Network Anomaly Detection by Means of Machine Learning: Random Forest Approach with Apache Spark

Hesamaldin HAJIALIAN[1], Cristian TOMA[2]
The Bucharest University of Economic Studies, Tehran, Iran
The Bucharest University of Economic Studies, Bucharest, Romania
hesamsheva7@gmail.com, cristian.toma@ie.ase.ro

Nowadays the network security is a crucial issue and traditional intrusion detection systems are not a sufficient way. Hence the intelligent detection systems should have a major role in network security by taking into consideration to process the network big data and predict the anomalies behavior as fast as possible. In this paper, we implemented a well-known supervised algorithm Random Forest Classifier with Apache Spark on NSL-KDD dataset provided by the University of New Brunswick with the accuracy of 78.69% and 35.2% false negative ratio. Empirical results show this approach is well in order to use for intrusion detection system as well as we seeking the best number of trees to be used on Random Forest Classifier for getting higher accuracy and lower cost for the intrusion detection system.
Keywords: *Random Forest, Network Security, Anomaly Detection, NSL-KDD, Apache Spark, Machine Learning, Intrusion Detection Systems (IDS)*

1 Introduction

Cyber-attacks in contrast with past have changed and threat the valuable information in financial, commerce, military, and industrial networks. The intrusion detection system (IDS) has a special place for preserving network from anomaly behaviors, a network security system for detecting vulnerability exploits against target computer or application [1]. There are two types of IDS: Network-based systems, Host-based systems. The network-based, monitoring system or systems on the network and determine this network traffic is an intrusion or acceptable but in the other hand, the Host-based systems monitoring on the system and examine the activity is an intrusion or not [2]. The methodology for detecting incidents are Anomaly-based detection, Signature-based detection, and Stateful protocol analysis. The Anomaly-based detection considering the definitions of normal activities and based on that determine this particular kind of activity is a deviation or not, this kind of methodology is very useful for detecting unknown activities [3]. Even with this situation, the cyber-attacks can penetrate in networks so the need for intelligent systems in order to decrease the cyber-attacks threats are increased and the old approach is not the best way anymore. Many recent studies have focused on to apply machine learning algorithms to improve the intrusion detection systems, the central issue in these studies is to evaluate the variant machine learning algorithms on datasets in order to identify their performance that which algorithms are more efficient in order to use in intrusion detection systems, upgrade their performance and make them more intelligent and also process fast as much as possible to investigate the anomalies. In this paper, we aim to propose a framework using Apache Spark the lightening-fast clustering and engine for big data processing in order to be used for anomaly detection in intrusion detection systems. In this part we review the related works, Farnnaz and Jabbar [4] used data mining on NSL-KDD by Weka software and evaluated the Random Forest model for detecting attacks and normal behavior, the proposed model was efficient with low false alarm rate. Kumari et al. [5] discussed implementing a K-mean model in order to determine and clustering anomalies in network traffic with Spark streaming on Cloudera virtual machine by streaming K-means algorithm in real-time. Rettig et al. [6] introduced a new system based on Pearson correlation and relative entropy for online anomaly detection over big data streams builds on Apache Spark, their

DOI: 10.12948/issn14531305/22.4.2018.08

studies show Pearson correlation is best-suited for detecting abrupt changes while the relative entropy is well-suited for detecting gradual changes. Sommer and Paxon [7] discussed the challenges of applying machine learning algorithms in order to find anomaly detection and they proposed guidelines for overcoming these challenges, they argued the capabilities and limitation from an operational point of view is important. Lee and Stolfo [8] outlined a data mining framework for intrusion detection, first they analyzed dataset and feature extraction then used a classification algorithm to compute the detection model, the experimental results were done on DARPA98 dataset. Bhuyan and Kalita [9] provided the structural overview of various facts of anomaly-based network intrusion detection also discussed the detection strategy and evaluations and presented various detection methods, strategy, and tools. Aggarwal and Sharma [10] analyzed KDD dataset with four classes Basic, Content, Traffic, and Host by means of Weka software in order to categorize all features and also the main target was to improve detection rate and false alarm rate by simulating Random Tree algorithm. In related research, some of them did not use Apache Spark for training, testing and detecting intrusion as fast as possible and some of them did not use Random Forest classifier for intrusion detection or just it was a survey about the tools or methods using in intrusion detection. In our research, the focus is to use Random Forest classifier algorithm by Apache Spark in order to compute huge datasets as fast as possible then train and test the algorithm on NSL-KDD dataset and evaluate this proposed approach for intrusion detection systems, also the main question in this research is how many trees should be used? For Random Forest classifier in this particular situation.

2 Problem Formulation

The objective of the article is to propose the model for intrusion detection with Random Forest Classifier algorithm by Apache Spark and also answer this question: how many trees should be used for getting the most accuracy in this case for detecting the anomalies? The acquired dataset is NSL-KDD from the University of New Brunswick improved version of KDD99 [11] dataset. This dataset is included by many intrusions simulated in the military network environment and also publicly accessible. The key outcome of the current article is to propose a model capable of detecting and predicting intrusions in the network area with high accuracy and low cost. The results also indicate the suggested amount for the number of trees in the Random Forest for detecting intrusions. By obtaining such data we could have a concrete and scalable model with high accuracy along with high-speed processing and low cost, also tuned and combined multiple algorithms just in a single step. In the reminder of this paper, first we create our model based on Random Forest Classifier by ML Pipeline component then evaluating the model after that investigate the Random Forest Classifier with various number of trees in order to identify how many trees are proper for our model to get best accuracy and cost for detecting intrusions or anomalies in the network via Apache Spark.

2.1 Random Forests

Random Forest algorithm is a part of tree-based classification algorithms and one of the most successful machine learning models, this algorithm is based on the ensembles of decision trees. The main concept of this algorithm is to increase the accuracy of the decision tree, in order to achieve this goal chooses the random subspaces of the features and build multiple trees for the randomness and this approach generalize and improve the classification then in order to predict a particular class aggregates the votes of tree and the class with the most votes is the prediction result. Because this algorithm uses multiple decision trees reduce the overfitting and it does not need any feature scaling and also could recognize the non-linearities features and feature interaction also it supports binary, multiclass classification and regressions both categorical and continuous features. The training of each tree doing separately, so could be done in parallel and also combining the predictions of

DOI: 10.12948/issn14531305/22.4.2018.08

each tree reduces the variance of the predictions and improving the predictions on the test data [12] [13].

2.2 NSL-KDD

The NSL-KDD dataset [14] is an improved version of the KDD'99 dataset. They solved the inherent problems of KDD'99, they cleaned dataset from the redundant or duplicated information in training data in order to avoid any biasing toward the frequent records also the duplicated records do not exist in the proposed test dataset that will affect the performance of prediction and not biasing towards the methods have better detection in frequent records. Another feature of this dataset is the test dataset and training dataset have the reasonable number of records that causes to have consistent and comparable results between different learning algorithm and also we do not need to make small portion or cross-validation or bagging in our training dataset in order to evaluate our models so we could run our experiments on complete train and test dataset also the selected records of different level group has the inversely proportional relation with the number of records in the original KDD99 dataset which help us to have more efficient and accurate evaluation for different machine learning algorithm [15].

3. Problem Solution

In this section, we discuss the implementation of the anomaly detection model or framework in the network environment based on the Random Forest Classifier and also investigate the best number of trees for the algorithm based on the cost and accuracy indicators also the easiness of the designing and speed of the computation are very important. Therefore, to meet these requirements, the Apache Spark is selected for creating and proposing such a model with the aforementioned criteria in order to detect anomalies. Apache Spark [16] is an engine for large-scale big data and lightning-fast clustering-computing framework which combined of SQL, streaming and complex analytics and included with libraries; SQL and DataFrames for structural data analyzing and processing, Spark Streaming for real-time streaming jobs processing and fault-tolerant, ML and MLlib for machine learning tasks that is 100x faster than MapReduce because of high-quality algorithms and excels better in iterative computations and also GraphX for graph computing. Apache Spark works interactively with various shells as Python, Java, Scala, SQL and also runs in various data sources as HDFS, Cassandra, Apache Hive and etc. Apache Spark has a concept the ML Pipeline that uniforms multiple algorithms on top of the DataFrames. The ML library represents the Pipelines such a workflow which consists of multiple stages. Hence by using this approach we could create models for anomaly detection easier and combine various algorithms just in a single pipeline, so it has a great effect on the performance of creating the models. In this study, the experiments are executed on the Toshiba Satellite L505-13x with Intel Core i3 2.13GHz, 8GB of RAM on Windows 7 64-bit OS and also the version of Apache Spark is 2.1.2 and Hadoop 2.7 with Python3 kernel, The Apache Spark ran with 2 worker threads on two cores locally.

4. Proposed Model

In this section, we present the workflow of the model based on the Apache Spark and evaluate the model with different evaluation metrics after that based on the model we had, investigating that how many trees needed for this model to have better accuracy? The experiment is done on the various number of trees in order to have a better conclusion.

4.1 Anomaly Detection Framework

Figure 1 shows the workflow of the proposed model. The training network traffic dataset (train NSLKDD.csv) with 14MB size is read by Apache Spark then the 42 selected features are considered for creating and implementing the Random Forest Classifier Algorithm; the two key values for the algorithm maxBins and numTrees are set 70 and 100, respectively.

DOI: 10.12948/issn14531305/22.4.2018.08

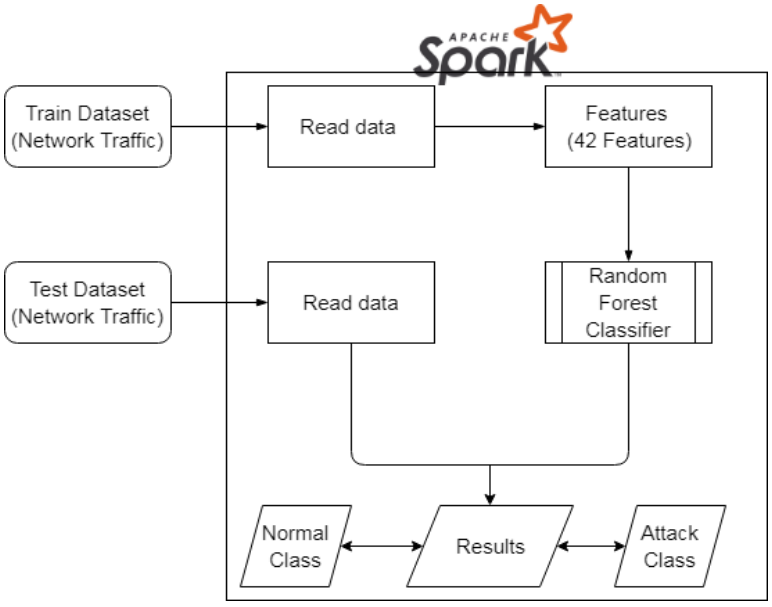

Fig. 1. Workflow of Model

The reasons are that the diff_srv_rate column has more than 70 distinct values and for the number of trees Oshiro and et al. [17] suggested that a ranged number of 64 to 128 are good in big data with great number of attributes to get the balance results for memory consuming and also the cost of processing time. So we chose the 100 in order to make the ideal anomaly detection model. As can be seen in Figure 1 in order to test the detection model, we need the separate test dataset to get more confident and accurate results. So the testing network traffic data (about 3MB) is read by Apache Spark and after that, the Random Forest algorithm is implementing on the test DataFrames to get the results. The Random Forest approach in this study is based on the binary classification so the results also split up to the normal and attack groups. When the model is ready, the analysis of model or framework is essential to elicit that the model is acceptable for use in intrusion detections or not?

4.2 Analysis of Empirical Results

The evaluation of the experimental results is the most important part, in order to see how the performance of the detection model is, in this study, we considered the results in binary classification; attack and normal classes. So in order to evaluate the model we used metrics, these metrics are TPR, TNR, FPR, and FNR. TPR stands for true positive ratio and also called the sensitivity or recall and shows in this study the attack is truly considered as an attack [18]. TNR stands for true negative ratio also called specificity in this study shows the ratio of normal behavior is considered as normal. FPR stands for false positive ratio and in this study considered as the attacks considered as normal behavior. FNR stands for the false negative ratio in this study shows the normal behavior is considered as an attack. Based on the above the formulas are [18] [19]:

$$Sensitivity = \frac{TP}{TP + FN} \quad (1)$$

$$Specificity = \frac{TN}{TN + FP} \quad (2)$$

$$FPR = \frac{FP}{FP + TN} \quad (3)$$

$$FNR = \frac{FP}{FP + TP} \quad (4)$$

where:

TP = number of attack label is predicted as attack;

TN = number of normal label is predicted as normal;

FP = number of normal label is predicted as attack;

FN = number of attack label is predicted as normal.

Table 1 shows the confusion matrix of the model. In addition, the accuracy, area under

DOI: 10.12948/issn14531305/22.4.2018.08

ROC curve, area under the precision-recall curve, test error, F1-Score, training time and testing time according to Pipeline approach also calculated in order to have stronger evaluations.

Table 1. Confusion Matrix of Model

Classes of the traffic packets		Predicted classes	
		attack	normal
Actual classes	attack	0.647	0.352
	normal	0.028	0.971

The accuracy is the most important metric in the evaluation that says the correctness ratio of predictions. The Area under ROC curve (AUC) measures the performance or accuracy of the test, in this case, differentiate the attacks from normal behaviors [20] [21]. The Area under precision-recall curve shows the success of the prediction with a concentration on the attacks [22]. F1-score shows the weighted average of the recall and precision and also comparing those, to each other [23]. The formulas of the above-mentioned metrics are [18]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$AUC = \int_0^1 \frac{TP}{P} d\left(\frac{FP}{N}\right) \quad (6)$$

$$AUPRC = \int_0^1 \frac{TP}{TP + FP} d\left(\frac{TP}{P}\right) \quad (7)$$

$$F(\beta) = (1 + \beta^2).\frac{(precision.recall)}{(\beta^2.precision + recall)} \quad (8)$$

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

where:
AUC=area under ROC curve;
P=sum of true positive and false negative;
N=sum of true negative and false positive;
AUPRC=area under precision-recall curve;
β=weight of precision;
Recall=true positive ration;
TP, TN, FP, and FN are same as previous explanations.
Table 2 shows the results of the aforementioned metrics for Random Forest Classifier algorithm.

Table 2 Experimental Results

Algorithm	Accuracy (%)	AUC (%)	AUPRC (%)	F1 (%)	Precision (%)
Random Forest	78.69	81	91	78.49	96.73

Listing 1 shows the sample code to run StringIndexer, VectorAssembler, and RandomForest Classifier, at the end put them all in the Pipeline. The StringIndexer algorithm is indexing string column's values in order to include them in our features as input data for creating Random Forest model, VectorAssembler assembling the all column's values in one vector and create a feature column. Random Forest Classifier needs the feature column that will be what we created on the VectorAssembler and also the label column of the dataset, in this study, the labels are normal and attack behavior, the number of trees is 100 and Pipeline which create a workflow for our algorithms.

DOI: 10.12948/issn14531305/22.4.2018.08

Listing 1 Sample of Code

```
//StringIndexer Algorithm
StringIndexer(inputCol=k, outputCol='outputColumn_pip', handleInvalid='error')

//VectorAssembler Transformer
VectorAssembler(inputCol=['duration',..], outputCol='features')

//IndexToString Algorithm
IndexToString(inputCol = 'prediction', outputCol='originalLabel', labels=pre-
dictIndexer.labels)

//RandomForest Classifier Algorithm
RandomForestClassifier(featuresCol='features', labelCol='attack_normal_index',
numTrees=100

//ML Piplines
pipeline = Pipeline(stages= 'StringIndexer', 'VectorAssembler', 'RandomForest',
'IndextToString')
model = pipeline.fit(train data)
predictions = model.transform(test data)
```

It can be seen from Listing 1 that 'k' in StringIndexer algorithm is the columns of strings that we want to index them in order to enter them in for training and also testing calculations, the handleInvalid set to error if the variable of training dataset does not exist in the testing dataset will raise an error. The indextostring algorithm returns the original labels of indexed results based on what the StringIndexer conducted for indexing the string values of the label column in NSL-KDD. Random forest based on the feature column and the indexed label column training dataset and after that make predictions with 100 number of trees. The Pipeline will create the workflow of the algorithm and the implementation done based on the ordering of passed variables that it means first the StringIndexer, second VectorAssembler, third RandomForest, fourth IndexToString and then the model training on the training dataset after that make a prediction on the testing dataset. Kato and Klyuev [24] proposed the anomaly detection system with Apache Spark and Hadoop and by use of Hive table and unsupervised learning algorithm like K-means and also GMM algorithm, this system capable of managing and detecting an enormous dataset about 90 GB quickly with low rate of false alarm and high value about 86% of accuracy. Gupta and Kulariya [25] proposed a framework for intrusion detection system based on Apache Spark, they used feature selections as correlation based and chi-squared with different algorithms such as Random Forest, Logistic Regression and other algorithms and evaluated the performance of each algorithm on NSL-KDD and KDD'99. In Table 3 and Table 4 we compared their Random Forest results on NSL-KDD with our results. Table 3 shows the accuracy of their framework is better also the sensitivity or true positive ratio is also higher than our proposed model but the specificity of the proposed model is higher than their model.

Table 3 Proposed model vs. Correlation based Results

Methods	Accuracy (%)	Sensitivity (%)	Specificity (%)
Our Results	78.69	64.76	97.11
[25]	82.35	72.72	95.07

Table 4 shows the proposed model's training time is more than the [25] proposed framework, the reasons are: we used the ML Pipeline so the pipeline is included by 4 algorithms and 100 trees for Random Forest Classifier

DOI: 10.12948/issn14531305/22.4.2018.08

but for their model, they did not mention how many trees are used and also the training and prediction time just including the Random Forest algorithm stage.

Table 4 Proposed Model Training and Testing Time

	Training Time	Prediction Time
Our Results	56.1	0.59
[25]	6.15	1.38

The prediction time of our model is less than their model and the reasons are: Runtime checking and topological order (DAG Pipelines) [26]. The main differences of our study with [25] are:

- We created our model with ML Pipeline.
- We did not use the feature selection on NSL-KDD because this model is improved and the records are selected logically in both: testing and training dataset and also the Random Forest has the ability to reduce overfitting and capture non-linearities.
- Our study focus is on binary classification and we used Random Forest binary classification with 100 trees for creating and testing mode but they did not mention what are their details.
- Also for strong evaluation, we evaluated our algorithm with more metrics such as AUC, F1, and AUPRC.
- Also in the reminder of this study, we are looking for the answer of this question how many trees for Random Forest? Based on cost, memory consuming and accuracy.

4.3 How many trees for Random Forest?

Random Forest Classifier algorithm takes some parameters, but the number of trees has important role for creating a model and affect the accuracy of the model, at the other hand the accuracy for intrusion detection is critical especially for important places so in this study we trying to find the number of trees that except the accuracy also considering the processing time and memory consuming. In order to find the answer, Oshiro and et al. [17] suggested that the range between 64 and 128 is ideal in order to get a good balance for time and memory consuming. Their experiments are done on 29 datasets and they concluded from 128 onwards there are no main differences even with a great number like 2048 or 4096. They also found the AUC has the inversely proportional relation with the doubled number of Random Forest trees.

Based on these experiments and their results we implement Pipelined Random Forest Classifier with a different number of trees in order to find the best accuracy, cost of processing time and memory consuming.

Table 5 Empirical Results of Different Number of Trees

Trees	128	64	32	16	8
Accuracy (%)	77.90	78.69	81.75	78.54	79.36
Time (s)	58.37	37.41	26.05	22.86	21.07

As can be seen from Table 5 when the number of trees being halved the accuracy is increased but at 16 the accuracy is decreased suddenly then at 8 trees is increased, it could be considered that the number of trees between 32 and 16 could be the best values, also the experiments are done on other numbers of trees as 50, 35, 42, 21, 30, 24, 40, 48, and 72. respectively the results of accuracy are 78.13%, 78.97%, 79.05%, 79.24%, 79.75%, 80.67%,

DOI: 10.12948/issn14531305/22.4.2018.08

78.19%, 79.73%, and 78.38%. So as can be seen if we want to have both criteria the time and the accuracy for choosing the best value, 32 number of trees is the result but if the cost of time and memory consuming are more important the 8 trees could be a best one. The worst value is 128 especially in the time and also with this greater number of trees, the accuracy is the worst one so if we choose the number of trees without logic just by increasing the number of trees, it could have critical damages on our anomaly detection model.

5. Conclusion

In this article, we proposed an intelligent model or framework based on the Random Forest Classifier for fast and precise anomaly-based intrusion detection system via Apache Spark. We used NSL-KDD train and test dataset from the University of New Brunswick respectively for training and testing the model. The model created with the ML Pipeline that decreases the prediction time even with more algorithms, based on the results and also it allows us to implement multiple algorithms just in the single step. The results show the high accuracy, low false alarm, low false negative and high precision which is promising for applying in anomaly detection systems. The second conclusion is that the number of trees in Random Forest Classifier has a direct impact on the performance and the accuracy of the Random Forest based model which in this study the 32 trees could be the best number in order to have more accuracy, also low cost and memory consuming for applying in anomaly detection systems.

Acknowledgment

This work is based on my dissertation thesis from Bucharest University Economic Studies, IT&C Security Master program, which has not been traditionally published.

References

[1] D. Elson, "Intrusion Detection, Theory and Practice," Symantec, 27 Mar 2000. [Online]. Available: https://www.symantec.com/connect/articles/intrusion-detection-theory-and-practice.

[2] T. Bradley, "Introduction to Intrusion Detection Systems (IDS)," LifeWire, 12 February 2018. [Online]. Available: https://www.lifewire.com/introduction-to-intrusion-detection-systems-ids-2486799.

[3] "What is IDS and IPS?," Juniper Networks, [Online]. Available: https://www.juniper.net/us/en/products-services/what-is/ids-ips/.

[4] N. Farnaaz and M. A. Jabbar, "Random Forest Modeling for Network Intrusion Detection System," Procedia Computer Science, vol. 89, p. 213 – 217, 2016.

[5] R. Kumari, Sheetanshu, M. K. Singh, R. Jha and N. Singh, "Anomaly Detection in Network Traffic using K-mean clustering," in Recent Advances in Information Technology (RAIT), Dhanbad, 2016.

[6] L. Rettig, M. Khayati, P. Cudr´e-Mauroux and M. Pi´orkowski, "Online Anomaly Detection over Big Data Streams," in IEEE International Conference on Big Data (Big Data), Santa Clara, 2015.

[7] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in IEEE Symposium on Security and Privacy, Berkeley/Oakland, 2010.

[8] W. LEE and S. J. STOLFO, "A Framework for Constructing Features and Models for Intrusion Detection," ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 4, p. 227–261, 2001.

[9] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303 - 336, 2013.

[10] P. Aggarwal and S. K. Sharma, "Analysis of KDD Dataset Attributes - Class wise For Intrusion Detection," Procedia Computer Science, vol. 57, p. 842 – 851, 2015.

[11] "KDD Cup 1999 Data," [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[12] "Random forest classifier," Apache Spark, [Online]. Available: https://spark.apache.org/docs/latest/ml-

DOI: 10.12948/issn14531305/22.4.2018.08

classification-regression.html#random-forest-classifier.
[13] T. K. Ho, "Random Decision Forest," in Proceedings of the Third International Conference on Document Analysis and Recognition, Montreal, Que., 1995.
[14] "NSL-KDD dataset," Canadian Institute for Cybersecurity (UNB), [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html.
[15] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, 2009.
[16] "Apache Spark," [Online]. Available: https://spark.apache.org/.
[17] T. M. Oshiro, P. S. Perez and J. A. Baranauskas, "How Many Trees in a Random Forest?," in Machine Learning and Data Mining in Pattern Recognition, New York, Springer International Publishing, 2017, pp. 154-168.
[18] "Evaluation Metrics - RDD-based API," [Online].Available: https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html.
[19] P. Pore, "How to evaluate a binary classifier," KDnuggets, April 2017. [Online]. Available: https://www.kdnuggets.com/2017/04/must-know-evaluate-binary-classifier.html.
[20] "ROC curve analysis," MEDCALC®, [Online]. Available: https://www.medcalc.org/manual/roc-curves.php.
[21] K. Markham, "ROC curves and Area Under the Curve explained," data school, 19 November 2014. [Online]. Available: http://www.dataschool.io/roc-curves-and-auc-explained/.
[22]"Precision-Recall," Sickit-learn, [Online]. Available: http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.
[23] T. G. Tape, "The Area Under ROC Curve," University of Nebraska Medical Center, [Online]. Available: http://gim.unmc.edu/dxtests/roc3.htm.
[24] K. Kato and V. Klyuev, "Development of a network intrusion detection system using Apache Hadoop and Spark," in IEEE Conference on Dependable and Secure Computing, Taipei, 2017.
[25] G. P. Gupta and M. Kulariya, "A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark," Procedia Computer Science, vol. 93, pp. 824-831, 2016.
[26] "ML Pipelines," Apache Spark, [Online]. Available: https://spark.apache.org/docs/latest/ml-pipeline.html.

Hesamaldin HAJIALIAN has graduated the Faculty of Economic Cybernetics, Statistics and Informatics of the Bucharest University of Economic Studies in 2018. He holds a Master diploma in Information Security and his work focuses on Machine Learning in Cybersecurity especially Intrusion Detection System (IDS) and Next-Generation Firewalls (NGFW).

Cristian TOMA has graduated from the Faculty of Cybernetics, Statistics and Economic Informatics, Economic Informatics specialization, within Bucharest University of Economic Studies in 2003. He has graduated from the BRIE master program in 2005 and PhD stage in 2008. In present, he is associate professor at Economic Informatics and Cybernetics Department and he is member in research structures such as ECO-INFOSOC. Since the beginning - 2005 - he is scientific secretary of IT&C Security Master Program from Bucharest University of Economic Studies and since 2006, he is in the editorial board of the SECITC – The Inter-

DOI: 10.12948/issn14531305/22.4.2018.08

national Conference on Security for Information Technology and Communications and JMEDS – Journal of Mobile, Embedded and Distributed Systems. His research areas are in: distributed and parallel computing, mobile applications, smart card programming, e-business and e-payment systems, network security, computer anti-viruses and viruses, secure web technologies and computational cryptography. He is teaching in Department of Economic Informatics and Cybernetics, and IT&C Security Master program. He has published 3 books and over 50 papers in indexed reviews and conferences proceedings.

DOI: 10.12948/issn14531305/22.4.2018.08