# A distributed intrusion detection system based on Scikit-learn applied on NSL-KDD Dataset

Djediden Mohamed Seghire Othman[1], Reguieg Hicham[1], Mekkakia Maaza Zoulikha[1]

[1]Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, USTO-MB, BP 1505, El Mnaouer, 31000 Oran, Algeria

mohamed.djediden@univ-usto.dz, hicham.reguieg@univ-usto.dz, zoulikha.mekkakia@univ-usto.dz

Corresponding Author: Djediden Mohamed Seghire Othman, Phone: +213-659-181-190; E-mail mohamed.djediden@univ-usto.dz

## Abstract

An intrusion detection system (IDS) is one of the important research areas in the field of network security. IDS becomes an essential part of building a computer network to capture any type of attack at an early stage because IDS works against all intruder attacks. Nowadays the number of users of computer networks has exploded; the captured network traffic has become more varied and more voluminous. This is why the detection of intrusion using traditional tools and methods is a very difficult and complicated task. Most existing IDS are deployed on a single server and have encountered several problems since the volume of data to be analyzed is larger. In our previous work, we create a distributed IDS based on an optimized version of the Scikit-learn library named SK-Dist. The proposed system has been evaluated on the UNSW-NB15.After the good results obtained in terms of accuracy, scalability and fault tolerance. We aim in this paper to prove the effectiveness of our previous approach by applying it to the NSL-KDD dataset. The proposed system is a combination of the features selection methods (Chi2 and RFE) from the Scikit-learn library, the classifiers integrated into the optimized Scikit-learn library named "Sk-Dist" and the Apache Spark framework. The results of comparisons made with other existing work have shown that our approach is much better in terms of accuracy, reduction of features and above all fault tolerance. The main contribution is that our IDS overcome the major limitation of the Sk-learn library (non-distributed processing) to benefit from the multiple-choice of these algorithms in terms of classification and feature selection. The results obtained proved that our approach is efficient with the two sets of data most used in the evaluation of IDS (UNSW-NB15 and NSL-KDD), from which we can say that it is suitable and compatible with any network traffic.

**Keywords:** *Intrusion Detection, Machine Learning, Big Data, Distributed Computing, Scikit-learn, NSL-KDD.*

## 1. Introduction

Network security is very important in today's data communications environment. Hackers and intruders can create many successful attempts to crash networks and web services by the unauthorized intrusion. New threats and associated solutions to prevent these threats are emerging as the secure system evolves. One such solution is intrusion detection systems (IDS). IDS becomes an essential part of building a computer network to capture this type of attack at an early stage because IDS works against all intruder attacks. (Sung et al., 2005)

IDS uses classification techniques to decide every packet pass through the network, whether it is a normal packet or an attack. With the explosion in the number of computer network users (Internet in particular), the captured data became more varied and voluminous (Big Data), making the intrusion detection process using traditional methods more complicated. That is why Big Data techniques are used in IDS to develop more accurate and efficient intrusion detection frameworks. (Buczak et al., 2016)

In general, Big Data refers to sets of data that cannot be acquired, managed and processed by traditional relational methods and hardware tools within a tolerable period. One of the most widely used big data frameworks is Apache Spark, this versatile framework known for its speed covers a wide range of big data workloads such as iterative algorithms, batch applications, streaming and interactive queries. (Chen et al., 2014; "Spark Overview", n.d).

The data captured in the computer networks are generally of a high dimensionality, which makes the analysis process very long and complicated. In machine learning, Feature selection (FS) is a crucial method in the data preprocessing stage. Feature selection reduces the dimensionality of data and enhances the performance of the classification process.

Sickit-learn is a machine-learning library widely used in the creation of anomaly-based IDS. Based on python, this library provides many features such as classification, regression, clustering, model selection and preprocessing. The limitation of this library is that it does not support parallelization, which means that a Scikit-learn program cannot be executed and distributed on a cluster. ( Pedregosa et al.,2011)

Recent research has aimed to introduce Big Data analysis techniques into the creation of IDS. The main goal is to create distributed fault-tolerant IDS that can analyze a fairly large and varied dataset with better precision. In this article, we use Apache Spark as the data processing framework.

In our previous research (Djediden et al., 2020), we create a distributed IDS based on an optimized version of the Scikit-learn library named SK-Dist. The proposed system has been evaluated on the UNSW-NB15. After good results are obtained, we aim in this paper to generalize and validate our approach by applying it to one of the most used data sets in the field of intrusion detection, which is NSL-KDD. This approach consists of creating a distributed IDS which supports big data analysis and which ensures better detection accuracy while using the minimum number of features. The proposed system is a combination of the features selection methods (Chi2 and RFE) from the Scikit-learn library, the classifiers integrated into the optimized Scikit-learn library named "Sk-Dist" and the Apache Spark framework. The IDS created selects the best subset of features ensuring higher accuracy. The main objective is to overcome the major limitation of the Sk-learn library (non-distributed processing).

For this purpose, this paper is organized as follows. In section 2, we introduce some related works on the application of ML and FS for IDS. In section 3, after the description of the chosen dataset (NSL-KDD), we introduced the proposed approach. In addition, each step in this method is described. Section 4 presents the proposed approach results. Finally, we conclude our work and describe the future work in section 5.

## 2. Related works

Many research describes the use of the ML classifiers and FS methods for intrusion detection. Buczak and Guven (2016) present a detailed survey of the use of ML algorithms for IDS.

In this section, we present several works that have created and developed IDS by applying machine learning algorithms and feature selection methods in undistributed environments with traditional tools like MATLAB, WEKA, and Scikit-learn. The works cited used the NSL-KDD and UNSW-NB15 data set to test the effectiveness of their approaches.

Divekar et al. (2018) propose an approach that focuses on data preprocessing using the SMOTE oversampling (Yap et al., 2013) to make the data more balanced before starting the detection process with ML classifiers (Neural Network (NN), Support Vector Machine (SVM), DT, Random Forest (RF), Naïve Bayes(NB) and K-Means) and selection methods from the Scikit-learn library (“Scikit-learn”, n.d). The authors tested their approach with the KddCup99, NSL-KDD, and UNSW-NB15 datasets.

Moustafa and Slay (2015a) propose a new hybrid features selection method, based on the central points (CP) of attribute values and Association Rule Mining (ARM). The approach aims to reduce the processing time overall by selecting the most frequent values and choosing the highest ranked features by removing irrelevant or noisy features. For intrusion detection, expectation-maximization (EM) clustering, Logistic Regression (LR) and NB classifiers are developed using Visual Studio C# 2008. This approach is applied to UNSW NB15 and NSL-KDD datasets. Experimental results show that the proposed model can improve accuracy and its processing time is extremely short.

In (Mukherjee et al., 2012), the authors combine the FS algorithm (based on the vitality of each feature)) with ML classifier (NB) to produce the optimal subset of features that can be used to classify the instances of NSK-KDD datasets. The approach is developed on Weka and the experimentation results show that the proposed approach achieving high recognition rates.

Dhanabal et al. (2015) analyze the NSL-KDD data using different ML classifiers (DT J48, SVM and NB) and exploring correlated features to improve intrusion detection. A new subset of features is proposed and compared with the previous work in the KDD'99 dataset. The results are obtained under the Weka framework, the new subset shows better intrusion detection rates with the DT J48 classifier.

Table 1 summarizes the different characteristics of the studied works. All the research cited in table 1 uses tools and frameworks (environment) which cannot be run on a cluster and which does not support fault tolerance, scalability, high availability, distribution. In this table, we present several works that have created and developed IDS by applying machine learning algorithms and feature selection methods in undistributed environments with traditional tools like WEKA, and Scikit-learn. All the works cited used the NSLKDD data set to test the effectiveness of their approaches.

**Table 1. NSL-KDD Related work comparative**

| **Reference** | FS | **ML Algorithm** | **ML (Best result)** | **Tools** |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| **Divekar et al. (2018)** | Gini Impurity Index | NN, SVM, DT, RF, NB and K-Means. | RF | Scikit-learn |
| **Moustafa et al. (2015a)** | CP and ARM | EM clustering, LR and NB | LR | Visual studio C# 2008 |
| **Mukherjee et al. (2012)** | **features vitality** | **NB** | **NB** | **WEKA** |
| **Dhanabal et al. (2015)** | **Correlation based Feature** | **DT J48, SVM and Naïve Bayes** | **DT J48** | **WEKA** |

These approaches cannot handle large and varied data sets, so since these approaches are undistributed, as soon as the server where they are developed will be down, the intrusion detection system will be stopped and the networks will be vulnerable to attacks. To deal with all these limitations, we integrate a new package called Sk-Dist that supports distribution and parallelization over spark clusters. The approach will be detailed in the next section

## 3. Proposed approach

This section describes the proposed IDS as well as the techniques and data set used. Figure 1 illustrates the data flow of our approach that are: load dataset and exports it into Data Frame in Apache Spark, preprocessing, feature selection, train the model, parameter tuning and evaluate the model with the testing dataset. In what follows, we detail the approach steps.

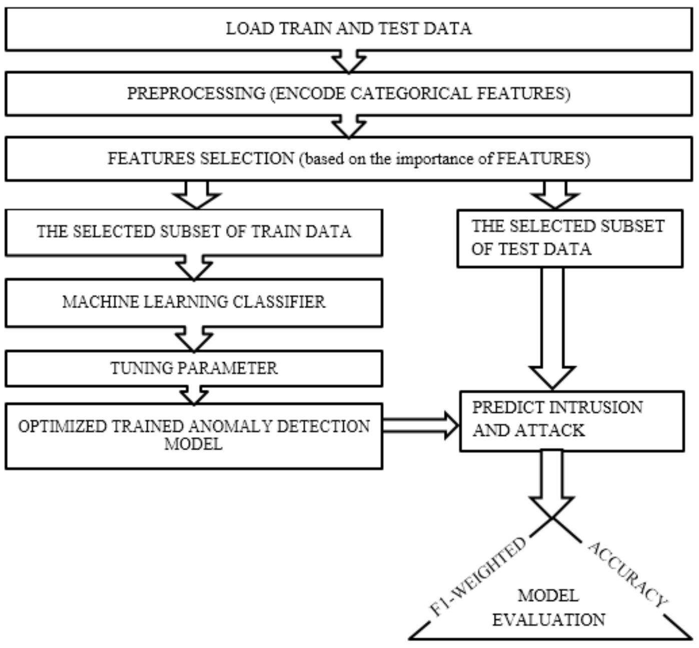

LOAD TRAIN AND TEST DATA
PREPROCESSING (ENCODE CATEGORICAL FEATURES)
FEATURES SELECTION (based on the importance of FEATURES)
THE SELECTED SUBSET OF TRAIN DATA
THE SELECTED SUBSET OF TEST DATA
MACHINE LEARNING CLASSIFIER
TUNING PARAMETER
OPTIMIZED TRAINED ANOMALY DETECTION MODEL
PREDICT INTRUSION AND ATTACK
F1-WEIGHTED
ACCURACY
MODEL EVALUATION

**Fig. 1. Proposed Approach data flow**

## 3.1 Description of NSL-KDD dataset

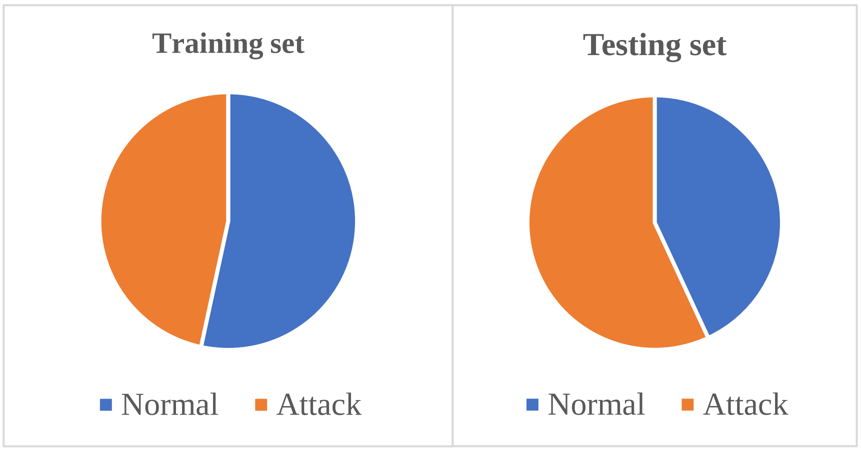
The NSL-KDD data set was brought into existence, as a revised, cleaned-up version of the KDD'99 from the University of New Brunswick. These data sets contain the records of the internet traffic seen by a simple intrusion detection network. The data set contains 43 features per record, with 41 of the features referring to the traffic input, and the last two are labels (whether it is a normal or attack) and Score (the severity of the traffic input itself). NSLKDD is comprised of four sub data sets: KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20Percent, although KDDTest-21 and KDDTrain+_20Percent are subsets of the KDDTrain+ and KDDTest+. (Tavallaee et al., 2009)

Table 2 and Figure 2 shows the distribution of normal and attacking instances in training and testing data sets.

**Table 2. NSL-KDD data set distribution**

| | Normal | Attack | Total |
|---|---|---|---|
| Training set | 13449 | 11743 | 25192 |
| Testing set | 9714 | 12830 | 22544 |

**Figure 2. NSL-KDD data set distribution**



### 3.2 Preprocessing

We start by loading the two NSLKDD datasets into Spark using the python library named pandas. Among all the features of NSL-KDD, there are three categorical features ('service', 'proto' and 'flag') that pose a problem for ML classifiers during training and prediction. To solve this problem we used the Label encoder function (from the Sklearn library) to convert these features to numeric features.

The label feature contains 39 attack sub-categories and normal traffic, since we aim in our approach to detect only attacks without citing their type (binary classification); we convert the label to a binary column where 0 is assigned to normal traffic and 1 for the attacks.

### 3.3 Feature Selection

The Sklearn library offers several methods for the selection of features, for our IDS we tested the two selectors known for their effectiveness (chi-square and RFE).

The Chi2 method computes chi-squared stats between each feature and the target. The calculation results are then used in the SelectKBest function to select only the features with the maximum chi-squared values. (sklearn.feature_selection.chi2, 2021)

RFE select features by recursively considering smaller sets. The estimator is trained on the initial set and the importance of the feature is calculated. Then, the least important features are deleted from the current set of features. This process is repeated until a specified number of features remains. (sklearn.feature_selection.RFE, 2021)

### 3.4 ML Classifier for intrusion detection (training the model)

Existing classifiers in the Sklearn library cannot be executed on a cluster (undistributed classifiers), this is why we integrated the Sk-Dist library in our proposed system.

Sk-dist is a python package built on top of Scikit-learn and can be defined as an optimized version of Scikit-learn, which supports distribution and parallelization over spark cluster. This allows distributed training

of two classifiers: random forest and extra tree without any constraint on the physical resources. ( GitHub, 2021)

We chose random forest as the classifier because all the works cited above have proven their effectiveness for binary classification.

### 3.5 Parameter Tuning

The random forest algorithm has several parameters; the choice of values for these parameters influences the accuracy of the model. To find the best combination of parameters that gives us the best accuracy, we used the Hypopt package. This python package designed to optimize the parameters of ML algorithms is known by the use of a validation set and Hypopt makes the process of obtaining the best combination faster by supporting distribution in a cluster. (Hypopt, 2019)

### 3.6 Evaluation Metrics

In order to analyze and evaluate the system performance, two evaluation metrics are used: accuracy and f1-weighted. Values for each metric are calculated from the confusion matrix of predictions.

**Accuracy:** the proportion of correct predictions made by the model, it is defined as well:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

**F1-weighted:** the F1-Scores average over all dataset classes, each weighted by its Support. The standard F-measure is F1, which gives equal importance to recall and precision. .F1-Score is the harmonic mean of recall and precision (Tharwat, 2018).

$$\mathrm{F1-Score} = 2 * \left(\frac{\mathrm{recall} \times \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}\right) \quad (2)$$

$$\mathrm{F1-Weighted} = \frac{\sum_{i=1}^{k} \mathrm{Support}_i . \mathrm{F1}_i}{\mathrm{Total}} \quad (3)$$

Where F1i is the F1-Score predicted for the ith target class.
These two metrics will be used to evaluate the proposed approach performance and to compare the results obtained with the related work.

## 4. Results and discussion

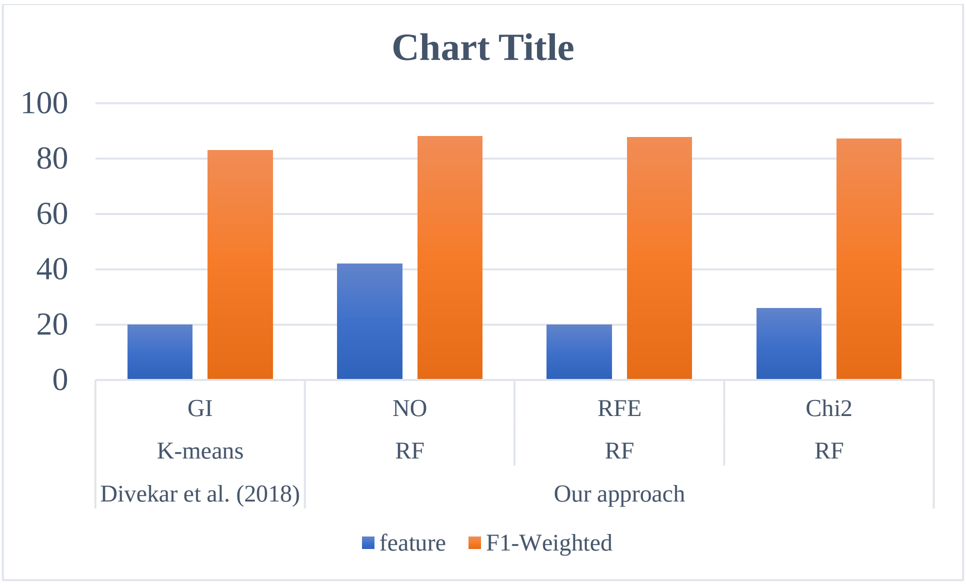
This section presents the proposed approach results. In addition, the comparison made with the other existing works will be discussed to prove that our proposed system ensures better accuracy with a reduced number of features.

Comparison with Divekar et al. (2018) in Table 3/ Figure 3 using F1-weighted metric prove that our proposed approach is better. The best result obtained is when using RFE as a feature selection method and RF as a classification algorithm. With this combination, we have an f1-weighted score equal to 87,71 with a

reduced subset of only 20 features. Which is much better than the result obtained by Divekar et al. (2018) (+4,18 in f1-weighted).

**Table 3: Comparison between f1-weighted of our approach and Divekar (2018)**

| Article | Machine learning classifier | FS Method | Number of feature | Tools / Framework | F1-Weighted |
|---|---|---|---|---|---|
| Divekar et al. (2018) | K-means | GI | 20 | Scikit-learn | 83 |
| Our approach | RF | NO | All 42 | Sk-dist in Spark | **88,02** |
| | RF | RFE | 20 | Sk-dist in Spark | **<u>87,71</u>** |
| | RF | Chi2 | 26 | Sk-dist in Spark | 87,18 |



To ensure the efficiency of our IDS, we carried out other comparisons with other related work using accuracy. Table 4 and Figure 4 illustrates the results of this comparison. Without the use of a selection method, and with our proposed IDS we achieved an accuracy of 87.97, with the use of the RFE method as a selector we reached an accuracy of 87.66 with only 20 features, which is much better compared to the reference Moustafa (2015), which found at most an accuracy of 82.1.

**Table 4: Accuracy of proposed approach vs (Moustafa, 2015)**

| **Reference** | **FS Method** | **ML classifier** | **Tools / Framework** | **Number of feature** | **Accuracy** |
|---|---|---|---|---|---|
| Proposed Approach | RF | NO | Sk-dist in Spark | 42 | 87,97 |
| Moustafa et al. (2015) | LR | CP and ARM | Visual studio C# 2008 | 11 | 82,1 |
| Proposed approach | RF | RFE | Sk-dist in Spark | 20 | 87,66 |

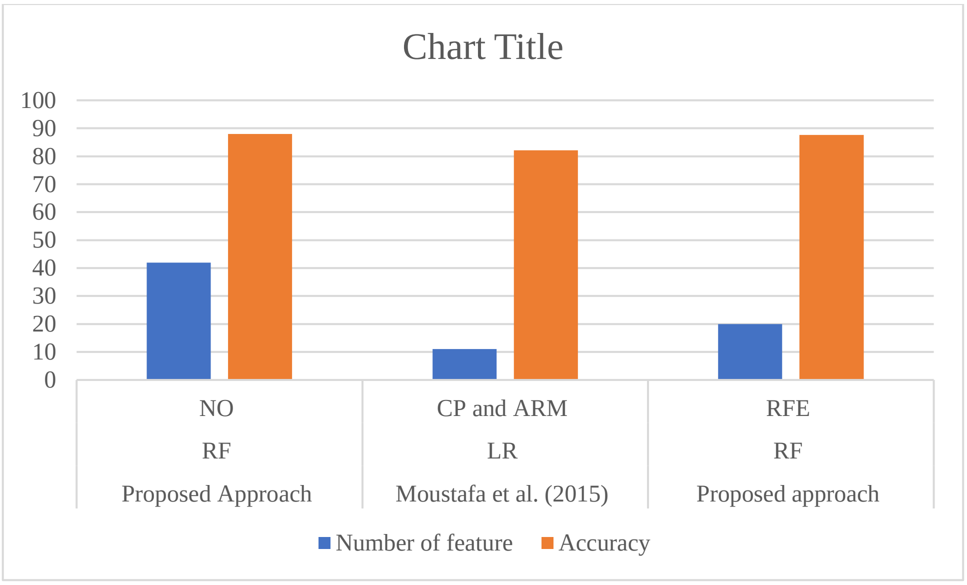


**Figure 4. Accuracy of proposed approach vs (Moustafa, 2015)**

The other essential point that differentiates our approach from all the works cited is that thanks to the use of the Sk-dist package, our approach supports distribution in a Spark cluster which guarantees high availability and fault tolerance of our IDS. The tests carried out in this point have shown that when two Spark “workers” are used and one of the two falls, our IDS works normally without any interruptions.

## 5. Conclusion

In this paper, we generalize and validate our approach by applying it to one of the most used data sets in the field of intrusion detection, which is NSL-KDD. The distributed intrusion detection system analyse massive data and ensure better accuracy while using the minimum number of features during the analysis. The proposed IDS overcome the major limitation of the Sk-learn library (non-distributed processing). From the comparison, we could see that our system is more efficient in terms of accuracy and speed and especially fault-tolerant. The results obtained proved that our approach is efficient with the two sets of data most used in the evaluation of IDS (UNSW-NB15 and NSL-KDD), from which we can say that it is suitable and compatible with any network traffic. As future work, we aim to test the machine-learning library integrated into Spark called Spark ML and compare the results with our system, also improve our IDS so that it detects the types of attack (multi-classification).

## References

1. Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176. doi:10.1109/comst.2015.2494502
2. Chen,M , Mao,S., and Liu,Y. “Big data: A survey,” Mobile Networks and Applications, pp. 171–209,vol. 19, Apr 2014.
3. Dhanabal,L. and Shantharajah,s., (2015) “A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms”, International Journal of Advanced Research in Computer and Communication Engineering, Vol 4, Issue 6, pp.
4. Divekar, A., Parekh, M., Savla, V., Mishra, R., and Shirole, M. (2018). Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives. 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS). doi:10.1109/cccs.2018.8586840
5. Djediden,MSO., Reguieg,H., Mekkakia Maaza,Z., "A Distributed Intrusion Detection System Based On Apache Spark and Scikit-learn Library" , International Conference on Research in Engineering and Fundamental Applied Sciences . (RFAS'2020), April 25-26, 2020, Barcelona-Spain
6. GitHub - Ibotta/sk-dist: Distributed scikit-learn meta-estimators in PySpark. (2021). Retrieved 5 August 2021, from https://github.com/Ibotta/sk-dist
7. hypopt. (2019). Retrieved 5 August 2021, from https://pypi.org/project/hypopt/
8. Moustafa, N., and Slay, J. (2015). A hybrid feature selection for network intrusion detection systems: Central points. ArXiv, abs/1707.05505
9. Pedregosa et al., "Scikit-learn: Machine Learning in Python", JMLR 12, pp. 2825-2830, 2011.
10. Saurabh Mukherjee, Neelam Sharma,Intrusion Detection using Naive Bayes Classifier with Feature Reduction, Procedia Technology,Volume 4,2012,Pages 119-128,ISSN 2212-0173,https://doi.org/10.1016/j.protcy.2012.05.017.
11. sklearn.feature_selection.chi2 — scikit-learn 0.24.2 documentation. (2021). Retrieved 5 August 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html
12. sklearn.feature_selection.RFE — scikit-learn 0.24.2 documentation. (2021). Retrieved 5 August 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
13. Spark Overview. (n.d.). Retrieved June 25, 2021, from https://spark.apache.org/docs/2.2.0/

14. Sung, A., Abraham, A., and Mukkamala, S. (2005). Cyber-Security Challenges. Enhancing Computer Security with Smart Technology, 125- 164. doi:10.1201/9781420031225.ch6
15. Tavallaee,M., Bagheri, E., Lu,W. and Ghorbani,A. , “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
16. Tharwat, A. (2018). Classification assessment methods. Applied Computing and Informatics. doi:10.1016/j.aci.2018.08.003