

An Efficient Real Time Intrusion Detection System for Big Data Environment

Faten Louati¹^a, Farah Barika Ktata²^b and Ikram Amous³^c

¹MIRACL Laboratory, FSEGS, Sfax University, Sfax, Tunisia

²MIRACL Laboratory, ISSATSo, Sousse University, Sousse, Tunisia

³MIRACL Laboratory, Enet'com, Sfax University, Sfax, Tunisia

Keywords: Intrusion Detection System, Big Data, Spark Streaming, Real Time Detection, Machine Learning.

Abstract: Nowadays, Security is among the most difficult issues in networks over the world. The problem becomes more challenging with the emergence of big data. Intrusion detection systems (IDSs) are among the most efficient solutions. However, traditional IDSs could not deal with big data challenges and are not able to detect attacks in real time. In this paper, a real time data preprocessing and attack detection are performed. Experiments on the well-known benchmark NSL KDD dataset show good results either in terms of accuracy rate or time of both training and testing and prove that our model outperforms other state-of-the-art solutions.

1 INTRODUCTION

All the data today could be considered as Big Data because of the rapid increase of the use of cloud/edge computing and 5G technologies which are utilized in all aspects of life such as economics, politics, culture, health-care to name a few. This rapid development brings along to big challenges namely, security and safety. The complexity of Big data makes the task of processing and handling data very hard. Hence, data are more vulnerable to different types of attacks. Since its invention by Anderson in 1980 (Anderson, 1980), intrusion detection systems (IDS) have been in continuous development and have been widely investigated by researchers as being among the most efficient solutions for networking security.


An IDS is a kind of software that monitors, analyzes networking traffics and sends an alert automatically once a malicious activity is detected (Louati and Ktata, 2020). There are two main techniques for IDSs: signature-based technique and anomaly-based technique. Signature-based IDS is based on a database of signatures of attacks which is used to decide if a given pattern is an attack or not. Although this approach achieves high accuracy and low false alarm rate, it is still unable to detect unseen attacks.


Thus, this technique is not suitable for big data context since there are new kinds of attacks appearing every day. On the other hand, anomaly-based approach tackles this limit and achieves good detection rate for known as well as unknown attacks, but the main drawback is that it causes a high false alarm rate i.e, it may trigger an alert for a benign pattern. For this reason, we used an enhanced anomaly-based intrusion detection by investigating Machine Learning (ML) algorithms, since they perform high level of accuracy and low value of false alarm rate in classification problem.


Because most of the existing IDSs are still unable to deal with the huge size of data in real time, in this paper we proposed a new solution that performs a real time intrusion detection system for big data environment. We address the challenges of big data such as velocity and volume. We achieved a real time data preprocessing and data classification.

For this purpose, we created two clusters; the role of the first is preparing and preprocessing the incoming streams of data in real time and in parallel way, then sending them to the second cluster to be classified in real time and in parallel way too. At this stage, we used the benchmark NSL KDD dataset to simulate network traffics. Experimental results show that our work outperforms other state-of-the-art solutions in term of accuracy as well as time of both preprocessing and detection,

The remaining part of the paper is organized in the

^a <https://orcid.org/0000-0002-8582-6092>

^b <https://orcid.org/0000-0001-5706-4548>

^c <https://orcid.org/0000-0002-5893-9833>

following way:

In section 2 we present same state-of-the-art solutions in big data context. Section 3 introduces our solution. Experimental results are described in section 4. Finally, section 5 concludes the paper and presents our future works.

2 RELATED WORKS

Only few papers focus on Intrusion detection systems in big data frameworks (Hassan et al., 2020). For instance (Terzi et al., 2017) used clustering algorithm to detects network anomaly from Netflow. Experiments performed on CTU-i3 dataset achieve 96% of accuracy but high false alarm rate.

(Hassan et al., 2020) exploited conventional neural network and weight dropped long short-term memory (WDLSTM) network to build the IDS. The work was tested on the UNSW-NB15 dataset and achieves an accuracy=97.17%.

(Mohamed et al., 2022) proposed an intrusion detection framework using Apache Spark for IoT. Three Spark's MLlib was tested in BoT-IoT dataset based on F1-measure, namely Random Forest, Decision Tree and Naive Bayes. Experiments show that Decision tree achieves the highest value of F1-measure in big data context i.e when using the whole dataset with 97.9% for binary classification and 79% for multi-classification.

(Liu et al., 2020) proposed a network intrusion detection system based on Deep Random Forest. The model was deployed in Spark environment. Four datasets were used in experimentation namely, NSL KDD, UNSW-NB15, CICIDS2017 and CICIDS2018 and good results were achieved.

(Al-Rawi, 2019) used two algorithms from Spark's MLlib; The first is Multi-Layer Perceptron which classifies the data into normal or attacks. Data classified as attacks are fitted to the second classifier, which is the Random Forest, for further verification. The proposed IDS performs an overall accuracy of 99.12% on UNSW-NB15 dataset.

Also (Kurt and Becerikli, 2018) performed a comparison between different machine learning algorithms provided by Spark's MLlib namely, Logistic Regression, Support Vector Machine, Naive Bayes and Random Forest. Experiments on KDD99 dataset show that Logistic Regression achieves the best accuracy with 99.1% . However Naive Bayes achieves the lowest training and prediction time.

(Vimalkumar and Radhika, 2017) presented an intrusion detection framework for smart grids using Apache spark and various machine learning tech-

niques namely, Deep Neural Networks, Support Vector Machines, Random Forest, Decision Trees and Naive Bayes. Also feature selection and dimensionality reduction algorithms are exploited. Experimentation are done on the synchrophasor dataset and the results are compared using useful metrics i.e accuracy, recall, false rate, specificity, and prediction time. Best results were achieved by Naive Bayes classifier with accuracy= 79.21%.

(Ouhssini et al., 2021) proposed a distributed IDS for cloud systems based on big data tools and machine learning algorithms. The system is composed of four components, namely network data collector, a streamer based on Kafka, preprocessing/data cleaning and data normalizing/feature selection using k-means algorithm. Different ML techniques are used for anomaly detection. After Comparison, authors chose decision Tree for their system because of its accuracy and detection time.

(Bagui et al., 2021) introduced an IDS based on Random Forest for a distributed big data environment using Apache Spark. The classifier is tested using the UNSW-NB15 dataset. Authors used information gain and principal components analysis (PCA) to address the issue of high dimensionality of the dataset. The highest accuracy was obtained by the binary classifier was 99.94%.

(Awan et al., 2021) applied machine learning approaches namely Random Forest (RF) and Multi-Layer Perceptron (MLP) through Spark ML library for the detection of Denial of Service (DoS) attacks. The model achieved a mean accuracy of 99.5%

(Jemili and Bouras, 2021) proposed an Intrusion Detection System based on big data fuzzy analytics. In fact, Fuzzy C-Means (FCM) is used to cluster and classify the training dataset. Experimentation are done with CTU-13 and UNSW-NB15 datasets and shows high performance in terms of accuracy (97.2%) and recall (96.4%).

Although works mentioned above are proposed for big data context, most of them didn't address same big data challenges such as velocity since data in big data environment are coming in very high speed, hence they should be treated at real time.

For this motivation, we introduce in this write-up a real time data preprocessing and detection within big data environment.

Table 1 summarizes and compares between those works and our solution based on experimental results especially the accuracy rate and the time of training and testing.

Table 1: Comparison of cited works for Big Data context.

Ref.	Approach	Dataset	Results	Training time	Testing time
(Jemili and Bouras, 2021)	big data fuzzy analytics	CTU-13 and UNSW-NB15	accuracy= 97.2%	-	-
(Awan et al., 2021)	Random Forest + Multi Layer Perceptron + Spark	The application layer DDoS dataset	accuracy= 99.5%	34.11 min	0.46 min
(Bagui et al., 2021)	Random Forest + Spark	UNSW-NB15 dataset	accuracy= 99.94%	-	-
(Ouhssini et al., 2021)	Decision Tree + spark + kafka	CIDDS-001 dataset	accuracy= 99.97%	-	-
(Vimalkumar and Radhika, 2017)	ML algorithms + spark	synchronphasor dataset	accuracy= 79.21%	-	18.23 sec for Random Forest
(Hassan et al., 2020)	Conventional Neural Network + Weight Dropped Long Short-Term Memory	UNSW-NB15	accuracy =97.17%	-	-
(Terzi et al., 2017)	Clustering algorithm	CTU-i3	accuracy= 96%	-	-
(Mohamed et al., 2022)	(Random Forest/ Decision Tree/ Naive Bayes) + Spark	BoT-IoT	f1 mesure= 97.9% for binary classification and 79% for multi-classification	-	-
(Liu et al., 2020)	Deep Random Forest + Spark	NSL KDD/ UNSW-NB15/ CI-CIDS2017/ CICIDS2018	For NSL KDD: Accuracy= 99.1%	-	16.1 sec for NSL KDD
(Al-Rawi, 2019)	Multi Layer Perceptron + Random Forest + Spark	UNSW-NB15	accuracy= 99.12%	-	-
(Kurt and Becerikli, 2018)	(Logistic Regression/ Support Vector Machine/ Naive Bayes/ Random Forest) + Spark	KDD	accuracy= 99.1%	4.041 hours	0.089 hours
Our solution	real time Streaming data preprocessing + Real time streaming data intrusion detection using spark	NSL KDD	accuracy=0.99%	32.043 sec	5.76 sec

3 THE PROPOSED SOLUTION

The main idea is to provide a solution that meets the challenge of Big Data velocity while maintaining a

high detection rate. Most, if not all, state-of-the-art solutions prepare the data first and then perform the detection/classification task. This approach takes twice, i.e. preparation time plus detection time. For

this motivation, our solution consists of reducing time by performing both preparation and detection tasks in parallel and in real time. This means that the data arriving at the system in a continuous stream is prepared and classified at the same time and in real time. As shown in Fig.1, our model is composed of two main components, the first one is responsible of data preparation and preprocessing in real time, the second one is responsible of intrusion detection in real time too. In this section, we give a brief background of the used concepts and we explain the solution with details .

3.1 Background

1. **Big Data:** The term big data dates back to 2005, it is typically defined by three words: volume, velocity and variety (Louati. et al., 2022). However, some researchers extends those known 3Vs of big data to 6Vs, namely:

- Volume: Size of the data
- Variety: Diversity of the data
- Velocity: Speed of the data
- Veracity: Uncertainty of the data
- Value : Usefulness of the data
- Variability: the way in with the data are used and formatted

2. **Apache Spark:** We used Apache spark (Spark, 2014) which is widely used framework for Big Data analysis, processing and parallel Machine Learning. Spark uses in-memory computation therefore, it executes all programs up to 100 xs faster in memory, (and 10 xs faster on disk) than Apache Hadoop.

Apache Spark system provides a high-level APIs accessible in several programming languages such as Scala, Java, Python and R and is composed with Spark core and higher level libraries such as Spark SQL which deals with SQL and structured data, Spark MLlib which contains a large set of ML and data mining algorithms, GraphX which helps with graph processing, and Spark Streaming for real-time stream processing . Spark has the ability to process a huge amounts of data in real time using big data analytic tools and streaming engine which leads to many benefits that can be exploited in intrusion detection field.

We tested four Spark MLlib algorithms, namely Decision Tree, Random Forest, Logistic Regression and Naive Bayes.

3. **Decision Tree:** Decision Tree (DT) is a powerful supervised machine learning algorithm. It consists of dividing the dataset into subsets based on

an attribute value test. Each node in the tree represents a test on an attribute, each branch represents the result of the test, and each leaf node represents a class label. The main advantages of Decision tree is that it can handle high dimensional data and has high accuracy rate.

4. **Random Forest:** Random Forest (Khan et al., 2021) consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most vote becomes the final model's prediction (Yiu, 2019).
5. **Logistic Regression:** Logistic regression algorithm gives a relationship between a dependent and one or more independent variables. It is usually used to makes predictions for continuous/real variables also for categorical variables.
6. **Naive Bayes:** A Naive Bayes (NB) classifier is a probabilistic machine learning model used for the classification task and is based on Bayes' theorem.

3.2 The Solution's Architecture

We created two spark clusters using docker (Docker, 2013) each cluster is built on one or more docker containers. The use of docker brings many advantages such as providing isolated environment for the applications. Thus, they could be deployed anywhere i.e in the cloud or local machines. Besides, Docker helps in data processing and analysis by providing packaging and management of dependencies e.g. python's libraries.

The first cluster (cluster preprocessor) works as an agent responsible for preparing and preprocessing of the incoming data streams in real time to be suitable for being fitted in machine learning model. The preprocessed streams are stored on a shared docker volume, The second cluster (cluster detector) takes those streams one by one and performs classification in real time. We used the four Spark Mllib algorithms explained in section3.1 i.e, Random Forest, Naive Bayes, Decision Tree and Logistic Regression to train the cluster detector and create models. Then, spark streaming is used for testing.

A comparison of those models is performed in terms of training time, testing time and performance metrics.

The application is divided into two phases:

The first phase consists of an offline training. This means that we train the model with batches (not streams) of the training dataset. For this purpose, a spark application was created in cluster preprocessor

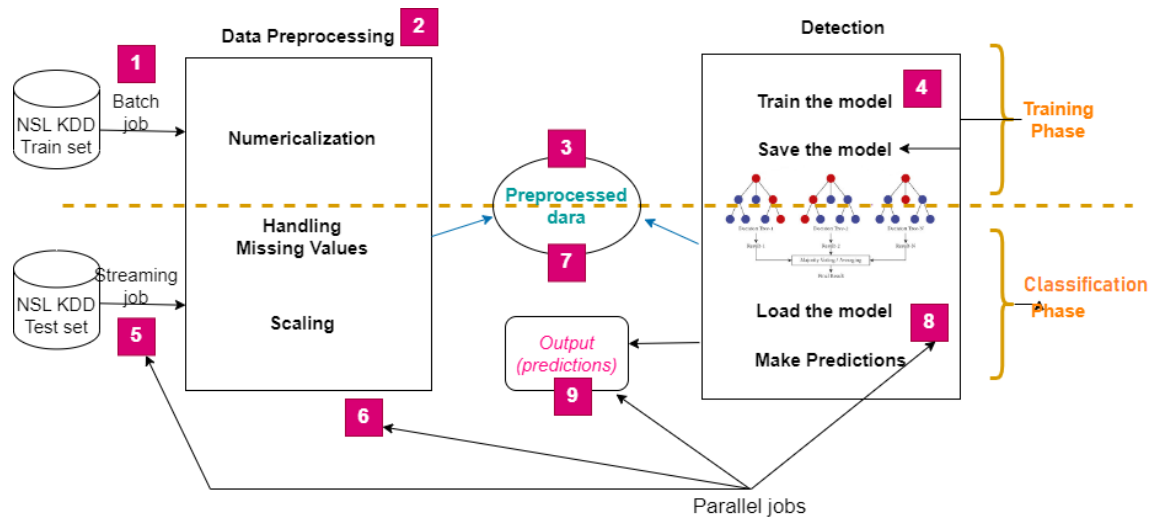


Figure 1: The proposed solution.

and ran as batch job. After being prepared, the new training set is saved in a docker volume. This volume is shared between the two clusters and could be accessed by both.

A second spark application was created in cluster detector and ran as batch job too. This cluster trains Random Forest classifier using the preprocessed training set. Once the training is completed, the cluster detector saves the model in the docker volume.

In the second phase, an online classification was performed. We used flow of data arriving in the form of streams. To simulate real time situation we divided the testing set into 30 parts where each part presents collected data flow.

In this phase, the cluster preprocessor performs data preprocessing and data cleaning to each stream of data as the same way as performed in the training phase and saves in the docker volume. At this time, the cluster detector is listening to the volume and checking if there are data arrived to be classified.

At this stage, the application uses NSL KDD dataset as input and reads the data from a local file. However, in the coming work we aim at improving the solution to be more real by reading the data from ingestion tools like Kafka.

The work-flow of the application is as follows:

1. Offline training:

- 1- Create and run spark application in cluster preprocessor as a batch job
- 2- Create and run spark application in cluster detector as spark job
- 3- Cluster preprocessor prepares and pre-processes the training dataset

4- Cluster preprocessor saves the preprocessed dataset in shared docker volume

5- Training cluster detector's model with pre-processed train set

2. Online classification:

6- Cluster preprocessor reads streaming data and preprocess them stream by stream in real time

7- Each preprocessed stream is saved in the shared docker volume

8- Cluster detector accesses to the saved streams and performs detection in real time

9- Output the results

Fig.1 depicts clearly the explained workflow.

To build the system we used docker compose which helps to create and run multiple related docker containers. As shown in Fig.2, our docker compose contains six containers:

- One container for the preprocessor cluster created from bitnami (Bitnami, 2022) spark image. It is a single node cluster composed of only the master i.e. no workers because data preparation does not need that. The container runs on the port 8888.
- Three containers for the detector cluster created from bitnami spark image. The cluster is composed of master and two workers and runs on ports 8080/4040. 1 core and 1G of memory are assigned to each worker.

Those two containers used the same volume which is bind mount to local directory. to visualize the results, we transform it to database to facilitate creating queries. For this reason we created two other containers:

Table 2: Number of samples of each class in the training set (KDDTrain+).

Class	Count
normal	67343
DoS	45927
Probe	11656
R2L	995
U2R	52
Total	125973

Table 3: Number of samples of each class in the testing set (KDDTest+).

Class	Count
normal	9711
DoS	7458
R2L	2754
Probe	2421
U2R	200
Total	22544

- One container for PostgreSQL (Postgres, 2022) which is a powerful, open source object-relational database system. The container is created from postgres image and run on port 5432.
- One container created from Adminer (Adminer, 2022) image which is a full-featured database management tool written in PHP and is available for MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Firebird, SimpleDB, Elasticsearch and MongoDB. The container runs on port 8088. Figure.3 depicts the results of random forest classifier shown by Adminer

4 EXPERIMENTAL RESULTS

4.1 NSL KDD Dataset

We evaluate the performance of our solution using NSL KDD dataset (Tavallae et al., 2009) which is widely used in intrusion detection researches.

NSL KDD dataset is an improved version of KDD99 dataset, composed of 42 features. Each sample of the dataset is labeled as either normal or a specific kind of networking attacks. All the attacks could be categorized as one of the main four classes of attacks i.e Denial of Service (DoS), Remote to Local (R2L), Probe and User to Root (U2R) (Table 2, Table 3)

NSL KDD dataset is provided with two files KDDTest+.txt and KDDTrain+.txt.. The first file represents the test set, as shown in table 3 the dataset is composed of 22544 samples where 9711 samples

Table 4: Comparison of both training and testing time of used MLlib algorithms.

Algorithm	Training time (s)	Testing time (s)
Random Forest	32.0438	5.7666
Decision Tree	38.2766	3.4811
Naive Bayes	1.4716	3.5905
Logistic Regression	6.1271	3.3405

are labeled as normal, 7458 samples are DoS attacks, 2754 samples are R2L attacks, 2421 samples are Probe attacks and 200 samples are U2R attacks.

The second file represents the train set, as shown in table 2, the dataset is composed of 125973 samples where 67343 samples are labeled as normal, 45927 samples are DoS attacks, 11656 samples are R2L attacks, 995 samples are Probe attacks and 52 samples are U2R attacks.

Hence, the dataset is very large therefore, it well represents the context of big data.

4.2 Discussion

To evaluate the solution we refer to usable metrics namely, accuracy, precision, recall, F1-measure, true positive rate, false positive rate, log Loss and hamming loss. All those metrics are measured using MultiClassificationEvaluator function from pyspark.ml.evaluator. Table 5 compares the results of the four used classifiers. As shown in the table, Random Forest performs the best results. For time, as shown in Table4, naive bayes gives the best time in training and Logistic regression in testing. However, Decision Tree classifier achieves the worst training time. Random Forest classifier achieves the worst testing time.

We choose Random Forest classifier in our solution because although it gives the worst testing time but it still efficient especially if we focus on the excellent results in terms of other metrics such as accuracy, precision recall etc. Furthermore, if we compare our results with other state-of-the art solutions, we can prove that our model outperforms other models in terms of accuracy and time of both training and testing as shown in Table1.

5 CONCLUSION

The characteristics of Big data generated in the networks such as high volume, high speed have made attack detection by traditional approaches very difficult. That is why the invention of new techniques able to perform big data analysis to make predictions and classification of large amount of data in real time, a persistent need. The purpose of this work is to provide a new solution that improves the efficiency and

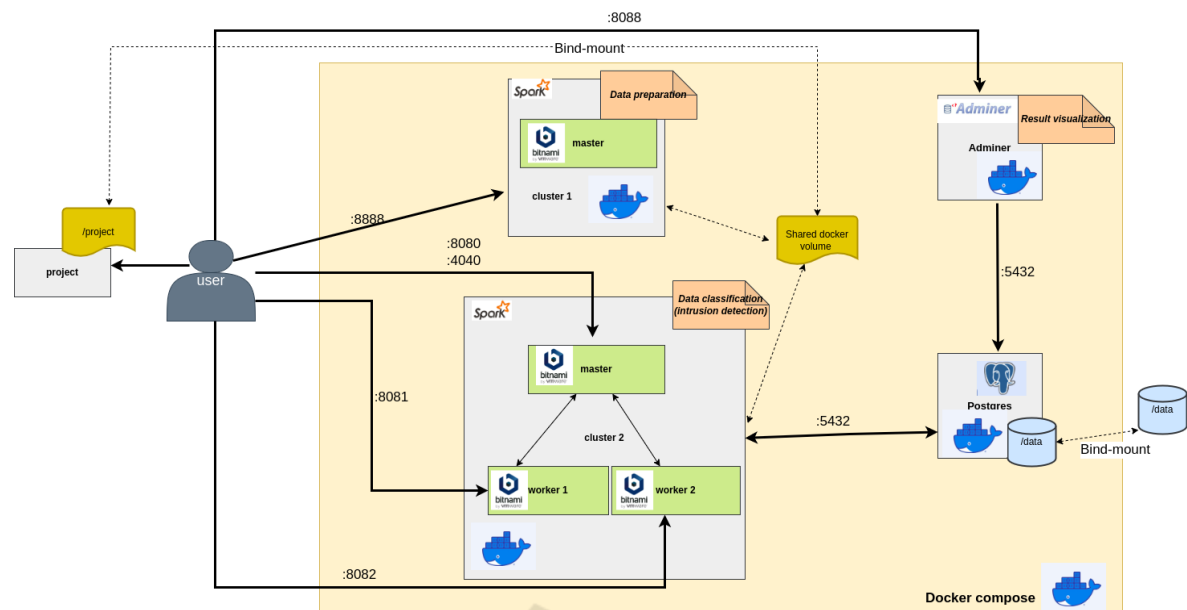


Figure 2: Docker containers architecture.

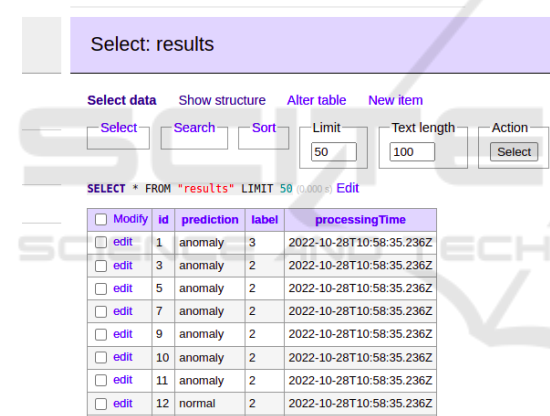


Figure 3: Visualization of classification's results with Adminer.

the rapidity of intrusion detection in the context of big data by performing a real time preprocessing and classification of incoming streams of data.

In summary, the main contributions of this work are:

- Building an intrusion detection system capable of dealing with big data streams in near-real time.
- Addressing the challenges of big data environment namely, velocity and volume by reducing the detection time.
- Providing a novel approach by executing preprocessing and classification at the same time as parallel jobs not in sequential manner as usually done in previous works. This novel approach improves

well the detection time as shown in Table.4.

- Dealing with big data in a secure way by using the docker technology.
- Taking advantages from the well-known ML algorithms in the detection task and providing comparison between them

Experimental results show that our solution performs the state-of-the art solution in term of speed (5.76 sec) and accuracy (0.99%).

In the future, we plan to use real network traffic instead of dataset also we aim at running other efficient algorithms within Spark that do not exist in Spark's MLlib such as algorithms that uses neural networks.

REFERENCES

- Adminer (2022). https://hub.docker.com/_/adminer.
- Al-Rawi, A. A. (2019). Intrusion detection system using apache spark analytic system.
- Anderson, J. (1980). Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington.
- Awan, M. J., Farooq, U., Babar, H. M. A., Yasin, A., Nobanee, H., Hussain, M., Hakeem, O., and Zain, A. M. (2021). Real-time ddos attack detection system using big data approach. *Sustainability*, 13(19).
- Bagui, S., Jason, S., Russell, P., Bennett, T. A., and Subhash, B. (2021). Classifying unsw-nb15 network traffic in the big data framework using random forest in spark. *International Journal of Big Data Intelligence and Applications*, 2.

Table 5: Comparison of experimental results of MLlib's algorithms.

Metric	Random Forest	Decision Tree	Naive Bayes	Logistic Regression
F1-measure	0.99	0.95	0.39	0.95
accuracy	0.99	0.95	0.43	0.95
weightedPrecision	0.99	0.95	0.71	0.95
weightedRecall	0.99	0.95	0.43	0.95
weightedTruePositiveRate	0.99	0.95	0.43	0.95
weightedFalsePositiveRate	0.01	0.03	0.18	0.04
weightedFMeasure	0.99	0.95	0.39	0.95
truePositiveRateByLabel	1.00	0.98	0.16	0.98
falsePositiveRateByLabel	0.01	0.06	0.00	0.07
precisionByLabel	0.99	0.95	1.00	0.93
recallByLabel	1.00	0.98	0.16	0.98
fMeasureByLabel	0.99	0.96	0.28	0.96
logLoss	0.04	0.17	19.29	0.16
hammingLoss	0.01	0.05	0.57	0.05

- Bitnami (2022). <https://hub.docker.com/r/bitnami/spark>.
- Docker (2013). <https://spark.docker.org/>.
- Hassan, M. M., Gumaei, A. H., Alsanad, A., Alrubaijan, M., and Fortino, G. (2020). A hybrid deep learning model for efficient intrusion detection in big data environment. *Inf. Sci.*, 513:386–396.
- Jemili, F. and Bouras, H. (2021). Intrusion detection based on big data fuzzy analytics. In Kakulapati, V., editor, *Open Data*, chapter 4. IntechOpen, Rijeka.
- Khan, M. Y., Qayoom, A., Nizami, M., Siddiqui, M. S., Wasi, S., and Syed, K.-U.-R. R. (2021). Automated prediction of good dictionary examples (gdex): A comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques. *Complexity*.
- Kurt, E. M. and Becerikli, Y. (2018). Network intrusion detection on apache spark with machine learning algorithms. In Pimenidis, E. and Jayne, C., editors, *Engineering Applications of Neural Networks*, pages 130–141, Cham. Springer International Publishing.
- Liu, Z., Su, N., Qin, Y., Lu, J., and Li, X. (2020). A deep random forest model on spark for network intrusion detection. *Mobile Information Systems*, 2020:1–16.
- Louati, F. and Ktata, F. (2020). A deep learning-based multi-agent system for intrusion detection. *SN Applied Sciences*, 2.
- Louati, F., Ktata, F., and Ben Amor, I. (2022). A distributed intelligent intrusion detection system based on parallel machine learning and big data analysis. In *Proceedings of the 11th International Conference on Sensor Networks - SENSORNETS*, pages 152–157. INSTICC, SciTePress.
- Mohamed, A., Mouhammd, A., Mohammad, A., and Muhannad, M. (2022). An accurate iot intrusion detection framework using apache spark.
- Ouhssini, M., Afdel, K., Idhammad, M., and Agherrabi, E. (2021). Distributed intrusion detection system in the cloud environment based on apache kafka and apache spark. In *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, pages 1–6.
- Postgres (2022). https://hub.docker.com/_/postgres.
- Spark (2014). <https://spark.apache.org/>.
- Tavallae, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6.
- Terzi, D. S., Terzi, R., and Sagioglu, S. (2017). Big data analytics for network anomaly detection from netflow data. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 592–597.
- Vimalkumar, K. and Radhika, N. (2017). A big data framework for intrusion detection in smart grids using apache spark. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 198–204.
- Yiu, T. (2019). Understanding random forest. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.