International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R

Zachariah Pelletier[1], Munther Abualkibash[2]
[1]School of Information Security and Applied Computing, Eastern Michigan University, Ypsilanti, Michigan, United States-48197
[2]School of Information Security and Applied Computing, Eastern Michigan University, Ypsilanti, Michigan, United States-48197

Abstract**—** *The analysis of network traffic is crucial to the design and implementation of Intrusion Detection Systems. The R language is a statistical computing environment capable of performing a variety of data analysis techniques. In this paper, we will use the R language to pre-process, analyze, and create a predictive model using the CIC IDS-2017 dataset capable of predicting whether or not network connections are malicious in nature. We will use the CIC IDS-2017 data to train both an Artificial Neural Network and Machine Learning algorithm and create a model capable of classifying labeled network data.*

Keywords**—** *R Language : Machine Learning : Neural Network : Network Traffic : nnet : Boruta : CIC IDS-2017 : randomForest : Intrusion Detection system.*

I. INTRODUCTION

The Intrusion Detection Evaluation Dataset (CICIDS2017) is a network traffic dataset comprised of both normal traffic and simulated abnormal data caused by intentional attacks on a test network. Intrusion Detection Systems and Intrusion Prevention Systems are critical in successfully detecting and mitigating attacks on computer networks from outside threats. Having up-to-date and relevant data from which to train these systems is imperative to the success rate of such systems and often makes the difference between a healthy and a compromised network. The IDS 2017 Dataset was created with the mitigation of contemporary attacks in mind while using a variety of realistic common net-work attacks to simulate attacks on a controlled network [1]. The focus lay on generating realistic and di-verse background data over the network while providing similarly realistic attack data. The team that generated this data set abstracted the behaviour and interactions of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols [1]. The creators of this dataset identified a total of eleven criteria that they deemed necessary for creating a reliable dataset for use as a benchmark, those being: a complete network configuration, complete traffic, a final dataset that is labelled, complete interaction, complete capture, available protocols, attack diversity, heterogeneity, feature-set with over 80 network features, and metadata [1].

R is a statistical computing language popular among data scientists. Among various statistical tools, R is also known for the availability of a wide selection of machine learning libraries and third-party tools. In this paper, we will analyse the IDS 2017 dataset using various machine learning techniques. We will use an R library called Boruta to gauge the relative importance of the network features in a systematic way. Further-more, we will create a predictive model using a Neural Network and machine learning methods. Since the IDS 2017 dataset is available in a variety of formats, we will use the comma separated value format of the network data to perform our analysis, although it is important to acknowledge that the full PCAP (descriptive net-work capture) files are also available for analysis [1].

II. DESCRIPTION OF THE DATASET

The CIC IDS-2017 Dataset was constructed using the NetFlowMeter Network Traffic Flow analyser [2]. The tool collects in excess of 80 network traffic features and supports Bi-directional flows [2]. Thakkar and Lohiya describe the setup for the data collection for the creation of the dataset [3]. It was captured over a duration of 5 days over which over 80 features and 15 classes were captured [3]. The attack infrastructure consisted of 4 PCs, 1 router, and 1 switch while the victim infrastructure was made up of 3 servers, 1 firewall, and 2 switches [3]. It differs from other traditional network datasets in that it uses different network profiles to generate ultra-realistic network data and attack data based on actual users [3]. Other network datasets such as NSL-KDD and the CAIDA rely on general labelled traffic with a limited number of attack types while the CIC IDS-2017 dataset uses a large variety of attack types and more pseudo-organic content from realistic user pro-files [3]. The entire dataset itself was designed and collected with the eleven characteristics of an ideal dataset in the forefront of the implementation [4]. While the CIC IDS-2017 dataset is considered a fairly recent and contemporary solution for supplementing intrusion detection systems, Thakkar and Lohiya have identified several shortcomings of the dataset and its usability. The first major issue that was identified was the sheer size of the data stored in each output file. Since there are a large number of data instances in each file, it is very cumbersome to process each file and load all of the data into memory. The size of the files also accounts for greater processing time and a decrease in the efficiency of many classification algorithms.

It was also identified that there are missing and redundant data records in this dataset, which can either skew and improperly weight the factors in a classification algorithm that uses the dataset to train a predictive model [3]. Finally, the CIC IDS-2017 dataset is prone to an issue called ‘high-class

Zachariah Pelletier and Munther Abualkibash, “Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R,” *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 2, pp. 187-191, 2020.

imbalance' which leads to low accuracy of the resulting predictive model [6]. While these issues continue to exist in this dataset, they can be eliminated or mitigated by more rigorously pre-processing the data before use.

III. Related Work

A. Deep Learning Approach for Intelligent Intrusion Detection System

The authors of this paper use a Deep Neural Network to detect anomalies related to attacks on a net-work system [5]. They use the IDS 2017 dataset as well as several other well known cyber security datasets as benchmark datasets for training their neural network and creating a predictive model that is capable of detecting unforeseen and unpredictable cyber attacks [5]. The authors of this paper aim to identify the best algorithm for effectively detecting future cyberattacks that is also flexible for use between multiple different datasets. The Deep Neural Network that is used to complete this classification was run with a cycle rate of 1000 epochs with a learning rate varying from 0.01 to 0.05 [5]. The resulting DNN that was built performed well when trained against the KDDCupp99 dataset as well as the NSL-KDD dataset [5].

The authors propose a scalable security framework that can scale with the systems it protects. The key instrument that allows this scalable architecture to function consists of several parallel machine-learning algorithms that are all optimized in different techniques for detection. This allows the system to handle a very large amount of network traffic into and between different facets of the system creating an increasingly complex fabrication of communication and tracing. To further enhance this security framework, the authors also worked to add modules for monitoring the DNS and BGP events in the networks [5].

B. A Detailed Analysis of CICIDS2017 Dataset for Designing Intrusion Detection Systems

In their paper analyzing the CICIDS2017 dataset for use with designing Intrusion Detection Systems, Ranjit Panhigrahi and Samarjeet Borah evaluate the dataset to identify its shortcomings and to attempt to relabel the dataset to reduce a high-class imbalance problem [6]. Both authors agree that the dataset includes quite a few redundant records, many of which seem to be irrelevant to the training of an IDS. This seems to be taken into consideration when the dataset was being constructed as most of the background (or benign traffic) is generated based upon individual user profiles. Due to the overwhelming number of benign records, however, the authors conclude that there are too few attack vectors to be useful to an intrusion detection system.

Another major shortcoming identified by the authors is missing or incomplete information in the dataset. They have observed 288602 instances of connections with missing class labels and 203 instances of connections with other missing information [6]. Finally, they touched on an issue that is prevalent in the CIC2017 dataset where the dataset is biased towards a majority class that is more prevalent in the dataset [6]. In this case, benign and DoS Hulk connections are much more prevalent in the dataset and which results in lower accuracy of the resulting IDS and a higher rate of false alarm. The authors proposed a relabeling scheme to solve this issue which involves combining the minority classes and splitting up the majority classes into smaller subclasses [6].

C. Improving AdaBoost-Based Intrusion Detection System (IDS) on CIC IDS2017 Dataset

In their paper, the authors attempt to address the high-class imbalance problem present in the CIC IDS 2017 dataset, allowing the dataset to perform better with AdaBoost machine-learning techniques [7]. In order to rebalance the classes in the dataset and mitigate the resulting bias, a strategy incorporating the Synthetic Minority Oversampling Technique (SMOTE) is proposed. [7]. As addressed in this paper, this technique is used to enhance the sensitivity of arrangement for the minority classes in the dataset [7]. In order to implement this system, the authors ran the data through the SMOTE algorithm to relabel enhance data using this technique. They then used the R programming language to select optimal features from the dataset, then fed the selected data into an AdaBoost machine learning algorithm to create a predictive model. In order to accurately select optimal features from the dataset to determine importance, they used the R package EFS [10].

IV. Data Pre-Processing

We used the comma separated values format of the CIC IDS 2017 Dataset for all of our processing. This dataset was available from the Canadian Institute for Cybersecurity [1]. This source provides both original PCAP files and a consolidated series of CSV files to work with. Since the R programming language is able to process CSV files quite well, we decided to use these instead of the PCAP files. A series of steps had to be taken to ensure that the data is in an appropriate format to be processed by our algorithms, in addition, we had to make sure that there were no anomalies in the data that would otherwise cause issues later. One area where there was an especially large amount of friction was attempting to automate the testing for importance. In order to determine which network features were most important in our dataset, we used a third-party library called Boruta [7]. Algorithms used to determine the importance of features are very specific, and it does not take much to throw off the results.

The first major issue we ran into regarding the format and structuring of the CSV data was the fact that there were duplicated columns in the dataset. After receiving a runtime error when trying to process the data using Boruta, we manually combed the data to find the superfluous column. It turned out that the column labeled "fwd_header_length" showed up as both column number 35 and column 56 and after scripting a solution to check the contents of both columns, it was determined that the data was identical and could therefore be removed entirely without sacrificing any of our vector features. We did not spend extra resources looking into whether this same data was replicated in the original PCAP data that was provided from the data source, but all of the

Zachariah Pelletier and Munther Abualkibash, "Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 2, pp. 187-191, 2020.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

CSV files were found to have this issue. Additionally, some values in the data set were either set as the string 'Infinity' or 'NaN' for Not a Number. It was not determined which overflow or non-numerical values caused these to be inserted into the dataset, but this became an issue when setting the datatype of each column. In order to process our data using the Boruta library, we had to replace all instances of these values with numerical values that would not skew our data. We decided to find the maximum value for the column that we found each instance of the 'Infinity value' and replace it with that value doubled. In each case of NaN we decided to take the mean of each column and use that value to replace all of these instances.

Additionally, as mentioned in related work [6], there is a large amount of the dataset that has missing information (either class information or other information). In the R language, these values result in an "N/A" or missing value for the data frames which halt some our algorithms, such as the Boruta test for importance. In order to successfully make an evaluation on the dataset, we first had to eliminate instances of packets with partial or missing information using the na.omit() function in R. Failure to properly handle this will result in several compiler errors from the system.

The last step that needed to be taken to ensure that all of the data could be processed properly was to actually combine all of the different labeled data into one file. As mentioned in [3], the dataset itself is separated by attack type so that each attack type has its own file along with benign data. In order to fully process a complete model (with all attack types), all of this data has to be combined into one complete file containing all records. This became very cumbersome very quickly as the resulting file was almost 1 gigabyte in size. Just reading the file into memory to be processed by our algorithms would take on average four to five minutes (the exact specifications of the computer we used can be found in table I).

TABLE I. Hardware Specifications

Hardware Specs On Test Computer		
Processor	**Memory**	**GPU**
2.0 GHz Intel Xeon Gold CPU 5117 8 Cores	64 GB	NVIDIA

V. Determination of Importance

In order to determine the relative importance of each individual network feature in the CIC IDS 2017 dataset (of which there were over 80), we ran the dataset through an automated test of importance in determining the packet type using a package called Boruta [8]. Due to the size of the complete dataset (which totaled 741 MB), we decided to cap the number of iterations under which the tests for importance would be run to 20. Even with the number of recursive iterations capped, the entire test of importance for every feature in the dataset to be determined took a total of 38.4 hours to complete. The determinacy of selection for each epoch took on average 2 hours to complete. The top 10 most importance features from this determination can be found in table II.

TABLE II. Importance Benchmarks for Network Features

Feature	**Importance Benchmarks for Network Features**		
	Mean Importance	**Min Importance**	**Max Importance**
Init_win_bytes_fwd	42.851426	40.324064	46.759915
Ack_flag_count	24.3363771	23.375138	25.527398
Fwd_packet_per_sec	24.0779123	22.208183	25.716682
Flow_packets_per_sec	23.1960694	21.509125	25.351291
Flow_iat_max	22.7701305	21.042116	23.934746
Flow_iat_min	22.5902703	18.070844	28.218225
Flow_duration	22.5343098	20.672436	24.731897
Init_win_bytes_bwd	22.4179293	20.433875	23.439661
Subflow_bwd_bytes	22.1918294	20.742846	24.335224
Flow_iat_mean	21.9582785	20.421413	23.388285

An interesting result to note from the relative importance feature selection, is that all of the different metrics for measure flow_iat were select within the top 15 most important features. The network connection feature with the highest importance (by far) was init_win_bytes_fwd. According to the NetFlowMeter homepage, init_win_bytes_fwd refers to the number of bytes sent in initial window in the forward direction [2]. The next important network connection feature with the ack_flag_count which refers to the number of times the ACK flag was set in packets traveling over the connection [2]. Overall, it will be important to consider all of the flow_iat metrics in our connection vectors moving forward, and it is worth noting that the init_win_bytes_fwd is a very strong predictor of the label type for a given connection vector. While these ten features were found to be the most important to determining the label of the data, the following features were determined to have no importance to the predictive model whatsoever. These features can be found in table III.

TABLE III. Least Important Benchmarks for Network Features

Feature	**Least Important Benchmarks for Network Features**		
	Mean Importance	**Min Importance**	**Max Importance**
Bwd_psh_flags	0.000000	0.000000	0.000000
Bwd_urg_flags	0.000000	0.000000	0.000000
Fwd_avg_bytes_per_bulk	0.000000	0.000000	0.000000
Fwd_avg_packets_per_bulk	0.000000	0.000000	0.000000
Fwd_avg_bulk_rate	0.000000	0.000000	0.000000
Bwd_avg_bytes_per_bulk	0.000000	0.000000	0.000000
Bwd_avg_packets_per_bulk	0.000000	0.000000	0.000000
Bwd_avg_bulk_rate	0.000000	0.000000	0.000000

VI. Building a Predictive Model

A. Implementation Using NNET

In order to create a predictive model and evaluate the efficiency of the CIC IDS 2017 dataset, a neural network was created and configured in R. In order to create and train an artificial neural network (ANN), we used the NNET library [9]. To start off, we created a neural network to check against the label for the data and added a subset of the dataset to start training for. We selected a randomized sample of 1 million connection vectors between lines 1 and 7 million, another million between lines 7 million one and 14 million, and another million lines between 14 million one and 23 million. We subsequently used these samples to train our neural network. Additionally, we configured our ANN to use 5

Zachariah Pelletier and Munther Abualkibash, "Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 2, pp. 187-191, 2020.

neurons to form the network and have a decay of 1.0-5. Finally, we configured the ANN to have 500 iterations. At this point, we made our first attempt to train the ANN and build our predictive model.

Upon initially loading our data into the program and running the setup for NNET to create the ANN, we encountered a problem with the R session. It seems that there was too much data to load in and successfully run the program. In order to achieve a running result, it was determined that we had to remove some of the data in order to do any analysis on it. In order to achieve this, we decided to try to limit the number of connections while simultaneously fixing the class imbalance issue discovered in [6]. Since the high-class imbalance presents itself and favors benign traffic, we decided to limit the amount of benign traffic by removing approximately 25% of the benign connection vectors. We also decided to randomize the order of the connection vectors as they were separated by attack type. In this way we were able to solve the issue with the R session crashing while also solving the high-class imbalance issue and making our raw data more unpredictable.

Our first running attempt resulted in several errors from the NNET package [9]. We got a compilation error stating that there were too many "weights" or factors in our training data. In order to attempt to fix this problem, we increased the maximum number of weights considered in the model as 500,000. This solution seemed to fix the previous error but resulted in an error pertaining to the neural network encountering missing or partial values in the connection vectors. When combing the data for incomplete data, we found some missing information in the connection vectors, but there were far too many to remove manually. The best implementation we found to remove the connection vectors that contained these partial values from the dataset altogether was to apply the na.omit() function to the entire dataset. This removed each instance in the dataset containing missing values, but these still could not be used successfully to train our ANN. In order to enforce a dataset that did not contain any empty vectors, we had to pass an argument into the NNET instance on creation called na.action and we set the action to na.omit. This allowed the ANN to automatically omit N/A values when running the training algorithm. The results from this classification can be found in table IV.

We first trained the ANN with a maximum iteration range of 50 iterations, then 500 iterations to compare the effect of more training sessions on the ability for the model to successfully predict network traffic. Our model trained on 500 iterations had an average success rate of 96.53% over a period of 89.163 minutes while our 50-iteration model's average success rate was only 87.79% over a period of 82.47 seconds. While the individual prediction rates by attack differed only slightly between the two levels of iteration ranges, the overall average prediction rate decreased by a significant amount. Overall, it seems that a high-class imbalance issue still exists in the dataset due to some types of attacks being more successfully predicted than others, but it does not seem as big of an issue here.

TABLE IV. Attack Detection Metrics for 500 Iteration Neural Network

Attack	Attack Detection Metrics For 500 iteration Neural Network		
	Number of Connections Correctly Predicted	Total Connections	% Correct
Benign (Normal Traffic)	1839789	1841023	99.933%
DDoS	128025	128027	99.998%
Portscan	158898	158930	99.980%
Bot	1960	1966	99.695%
Infiltration	34	36	94.444%
Web Attack: Brute Force	1505	1507	99.867%
Web Attack: XXS	652	652	100%
Web Attack: SQL Injection	19	21	90.476%
DoS Slowloris	5582	5796	96.308%
DoS Slowhttptest	5381	5499	97.854%
DoS Hulk	200067	231073	86.582%
DoS GoldenEye	10175	10293	98.854%
Heartbleed	10	11	90.909%

B. Implementation Using randomForest and Caret

RandomForest is a machine learning library that 'random forests' of input for classification and regression testing [11]. Caret is an R library capable of classification and regression training [12]. It is also capable of plotting regression and classification models in a visual aspect. Between both libraries, we were able to create a predictive model that was trained using our CIC IDS-2017 data and compare the results across the different attack vectors. Our goal was to use machine-learning algorithms to create a predictive model with the same data and compare the results against those from our ANN. We decided to use the same subsets of data for training and compare our results by testing our model against the complete data set.

The configuration for Caret was relatively simple with the only two requirements for use being the setup of a data partition and a training control initialization. For the data partition, we passed in an equation looking primarily at the label of the data and we used 10 epochs for the training control. In order to train our predictive model, we used the train method from Caret passing in our data, the method to be used for training (randomForest), and the metric which accounts for accuracy. After attempting to run the algorithm with our data, we received several errors regarding missing data as we did when using the ANN. However, this time simply using the na.omit() function to omit all of the data was not effective in actually filtering the data in a way that would allow the algorithm to run.

It seemed that data was still present in the dataset that had incomplete cases. In order to run the data through the algorithm successfully, an additional step had to be taken to ensure that every connection vector did not have any missing data, we had to filter the data using the complete.cases() function and pass only those cases into the training algorithm. Finally, we had to explicitly tell R that in the cases where incomplete data was found, the entire connection vector should not be included in the training data. In order to do this, we had to pass the na.action parameter into the training

Zachariah Pelletier and Munther Abualkibash, "Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 2, pp. 187-191, 2020.

function and assign it to "Exclude". After making this last configuration, we were able to successfully run the function to train the predictive model and get out results. The results from this training method and classification can be found in table V.

TABLE V. Attack Detection Metrics for randomForest Training

Attack	Attack Detection Metrics For randomForest Training		
	Number of Connections Correctly Predicted	Total Connections	% Correct
Benign (Normal Traffic)	1834784	1841023	99.661%
DDoS	127844	128027	99.857%
Portscan	158798	158930	99.917%
Bot	1912	1966	97.253%
Infiltration	33	36	91.167%
Web Attack: Brute Force	1477	1507	98.009%
Web Attack: XXS	649	652	99.540%
Web Attack: SQL Injection	20	21	95.238%
DoS Slowloris	5640	5796	97.309%
DoS Slowhttptest	5486	5499	99.764%
DoS Hulk	217813	231073	94.262%
DoS GoldenEye	10018	10293	97.328%
Heartbleed	9	11	81.818%

After running the training program and checking the results against the actual labels for the data, the randomForest machine learning algorithm was found to have an average accuracy rate of 96.24% and took 68.35 hours to execute. Comparing our results, individual attack types seemed to have differing results between our two methods of classification, but the overall average classification rate was comparable. We found that the results from the randomForest machine learning algorithm were more consistent across different attack types.

VII. Conclusion

A full analysis and review of the CIC IDS 2017 dataset used for training Intrusion Detection Systems has been performed and evaluated with several machine learning methods. We have found and examined many of the advantages and shortcomings that this dataset brings with it. Overall, we have found that the dataset excels in generating realistic background (benign) traffic using a variety of user profiles and has a good array of different contemporary attacks. It has also been determined that the dataset suffers from overall size and it is often too bulky to use for performing any quick tasks. In many cases, the actual virtualization of the software used to process the data cannot handle the size of the dataset file and often changes needed to be made to accommodate these issues.

It has also been found that the data itself is not always reliable when working with processes that require complete data rows. There are many cases where the data cells read 'NaN', 'Infinity', or simply cease to exist. Along with the inconsistent data, the CIC IDS 2017 dataset has been found to suffer from a high-level class imbalance, leading to the major (most common classes) overpowering the minority classes. Without actually eliminating data, the best proposed method for solving this issue is to relabel and reallocate the data to give the minority classes a more proportional amount of data for the predictive model to use for training. We used the Boruta test for importance to determine which features the in the dataset were most important, and which of the features were least important.

Finally, we build an Artificial Neural Network and used machine learning algorithms from third-party R libraries to build several predictive models using the data. We have found that the success rate of prediction for the connection vector differs by attack type, but the average rate has turned out be highly successful (about the 96th percentile in both cases). Overall, we have determined what good features lie in both the dataset and the R programming language, and what applications best fit this solution for Intrusion Detection Systems.

References

[1] I. Sharafaldin. *Intrusion Detection Evaluation Dataset (CICIDS2017)*, Canadian Institute for Cybersecurity, January, 2018. Accessed on May 12, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[2] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features", In the proceedings of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal, 2017.

[3] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In the proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), p. 407-414, Italy, 2016.

[4] A. Thakkar, R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets", In Procedia Computer Science issue 167 (2020), p. 636-645.

[5] A. Gharib, I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani. "An evaluation framework for intrusion detection dataset", In: 2016 International Conference on Information Science and Security (ICISS). IEEE; 2016. p. 1–6.

[6] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman. "Deep Learning Approach For Intelligent Intrusion Detection System", In: IEEE Access; Volume 7, 2019. P. 41525-41550.

[7] R. Panigrahi, S. Boorah. "A Detailed Analysis of CICIDS2017 Dataset for Designing Intrusion Detection Systems", In the International Journal of Engineering and Technology; Volumne 7, 2018. p. 479-482.

[8] A. Yulianto, P. Sukarno, N. A. Suwastika. "Improving AdaBoost-Based Intrusion Detection Systems (IDS) on CIC IDS 2017 Dataset", In J. Phys.: Conf. Ser. 1192 012018.

[9] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

[10] "*Package 'Boruta',*" Jul. 17, 2018. Accessed on April. 25, 2020. [Online]. Available: https://cran.r-project.org/web/packages/Boruta/Boruta.pdf

[11] *"Package 'nnet',"* Feb. 25, 2020. Accessed on April. 20, 2020. [Online]. Available: https://cran.r-project.org/web/packages/nnet/nnet.pdf

[12] "Package 'EFS: Ensemble Feature Selection'," Accessed on April 30, 2020. [Online]. Available: https://cran.rproject.org/web/packages/EFS/index.html

[13] *"Package 'randomForest',"* Mar. 25, 2018. Accessed on May. 3, 2020. [Online]. Available: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[14] *"Package 'Caret',"* Mar.20, 2020. [Online]. Available: https://cran.r-project.org/web/packages/caret/caret.pdf