anomalous) traffic. Our survey work contributes several key findings. We determined that the best performance scores for each study, where available, were unexpectedly high overall, which may be due to overfitting. We also found that most of the works did not address class imbalance, the effects of which can bias results in a big data study. Lastly, we discovered that information on the data cleaning of CSE-CIC-IDS2018 was inadequate across the board, a finding that may indicate problems with reproducibility of experiments. In our survey, major research gaps have also been identified.

Keywords: CSE-CIC-IDS2018, Big data, Intrusion detection, Machine learning, Class imbalance

Introduction

Intrusion detection is the accurate identification of various attacks capable of damaging or compromising an information system. An (IDS) can be host-based, network-based, or a combination of both. A host-based IDS is primarily concerned with the internal monitoring of a computer. Windows registry monitoring, log analysis, and file integrity checking are some of the tasks performed by a host-based IDS [1]. A network-based IDS monitors and analyzes network traffic to detect threats that include *Denial-of-Service* (DoS) attacks, SQL injection attacks, and password attacks [2]. The rapid growth of computer networks and network applications worldwide has encouraged an increase in cyberattacks [3]. In 2019, business news channel CNBC reported that the average cost of a cyberattack was $200,000 [4].

An IDS can also be categorized as signature-based or anomaly-based. A signature-based IDS contains patterns for known attacks and is unable to detect unknown attacks. This means that the database of a signature-based IDS must be updated ad nauseam to keep up with all known attack signatures. By contrast, an anomaly-based IDS identifies deviations from normal traffic behavior. Since various machine learning approaches can generally be successfully applied to anomaly detection, it makes intuitive sense that anomaly-based intrusion detection is a productive research area.

Datasets such as CSE-CIC-IDS2018 [5] were created to train predictive models on anomaly-based intrusion detection for network traffic. CSE-CIC-IDS2018 is not an entirely new project, but part of an existing project that produces modern, realistic datasets in a scalable manner [6]. In the next three paragraphs we trace the development of this project, from the foundational dataset (ISCXIDS2012 [7]) to CSE-CIC-IDS2018.

Created in 2012 by the Information Security Centre of Excellence (ISCX) at the University of New Brunswick (UNB) over a seven-day period, ISCXIDS2012 contains both normal and anomalous network traffic. The dataset contains several attack types (e.g. DoS, Distributed Denial-of-Service (DDoS), and brute force), but these have all been labeled as "attack" [8]. ISCXIDS2012 is big data, with 20 independent features and 2,450,324 instances, of which roughly 2.8% typifies attack traffic. Big data is associated with specific properties, such as volume, variety, velocity, variability, value, and complexity [9]. These properties may make classification more challenging for learners trained on big data. Hereafter, "ISCXIDS2012" will be referred to as "ISCX2012" throughout the text.

In 2017, the creators of ISCX2012 and the Canadian Institute of Cybersecurity (CIC) acted on the fact that the dataset was limited to only six traffic protocols (HTTP, SMTP, SSH, IMAP, POP3, FTP). A case in point was the lack of representation of HTTPS, an important protocol accounting for about 70% of current network traffic in the real world [5]. Also, the distribution of simulated attacks did not conform to reality. CIC-IDS2017, which contains five days of network traffic, was released to remedy the deficiencies of its predecessor. Among the many benefits of this new dataset, the high number of features (80) facilitates machine learning. CICIDS2017 contains 2,830,743 instances, with attack traffic amounting to about 19.7 % of this total number. The dataset has a class imbalance and a wider range of attack types than ISCX2012. Class imbalance, which is a phenomenon caused by unequal distribution between majority and minority classes, can skew results in a big data study. At a granular level, CICIDS2017 has a high class imbalance with respect to some of the individual attack types. High class imbalance is defined by a majority-to-minority ratio between 100:1 and 10,000:1 [10].

The Communications Security Establishment (CSE) joined the project, and in 2018, the latest iteration of the intrusion detection dataset was released, CSE-CIC-IDS2018. The updated version also has a class imbalance and is structurally similar to CICIDS2017. However, CSE-CIC-IDS2018 was prepared from a much larger network of simulated client-targets and attack machines [11], resulting in a dataset that contains 16,233,002 instances gathered from 10 days of network traffic. About 17% of the instances is attack traffic. Table 1 shows the percentage distribution for the seven types of network traffic represented by CSE-CIC-IDS2018. Hereafter, "CSE-CIC-IDS2018" and "CICIDS2018" will be used interchangeably throughout the text. The dataset is distributed over ten

Table 1 CICIDS2018: Network traffic distribution

Traffic type	Distribution (%)
Benign	83.070
DDoS	7.786
DoS	4.031
Brute force	2.347
Botnet	1.763
Infiltration	0.997
Web attack	0.006

CSV files that are downloadable from the cloud.[1] Nine files consist of 79 independent features, and the remaining file consists of 83 independent features.

Our exhaustive search for relevant, peer-reviewed papers ended on September 22, 2020. To the best of our knowledge, this is the first survey to exclusively present and analyze intrusion detection research on CICIDS2018 in such detail. CICIDS2018 is the most recent intrusion detection dataset that is big data, publicly available, and covers a wide range of attack types. The contribution of our survey centers around three important findings. In general, we observed that the best performance scores for each study, where provided, were unusually high. This may be a consequence of overfitting. The second finding deals with the apparent lack of concern in most studies for the class imbalance of CICIDS2018. Finally, we note that for all works, the data cleaning of CICIDS2018 has been given little attention, a shortcoming that could hinder reproducibility of experiments. Data cleaning involves the modification, formatting, and removal of data to enhance dataset usability [12].

The remainder of this paper is organized as follows: "Research papers using CICIDS2018" section describes and analyzes the compiled works; "Discussion of surveyed works" section discusses survey findings, identifies gaps in the current research, and explains the performance metrics used in the curated works; and "Conclusion" section concludes with the main points of the paper and offers suggestions for future work.

Research papers using CICIDS2018

In this section, we examine research papers that use CICIDS2018. Works of research are presented in alphabetical order by author. All scores obtained from metrics (accuracy, recall, etc.) are the best scores in each study for binary classification [13].

Table 2 provides an alphabetical listing by author of the papers discussed in this section, along with the best respective performance score(s) for CICIDS2018. Comparisons between scores for separate works of research or separate experiments in the same paper may not be valid. This is because datasets may differ in the number of instances and features, and possibly the choice of computational framework. Furthermore, variations of an original experiment may be performed on the same dataset. However, providing

[1] https://www.unb.ca/cic/datasets/ids-2018.html.

Table 2 CICIDS2018: Performance scores

Authors	Accuracy[a]	Precision[a]	Recall[a]	AUC
Atefinia and Ahmadi [14]	100.00	100.00	100.00	n/a
Basnet et al. [15]	99.00	n/a	n/a	n/a
Catillo et al. [16]	99.20	95.00	98.90	n/a
Chadza et al. [17]	97.00	n/a	n/a	n/a
Chastikova and Sotnikov [18]	n/a	n/a	n/a	n/a
D'hooge et al. [19]	96.00	99.00	79.00	n/a
Ferrag et al. [20]	97.38	n/a	98.18	n/a
Filho et al. [21]	n/a	100.00	100.00	n/a
Fitni and Ramli [22]	98.80	98.80	97.10	0.94
Gamage and Samarabandu [23]	98.40	97.79	98.27	n/a
Hua [24]	98.37	98.14	98.37	n/a
Huancayo Ramos et al. [25]	99.99	100.00	99.99	n/a
Kanimozhi and Jacob [26]	100.00	100.00	100.00	1
Kanimozhi and Jacob [27]	99.97	99.96	100.00	1
Karatas et al. [28]	99.69	99.70	99.69	n/a
Kim et al. [29]	99.99	81.75	82.25	n/a
Li et al. [30]	n/a	n/a	100.00	1
Lin et al. [31]	96.20	96.00	96.00	n/a
Zhao et al. [32]	97.90	98.00	98.00	n/a

[a] All scores shown are percentages

Table 3 CICIDS2018: Proposed models

Authors	Proposed model(s)
Atefinia and Ahmadi [14]	Modular Deep Neural Network
Basnet et al. [15]	MLP
Catillo et al. [16]	Deep Autoencoder
Chadza et al. [17]	Baum Welch, Viterbi
Chastikova and Sotnikov [18]	LSTM
D'hooge et al. [19]	XGBoost
Ferrag et al. [20]	RNN, Deep Autoencoder
Filho et al. [21]	Random Forest
Fitni and Ramli [22]	Logistic Regression + Decision Tree + Gradient Boosting
Gamage and Samarabandu [23]	Deep Feed-forward Neural Network
Hua [24]	LightGBM
Huancayo Ramos et al. [25]	Random Forest, Decision Tree
Kanimozhi and Jacob [26]	MLP
Kanimozhi and Jacob [27]	MLP
Karatas et al. [28]	Adaboost
Kim et al. [29]	CNN
Li et al. [30]	Deep Autoencoder
Lin et al. [31]	LSTM
Zhao et al. [32]	Deep Autoencoder

Table 4 CICIDS2018: Computing environment

Authors	Computing environment
Atefinia and Ahmadi [14]	Python, Scikit-learn
Basnet et al. [15]	fast.ai, GPU, Python
Catillo et al. [16]	Keras, Python, Tensorflow
Chadza et al. [17]	MATLAB 2019a, Snort IDS
Chastikova and Sotnikov [18]	n/a
D'hooge et al. [19]	Python, Scikit-learn, XGBoost
Ferrag et al. [20]	Google Colab, GPU, Python,Tensorflow
Filho et al. [21]	Python, Scikit-learn
Gamage and Samarabandu [23]	GPU (on some PCs)
Fitni and Ramli [22]	Python, Scikit-learn
Hua [24]	Scikit-learn, Tensorflow
Huancayo Ramos et al. [25]	Python, Scikit-learn
Kanimozhi and Jacob [26]	Python, Scikit-learn
Kanimozhi and Jacob [27]	Python, Scikit-learn
Karatas et al. [28]	GPU, Keras, Python, Scikit-learn, Tensorflow
Kim et al. [29]	Python, Tensorflow
Li et al. [30]	Python
Lin et al. [31]	Tensorflow
Zhao et al. [32]	Python, Tensorflow

these scores may be valuable for future comparative research. Table 3 provides an alphabetical listing by author of the papers discussed in this section, along with the proposed respective model(s) for CICIDS2018, and Table 4 shows the same ordered listing by author coupled with the associated computing environment(s) for CICIDS2018.

Atefinia and Ahmadi [14] (Network intrusion detection using multi-architectural modular deep neural network)

Using an aggregator module [33] to integrate four network architectures, the authors aimed to obtain a higher precision rate than any of those produced in the related works described in their paper. The four network components included a restricted Boltzmann machine [34], a deep feed-forward neural network [35], and two Recurrent Neural Networks (RNNs) [36], specifically a Long Short-term Memory (LSTM) [37] and a Gated Recurrent Unit (GRU) [38]. The models were implemented with Python and the Scikit-learn[2] library. Data preprocessing involved the removal of source and destination IP addresses and also source port numbers. Labels with string values were one-hot encoded, and feature scaling was used to normalize the feature space of all the attributes between a range of 0 and 1. Rows with missing values and columns with too many missing values were dropped from CICIDS2018. However, no information is provided on how many rows and columns were removed. Stratified sampling with a train to test

[2] https://scikit-learn.org.

ratio of 80-20 was performed on each of the four modules. Information was then fed to the aggregator module, which used a weighted averaging technique to produce the output for the modular network. The highest accuracies (100%) were obtained for the DoS, DDoS, and brute force attack types. These accuracies were associated with precision and recall scores of 100%. One drawback is the authors' comparison of results from their study with results from the related works. A better approach is to empirically evaluate at least two models, one of which would be the proposed modular network. Another shortcoming relates to the non-availability of performance scores that cover the collective attack types. In other words, the scores of precision, recall, etc. for the combination of attacks could provide additional insight. This does not detract from the usefulness of reporting precision, recall, etc. for each attack type.

Basnet et al. [15] (Towards detecting and classifying network intrusion traffic using deep learning frameworks)

The authors experimented with various deep learning frameworks (fast.ai,[3] Keras,[4] PyTorch,[5] TensorFlow,[6] Theano[7]) to detect network intrusion traffic and classify attack types. For preprocessing, samples with "Infinity", "NaN", or missing values were dropped and timestamps converted to Unix epoch numeric values (number of seconds since January 1, 1970). About 20,000 samples were dropped after the data cleanup process. The destination port and protocol features were treated as categorical data, and the remainder were treated as numerical data. Ten-fold cross validation with either an 80–20 or 70–30 split was used for training and testing. Both binary class and multi-class classification [39] were considered. A Multilayer Perceptron (MLP) [40] served as the only classifier. With the aid of GPU acceleration, the authors observed that fast.ai outperformed the other frameworks consistently among all the experiments, yielding an optimum accuracy of 99% for binary classification. The main limitation of this study is the use of only one classifier.

Catillo et al. [16] (2L-ZED-IDS: a two-level anomaly detector for multiple attack classes)

Based on an extension of previous research with CICIDS2017, this study trained a deep autoencoder [41] on CICIDS2017 and CICIDS2018. In the preprocessing stage, the Flow_ID and Timestamp features of the datasets were not selected because they were deemed not relevant to the study. The autoencoder was implemented with Python, Keras, and TensorFlow and trained on normal and DoS attack traffic. The train to test ratio was 80–20 for both datasets. The highest accuracy for CICIDS2018 (99.2%) was obtained for the botnet attack type, corresponding to a precision of 95.0% and a recall of 98.9%. The highest accuracy (99.3%) of the entire study was obtained for CICIDS2017 (botnet attack type), coupled with a precision of 94.8% and a recall of 98.6%. One drawback of the study is the non-availability of an accuracy score for the collective attack types. Another disadvantage is the use of only one classifier.

[3] https://docs.fast.ai/.

[4] https://github.com/keras-team/keras.

[5] https://pytorch.org/.

[6] https://www.tensorflow.org/.

[7] http://deeplearning.net/software/theano/.

Chadza et al. [17] (Contemporary sequential network attacks prediction using hidden Markov Model)

By way of MATLAB software, two conventional Hidden Markov Model (HMM) training algorithms, namely Baum Welch [42] and Viterbi [43], were applied to CIC-IDS2018. HMM is a probabilistic machine learning framework that generates states and observations. For this study, information is clearly lacking on data preprocessing. About 457,550 records (selection criteria set in Snort Intrusion Detection System [44]) were selected from CICIDS2018. From that sample of records, 70% were allocated to training and the remainder to testing. The authors found that the highest accuracy of about 97% was achieved by both the Baum Welch and Viterbi algorithms. This paper is only three pages in length. The main shortcoming of this work is the lack of detail on the experiments performed.

Chastikova and Sotnikov [18] (Method of analyzing computer traffic based on recurrent neural networks)

This highly theoretical study, which was submitted to the Journal of Physics Conference series, does not give any empirical results and is extremely short on details. It merely proposes a LSTM model to analyze computer network traffic using CIC-IDS2018. The authors note that their use of the Focal Loss function [45] (initially developed by Facebook AI research) addresses class imbalance. The fact that no metrics have been used and no computing environment was provided is a major drawback of this six-page paper.

D'hooge et al. [19] (Inter-dataset generalization strength of supervised machine learning methods for intrusion detection)

By including both CICIDS2017 and CICIDS2018, this study investigates how efficiently the results of an intrusion detection dataset can be generalized. For performance evaluation, the authors used 12 supervised learning algorithms from different families: *Decision Tree* (DT) [46], *Random Forest* (RF) [47], DT-based Bagging [48], Gradient Boosting [49], Extratree [50], Adaboost [51], XGBoost [52], *k-Nearest Neighbor* (*k*-NN) [53], Ncentroid [54], linearSVC [55], RBFSVC [56], and Logistic Regression [57]. The models were built within a Python framework with Scikit-learn and XGBoost modules. Feature scaling was used to normalize the feature space of all the attributes. The tree-based classifiers performed best, and among them, XGBoost ranked first. The top accuracy, precision, and recall scores for CICIDS2018 were 96%, 99%, and 79%, respectively. For intrusion detection, the authors concluded that a model trained on one dataset (CICIDS2017) cannot generalize to another dataset (CICIDS2018). One shortcoming of this work is the assumption that certain categorical features, such as destination port, have the same number of unique values for both datasets. Drawing upon an example, we expound on this limitation: a classifier trained on a dataset with feature 'X' of {car, boat}

is not expected to generalize well to a related dataset with feature 'X' of {car, boat, train, plane}.

Ferrag et al. [20] (Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study)

Seven deep learning models were evaluated on CICIDS2018 and the Bot-IoT dataset [58]. The models included RNNs, deep neural networks [59], restricted Boltzmann machines, deep belief networks [60], Convolutional Neural Networks (CNNs) [61], deep Boltzmann machines [62], and deep autoencoders. The Bot-IoT dataset is a 2018 creation from the University of New South Wales (UNSW) that incorporates about 72,000,000 normal and botnetInternet of Things (IoT) instances with 46 features. The experiment was performed on Google Colaboratory[8] using Python and TensorFlow with GPU acceleration. Only 5% of the instances were used in this study, as determined by [58]. The highest accuracy for the Bot-IoT dataset (98.39%) was obtained with a deep autoencoder, while the highest accuracy for CICIDS2018 (97.38%) was obtained with an RNN. The highest recall for the Bot-IoT dataset (97.01%) came from a CNN, whereas the highest recall for CICIDS2018 (98.18%) came from a deep autoencoder. The bulk of this paper deals with classifying 35 cyber datasets and describing the seven deep learning models. Only the last three pages discuss the actual experimentation, which is lacking in detail. This is a major shortcoming of the study.

Filho et al. [21] (Smart detection: an online approach for DoS/DDoS attack detection using machine learning)

The authors used a customized dataset and four well-known ones (CIC-DoS [63], ISCX2012, CICIDS2017, and CICIDS2018) to obtain online random samples of network traffic and classify them as DoS attacks or normal. There were 33 features obtained for each dataset. These features were derived from source and destination ports, transport layer protocol, IP packet size, and TCP flags. The individual datasets were combined into one unit containing normal traffic (23,088 instances), TCP flood attacks (14,988 instances), UDP flood (6,894 instances), HTTP flood (347 instances), and HTTP slow (183 instances). For the combined dataset, the authors noted that ISCX2012 was only used to provide data traffic with normal activity behavior. Recursive Feature Elimination with Cross Validation [64] was performed on six learners (RF, DT, Logistic Regression, SGDClassifier [65], Adaboost, MLP). The learners were built with Scikit-learn. With regard to the combined dataset, Random Forest (20 features selected) had the highest accuracy among the learners. For CICIDS2018, the precision and recall for RF were both 100%. One shortcoming of this study is the use of ISCX2012 to provide normal traffic for the combined dataset. ISCX2012 is outdated, and as we previously pointed out, it is limited to only six traffic protocols.

Fitni and Ramli [22] (Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems)

Adopting an ensemble model approach, this study compared seven single learners to evaluate the top performers for integration into a classifier unit. The seven learners are as follows: RF, Gaussian Naive Bayes [66], DT, Quadratic Discriminant Analysis [67],

[8] https://colab.research.google.com/.

Gradient Boosting, and Logistic Regression. The models were built with Python and Scikit-learn. During preprocessing, samples with missing values and infinity were removed. Records that were actually a repetition of the header rows were also removed. The dataset was then divided into training and testing validation sets in an 80-20 ratio. Feature selection [68], a technique for selecting the most important features of a predictive model, was performed using the Spearman's rank correlation coefficient [69] and Chi-squared test [70], resulting in the selection of 23 features. After the evaluation of the seven learners with these features, Gradient Boosting, Logistic Regression, and DT emerged as the top performers for use in the ensemble model. Accuracy, precision, and recall scores for this model were 98.80%, 98.80%,and 97.10%, respectively, along with an Area Under the Receiver Operating Characteristic Curve (AUC) of 0.94. We believe that the performance of the ensemble model could be improved by substituting a Catboost [71], LightGBM [72], or XGBoost classifier for the Gradient Boosting classifier. The three possible substitutions are enhanced gradient boosting variants [73].

Gamage and Samarabandu [23] (Deep learning methods in network intrusion detection: a survey and an objective comparison)

This work introduces a taxonomy of deep learning models for intrusion detection and summarizes relevant research papers. Four deep learning models (feed-forward neural network, autoencoder, deep belief network, LSTM) were then evaluated on the KDD Cup 1999 [74], NSL-KDD [75], CICIDS2017, and CICIDS2018 datasets. The KDD Cup 1999 dataset, which was developed by Defense Advanced Research Project Agency (DARPA), contains four categories of attacks, including the DoS category. Preprocessing of the datasets consisted of removing invalid flow records (missing values, strings, etc.) and feature scaling. One-hot encoding was done for the protocol type, service, and flag features, three attributes that are categorical and non-numerical. The full datasets were split into train and test sets, with hyperparameter tuning applied. Results show that the feed-forward neural networks performed well on all four datasets. For this classifier, the highest accuracy (99.58%) was obtained on the CICIDS2017 dataset. This score is associated with a precision of 99.43% and a recall of 99.45%. With respect to CICIDS2018, the highest accuracy for the feed-forward network was 98.4%, corresponding to a precision of 97.79% and a recall of 98.27%. GPU acceleration was used on the some of the PCs involved in the experiments. One shortcoming of this study stems from the use of KDD Cup 1999 and NSL-KDD, both of which are outdated and have known issues. The main problem with KDD Cup 1999 is its significant number of duplicate records [75]. NSL-KDD is an improved version that does not have the issue of redundant instances, but it is far from ideal. For example, there are some attack classes without records in the test dataset of NSL-KDD [12].

Hua [24] (An efficient traffic classification scheme using embedded feature selection and LightGBM)

To tackle the class imbalance of CICIDS2018, the author incorporates an undersampling and embedded feature selection approach with a LightGBM classifier. LightGBM contains algorithms that address high numbers of instances and features in datasets.

Undersampling [76] randomly removes majority class instances to affect distribution. During the data cleaning stage of this study, missing values and useless features were removed from the full dataset, resulting in a modified set of 77 features. String labels were consequently converted to integer labels, which were then one-hot encoded. Six other learners were evaluated in this research work: Support Vector Machine (SVM) [47], RF, Adaboost, MLP, CNN, and Naive Bayes [77]. Learners were implemented with Scikit-learn and TensorFlow. The train to test ratio was 70–30, and the XGBoost algorithm was used to perform feature selection. The LightGBM classifier had the best performance of the group, with an optimum accuracy of 98.37% when the sample size was three million and the top ten features were selected. For this accuracy, the precision and recall were 98.14% and 98.37%, respectively. LightGBM also had the second fastest training time among the classifiers. Although this study provides more information about data preprocessing than other works in our survey, it deals with data cleaning in a superficial matter.

Huancayo Ramos et al. [25] (Benchmark-based reference model for evaluating Botnet detection tools driven by traffic-flow analytics)

Five learners were evaluated on two datasets (CICIDS2018 and ISOT HTTP Botnet [78]) to determine the best botnet classifier. The ISOT HTTP Botnet dataset contains malicious and benign instances of Domain Name System (DNS) traffic. The learners in the study include RF, DT, *k*-NN, Naive Bayes, and SVM. Feature selection was performed using various techniques, including the feature importance method [79] of RF. After feature selection, CICIDS2018 had 19 independent attributes while ISOT HTTP had 20, with destination port number, source port number, and transport protocol among the selected features. The models were implemented with Python and Scikit-learn. Five-fold cross-validation was applied to a training set comprising 80% of the botnet instances. The remainder of the botnet instances served as the testing set. For optimization, the Grid Search algorithm [80] was used. With regard to CICIDS2018, the RF and DT learners scored an accuracy of 99.99%. Tied to this accuracy, the precision was 100% and the recall was 99.99% for both learners. The RF and DT learners also had the highest accuracy for ISOT HTTP (99.94% for RF and 99.90% for DT). One limitation of this paper is the inadequate information provided on data preprocessing.

Kanimozhi and Jacob [26] (Artificial litelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing)

The authors trained a two-layer MLP, implemented with Python and Scikit-learn, to detect botnet attacks. GridSearchCV [81] performed hyper-parameter optimization, and L2 regularization [82] was used to prevent overfitting. Overfitting refers to a model that has memorized training data instead of learning to generalize it [83]. The MLP classifier was trained only on the botnet instances of CICIDS2018, with ten-fold cross validation [84] implemented. For this study the AUC was 1, which is a perfect score. Related accuracy, precision, and recall scores were all 100%. The paper is four pages long (with two references), and one major shortcoming is an obvious lack of detail. Another drawback is the use of only one classifier to evaluate performance.

Kanimozhi and Jacob [27] (Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing)

The purpose of this study was to determine the best classifier out of six candidates (MLP, RF, *k*-NN, SVM, Adaboost, Naive Bayes). The models were developed with Python and Scikit-learn. A calibration curve was used, which is a graph showing the deviation of classifiers from a perfectly calibrated plot. Botnet instances of CIC-IDS2018 were split into train and test instances, with no information provided on the ratio of train to test instances. The MLP model emerged as the top choice with an AUC of 1. Accuracy, precision, and recall scores associated with this perfect AUC score were 99.97%, 99.96%, and 100%, respectively. No information was provided on the MLP classifier, but it is most likely the same two-layer network as in [26]. The main shortcoming of this paper is the lack of detail.

Karatas et al. [28] (Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset)

Using the Synthetic Minority Oversampling Technique (SMOTE) [85] algorithm to address class imbalance, the authors evaluated the performance of six learners on CICIDS2018. The classifiers involved were *k*-NN, RF, Gradient Boosting, Adaboost, DT, and Linear Discriminant Analysis [86]. The learners were developed in a Python environment using Keras, TensorFlow, and Scikit-learn. According to the authors, CICIDS2018 contains about 5,000,000 samples. However, the full dataset inarguably contains about 16,000,0000 instances, so the authors should clearly indicate that a subset was used. The dataset was preprocessed to address issues such as missing values and "Infinity." In addition, one-hot encoding was used, and rows were shuffled for randomness. Five-fold cross-validation was applied to a training set comprising 80% of the instances. The remaining instances served as the test set. After SMOTE was applied, the total dataset size increased by 17%. The Adaboost learner was shown to be the best performer, with an accuracy of 99.69%, along with precision and recall scores of 99.70% and 99.69%, respectively. In our opinion, this study should have gone into a little more detail on data cleaning. Nevertheless, among the surveyed works, this paper has done the best job at covering data cleaning.

Kim et al. [29] (CNN-based network intrusion detection against denial-of-service attacks)

In this study, the authors trained a CNN on DoS datasets from KDD Cup 1999 and CICIDS2018. The model was implemented with Python and TensorFlow. For both datasets, the train to test ratio was 70–30. In the case of KDD, the authors used about 283,000 samples, and for CICIDS2018, about 11,000,000. Image datasets were subsequently generated, and binary and multi-class classification was performed. The authors established that for the two datasets, the accuracy was about 99% for binary classification, which corresponded to precision and recall scores of 81.75% and 82.25%, respectively. An RNN model was subsequently introduced into the study for comparative purposes. The main drawback of this work arises from the use of the

KDD Cup 1999 dataset, which, as previously discussed, is an outdated dataset with a high number of redundant instances.

Li et al. [30] (Building auto-encoder intrusion detection system based on random forest feature selection)

In this online real-time detection study, unsupervised clustering and feature selection play a major role. For preprocessing, "Infinity" and "NaN" values were replaced by 0, and the data was then divided into sparse and dense matrices, normalized by L2 regularization. A sparse matrix has a majority of elements with value 0, while a dense matrix has a majority of elements with non-zero values. The model was built within a Python environment. The best features were selected by RF, and the train to test ratio was set as 85–15. The Affinity Propagation (AP) clustering [87] algorithm was subsequently used on 25% of the training dataset to group features into subsets, which were relayed to the autoencoder. Recall rates for all attack types for the proposed model were compared with those of another autoencoder model called Kitnet [88]. Several attack types for both models had a recall of 100%. Only the proposed model was evaluated with the AUC metric, with several attack types yielding a score of 1. Based on detection time results, the authors showed that their model has a faster detection time than KitNet. The authors provided performance scores for AUC and recall for each attack type of CICIDS2018. This is a deficiency of the study as scores covering the collective attack types could provide additional insight. The absence of AUC values for Kitnet is another shortcoming.

Lin et al. [31] (Dynamic network anomaly detection system by using deep learning techniques)

The authors investigated the use of Attention Mechanism (AM) [89] with LSTM to improve performance. Attention mechanism imitates the focus mechanism of the human brain, extracting and representing information most relevant to the target through an automatic weighing scheme. The model was built with TensorFlow and further optimized with Adam Gradient Descent [90], a replacement algorithm for Stochastic Gradient Descent [91]. Seven other learners (DT, Gaussian Naive Bayes, RF, *k*-NN, SVM, MLP, LSTM without AM) were also evaluated. Preprocessing of a CICIDS2018 subset (about 50% of the original size) involved removing the timestamp feature and IP address feature. The dataset was then divided into training, test, and validation sets in the ratios of 90%, 9%, and 1%. Normal dataset traffic was randomly undersampled to obtain 2,000,000 records, while Web and infiltration attacks were oversampled with SMOTE to address class imbalance. The LSTM model with AM outperformed the other learners with an accuracy of 96.2% and a precision and recall of 96%. The contribution of this useful study is limited by the inadequate information provided on data cleaning. Another shortcoming is the omission of the oversampling rate for SMOTE.

Zhao et al. [32] (A semi-self-taught network intrusion detection system)

The authors used a denoising autoencoder [92] with a heuristic method of class separation based on the fuzzy c-means algorithm [93]. This approach was adopted to get rid of samples with problems such as missing values and redundant data. However, it is ineffective against class noise. Class noise is caused either by different class labels for

duplicate instances or by misclassified instances [94]. The autoencoder was developed using Python and TensorFlow. Training, validation, and test sets comprised 70%, 15%, and 15% of the data, respectively. The highest accuracy obtained was 97.9%, accompanied by a score of 98.0% for both precision and recall. One limitation of this study is a lack of details about the experiments. Another limitation is the use of only one learner.

Discussion of surveyed works

In general, the best performance scores are unusually high for studies where scores are provided. This finding is notable. Accuracy scores are between 96 (D'hooge et al., 2020) and 100 (Atefinia & Ahmadi, 2020; Kanimozhi & Jacob, 2019a). Several papers show recall scores of 100 (Atefinia & Ahmadi, 2020; Kanimozhi & Jacob, 2019a; Kanimozhi & Jacob, 2019b; Li et al., 2020; Filho et al., 2019) and also precision scores of 100 (Atefinia & Ahmadi, 2020; Kanimozhi & Jacob, 2019a; Huancayo Ramos et al., 2020; Filho et al., 2019). In addition, three studies show a perfect AUC score (Kanimozhi & Jacob, 2019a; Kanimozhi & Jacob, 2019b ;Li et al., 2020). These noticeably high scores for the various metrics may be due to overfitting.

Surprisingly, use of the accuracy metric is prevalent throughout the surveyed works, while use of the AUC metric has only been used in four studies (Fitni & Ramli, 2020; Kanimozhi & Jacob, 2019a; Kanimozhi & Jacob, 2019b; Li et al., 2020). This observation relates to the class imbalance of CICIDS2018. The high imbalance makes identification of the minority class more burdensome for learners, especially in the case of big data, and tends to introduce a bias in favor of the majority class. Hence, the use of accuracy alone may not be beneficial since a deceptively high score could be obtained when the influence of the minority class is greatly reduced. It is always better to provide accuracy along with other metrics, such as precision and recall, and in all fairness, most of the works have shown this. We point out that the use of AUC as a robust, standalone metric for class imbalance has been demonstrated in several studies [95–97]. Please see "Performance metrics" for an explanation of the various metrics provided.

As mentioned previously, the CICIDS2018 dataset has a class imbalance. The effects of this imbalance can be mitigated by techniques at the data level (e.g. random undersampling, feature selection) and algorithm level (e.g. cost-sensitive classification, ensemble techniques) [9]. We make the important observation that less than half of the curated papers discuss techniques for addressing the high imbalance of CICIDS2018. Hua, 2020, for example, has highlighted the use of embedded feature selection and undersampling with a LightGBM classifier.

None of the papers satisfactorily discuss the data cleaning of CICIDS2018. This is a significant revelation. About 60% of data scientists believe that no task is more time-consuming than data cleaning [12]. A discussion of data cleaning in a research paper should provide detailed information on all rows and columns of a dataset that have been dropped or modified, along with a rationale for these actions. Insufficient information on data cleaning in a study can make duplication of an experiment problematic for outside researchers. Data cleaning is a subset of data preprocessing, a task that makes a dataset more usable. It is important to note that data preprocessing should be performed on a dataset such as CICIDS2018 before learners are trained, as failure to do so could lead to inaccurate analytics.

Another important consideration pertains to the use of outdated datasets, such as KDD 1999, NSL-KDD, and ISCX2012, alongside CICIDS2018 in a study. For some or all attack traffic embodied within these older datasets, patches have long been issued and updated software versions hardened. A much greater concern, however, are the issues (discussed in Section 2) associated with these datasets. Researchers using intrusion detection datasets that are outdated should thoroughly understand how these known issues could affect the outcome of experiments.

Finally, our survey shows that statistical analysis of performance scores appears to have been overlooked. Determining the statistical significance of these scores provides clarity, and there are some established techniques for doing this, such as ANalysis Of VAriance (ANOVA) [98] and Tukey's Honestly Significant Difference (HSD) [99]. ANOVA reveals whether the means of one or more independent factors are significant. Tukey's HSD ascribes group letters to means that are significantly different from each other.

Gaps in current research

Significant gaps exist in intrusion detection research with CICIDS2018. Topics such as big data processing frameworks, concept drift, and transfer learning are missing from the literature. We explain further in the following paragraphs.

There are specialized frameworks for handling the processing and analysis of big data, where computations are enhanced by the utilization of computing clusters and parallel algorithms. One example is Apache Hadoop, an open source variant of the MapReduce framework, which divides a dataset into subsets for easier processing and then recombines the partial solutions [100]. The Apache Spark framework, another example, enables faster distributed computing by using in-memory operations [101]. Apache Spark is currently one of the most popular engines for big data processing, and we encourage researchers to evaluate learner performance on CICIDS2018 with this framework.

Concept drift is the variation of data distributions over time [102]. For example, a model trained today on CICIDS2018 may have a lower optimum recall score in five or ten years when tested against an up-to-date intrusion detection dataset. As discussed previously, some of the attack instances in a modern dataset would be rendered ineffective in the future (patches, updated software, etc.) and not reflect current reality. Research examining the effect of time on intrusion detection models is a promising area.

Transfer learning attempts to boost the performance of target learners on target domains by transferring knowledge from related but different source domains [103]. The aim is to construct models with a reduced number of target data instances. Within the context of intrusion detection, Singla et al. [104] note that models are better able to identify new attacks, through transfer learning, when the training data is limited. We surmise that CICIDS2018, with its ample supply of instances, could serve as an ideal source domain.

Performance metrics

In order to explain the metrics provided in this survey, it is necessary to start with the fundamental metrics and then build on the basics. Our list of applicable performance metrics is explained as follows:

- True Positive (TP) is the number of positive instances correctly identified as positive.
- True Negative (TN) is the number of negative instances correctly identified as negative.
- False Positive (FP), also known as Type I error, is the number of negative instances incorrectly identified as positive.
- False Negative (FN), also known as Type II error, is the number of positive instances incorrectly identified as negative.

Based on these fundamental metrics, the other performance metrics are derived as follows:

- *Recall*, also known as sensitivity or True Positive Rate (TPR), is equal to $TP/(TP + FN)$.
- *Precision*, also known as positive predictive value, is equal to $TP/(TP + FP)$.
- *Fall-Out*, also known as False Positive Rate (FPR), is equal to $FP/(TP + FN)$.
- *Accuracy* is equal to $(TP + TN)/(TP + TN + FP + FN)$.
- AUC provides the area under the Receiver Operating Characteristic (ROC) curve, which plots TPR against FPR for various classification cut-offs. The behavior of a classifier is shown across all thresholds of the ROC curve. AUC is a popular metric that counters the adverse effects of class imbalance. A model with 100% correct predictions has an AUC of 1, while a model with 100% incorrect predictions has an AUC of 0.

Conclusion

A marked increase in cyberattacks has shadowed the rapid growth of computer networks and network applications. In light of this, several intrusion detection datasets, including CICIDS2018, have been created to train predictive models. CICIDS2018 is multi-class, contains about 16,000,000 instances, and is class-imbalanced. As late as September 22, 2020, we aggressively searched for relevant studies based on this dataset.

For the most part, we observed that the best performance scores for each study, where provided, were unusually high. This may be attributable to overfitting. Furthermore, we note that only a few of the surveyed works explored treatment for the class imbalance of CICIDS2018. Class imbalance, particularly for big data, can skew the results of an experiment. As a final point, we emphasize that the level of detail paid to the data cleaning of CICIDS2018 failed to meet our expectations. This concern has a bearing on the reproducibility of experiments.

Several gaps have been identified in the current research. Topics such as big data processing frameworks, concept drift, and transfer learning are missing from the literature. Future work should address these gaps.

Abbreviations

AM: Attention Mechanism; ANOVA: ANalysis Of VAriance; AP: Affinity Propagation; AUC: Area Under the Receiver Operating Characteristic Curve; CIC: Canadian Institute of Cybersecurity; CNN: Convolutional Neural Network; CSE: Communications Security Establishment; DARPA: Defense Advanced Research Project Agency; DNS: Domain Name System; DDoS: Distributed Denial-of-Service; DoS: Denial-of-Service; DT: Decision Tree; FN: False Negative; FNR: False Negative Rate; FP: False Positive; FPR: False Positive Rate; GRU: Gated Recurrent Unit; HMM: Hidden Markov Model; HSD: Honestly Significant

Difference; IDS: Intrusion Detection System; IoT: Internet of Things; ISCX: Information Security Centre of Excellence; *k*-NN: k-Nearest Neighbor; LSTM: Long Short-term Memory; MLP: Multilayer Perceptron; NSF: National Science Foundation; RF: Random Forest; RNN: Recurrent Neural Network; ROC: Receiver Operating Characteristic; SMOTE: Synthetic Minority Oversampling Technique; SRBMM: Synthetic Minority Oversampling Technique; SVM: Support Vector Machine; TN: True Negative; TNR: True Negative Rate; TP: True Positive; TPR: True Positive Rate; UNB: University of New Brunswick; UNSW: University of New South Wales.

Acknowledgements
We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University. Additionally, we acknowledge partial support by the *National Science Foundation* (NSF) (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are the authors' and do not reflect the views of the NSF.

Authors' contributions
JLL performed the literature review and drafted the manuscript. TMK worked with JLL to develop the article's framework and focus. TMK introduced this topic to JLL. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 21 October 2020 Accepted: 15 November 2020
Published online: 23 November 2020

References

1. Singh AP, Singh MD. Analysis of host-based and network-based intrusion detection system. IJ Comput Netw Inf Secur. 2014;8:41–7.
2. Patil A, Laturkar A, Athawale S, Takale R, Tathawade P. A multilevel system to mitigate ddos, brute force and sql injection attack for cloud security. In: International Conference on Information, Communication, Instrumentation and Control (ICICIC), 2017. p. 1–7. IEEE.
3. Saxena AK, Sinha S, Shukla P. General study of intrusion detection system and survey of agent based intrusion detection system. In: 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017. p. 471–421. IEEE.
4. CNBC: Cyberattacks now cost companies $200,000 on average, putting many out of business. https://www.cnbc.com/2019/10/13/cyberattacks-cost-small-companies-200k-putting-many-out-of-business.html.
5. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, 2018. p. 108–116.
6. D'hooge L, Wauters T, Volckaert B, De Turck F. In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats. In: Proceedings of the 4th International Conference on Internet of Things, Big Data and Security; 2019.
7. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers Secur. 2012;31(3):357–74.
8. Bouteraa I, Derdour M, Ahmim A. Intrusion detection using data mining: A contemporary comparative study. In: 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), 2018. p. 1–8. IEEE.
9. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data. 2018;5(1):42.
10. He H, Garcia EA. Learning from imbalanced data. IEEE Trans knowl Data Eng. 2009;21(9):1263–84.
11. Thakkar A, Lohiya R. A review of the advancement in intrusion detection datasets. Procedia Comput Sci. 2020;167:636–45.
12. Groff Z, Schwartz S. Data preprocessing and feature selection for an intrusion detection system dataset. In: 34th Annual Conference of The Pennsylvania Association of Computer and Information Science Educators, 2019. p. 103–110.
13. Menon AK, Williamson RC. The cost of fairness in binary classification. In: Conference on Fairness, Accountability and Transparency, 2018. p. 107–118.
14. Atefinia R, Ahmadi M. Network intrusion detection using multi-architectural modular deep neural network. J Supercomput. 2020. https://doi.org/10.1007/s11227-020-03410-y
15. Basnet RB, Shash R, Johnson C, Walgren L, Doleck T. Towards detecting and classifying network intrusion traffic using deep learning frameworks. J Internet Serv Inf Secur. 2019;9(4):1–17.
16. Catillo M, Rak M, Villano U. 2l-zed-ids: A two-level anomaly detector for multiple attack classes. In: Workshops of the International Conference on Advanced Information Networking and Applications. 2020. p. 687–696.

17. Chadza T, Kyriakopoulos KG, Lambotharan S. Contemporary sequential network attacks prediction using hidden markov model. In: 2019 17th International Conference on Privacy, Security and Trust (PST), 2019. p. 1–3.
18. Chastikova V, Sotnikov V. Method of analyzing computer traffic based on recurrent neural networks. J Phys. 2019;1353:012133.
19. D'hooge L, Wauters T, Volckaert B, De Turck F. Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. J Inf Secur Appl. 2020;54:102564.
20. Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J Inf Secur Appl. 2020;50:102419.
21. Lima Filho FSd, Silveira FA, de Medeiros Brito Junior A, Vargas-Solar G, Silveira LF. Smart detection: an online approach for dos/ddos attack detection using machine learning. Security and Communication Networks 2019; 2019.
22. Fitni QRS, Ramli K. Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2020. p. 118–124.
23. Gamage S, Samarabandu J. Deep learning methods in network intrusion detection: a survey and an objective comparison. J Netw Comput Appl. 2020;169:102767.
24. Hua Y. An efficient traffic classification scheme using embedded feature selection and lightgbm. In: 2020 Information Communication Technologies Conference (ICTC), 2020. p. 125–130.
25. Huancayo Ramos KS, Sotelo Monge MA, Maestre Vidal J. Benchmark-based reference model for evaluating botnet detection tools driven by traffic-flow analytics. Sensors. 2020;20(16):4501.
26. Kanimozhi V, Jacob TP. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In: 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, p. 0033–0036.
27. Kanimozhi V, Jacob TP. Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. Int J Eng Appl Sci Technol. 2019;4(6):2143–455.
28. Karatas G, Demir O, Sahingoz OK. Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. IEEE Access. 2020;8:32150–62.
29. Kim J, Kim J, Kim H, Shim M, Choi E. Cnn-based network intrusion detection against denial-of-service attacks. Electronics. 2020;9(6):916.
30. Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur. 2020;95:101851.
31. Lin P, Ye K, Xu C-Z. Dynamic network anomaly detection system by using deep learning techniques. In: International Conference on Cloud Computing. Springer; 2019, 161–176.
32. Zhao F, Zhang H, Peng J, Zhuang X, Na S-G. A semi-self-taught network intrusion detection system. Neural Comput Appl. 2020;32:17169–79.
33. Happel BL, Murre JM. Design and evolution of modular neural network architectures. Neural Netw. 1994;7(6–7):985–1004.
34. Lu N, Li T, Ren X, Miao H. A deep learning scheme for motor imagery classification based on restricted Boltzmann machines. IEEE Trans Neural Syst Rehab Eng. 2016;25(6):566–76.
35. Varsamopoulos S, Criger B, Bertels K. Decoding small surface codes with feedforward neural networks. Quantum Sci Technol. 2017;3(1):015004.
36. De Mulder W, Bethard S, Moens M-F. A survey on the application of recurrent neural networks to statistical language modeling. Comput Speech Lang. 2015;30(1):61–98.
37. Madan A, George AM, Singh A, Bhatia M. Redaction of protected health information in ehrs using crfs and bi-directional lstms. In: 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2018. p. 513–517.
38. Lee K, Filannino M, Uzuner Ö. An empirical test of grus and deep contextualized word representations on de-identification. Stud Health Technol Inform. 2019;264:218–22.
39. Chaudhary A, Kolhe S, Kamal R. An improved random forest classifier for multi-class classification. Inf Process Agric. 2016;3(4):215–22.
40. Rynkiewicz J. Asymptotic statistics for multilayer perceptron with Relu hidden units. Neurocomputing. 2019;342:16–23.
41. Chen J, Xie B, Zhang H, Zhai J. Deep autoencoders in pattern recognition: A survey. Bio-inspired Computing Models And Algorithms. World Scientific;2019. 229–55.
42. Joshi J, Kumar T, Srivastava S, Sachdeva D. Optimisation of hidden Markov model using Baum-Welch algorithm for prediction of maximum and minimum temperature over Indian Himalaya. J Earth Syst Sci. 2017;126(1):3.
43. Lember J, Sova J. Regenerativity of viterbi process for pairwise markov models. J Theor Probab. 2020;. https://doi.org/10.1007/s10959-020-01022-z.
44. Shah SAR, Issac B. Performance comparison of intrusion detection systems and application of machine learning to snort system. Future Gener Comput Syst. 2018;80:157–70.
45. Pasupa K, Vatathanavaro S, Tungjitnob S. Convolutional neural networks based focal loss for class imbalance problem: A case study of canine red blood cells morphology classification. J Ambient Intell Human Comput. 2020;. https://doi.org/10.1007/s12652-020-01773-x.
46. Chen W, Zhang S, Li R, Shahabi H. Performance evaluation of the gis-based data mining techniques of best-first decision tree, random forest, and naïve bayes tree for landslide susceptibility modeling. Sci Total Environ. 2018;644:1006–188.
47. Ahmad I, Basheri M, Iqbal MJ, Rahim A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access. 2018;6:33789–95.
48. Taşer PY, Birant KU, Birant D. Comparison of ensemble-based multiple instance learning approaches. In: 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), 2019. p. 1–5.

49. Ayyadevara VK. Gradient boosting machine. In: Pro Machine Learning Algorithms. Berkeley, CA: Apress; 2018. https://doi.org/10.1007/978-1-4842-3564-5_6.
50. Wang R, Zeng S, Wang X, Ni J. Machine learning for hierarchical prediction of elastic properties in fe-cr-al system. Comput Mater Sci. 2019;166:119–23.
51. Baig MM, Awais MM, El-Alfy E-SM. Adaboost-based artificial neural network learning. Neurocomputing. 2017;248:120–6.
52. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016. p. 785–794.
53. Vajda S, Santosh K. A fast k-nearest neighbor classifier using unsupervised clustering. In: International Conference on Recent Trends in Image Processing and Pattern Recognition, 2016. p. 185–193.
54. Saikia T, Brox T, Schmid C. Optimized generic feature learning for few-shot classification across domains. arXiv preprint arXiv:2001.07926 2020.
55. Sulaiman S, Wahid RA, Ariffin AH, Zulkifli CZ. Question classification based on cognitive levels using linear svc. Test Eng Manag. 2020;83:6463–70.
56. Rahman MA, Hossain MA, Kabir MR, Sani MH, Awal MA et al.. Optimization of sleep stage classification using single-channel eeg signals. In: 2019 4th International Conference on Electrical Information and Communication Technology (EICT), 2019. p. 1–6.
57. Rymarczyk T, Kozłowski E, Kłosowski G, Niderla K. Logistic regression for machine learning in process tomography. Sensors. 2019;19(15):3400.
58. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Gener Comput Syst. 2019;100:779–96.
59. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of deep neural network architectures and their applications. Neurocomputing. 2017;234:11–26.
60. Li J, Xi B, Li Y, Du Q, Wang K. Hyperspectral classification based on texture feature enhancement and deep belief networks. Remote Sensing. 2018;10(3):396.
61. Zhao Y, Li H, Wan S, Sekuboyina A, Hu X, Tetteh G, Piraud M, Menze B. Knowledge-aided convolutional neural network for small organ segmentation. IEEE J Biomed Health Inform. 2019;23(4):1363–73.
62. Taherkhani A, Cosma G, McGinnity TM. Deep-fs: A feature selection algorithm for deep boltzmann machines. Neurocomputing. 2018;322:22–37.
63. Jazi HH, Gonzalez H, Stakhanova N, Ghorbani AA. Detecting http-based application layer dos attacks on web servers in the presence of sampling. Comput Netw. 2017;121:25–36.
64. Akhtar F, Li J, Pei Y, Xu Y, Rajput A, Wang Q. Optimal features subset selection for large for gestational age classification using gridsearch based recursive feature elimination with cross-validation scheme. In: International Conference on Frontier Computing, 2019. p. 63–71.
65. Scikit-learn: SGDClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
66. Fadlil A, Riadi I, Aji S. Ddos attacks classification using numeric attribute based Gaussian Naive Bayes. Int J Adv Comput Sci Appl. 2017;8(8):42–50.
67. Elkhalil K, Kammoun A, Couillet R, Al-Naffouri TY, Alouini M-S. Asymptotic performance of regularized quadratic discriminant analysis based classifiers. In: 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), 2017. p. 1–6.
68. Abd Elrahman SM, Abraham A. A review of class imbalance problem. J Netw Innov Comput. 2013;1(2013):332–40.
69. Zhang W-Y, Wei Z-W, Wang B-H, Han X-P. Measuring mixing patterns in complex networks by spearman rank correlation coefficient. Phys A Stat Mech Appl. 2016;451:440–50.
70. Shi D, DiStefano C, McDaniel HL, Jiang Z. Examining chi-square test statistics under conditions of large model size and ordinal data. Struct Equ Model. 2018;25(6):924–45.
71. Hancock J, Khoshgoftaar TM. Medicare fraud detection using catboost. In: 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), 2020. p. 97–103. IEEE Computer Society.
72. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, 2017. p. 3146–3154.
73. Bentéjac C, Csörgő A, Martínez-Muñoz G. A comparative analysis of gradient boosting algorithms. Artif Int Rev. 2020;1–31.
74. KDD: KDD Cup. https://kdd.ics.uci.edu/databases/kddcup99/task.html/.
75. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009. p. 1–6. IEEE.
76. Yap BW, Abd Rani K, Abd Rahman HA, Fong S, Khairudin Z, Abdullah NN. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In: Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013), 2014. p. 13–22. Springer.
77. Saritas MM, Yasar A. Performance analysis of ann and Naive Bayes classification algorithm for data classification. Int J Intell Syst Appl Eng. 2019;7(2):88–91.
78. Alenazi A, Traore I, Ganame K, Woungang I. Holistic model for http botnet detection based on dns traffic analysis. In: International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, 2017. p. 1–18.
79. Gupta V, Bhavsar A. Random forest-based feature importance for hep-2 cell image classification. In: Annual Conference on Medical Image Understanding and Analysis, 2017. p. 922–934. Springer.
80. Yuanyuan S, Yongming W, Lili G, Zhongsong M, Shan J. The comparison of optimizing svm by ga and grid search. In: 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), 2017. p. 354–360.
81. Ranjan G, Verma AK, Radhika S. K-nearest neighbors and grid search cv based real time fault monitoring system for industries. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019. p. 1–5.
82. Bilgic B, Chatnuntawech I, Fan AP, Setsompop K, Cauley SF, Wald LL, Adalsteinsson E. Fast image reconstruction with l2-regularization. J Magn Reson Imaging. 2014;40(1):181–91.

83. Meyer H, Reudenbach C, Hengl T, Katurji M, Nauss T. How to detect and avoid overfitting in spatio-temporal machine learning applications. In: EGU General Assembly Conference Abstracts, vol. 20, 2018. p. 8365.
84. Yadav S, Shukla S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: 2016 IEEE 6th International Conference on Advanced Computing (IACC), 2016. p. 78–83.
85. Fernández A, Garcia S, Herrera F, Chawla NV. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res. 2018;61:863–905.
86. Negi S, Kumar Y, Mishra V. Feature extraction and classification for emg signals using linear discriminant analysis. In: 2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall), 2016. p. 1–6.
87. Wei Z, Wang Y, He S, Bao J. A novel intelligent method for bearing fault diagnosis based on affinity propagation clustering and adaptive feature selection. Knowl Based Syst. 2017;116:1–12.
88. Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089 2018.
89. Chorowski JK, Bahdanau D, Serdyuk D, Cho K, Bengio Y. Attention-based models for speech recognition. In: Advances in Neural Information Processing Systems, 2015. p. 577–585.
90. Zhang Z. Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), 2018. p. 1–2.
91. Sharma A. Guided stochastic gradient descent algorithm for inconsistent datasets. Appl Soft Comput. 2018;73:1068–80.
92. Chiang H-T, Hsieh Y-Y, Fu S-W, Hung K-H, Tsao Y, Chien S-Y. Noise reduction in ECG signals using fully convolutional denoising autoencoders. IEEE Access. 2019;7:60806–133.
93. Deng Z-H, Qiao H-H, Song Q, Gao L. A complex network community detection algorithm based on label propagation and fuzzy c-means. Phys A Stat Mech Appl. 2019;519:217–26.
94. Zhu X, Wu X, Chen Q. Eliminating class noise in large datasets. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003. p. 920–927.
95. Lee J-S. Auc4. 5: Auc-based c4. 5 decision tree algorithm for imbalanced data classification. IEEE Access. 2019;7:106034–42.
96. Sulam J, Ben-Ari R, Kisilev P. Maximizing auc with deep learning for classification of imbalanced mammogram datasets. In: VCBM, 2017. p. 131–135.
97. Buda M, Maki A, Mazurowski MA. A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. 2018;106:249–59.
98. Iversen GR, Wildt AR, Norpoth H, Norpoth HP. Analysis of Variance. Thousand Oaks: Sage; 1987.
99. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5:99–114.
100. Del Río S, López V, Benítez JM, Herrera F. On the use of map reduce for imbalanced big data using random forest. Inf Sci. 2014;285:112–37.
101. Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F. Evolutionary undersampling for extremely imbalanced big data classification under apache spark. In: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016. p. 640–647. IEEE.
102. Moreno-Torres JG, Raeder T, Alaiz-RodríGuez R, Chawla NV, Herrera F. A unifying view on dataset shift in classification. Pattern Recogn. 2012;45(1):521–30.
103. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q. A comprehensive survey on transfer learning. Proceedings of the IEEE. 2020.
104. Singla A, Bertino E, Verma D. Overcoming the lack of labeled data: training intrusion detection models using transfer learning. In: 2019 IEEE International Conference on Smart Computing (SMARTCOMP), 2019. p. 69–74.

Publisher's Note