Leevy *et al. J Big Data* (2021) 8:38
https://doi.org/10.1186/s40537-021-00426-w

Journal of Big Data

RESEARCH Open Access

Detecting cybersecurity attacks across different network features and learners

Joffrey L. Leevy*, John Hancock, Richard Zuech and Taghi M. Khoshgoftaar

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

Abstract

Machine learning algorithms efficiently trained on intrusion detection datasets can detect network traffic capable of jeopardizing an information system. In this study, we use the CSE-CIC-IDS2018 dataset to investigate ensemble feature selection on the performance of seven classifiers. CSE-CIC-IDS2018 is big data (about 16,000,000 instances), publicly available, modern, and covers a wide range of realistic attack types. Our contribution is centered around answers to three research questions. The first question is, "Does feature selection impact performance of classifiers in terms of Area Under the Receiver Operating Characteristic Curve (AUC) and F1-score?" The second question is, "Does including the Destination_Port categorical feature significantly impact performance of LightGBM and Catboost in terms of AUC and F1-score?" The third question is, "Does the choice of classifier: Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), Catboost, LightGBM, or XGBoost, significantly impact performance in terms of AUC and F1-score?" These research questions are all answered in the affirmative and provide valuable, practical information for the development of an efficient intrusion detection model. To the best of our knowledge, we are the first to use an ensemble feature selection technique with the CSE-CIC-IDS2018 dataset.

Keywords: Feature selection, Intrusion detection, Catboost, XGBoost, LightGBM, SlowlorisBig, Big data, CSE-CIC-IDS2018

Introduction

CSE-CIC-IDS2018 [1], also referred to as the 2018 dataset throughout this text, is an intrusion detection dataset with normal and anomalous instances of network traffic. Machine learning models efficiently trained on CSE-CIC-IDS2018 can detect network traffic capable of compromising an information system. This dataset is the most recent iteration of ISCXIDS2012 [2], a scalable project designed to produce modern, realistic datasets. CSE-CIC-IDS2018 data originated from an extensive network of victim and attack machines [3], yielding an aggregate of 16,233,002 instances. Six classes of attack traffic (percentage distribution shown in Table 1) are represented by about 17% of these instances.

The 2018 dataset has a binary class imbalance vis-à-vis the non-attack instances to the total number of attack instances. The dataset is distributed over ten CSV files that are

Table 1 CSE-CIC-IDS2018 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

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

Machine learning is greatly facilitated by the high number of features in CSE-CIC-IDS2018. Machine learning algorithms typically outperform traditional statistical methods in classification tasks [4, 5]. However, the threshold settings of some learners may not be appropriately set for imbalanced data, thus rendering these algorithms inefficient at distinguishing majority and minority classes in a highly imbalanced environment. The learners will consequently fail to properly model the distribution of the positive (minority) class and become biased in favor of the negative (majority) class. Therefore, one must employ metrics that safeguard against this outcome. The two metrics used in this study, F1-score and Area Under the Receiver Operating Characteristic (ROC) Curve (AUC), are suitable for evaluating classifier performance on imbalanced datasets [6, 7]. We note that class imbalance is more noticeable in big data as the number of majority class instances is disproportionately high in that environment [8, 9].

The ensemble feature selection [10, 11] approach in this paper is tailored toward improving classifier performance by using a relevant subset of variables from CSE-CIC-IDS2018. It is worth noting that feature selection also provides data clarity and reduces computation requirements. In our study, we utilize both supervised and filter-based [12] feature ranking techniques, and the last stage of our ensemble approach is the selection of common features from these techniques.

The specific properties of big data can make classification more challenging for learners trained on the 2018 dataset. These properties include volume, variety, velocity, variability, value, and complexity [8]. Traditional methods may have difficulty handling the high data volume, the diversity of data formats, the speed of data originating from different sources, data flow inconsistencies, the filtering of important data, and data linking and transformation.

Classifier performance in our case study is based on the training and testing of the following learners: Decision Tree (DT) [13], Random Forest (RF) [14], Naive Bayes (NB) [15], Logistic Regression (LR) [16], Catboost [17], LightGBM [18], and XGBoost [19]. These learners are selected for their good coverage of several Machine Learning (ML) model families and are viewed favorably in terms of performance [20]. The seven classifiers are further discussed in "Classifier development and metrics" section.

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

To the best of our knowledge, this study is the most comprehensive analysis on CSE-CIC-IDS2018 to date. Our work uniquely uses the 2018 dataset to investigate ensemble feature selection on the performance of seven classifiers. Our contribution is defined by our responses to three research questions: The first question is, "Does feature selection impact performance of classifiers in terms of AUC and F1-score?" The second question is, "Does including the Destination_Port categorical feature significantly impact performance of LightGBM and Catboost in terms of AUC and F1-score?" And, our third question is, "Does the choice of classifier: DT, RF, NB, LR, Catboost, LightGBM, or XGBoost, significantly impact performance in terms of AUC and F1-score?" The answers to these research questions provide valuable and practical information for the development of an efficient intrusion detection model.

The remainder of this paper is organized as follows: "Related work" provides an overview of literature that manipulates features of CSE-CIC-IDS2018; "Methodology" section describes the cleaning process of the 2018 dataset, our unique ensemble approach for feature selection, the classifiers and metrics used in the study, and the training and testing procedure for these classifiers; "Results and discussion" section presents and discusses our empirical results; "Conclusion" section concludes our paper with a summary of the work presented and suggestions for related future work.

Related work

In this section, we highlight studies that modify features of CSE-CIC-IDS2018 to improve classification results. However, to the best of our knowledge, none of these studies use an ensemble feature selection approach.

To address the high class imbalance of the 2018 dataset, Hua [21] uses an undersampling and embedded feature selection approach with a LightGBM classifier. Undersampling [22] randomly removes majority class instances to alter class distribution. During the data cleaning stage, missing values and useless features were removed, resulting in a modified set of 77 features. String labels were converted to integer labels, which were then one-hot encoded. In addition to LightGBM, six other learners were evaluated in this research work: Support Vector Machine (SVM) [23], RF, Adaboost [24], Multilayer Perceptron (MLP) [25], Convolutional Neural Network (CNN) [26], and Naive Bayes. Learners were implemented with Scikit-learn [27] and TensorFlow [28]. The train to test data ratio was 70 to 30, and XGBoost was used to perform feature selection. LightGBM 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.

In another related work of research [29], five learners were evaluated on two datasets (CSE-CIC-IDS2018 and ISOT HTTP Botnet [30]) 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-Nearest Neighbor (*k*-NN) [31], Naive Bayes, and SVM. Feature selection was performed using various techniques, including the feature importance method [32] of RF. Subsequent to feature selection, CSE-CIC-IDS2018 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. About 80% of botnet instances were used for training, where five-fold cross-validation was applied. The remaining botnet instances served as the testing set. For optimization, the Grid Search algorithm [33] was used. With regard to CSE-CIC-IDS2018, 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).

Li et al. [34], in a third related study, apply clustering and feature selection to CSE-CIC-IDS2018. This unsupervised learning study involves online real-time detection with an autoencoder classifier. An autoencoder encodes data in a way that usually results in dimensionality reduction [35]. For preprocessing, "Infinity" and "NaN" values were replaced by 0, and the data was subsequently 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 data ratio was set as 85 to 15. The Affinity Propagation (AP) clustering [36] algorithm was subsequently used on 25% of the training dataset to group features into subsets, which were sent to the autoencoder. Recall rates for all attack types for the proposed model were compared with those of another autoencoder model called Kitnet [37]. 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.

Fitni and Ramli [38] adopt an ensemble model approach to compare seven single learners for integration into a classifier unit. The seven learners are as follows: RF, Gaussian Naive Bayes [39], DT, Quadratic Discriminant Analysis [40], 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 [41], a technique for selecting the most important features of a predictive model, was performed using the Spearman's rank correlation coefficient [42] and Chi-squared test [43], 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 AUC of 0.94.

Finally, D'hooge et al. include both CICIDS2017 and CSE-CIC-IDS2018 in a study investigating how efficiently the results of an intrusion detection dataset can be generalized [44]. CICIDS2017 is the predecessor of the 2018 dataset. For performance evaluation, the authors used 12 supervised learning algorithms from various families: DT, RF, Bag [45], gradient-boosted decision tree (GBDT), Extratree [46], Adaboost, XGBoost, *k*-NN, Ncentroid [47], linearSVC [48], RBFSVC [49], and Logistic Regression. The models were built with the Scikit-learn and XGBoost modules in Python. The authors used feature scaling, which is different from feature selection. Feature scaling attempts to normalize the feature space of all attributes. Results show that the tree-based classifiers

yielded the best performance, and among them, XGBoost ranked first with many perfect values for F1-score and AUC. D'hooge et al. hinted overfitting might have been a problem and "further analysis" was warranted. We note that their source code indicated hyperparameter values of max-depth = 35 for some of their tree-based learners. Such values are prone to overfitting. For intrusion detection, the authors concluded that a model trained on one dataset (CICIDS2017) cannot generalize to another dataset (CSE-CIC-IDS2018).

In summary, the related works exhibit shortcomings with nearly perfect classification performance values typically associated with overfitting. We discovered additional shortcomings, such as errors in preparation (e.g. using Destination_Port as a numeric value instead of categorical value) and in data cleaning. Ambiguous specifications are also an issue with regard to reproducibility of the studies.

Methodology

Data cleaning

Removing certain fields from CSE-CIC-IDS2018 was our first step in the data cleaning stage. We dropped the Protocol field because it is redundant, since the Dst Port (Destination_Port) field mostly contains equivalent Protocol values for each Destination_Port value. We dropped the Timestamp field as we wanted the learners to not discriminate between attack predictions based on time, especially with more stealthy attacks in mind. In other words, the learners should be able to distinguish attacks regardless of whether they are high volume or slow and stealthy. Dropping the Timestamp field also allows us the convenience of combining or dividing the datasets into ways more compatible with our experimental frameworks.

We removed 59 records that were actually a repetition of the header rows. These were easily found and removed by filtering records based on a white list of valid label values.

The fourth downloaded file "Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv" was different than the other 9 files for the 2018 dataset. This file contained 4 extra columns: Flow ID, Src IP, Src Port, and Dst IP. We dropped these 4 additional fields.

Certain fields contained negative values which did not make sense, and so we dropped those instances with negative values for the Fwd_Header_Length, Flow_Duration, and Flow_IAT_Min fields. In particular, the negative values from the Fwd_Header_Length field occur with extreme values in other fields. These extreme values skew statistics that are sensitive to outliers.

Eight fields contained values of zero for every instance. Prior to the start of machine learning, we filtered out the following list of fields:

1. Bwd_PSH_Flags
2. Bwd_URG_Flags
3. Fwd_Avg_Bytes_Bulk
4. Fwd_Avg_Packets_Bulk
5. Fwd_Avg_Bulk_Rate
6. Bwd_Avg_Bytes_Bulk
7. Bwd_Avg_Packets_Bulk
8. Bwd_Avg_Bulk_Rate

We also excluded the Init_Win_bytes_forward and Init_Win_bytes_backward fields, since about half of the total instances contained negative values for these two fields. Similarly, we did not use the Flow_Duration field as some of those values were unreasonably low with zero values. The Flow Bytes/s and Flow Packets/s fields contained some "Infinity" and "NaN" values (with less than 0.6% of the records containing these values). We dropped these instances (total of 95,760) where either Flow Bytes/s or Flow Packets/s contained "Infinity" or "NaN" values.

Ensemble feature selection

After data cleaning, the number of independent features in CSE-CIC-IDS2018 is reduced to 66. We then adopt an ensemble approach for feature selection. Ensemble feature selection is derived from the concept of ensemble learning, which demonstrates that the combination of multiple learning approaches outperforms a single approach for the classification of instances. This intuitive concept has been extended from an ensemble of learners to an ensemble of feature ranking techniques, where distinct feature ranking methods are integrated to provide one ranking.

In our case study, we use seven ranking techniques to generate seven lists of features and subsequently process the resulting lists to select features. The ranking techniques assign a number to each of the 66 usable features in the 2018 data. We use this number to place an ordering on the features. When we apply a feature ranking technique, we select, at most, the top 20 highest ranked features. Our decision to use 20 features was motivated by two factors. First, we wanted to select a list of features long enough that there would be a good chance for different rankings to have elements in common. Second, due to hyper-parameter tuning to avoid overfitting, we set the maximum depth of CatBoost at 5, which causes CatBoost to construct constituent DTs that use only 14 features. Therefore, to keep CatBoost's ranking relevant, we settled on taking a maximum of 20 features from other rankers.

We employ both filter-based and supervised ranking techniques. Filter-based feature ranking techniques create a list of features by a statistic that we calculate for each variable. Supervised feature ranking techniques leverage the structures of constituent DTs of ensemble classifiers to generate a feature importance list. We refer to these ordered lists as "rankings." Selected features appear in at least four out of the seven ranking techniques.

Filter-based techniques

The filter-based feature ranking techniques we use are based on the Information Gain (IG) (also known as Mutual Information) [50], Gain Ratio (GR) [51], and Chi-Squared (CS) [52] statistics. We use the value of the statistic calculated for each feature to filter the list of all features to a reduced list.

To calculate IG and GR statistics, we use the "info_gain" and "info_gain_ratio" functions from the info_gain Python library. To calculate the CS statistic, we use the "chi2" function that is included as part of the Scikit-learn library. One may employ the same method to rank features of a dataset with any of these 3 functions. We do not supply configuration parameters to the IG, GR, or CS functions when we invoke them. Each of the three functions accepts two arrays of data for input. We employ all 66 usable features

Table 2 Top 20 features by filter-based ranking technique for CSE-CIC-IDS2018

CS	Information gain	Gain ratio
Fwd_IAT_Total	Fwd_Packets_s	Fwd_Packets_s
Bwd_IAT_Total	Flow_Packets_s	Flow_Packets_s
Bwd_IAT_Max	Flow_IAT_Mean	Flow_IAT_Mean
Fwd_IAT_Max	Destination_Port	Destination_Port
Flow_IAT_Max	Bwd_Packets_s	Bwd_Packets_s
Fwd_IAT_Std	Flow_Bytes_s	Flow_Bytes_s
Idle_Max	Fwd_IAT_Mean	Fwd_IAT_Mean
Idle_Mean	Flow_IAT_Max	Flow_IAT_Max
Idle_Min	Fwd_Packet_Length_Mean	Fwd_Packet_Length_Mean
Flow_IAT_Std	Avg_Fwd_Segment_Size	Avg_Fwd_Segment_Size
Bwd_IAT_Std	Packet_Length_Std	Packet_Length_Std
Bwd_IAT_Mean	Fwd_Header_Length	Fwd_Header_Length
Fwd_IAT_Mean	Packet_Length_Variance	Packet_Length_Variance
Flow_Bytes_s	Total_Length_of_Fwd_Packets	Total_Length_of_Fwd_Packets
Active_Max	Subflow_Fwd_Bytes	Subflow_Fwd_Bytes
Bwd_Packets_s	Packet_Length_Mean	Packet_Length_Mean
Flow_Packets_s	Fwd_Packet_Length_Max	Fwd_Packet_Length_Max
Flow_IAT_Mean	Bwd_Packet_Length_Mean	Bwd_Packet_Length_Mean
Active_Mean	Avg_Bwd_Segment_Size	Avg_Bwd_Segment_Size
Active_Min	Total_Length_of_Bwd_Packets	Total_Length_of_Bwd_Packets

of the 2018 dataset as the source of data for the first input array, and the "label" value of the 2018 data for the second input array. All three functions also return a list, l, of numbers where we use the value of the ith element of the list to determine its rank, r, relative to the values of the other elements in the list. To be concrete, we create a list l' of pairs (r_i, i) from l, and then sort l' in decreasing order of r_i. After sorting l', we truncate it at the 20th element. If the feature ranking technique assigns an importance of zero to a feature, we do not include it in the list of ranked features. For instance, we find CatBoost assigns an importance greater than 0 to fewer than 20 features.

Some readers may have reservations about the applicability of IG, GR, or CS to categorical or numeric features. We apply IG, GR, and CS feature selection techniques to CSE-CIC-IDS2018 network traffic data in a manner similar to Singh et al. in [53]. In their study, Singh et al. apply these techniques to the KDD CUP 1999 network traffic dataset. This dataset is similar to the 2018 dataset in that it contains numeric and categorical features. Therefore, we are comfortable applying these filter-based feature ranking techniques to the 2018 dataset. Table 2 contains the rankings for the three filter-based feature ranking techniques.

Through filter-based feature ranking, we obtain three out of the seven lists used to select features for our models. The remaining four lists of features are obtained with supervised feature selection techniques that are discussed in the next subsection.

Supervised feature ranking techniques

For the supervised feature ranking techniques, we use the feature importance lists from the RF, CatBoost, XGBoost, and LightGBM Python libraries. CatBoost, LightGBM, and XGBoost are Python libraries of their own. The RF implementation we use is part of the

Table 3 Supervised ranking classifier Random Forest initialization options

Parameter/Value	Comment
n_estimators = 5	Prevent overfitting
max_depth = 6	Limit number maximum number of features ranked to 32

Table 4 Supervised ranking classifier LightGBM initialization options

Parameter/Value	Comment
learning_rate = 0.1	Selected to prevent overfitting

Table 5 Supervised ranking classifier XGBoost initialization options

Parameter/Value	Comment
objective = 'binary_logistic'	Specify binary classification
n_jobs = 8	Take advantage of parallel processing functionality
n_estimators = 4	Prevent overfitting
max_depth = 5	Limit maximum number of features ranked to 32

Table 6 Supervised ranking classifier CatBoost initialization options

Parameter/Value	Comment
thread_count = 8	Take advantage of parallel processing functionality
iterations = 4	Prevent overfitting
max_depth = 5	Limit maximum number of features ranked to 32

Scikit-learn Python library. Here, we discuss how we use elements in common to the implementations of RF, CatBoost, XGBoost, and LightGBM to employ them in ranking techniques.

All four libraries have classifier objects. These objects have initialization (constructor) functions. One may pass configuration options to the initialization functions. Please see Tables 3, 4, 5, and 6 for the configuration options we use for each classifier object. Each object also has a "fit" function. After the classifiers' fit function is successfully invoked, the classifier object has a list attribute "feature_importances_". We use the feature_importances_ list in the same way we use the list of values l returned by the functions for filter-based ranking techniques discussed in the previous subsection. Hereafter, we refer to the feature selection technique of using the feature importance values from CatBoost, LightGBM, XGBoost, and RF classifiers by the names of the classifiers, where not ambiguous.

As discussed in the previous subsection, CSE-CIC-IDS2018 has one categorical feature: Destination_Port. We found this feature has 53,760 possible values in the 2018

Table 7 Top 20 CSE-CIC-IDS2018 features by supervised ranking technique: XGBoost, Random Forest; due to hyper-parameter settings, CatBoost uses only 14 features

XGBoost	CatBoost	Random Forest
Fwd_Packet_Length_Max	Destination_Port	Fwd_Packet_Length_Mean
Fwd_IAT_Max	Flow_IAT_Min	Max_Packet_Length
Total_Length_of_Fwd_Packets	Bwd_Packet_Length_Std	min_seg_size_forward
Fwd_IAT_Std	min_seg_size_forward	Flow_IAT_Min
Fwd_Packets_s	Fwd_Packet_Length_Max	Fwd_Packets_s
Fwd_Header_Length	Fwd_IAT_Total	Total_Length_of_Fwd_Packets
Fwd_Packet_Length_Mean	Fwd_Header_Length	Fwd_IAT_Min
min_seg_size_forward	Fwd_Packets_s	Fwd_IAT_Total
Bwd_IAT_Total	ACK_Flag_Count	Fwd_IAT_Max
Idle_Min	Fwd_IAT_Min	Bwd_Packet_Length_Max
Fwd_IAT_Total	Flow_Bytes_s	Flow_IAT_Mean
Flow_IAT_Std	Avg_Fwd_Segment_Size	Fwd_Packet_Length_Max
Bwd_Packet_Length_Std	Flow_IAT_Std	Total_Length_of_Bwd_Packets
act_data_pkt_fwd	Flow_IAT_Max	Fwd_Header_Length
PSH_Flag_Count		PSH_Flag_Count
Bwd_Packets_s		Flow_Packets_s
Max_Packet_Length		Down_Up_Ratio
RST_Flag_Count		Bwd_Packets_s
ACK_Flag_Count		Flow_IAT_Max
Flow_IAT_Max		Flow_IAT_Std

dataset. Hence, we concluded that finding an appropriate encoding technique for this feature is outside the scope of our study. However, CatBoost and LightGBM have built-in support for categorical features, so we include Destination_Port as a candidate for ranking for CatBoost and LightGBM, but not for XGBoost or Random Forest.

All supervised ranking techniques yield 20 features, except CatBoost. Due to implementation details and the hyper-parameter settings we use, CatBoost will not construct DTs with a number of nodes sufficient to utilize 20 or more features in the training data. Hence, we find CatBoost provides rankings with fewer than 20 features. Tables 7 and 8 contain the top 20 features in all supervised rankings, except CatBoost, which ranks only 14 features.

We use supervised ranking techniques to generate 4 out of 7 rankings, and filter-based ranking techniques to generate the remaining 3 out of 7 rankings. We use the 7 rankings to conduct ensemble feature selection. In the following subsection, we cover the specifics of our feature selection techniques.

Feature selection

After obtaining the 7 rankings, our feature selection techniques are to select features that appear in k out of 7 rankings, where k has the value 4, 5, 6, or 7. Hence, we have 4 feature selection techniques, based on an ensemble of 7 feature ranking techniques. We refer to the set of features that appear in 4 out of 7 rankings as "feature group 1." This is our first ensemble feature selection technique. Since feature group 1 contains the Destination_Port categorical feature which some learners that we use cannot consume directly, "feature group 1A" is the set of all features in feature group 1, but

Table 8 Top 20 CSE-CIC-IDS2018 features by supervised ranking technique LightGBM

Destination_Port
Packet_Length_Std
min_seg_size_forward
Flow_IAT_Min
Fwd_Packet_Length_Std
Packet_Length_Variance
Fwd_IAT_Min
Idle_Max
Fwd_Packet_Length_Mean
Fwd_Packets_s
Bwd_Packet_Length_Mean
Flow_IAT_Max
Bwd_Packet_Length_Std
Fwd_Packet_Length_Max
Fwd_Header_Length
Fwd_IAT_Total
RST_Flag_Count
Bwd_IAT_Total
Bwd_Packets_s
Fwd_IAT_Max

excluding Destination_Port. We believe Destination_Port is a valuable categorical feature, but is only usable for 5 out of 7 of our rankers (CS, IG, GR, CatBoost, and LightGBM). This puts Destination_Port at a disadvantage for getting selected as a feature. In later experiments, we would like to know how classifiers that can handle categorical features perform with or without Destination_Port. Therefore, for every set of features that a feature selection technique produces, we add or remove Destination_Port as necessary to end up with two sets of features—one that has Destination_Port, and one that does not. For CSE-CIC-IDS2018, when we inspect the 7 rankings for features in common, we find 15 features in feature group 1.

We follow the naming convention similar to that of the feature selection technique where 4 out of 7 ranking techniques agree on a feature. Namely, if the result of a feature selection technique ends in 'A', we mean it does not contain Destination_Port, and if it does not end in 'A', it contains Destination_Port. Since Destination_Port does not appear in 5 out of 7 rankings, "feature group 2" is the result of applying the feature selection technique where 5 out of 7 rankers agree on a feature, then augmenting the result with Destination_Port. "Feature group 2A" is the set of features that appear in 5 out of 7 rankings. Similarly, we have "feature group 3", "feature group 3A", "feature group 4", and "feature group 4A" that we form in a manner similar to feature group 2 or feature group 2A. Feature group 3 and feature group 3A correspond to the case where 6 out of 7 rankings agree on a feature. Feature group 4 and feature group 4A correspond to the case where 7 out of 7 rankings agree on a feature. Hence, our 4 feature selection techniques net 8 groups of features depending on whether we include or exclude Destination_Port. Please refer to Tables 9, 10, 11, and 12 to see the resulting feature groups for all rounds of feature selection.

Table 9 Features appearing in at least 4 out of 7 rankings of CSE-CIC-IDS2018, referred to as "feature group 1"; we remove Destination_Port to form "feature group 1A"

Fwd_Packet_Length_Mean
Fwd_Packet_Length_Max
Flow_IAT_Mean
Total_Length_of_Fwd_Packets
Bwd_Packets_s
Fwd_Packets_s
Flow_Bytes_s
Fwd_IAT_Max
Fwd_IAT_Total
Flow_IAT_Std
Flow_IAT_Max
* Destination_Port
min_seg_size_forward
Flow_Packets_s
Fwd_Header_Length

* Indicates Destination_Port not included in feature group 1A

Table 10 Features appearing in at least 5 out of 7 rankings of CSE-CIC-IDS2018, referred to as "feature group 2A"; we add Destination_Port to form "feature group 2"

Fwd_Packets_s
Fwd_Header_Length
Fwd_Packet_Length_Mean
Fwd_IAT_Total
Flow_IAT_Max
Bwd_Packets_s
Fwd_Packet_Length_Max
* Destination_Port

* Indicates Destination_Port not included in feature group 2A

Table 11 Features appearing in at least 6 out of 7 rankings of CSE-CIC-IDS2018, referred to as "feature group 3A"; we add Destination_Port to form "feature group 3"

Fwd_Packets_s
Fwd_Header_Length
Flow_IAT_Max
Bwd_Packets_s
Fwd_Packet_Length_Max
* Destination_Port

* Indicates Destination_Port not included in feature group 3A

Table 12 Features appearing in at 7 out of 7 rankings of CSE-CIC-IDS2018, referred to as "feature group 4A"; we add Destination_Port to form "feature group 4A"

Flow_IAT_Max
* Destination_Port

* Indicates Destination_Port not included in feature group 4A

After selecting features listed in Tables 9, 10, 11, and 12, the datasets are suitable for training and testing classifiers. The reader should not attach any significance to the order of features in Tables 9, 10, 11, and 12.

We create a total of 11 datasets. Four are the result of applying the four feature selection techniques, and another four are the result of adding or removing the Destination_Port categorical feature as needed. In order to assess the impact of feature selection, we require two more datasets, one that contains all 66 usable features, and a similar dataset with all features except Destination_Port. We call these datasets "all features" and "all features A", respectively. Finally, we have one dataset that contains only the Destination_Port feature, which we call "Destination_Port only". In the next subsection we review classifiers that we train and test with the 2018 dataset.

Classifier development and metrics

Classifier development

After feature selection, we use the resulting datasets as input to 7 classifiers: DT, RF, NB, LR, CatBoost, XGBoost, and LightGBM. A DT is a simple representation of observed data. The tree can be easily visualized, with nodes or leaves representing class labels and branches representing observations. RF is an ensemble approach building multiple decision trees. The classification results are calculated by combining the results of the individual trees, typically using majority voting. NB uses Bayes' theorem of conditional probability to determine the probability that an instance belongs to a particular class. It is considered "naive" because of the strong assumption of independence between features. LR uses a sigmoidal, or logistic, function to generate values from [0,1] that can be interpreted as class probabilities. LR is similar to linear regression but uses a different hypothesis class to predict class membership. CatBoost, XGBoost, and LightGBM are GBDTs [54], an ensemble of DTs sequentially trained. Catboost uses Ordered Boosting, which imposes an order on the samples that CatBoost uses to fit constituent decision trees. XGBoost uses a sparsity-aware algorithm and a weighted quantile sketch. Sparsity is the quality of having many missing or zero values, while a weighted quantile sketch uses approximate tree learning [55] to support merge and prune operations. LightGBM uses Gradient-based One-Side Sampling and Exclusive Feature Bundling to handle large numbers of data instances and features. One-Side Sampling ignores a substantial portion of data instances with small gradients, while Exclusive Feature Bundling groups mutually exclusive features to reduce variable count.

As stated in the discussion on feature ranking, LightGBM and CatBoost handle encoding of categorical features automatically, so we take advantage of that, and use Destination_Port as a feature for LightGBM and CatBoost. Since we pass the array of features to LightGBM as a Pandas DataFrame [56], we indicate to LightGBM that Destination_Port is a categorical feature by setting the data type of the Destination_Port column to "category". In order to direct CatBoost to treat Destination_Port as a categorical feature, when we call the CatBoost classifier's initialization (constructor) function, we set the cat_features parameter to the value of a one-element list containing the string "Destination_Port".

Table 13 LightGBM classifier initialization options

Parameter/Value	Comment
bagging_fraction = 0.01	Faster training and prevent overfitting;
bagging_freq = 10	Necessary to specify along with bagging_fraction
max_bin = 32	Prevent overfitting

Table 14 XGBoost classifier initialization options

Parameter/Value	Comment
objective = 'binary_logistic'	Specify objective function for binary classification
n_jobs = 8	Take advantage of parallel processing functionality
n_estimators = 4	Prevent overfitting
max_depth = 5	Prevent overfitting

Table 15 Supervised ranking classifier Random Forest initialization options

Parameter/Value	Comment
n_estimators = 5	Prevent overfitting
max_depth = 6	Prevent overfitting

Table 16 Supervised ranking classifier CatBoost initialization options

Parameter/Value	Comment
thread_count = 8	Take advantage of parallel processing functionality
iterations = 4	Prevent overfitting
max_depth = 5	Prevent overfitting

We do one set of experiments where Destination_Port is the only feature. For these experiments we use CatBoost, LightGBM, and Scikit-learn's NB classifier for categorical data, CategoricalNB [57].

Before we train and test our classifiers, we initialize them with certain parameters. The settings of these parameters were selected based on experimentation. We list these initialization parameters in Tables 13, 14, 15, 16, and 17. We do not provide tables for initialization parameters for Naive Bayes or Logistic regression constructors because we did not set any for those two classifiers.

Table 17 Supervised ranking classifier Decision Tree initialization options

Parameter/Value	Comment
max_depth = 5	Prevent overfitting

Table 18 Confusion matrix

Actual class	Predicted class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN) (Type II error)
Negative	False Positive (FP) (Type I error)	True Negative (TN)

Classifier metrics

Our work records the confusion matrix (Table 18) for a binary classification problem, where the class of interest is usually the minority class and the opposite class is the majority class, i.e. positives and negatives, respectively. A related list of simple performance metrics [58] is explained as follows:

- True Positive (TP) is the number of positive samples correctly identified as positive.
- True Negative (TN) is the number of negative samples 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, other performance metrics are derived as follows:

- *Recall*, also known as True Positive Rate (TPR) or sensitivity, is equal to $TP/(TP + FN)$.
- *Precision*, also known as positive predictive value, is equal to $TP/(TP + FP)$.
- *Specificity*, also known as True Negative Rate (TNR), is equal to TN/(TN + FP).

In our study, we used more than one performance metric to better understand the challenge of evaluating machine learning models with severely imbalanced data. The metrics are explained below:

- *F1-score* (traditional), also known as the harmonic mean of precision and recall, is equal to $2 \cdot Precision \cdot Recall/(Precision + Recall)$.
- *AUC* is equal to the area under the Receiver Operating Characteristic (ROC) curve, which graphically shows recall versus (1-specificity) across all classifier

Table 19 Mean performance of CatBoost, LightGBM and CategoricalNB in terms of AUC and F1-score on a one-feature dataset of Destination_Port only

Classifier	AUC	SD AUC	F1-Score	SD F1
CatBoost	0.9073306	0.0002258	0.7357801	0.0003923
LightGBM	0.9073228	0.0002276	0.7357771	0.0003934
CNB	*0.9073425*	0.0002269	*0.7357834*	0.0003924

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

decision thresholds. From this curve, the AUC obtained is a single value that ranges from 0 to 1, with a perfect classifier having a value of 1.

Classifier training and testing

We train and test all classifiers using stratified fivefold cross-validation (CV). For training and testing any model, we use the "label" column of CSE-CIC-IDS2018 as the label. For each of the datasets that are appropriate for the classifiers, we do 10 iterations of 5-fold CV for all classifiers. Only CatBoost and LightGBM have built in support for categorical features, so all datasets are appropriate for CatBoost and LightGBM. However, we felt finding the optimal encoding technique as a preprocessing step is out of scope for this work, so it is not appropriate for NB, LR, DT, RF or XGBoost to be trained and tested with datasets that contain the Destination_Port categorical feature. We are interested in models that use solely the Destination_Port, and therefore we perform some experiments with the CategoricalNB classifier. CategoricalNB is appropriate for datasets that contain only categorical features; therefore, "Destination_Port only" dataset is appropriate for CategoricalNB.

For each unique combination of dataset and classifier, since we do ten iterations of fivefold CV, we record 50 measurements of AUC and F1-score values. The AUC and F1-score figures we report are mean values of 50 measurements. After training and testing all models with the various datasets, we group the performance metrics according to experimental factors at different levels, to perform ANalysis Of VAriance (ANOVA) [59] tests. When the outcomes of the ANOVA tests indicate that factors explain variance in performance metrics, we perform Tukey's Honestly Significant Difference (HSD) [60] tests to determine which levels of factors are significantly different, and which are associated with highest mean AUC and F-1 score values. We then use the outcomes of the ANOVA and Tukey's HSD tests to answer research questions Q1, Q2, and Q3. The first results we report are the mean AUC and F1-scores for CatBoost, LightGBM, and CategoricalNB with their respective datasets. We present these results in Table 19. We report results on ANOVA and Tukey's HSD tests in the next section, but mention them here since the data in the tables below are used in the tests later.

For reasons given earlier, we do not train or test CategoricalNB on datasets with numeric and categorical features. However, it is appropriate to train and test CatBoost and LightGBM with datasets consisting of categorical and numeric features. We report the outcome of the first such experiments in Table 20.

Table 20 Mean performance of CatBoost and LightGBM in terms of AUC and F1-score on datasets with features from feature group 1

Classifier	Feature group 1			
	AUC	SD AUC	F1	SD F1
CatBoost	0.95306	0.00360	0.93301	0.00327
LightGBM	*0.96694*	0.00107	*0.95880*	0.00066

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 21 Mean performance of CatBoost and LightGBM in terms of AUC and F1-score on datasets with features from feature group 2

Classifier	Feature group 2			
	AUC	SD AUC	F1	SD F1
CatBoost	0.94956	0.00167	0.92466	0.00308
LightGBM	*0.96728*	0.00118	*0.95681*	0.00104

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 22 Mean performance of CatBoost and LightGBM in terms of AUC and F1-score on datasets with features from feature group 3

Classifier	Feature group 3			
	AUC	SD AUC	F1	SD F1
CatBoost	0.94010	0.01376	0.92074	0.01270
LightGBM	*0.96722*	0.00038	*0.95585*	0.00126

best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 23 Mean performance of CatBoost and LightGBM in terms of AUC and F1-score on datasets with features from feature group 4

Classifier	Feature group 4			
	AUC	SD AUC	F1	SD F1
CatBoost	0.94169	0.00056	*0.91140*	0.00058
LightGBM	*0.94208*	0.00046	0.90989	0.00085

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 24 Mean performance of CatBoost and LightGBM in terms of AUC and F1-score on datasets with features from feature group all features

	All features			
CatBoost	0.96389	0.00108	0.95054	0.00187
LightGBM	*0.96890*	0.00019	*0.96134*	0.00021

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 25 Mean performance of 7 classifiers in terms of AUC and F1-score on datasets with features from feature group 1A

Classifier	Feature group 1A			
	AUC	SD AUC	F1	SD F1
CatBoost	0.94200	0.00870	0.91650	0.01075
LightGBM	*0.96147*	0.00087	*0.94690*	0.00062
Decision Tree	0.90891	0.00022	0.88583	0.00031
Logistic Regression	0.66772	0.00134	0.49471	0.00288
Naive Bayes	0.56711	0.00023	0.24319	0.00061
Random Forest	0.95132	0.00782	0.92949	0.00845
XGBoost	0.95385	0.00020	0.93647	0.00032

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 26 Mean performance of 7 classifiers in terms of AUC and F1-score on datasets with features from feature group 2A

Classifier	Feature group 2A			
	AUC	SD AUC	F1	SD F1
CatBoost	0.89547	0.01241	0.86477	0.01188
LightGBM	*0.95883*	0.00157	*0.94159*	0.00127
Decision Tree	0.89876	0.00026	0.86111	0.00032
Logistic Regression	0.55609	0.00018	0.20412	0.00056
Naive Bayes	0.56993	0.00018	0.24736	0.00055
Random Forest	0.93240	0.02058	0.90543	0.02154
XGBoost	0.94174	0.00116	0.91419	0.00057

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 27 Mean performance of 7 classifiers in terms of AUC and F1-score on datasets with features from feature group 3A

Classifier	Feature group 3A			
	AUC	SD AUC	F1	SD F1
CatBoost	0.88616	0.00304	0.85593	0.00363
LightGBM	*0.95832*	0.00088	*0.93869*	0.00152
Decision Tree	0.88518	0.01677	0.83763	0.01621
Logistic Regression	0.55352	0.00021	0.19546	0.00066
Naive Bayes	0.56753	0.00287	0.24000	0.00887
Random Forest	0.93496	0.01585	0.90416	0.01719
XGBoost	0.94780	0.00020	0.91189	0.00031

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

In Tables 21, 22, 23 and 24 we report the results of further experiments involving CatBoost and LightGBM. In these tables, we train and test models on data with features in feature groups 2, 3, 4, and all features.

In Tables 25, 26, 27, 28 and 29 we report performance of the 7 classifiers CatBoost, LightGBM, DT, LR, NB, RF, and XGBoost as we train and test them on datasets with

Table 28 Mean performance of 7 classifiers in terms of AUC and F1-score on datasets with features from feature group 4A

Classifier	Feature group 4A			
	AUC	SD AUC	F1	SD F1
CatBoost	0.62869	0.00346	0.40433	0.00786
LightGBM	0.62282	0.00243	0.38899	0.00566
Decision Tree	0.64309	0.00031	0.42947	0.00059
Logistic Regression	0.50000	0.00000	0.00000	0.00000
Naive Bayes	0.50000	0.00000	0.00000	0.00000
Random Forest	*0.65889*	0.00030	*0.46665*	0.00061
XGBoost	0.64208	0.00030	0.42912	0.00065

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

Table 29 Mean performance of 7 classifiers in terms of AUC and F1-score on datasets with features from feature group all features A

Classifier	All features A			
	AUC	SD AUC	F1	SD F1
CatBoost	*0.95602*	0.00271	*0.93653*	0.002
Decision Tree	0.91190	0.00030	0.88740	0.000
LightGBM	0.95192	0.00616	0.92988	0.007
Naive Bayes	0.55359	0.00038	0.31448	0.000
Random Forest	0.95522	0.00474	0.93234	0.004
LightGBM	0.91324	0.00024	0.88905	0.000

Best metrics are highlighted in italics; SD AUC is the standard deviation of AUC and SD F1 is the standard deviation of the F1-score

features from feature groups 1A through 4A, and feature group all features A. Since these datasets do not contain the Destination_Port categorical feature, more classifiers are available for us to experiment with.

Results and discussion

We inspect Tables 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29, then use the same data we used to obtain the mean values in them to conduct ANOVA and Tukey's HSD tests to answer research questions Q1, Q2, and Q3. The confidence levels we use for all tests is 99%. For ANOVA tests, we build models where the dependent variable is the experiment outcome (AUC or F1-score), and the independent variables (classifier, feature selection technique, *etc.*) are factors in the experiments. Therefore, we include box plots of data used to build the models for ANOVA to aid in understanding groupings the Tukey's HSD tests identify.

Research Question Q1: Does feature selection impact performance of classifiers in terms of AUC and F1-score?

Our first step in answering Q1 is to inspect Tables 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. In Table 20, we see the LightGBM model yields an AUC value of 0.96694 and an F1-score of 0.95880 when trained on the 15 features (including Destination_Port) from feature group 1. However, in Table 24, we see that, when trained with all features,

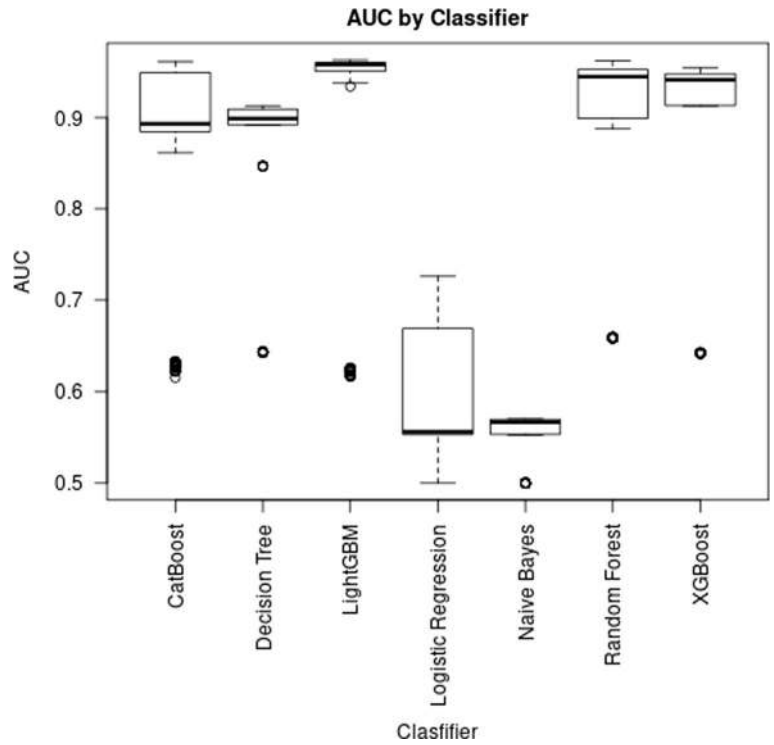

Fig. 1 Box plots of AUC grouped by classifier; here datasets do not include destination port; Tukey's HSD test indicates LightGBM, Random Forest and XGBoost in group a, CatBoost, Decision Tree in group b, Logistic Regression in group c and Naive Bayes in group d (factors in the same group are not significantly different)

LightGBM yields an AUC value of 0.96890 and an F1-score of 0.96134. We see analogous patterns of similar or better performance for other classifiers in Tables 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29 for other classifiers and datasets.

Therefore, we perform two-factor ANOVA tests with classifiers and datasets as the factors, and AUC or F1-score as the dependent variable. In all cases, with one exception, the p-values for the ANOVA tests are zero, so we conclude that classifier and dataset are significant factors affecting the outcome of experiments. The exception is for experiments involving the dataset with one feature of Destination_Port only. For this one-feature dataset, the classifier choice is not significant. We report the results of these experiments with the one-feature Destination_Port only dataset in Table 19, where we are forced to report results to 8 decimal places instead of the usual 5 to show any difference in performance when we use different classifiers. Otherwise, Tukey's HSD tests are appropriate for both classifier and dataset factors.

The dataset factor (feature group 1, 1A, *etc.*) in an experiment is equivalent to the application of a feature selection technique. In order to get a sense of the impact of feature selection and classifier choice, we conduct Tukey's HSD tests at a 99% confidence level for the dataset and classifier factors in order to gauge the effect of feature selection. We see in Figs. 1 and 3 that performance in terms of AUC and F1-score is influenced by the classifier. Reflected in Figs. 2 and 4, and according to the groupings the Tukey's HSD test yields, there is no significant difference performance in terms of AUC for group a, which consists of the feature selection technique where

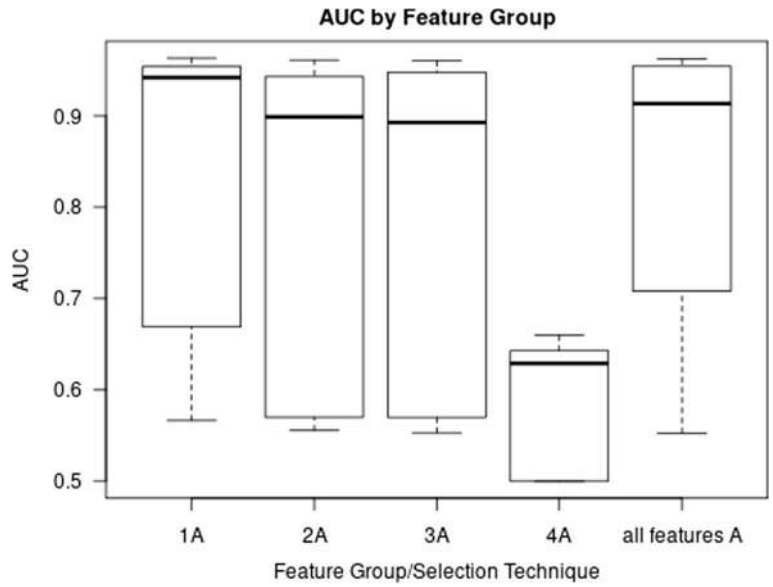

Fig. 2 Box plots of AUC grouped by feature group/selection technique; here datasets do not include destination port; Tukey's HSD test indicates feature groups/selection techniques 1A, all features A in group a, 2A 3A in group b and 4A in group c

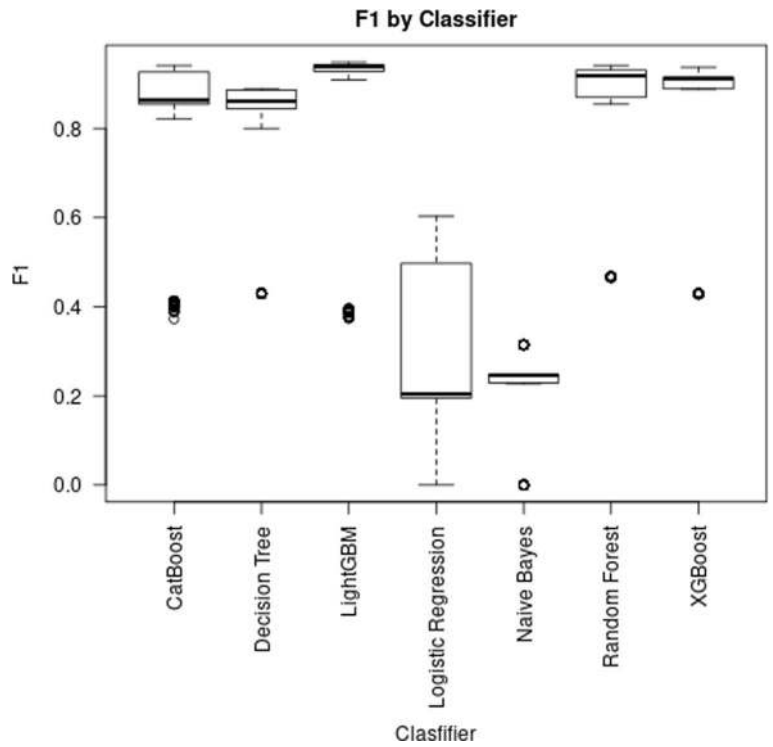

Fig. 3 Box plots of F1-score grouped by classifier; here datasets do not include destination port; Tukey's HSD test indicates LightGBM, Random Forest, XGBoost in group a, CatBoost and Decision Tree in group b, Logistic Regression in group c, Naive Bayes in group d

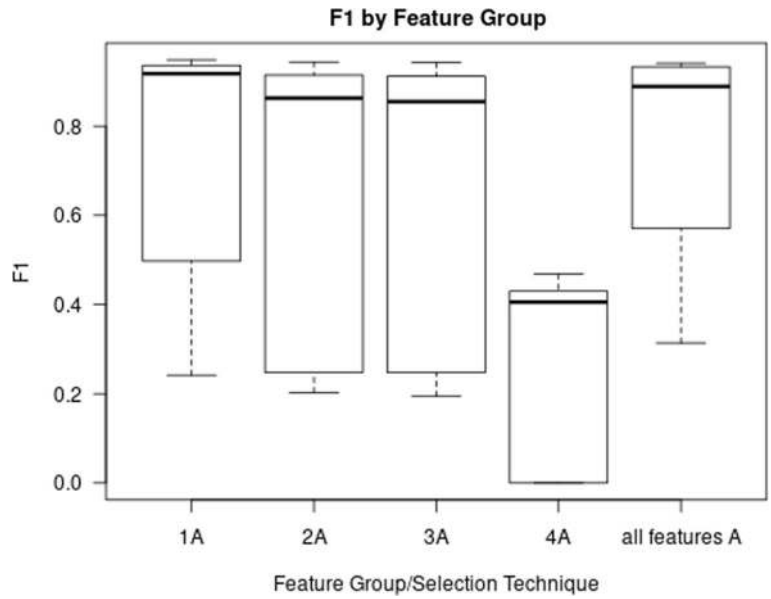

Fig. 4 Box plots of F1-score grouped by feature group/selection technique; here datasets do not include destination port; Tukey's HSD test indicates all features A in group a, 1A in group b, 2A in group c, 3A in group c and 4A in group d

we use features 4 out of 7 classifiers agree on (feature group 1A), and when we use all features (feature group all features A). This is an ideal result since it implies we obtain similar performance with a smaller dataset. However, in terms of F1-score, we do not obtain the ideal result, but one where performance in terms of F1-score is similar. We see in Fig. 4 that the F1-scores for classifiers trained on all features in feature group 1A are very close to the F1-scores that classifiers trained with data from feature group 1 yield. In fact, the adjusted p-value for the Tukey's HSD test for the difference in F1-score for feature groups 1 and all features is 0.0100414. We cite this adjusted p-value as another reason to claim that performance in terms of F1-score for classifiers trained with feature group 1A is similar, or better than the performance of classifiers trained with all features from CSE-CIC-IDS2018. However, results for the feature selection techniques 2, 3, 4, 2A, 3A, or 4A do not show the same conclusion.

Only CatBoost and LightGBM have built-in support for categorical features. Therefore, we deem it out of scope to address encoding techniques for the Destination_Port categorical feature in the 2018 dataset. As a result, we perform separate experiments to assess the impact of feature selection to further answer research question Q1. We conduct ANOVA to determine if the classifier and feature selection technique have an impact on the results for AUC and F1-score. Since p-values for the classifier and feature selection technique factors are nearly zero for the ANOVA tests, we conduct Tukey's HSD tests to check the levels for factors that yield the best performance. Box plots of results, grouped by factors analyzed in ANOVA and HSD tests, are depicted in Figs. 5, 6, 7 and 8.

In Figs. 5 and 7, we see the performance of LightGBM or CatBoost trained on feature group 1 is similar to the performance of LightGBM or CatBoost trained on all

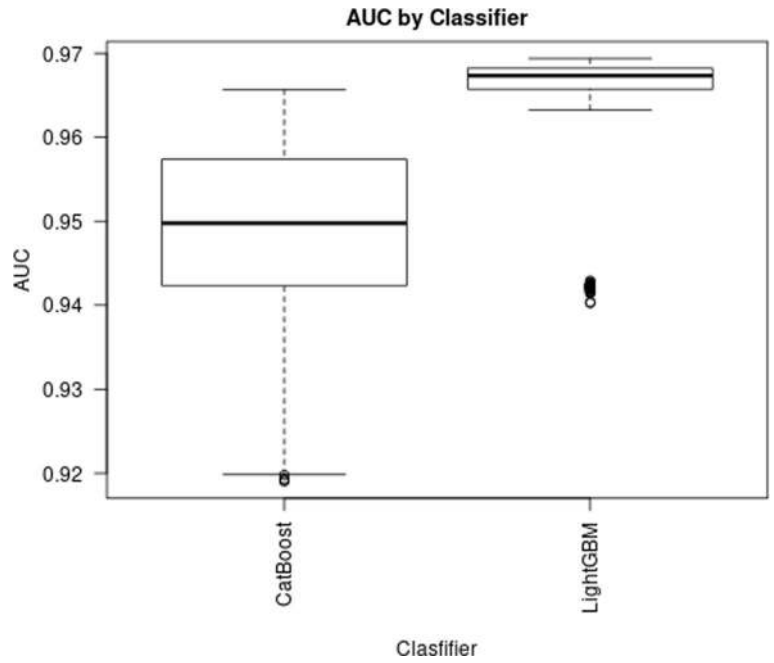

Fig. 5 Box plots of AUC values grouped by classifier; here datasets include destination port; Tukey's HSD test indicates each factor (classifier) is in its own group

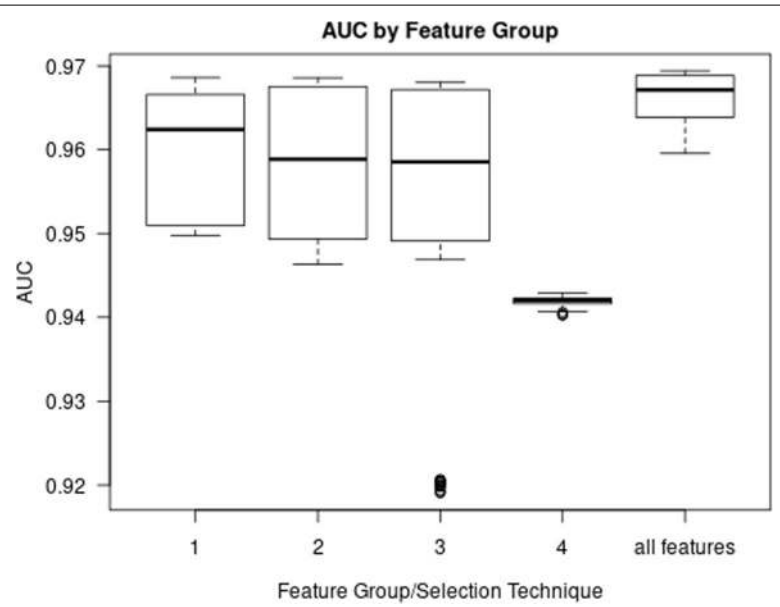

Fig. 6 Box plots of AUC grouped by feature group/selection technique; here datasets include destination port; Tukey's HSD test indicates feature selection techniques 1 and 2 are not significantly different, and other techniques are in groups of their own

features. Moving on to performance grouped by feature selection technique, we see in Figs. 6 and 8 that for models trained with feature group 1, the Tukey's HSD test yields a mean AUC of approximately 0.95999. For models trained with all features, the

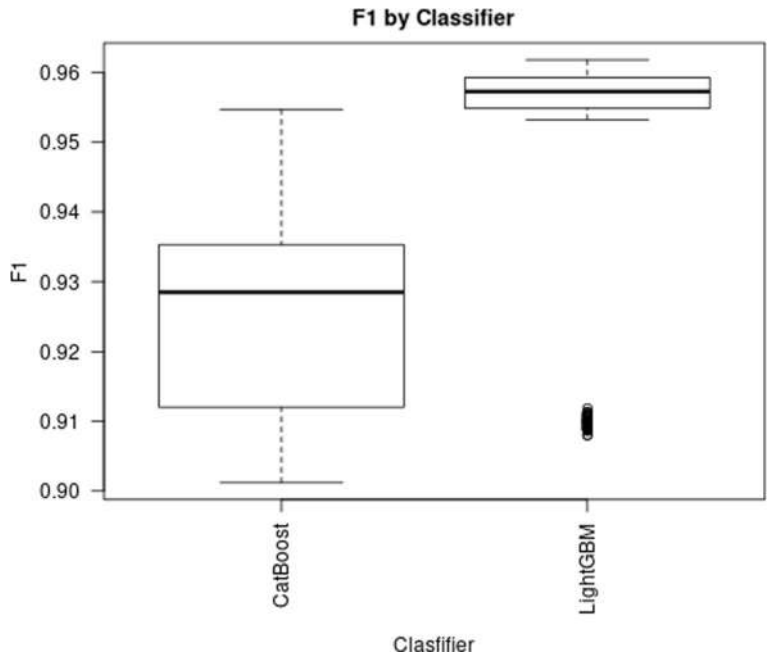

Fig. 7 Box plots of F1-score grouped by classifier; here datasets include destination port; Tukey's HSD test indicates each factor (classifier) is in its own group

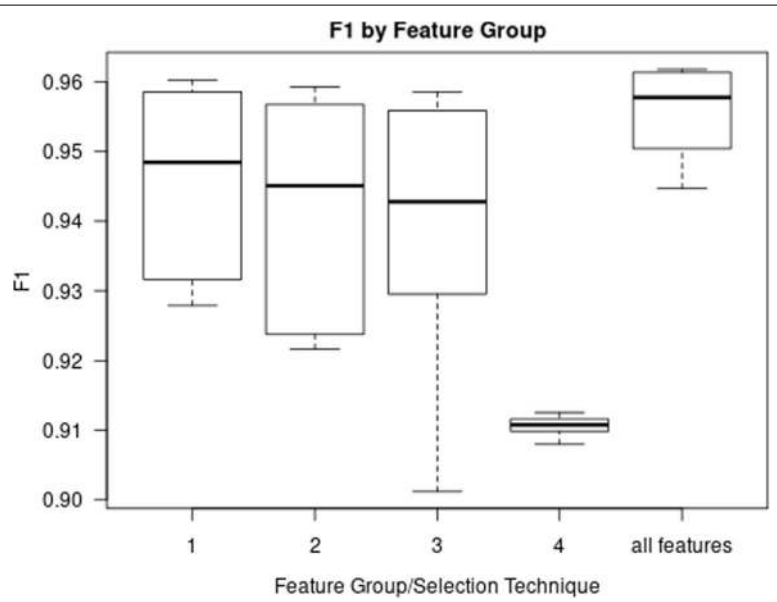

Fig. 8 Box plots of F1-score grouped by feature group/ selection technique; here datasets include destination port; Tukey's HSD test indicates feature selection techniques 2 and 3 are not significantly different, other techniques are in groups by themselves

Tukey's HSD test yields a mean AUC of 0.96640 (a difference of 0.00641). Likewise, the mean F1-scores for CatBoost and LightGBM are similar for models trained on feature group 1 and all features. In this case the Tukey's HSD adjusted mean F1-score

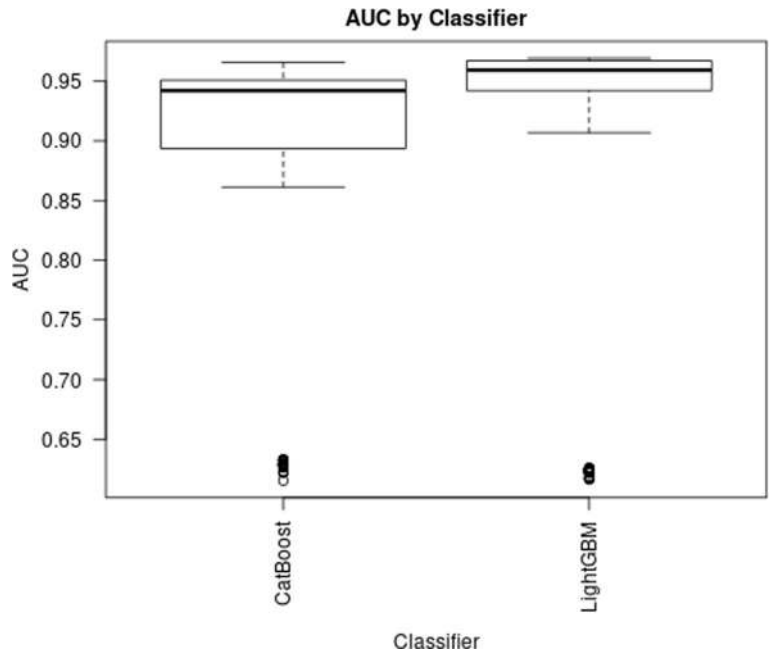

Fig. 9 Box plots of AUC grouped by classifier; Tukey's HSD test indicates classifiers are significantly different

is 0.94591 for models trained with data from feature group 1, and 0.95594 for models trained with all features (a difference of 0.0103).

Research Question Q1 Answer: Yes, our ensemble feature selection technique yields performance similar to, or better than, using all features. More specifically, the variant of our technique where 4 out of 7 classifiers agree on a feature is the criterion for feature selection that yields performance similar to, or better than, using all features.

Research question Q2: Does including the Destination_Port categorical feature significantly impact performance of LightGBM and CatBoost in terms of AUC and F1-score?

To answer research question Q2, we use results of experiments where classifier: CatBoost or LightGBM, is a factor, and the datasets' having, or not having the Destination_Port feature is another factor. We perform ANOVA tests on the results of experiments grouped by these factors. The p-values associated with classifier and dataset factors for the ANOVA tests are both zero. Therefore, Tukey's HSD tests are appropriate. We report the results of those tests in Figs. 9, 10, 11 and 12.

It is interesting to note that the ranges of values of both AUC and F1-score are smaller when we use a dataset that includes destination port. So, not only do the ANOVA and HSD tests confirm that including Destination_Port is a significant factor in the performance of models for identifying attacks, but our results here also show greater stability in the values of results. These results enable us to answer our second research question.

Research question Q2 Answer: Yes, including the Destination_Port feature has a significant impact on performance in terms of AUC and F1-score.

Research question Q3: Does the choice of classifier: RF, DT, NB, LR, CatBoost, LightGBM, or XGBoost, significantly impact performance in terms of AUC and F1-score?

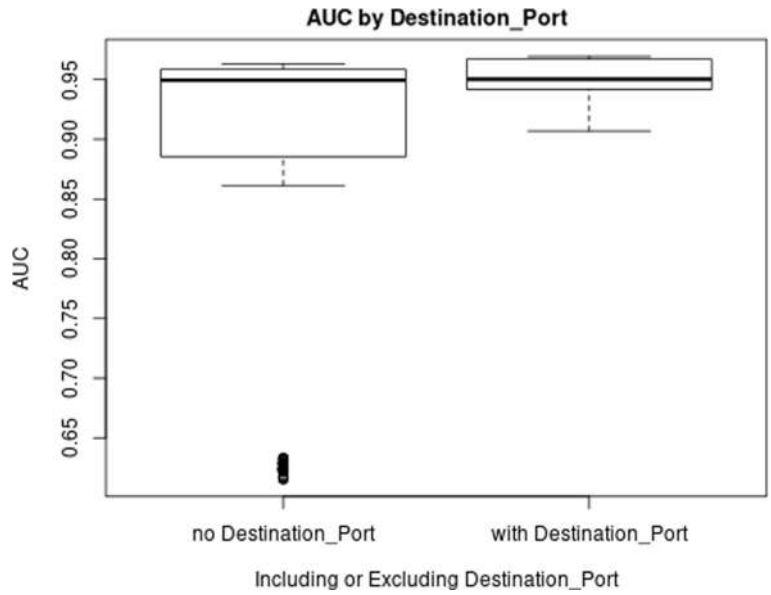

Fig. 10 Box plots of AUC grouped by whether the dataset contains the Destination Port feature; Tukey's HSD test indicates including Destination Port produces significantly different results

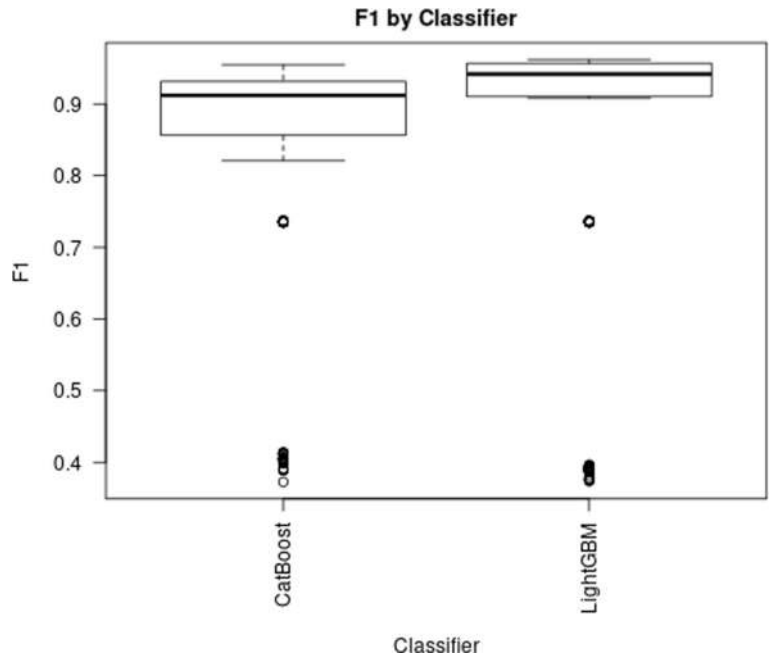

Fig. 11 Box plots of performance in terms of F1-score grouped by classifier; Tukey's HSD test indicates classifiers are significantly different

To answer the third research question, we note that all ANOVA tests we conduct show that classifier is a significant factor in experiments—the p-values associated with the classifier factor are 0. So, we perform Tukey's HSD tests to determine how classifiers may be grouped in terms of their performance. The groupings enable a

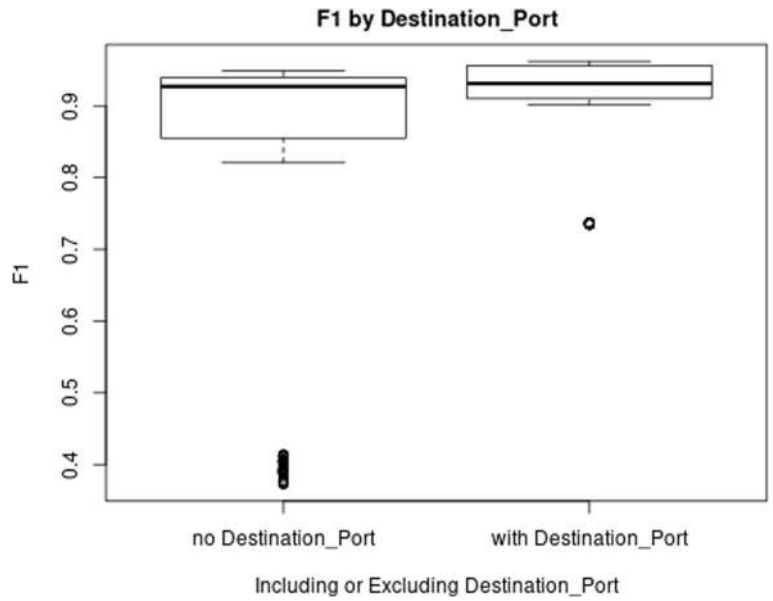

Fig. 12 Box plots of F1-score grouped by whether the dataset contains the Destination Port feature; Tukey's HSD test indicates including Destination Port produces significantly different results

conclusion to be drawn on how the choice of classifier impacts the outcome of an experiment. We refer to groupings we report from conducting Tukey's HSD tests in Figs. 1, 3, 5, 7, 9, and 11. Taking a closer look at Figures 1 and 3, we see that our HSD test results indicate that some classifiers do not have significantly different performance. For example, in Fig. 1 we report that the classifier performance of LightGBM, RF, and XGBoost in terms of AUC is not significantly different. However, the same HSD test indicates seven classifiers fall into four distinct groups. In Figs. 5, 7, 9, and 11, the Tukey's HSD test results reported indicate the classifier is an important factor in the outcome of experiments, in terms of AUC or F1-Score. We also note that LightGBM is consistently in the group that the HSD test identifies with the best value of AUC or F1-Score in all cases.

Research question Q3 Answer: Yes, the choice of classifier significantly impacts performance.

Conclusion

The results in Tables 19 through 24, as well as the results from Tukey's HSD tests depicted in Figs. 1 through 11, and the answer to research question Q1 show that the feature selection technique that produces feature group 1A performs similar to or better than using all features. These results demonstrate that our ensemble feature selection technique should be used with classifiers to detect anomalies in CSE-CIC-IDS2018, since training a model with the reduced feature set consumes fewer computing resources.

We may also draw conclusions from the results of the ANOVA and Tukey's HSD tests to answer research questions Q2 and Q3. Test results for research question Q2 indicate that Destination_Port is a useful feature for classifiers. Hence, we conclude one should encode it for use with a classifier, if the classifier does not handle categorical features automatically. Test results for research question Q3 reveal that LightGBM performs

similar to, or better than, any other classifier of CSE-CIC-IDS2018, even when we do not use Destination_Port as a feature for LightGBM.

Since our current study is limited to comparing CatBoost and LightGBM when we include Destination_Port as a categorical feature, we have an opportunity for future research to investigate whether another classifier might yield better performance in conjunction with a technique for encoding Destination_Port. There is also an opportunity to evaluate classifier performance with other network intrusion detection datasets. Another subject we have not broached here that deserves attention deals with techniques for addressing class imbalance, such as Random Undersampling (RUS) [61].

Abbreviations

ANOVA: ANalysis Of VAriance; AP: Affinity propagation; APT: Application programming interface; AUC: Area Under the Receiver Operating Characteristic (ROC) Curve; AUC: Area Under the Receiver Operating Characteristic Curve; CNN: Convolutional Neural Network; CS: Chi-Squared; CV: Cross-validation; DNS: Domain Name System; DT: Decision Tree; FAU: Florida Atlantic University; FN: False Negative; FP: False Positive; GBDT: Gradient-boosted decision tree; GPU: Graphics Processing Unit; GR: Gain Ratio; HSD: Honestly Significant Difference; IG: Information Gain; *k*-NN: k-Nearest Neighbor; LR: Logistic Regression; ML: Machine Learning; MLP: Multilayer Perceptron; NB: Naive Bayes; NSF: National Science Foundation; ODT: Oblivious Decision Tree; RF: Random Forest; ROC: Receiver Operating Characteristic; RUS: Random Undersampling; SVM: Support Vector Machine; TN: True Negative; TNR: True Negative Rate; TP: True Positive; TPR: True Positive Rate; TS: Target Statistic; AUC: Area Under the Receiver Operating Characteristic Curve; LR: Logistic Regression; NB: Naive Bayes; RF: Random Forest; DT: Decision Tree.

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 prepared the manuscript and the primary literary review for this work. RZ performed the data cleaning. JH performed the statistical analyses. All authors provided feedback to TMK and helped shape the research. TMK introduced this topic to JLL, and helped to complete and finalize this work. All authors read and approved the final manuscript.

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: 17 December 2020 Accepted: 11 February 2021
Published online: 23 February 2021

References

1. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP; 2018. p. 108–116.
2. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput Secur. 2012;31(3):357–74.
3. Thakkar A, Lohiya R. A review of the advancement in intrusion detection datasets. Proc Comput Sci. 2020;167:636–45.
4. Wald R, Khoshgoftaar TM, Zuech R, Napolitano A. Network traffic prediction models for near-and long-term predictions. In: 2014 IEEE International Conference on Bioinformatics and Bioengineering. IEEE; 2014. p. 362–68
5. Najafabadi MM, Khoshgoftaar TM, Kemp C, Seliya N, Zuech R. Machine learning for detecting brute force attacks at the network level. In: 2014 IEEE International Conference on Bioinformatics and Bioengineering. IEEE; 2014. p. 379–85.
6. Bekkar M, Djemaa HK, Alitouche TA. Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl. 2013;3(10).
7. Wald R, Villanustre F, Khoshgoftaar TM, Zuech R, Robinson J, Muharemagic E. Using feature selection and classification to build effective and efficient firewalls. In: Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014). IEEE; 2014. p. 850–54.

8. 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.
9. Leevy JL, Khoshgoftaar TM. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. J Big Data. 2020;7(1):1–19.
10. Wang H, Khoshgoftaar TM, Napolitano A. A comparative study of ensemble feature selection techniques for software defect prediction. In: 2010 Ninth International Conference on Machine Learning and Applications. IEEE; 2010. p. 135–40.
11. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In: 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI). IEEE; 2020. p. 184–91.
12. Najafabadi MM, Khoshgoftaar TM, Seliya N. Evaluating feature selection methods for network intrusion detection with Kyoto data. Int J Reliab Qual Saf Eng. 2016;23(01):1650001.
13. Lee J-S. Auc4. 5: Auc-based c4. 5 decision tree algorithm for imbalanced data classification. IEEE Access. 2019;7:106034–42.
14. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
15. 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.
16. Rymarczyk T, Kozłowski E, Kłosowski G, Niderla K. Logistic regression for machine learning in process tomography. Sensors. 2019;19(15):3400.
17. Hancock J, Khoshgoftaar TM. Medicare fraud detection using catboost. In: 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI). IEEE Computer Society; 2020. p. 97–103.
18. Hancock JT, Khoshgoftaar TM. Catboost for big data: an interdisciplinary review. J Big Data. 2020;7(1):1–45.
19. Hancock J, Khoshgoftaar TM. Performance of catboost and xgboost in medicare fraud detection. In: 19th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE; 2020.
20. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A. Mining data with rare events: a case study. In: 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol 2. IEEE; 2007. p. 132–139.
21. Hua Y. An efficient traffic classification scheme using embedded feature selection and lightgbm. In: 2020 Information Communication Technologies Conference (ICTC). IEEE; 2020. p. 125–30.
22. 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). Springer; 2014. p. 13–22.
23. 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.
24. Baig MM, Awais MM, El-Alfy E-SM. Adaboost-based artificial neural network learning. Neurocomputing. 2017;248:120–6.
25. Rynkiewicz J. Asymptotic statistics for multilayer perceptron with Relu hidden units. Neurocomputing. 2019;342:16–23.
26. 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.
27. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
28. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org 2015. https://www.tensorflow.org/
29. 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.
30. 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. Springer; 2017. p. 1–18.
31. 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. Springer; 2016. p. 185–193.
32. Gupta V, Bhavsar A. Random forest-based feature importance for hep-2 cell image classification. In: Annual Conference on Medical Image Understanding and Analysis. Springer; 2017. p. 922–934.
33. 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). IEEE; 2017. p. 354–60.
34. 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.
35. Chen J, Xie B, Zhang H, Zhai J. Deep autoencoders in pattern recognition: a survey. Bio-inspired Computing Models And Algorithms. 2019;229.
36. 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.
37. Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: an ensemble of autoencoders for online network intrusion detection 2018. arXiv preprint arXiv:1802.09089
38. 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). IEEE; 2020. p. 118–24.
39. Fadlil A, Riadi I, Aji S. Ddos attacks classification using numeric attribute based Gaussian Naive Bayes. Int J Adv Comput Sci Appl (IJACSA). 2017;8(8):42–50.

40. 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). IEEE; 2017. p. 1–6.
41. Abd Elrahman SM, Abraham A. A review of class imbalance problem. J Netw Innov Compu. 2013;1(2013):332–40.
42. 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 Statist Mech Appl. 2016;451:440–50.
43. 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.
44. D'hooge L, Wauters T, Volckaert B, De Turck FF. Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. J Inf Secur Appl. 2020;54:102564.
45. 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). IEEE; 2019. p. 1–5.
46. 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.
47. Saikia T, Brox T, Schmid C. Optimized generic feature learning for few-shot classification across domains; 2020. arXiv preprint arXiv:2001.07926
48. Sulaiman S, Wahid RA, Ariffin AH, Zulkifli CZ. Question classification based on cognitive levels using linear svc. Test Eng Manag. 2020;83:6463–70.
49. 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). IEEE; 2019. p. 1–6.
50. Zuech R, Khoshgoftaar TM. A survey on feature selection for intrusion detection. In: Proceedings of the 21st ISSAT International Conference on Reliability and Quality in Design; 2015. p. 150–155.
51. Witten IH, Frank E, Hall MA. Data mining: practical machine learning tools and techniques. 3rd ed. San Francisco: Morgan Kaufmann Publishers Inc.; 2011.
52. Agresti A. Categorical data analysis. Wiley Series in Probability and Mathematical Statistics. Applied probability and statistics, applied probability and statistics. Hoboken: Wiley; 1990. p. 42–3.
53. Singh R, Kumar H, Singla R. Analysis of feature selection techniques for network traffic dataset. In: 2013 International Conference on Machine Intelligence and Research Advancement. IEEE; 2013. p. 42–46.
54. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;1189–1232.
55. Gupta A, Nagarajan V, Ravi R. Approximation algorithms for optimal decision trees and adaptive tsp problems. Math Oper Res. 2017;42(3):876–96.
56. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference; 2010. p. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a
57. Rish I, et al. An empirical study of the naive bayes classifier. In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol 3; 2001. p. 41–46.
58. Seliya N, Khoshgoftaar TM, Van Hulse J. A study on the relationships of classifier performance metrics. In: Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference On. IEEE; 2009. p. 59–66.
59. Iversen GR, Wildt AR, Norpoth H, Norpoth HP. Analysis of variance. Thousand Oak: Sage; 1987.
60. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5:99–114.
61. Liu B, Tsoumakas G. Dealing with class imbalance in classifier chains via random undersampling. Knowl Based Syst. 2020;192:105292.

Publisher's Note