Hybrid Machine Learning-Based Approach for Anomaly Detection using Apache Spark

Hanane Chliah[1], Amal Battou[2], Maryem Ait el hadj[3], Adil Laoufi[4]
RF-SIC Laboratory-Faculty of Science, Ibn Zohr University, Agaadir, Morocco[1, 2]
Laboratory for Sustainable Innovation and Applied Research (L.I.D.R.A), Universiapolis, Agadir, Morocco[3]
Equipe Systèmes Intelligents et Communicants (SIC), Ibn Zohr University, Agaadir, Morocco[4]

***Abstract*—Over the past few decades, the volume of data has increased significantly in both scientific institutions and universities, with a large number of students enrolled and a high volume of related data. Furthermore, network traffic has increased with post-pandemic and the use of online learning. Therefore, processing network traffic data is a complex and challenging task that increases the possibility of intrusions and anomalies. Traditional security systems cannot deal with such high-speed and big data traffic. Real-time anomaly detection should be able to process data as quickly as possible to detect abnormal and malicious data. This paper proposes a hybrid approach consisting of supervised and unsupervised learning for anomaly detection based on the big data engine Apache Spark. Initially, the k-means algorithm was implemented in Sparks MLlib for clustering network traffic, then for each cluster, K-nearest neighbors algorithm (KNN) was implemented for classification and anomaly detection. The proposed model was trained and validated against a real dataset from *Ibn Zohr* University. The results indicate that the proposed model outperformed other well-known algorithms in detecting anomalies based on the aforementioned dataset. The experimental results show that the proposed hybrid approach can reach up to 99.94 % accuracy using the k-fold cross-validation method in the complete dataset with all 48 features.**

***Keywords*—Anomaly detection; big data; Apache Spark; k-means; KNN**

I. Introduction

The growth of digital technology and the internet has resulted in an explosion of data creation and consumption in various fields, including science and education. Universities and research institutions are generating and collecting a vast amount of data, including research findings, academic papers, student records, and administrative data.

In addition, the COVID-19 pandemic has significantly impacted the way education is delivered, with many institutions transitioning to online learning to maintain social distancing and reduce the spread of the virus. This shift has led to an increase in network traffic, as students and faculty members access online resources, participate in virtual classes, and communicate through online platforms. Processing such network traffic data is a complex and challenging task that increases the possibility of intrusions and anomalies. Therefore, universities and research institutions need to ensure that their digital infrastructure can handle the increased data volume and network traffic, while maintaining data security and privacy, but these pillars are not sufficient without anomaly detection.

Several security strategies have been implemented to secure networks; firewalls are an example of strategies used as a basic packet filter. Studies demonstrate that firewalls are not sufficient in providing secure environment [1][2]. Therefore, combining anomaly detection with firewalls can provide a safer network.

In machine learning (ML)-based anomaly detection, the anomaly is represented through a set of features that comprises the expected behavior of a system. Therefore, the ML model is expected to classify other events correctly if they present the same behavior during the training phase. Although, in a real-world environment, such as scientific institutions and universities, where the network traffic changes daily, either due to new attack discoveries or responding to new students' requests. Traditional security systems cannot deal with such high-speed and big data traffic. Real-time anomaly detection should be able to process data as quickly as possible to detect abnormal and malicious data.

The contribution of this paper is proposing a hybrid approach consisting of supervised and unsupervised learning for anomaly detection. Furthermore, the big data engine Apache Spark is used to provide continuous monitoring through real-time network traffic processing. The proposed approach is divided to three main components. (1) Data preprocessing to prepare data for feature extraction in machine learning. The dataset used in this analysis includes both numerical and symbolic representations, with 46 numerical attributes and 2 symbolic attributes. (2) Features selection based on K-means clustering in order to partitionate the dataset into K non-overlapping clusters based on their feature values. (3) Anomaly detection through clustering-based methods which are commonly used by analyzing the relationship between data instances and clusters. The existence of a large distance between an instance and the clusters can be used to identify an anomaly. In the suggested approach, the K-Nearest Neighbors (KNN) algorithm was applied with cross-validation techniques to detect network traffic anomalies.

The proposed model was trained and validated against a real dataset from *Ibn Zohr* University, and aimed to minimize false negatives and optimize true positives. The results indicate that the proposed model outperformed other well-known algorithms; in detecting anomalies based on the aforementioned dataset; compared to other well-known algorithms such as Random Forest [3], Support Vector Machine (SVM) [4], Naive Bayes [4] and Gradient Boosting [5]. The experimental results show that the proposed hybrid approach can reach up to 99.94 % accuracy using the k-fold cross-validation method in the

complete dataset with all 48 features.

The rest of this paper is organized as follows: Section II, present related work. Section III presents the proposed approach. Section IV reports and analyze experimental results. Section V presents general observations and considerations. Finally, the conclusion and expected future work are given in Section VI.

II. Related Work

The issue of designing an efficient intrusion detection system has been discussed in many articles [6]. In order to achieve this goal, researchers have been working for many years on different aspects. Mainly, improving the accuracy of intrusion detection by minimizing the false detection rate. Another aspect concerns real-time intrusion detection, especially in a big data environment [7]. In this direction, this section reviews the prior work on applying machine learning techniques to support real-time processing.

Vimalkumar and Radhika [8] have designed a big data framework for intrusion detection using classification methods and Apache Spark as a platform for implementing intrusion detection in smart grids using big data analytics. However, the accuracy obtained is less than 80% and the DNN prediction time is higher compared to other models.

The authors in [9] have evaluated multiple classification algorithms using Apache Spark on the full UNSW-NB15 dataset. The performance achieved 97.49 % accuracy for Random Forest. Nevertheless, with the absence of any computation platform details, it is hard to validate the extremely high accuracies reported. Zhang et al. [10] proposed a real-time intrusion detection system for high-speed network environments using a distributed Random Forest detection model based on Spark. Their experimental results showed that the framework has a shorter detection time, achieves higher accuracy, and can realize real-time intrusion detection in a high-speed network environment. Gupta and Kulariya [11] proposed the use of Apache Spark for fast and efficient network intrusion detection. The authors compared the performance of five classifiers on the datasets KDD99 and NSL-KDD. The Random Forest method achieved the best accuracy on both datasets. In the same direction of using machine learning techniques to improve IDSs and to reduce the FPR and FNR, the authors of [12] developed a prototype IDS using the k-means algorithm implemented in Sparks MLib.

In recent years, deep learning technologies are widely conducted in the field of intrusion detection [13][14]. For example, Mighan and Kahani [15] proposed a scheme that combines the advantages of a deep network and ML algorithms on Apache Spark. They used stacked auto-encoding network for feature extraction followed by several classification algorithms on The ISCX 2012 dataset. In the same direction, Chen et al. [16] proposed a method which uses the fusion convolutional neural network (FCNN) for feature extraction and stacked ensemble (SE) for classification.

Table I compares the related works in terms of data streams feature, ML algorithms and datasets that they have used, and the metrics used for evaluation. Some features in Table I such evaluation metrics are assigned * since there is no information available for that feature in related paper.

III. The Proposed Model

This section presents the proposed hybrid approach consisting of supervised and unsupervised learning for anomaly detection. Let us recall that the aim of the proposed model is to provide continuous monitoring trough real-time network traffic processing based on the big data engine Apache Spark. The general architecture can be described in Fig. 1.

The system presented in this article is designed as follows: for each new instance data captured in NetFlow format, Apache Spark Streaming is used as network traffic processing tool. The captured traffic from the input data is collected and streamed with Apache Kafka to Spark for processing. The captured packets from the input data as well as data transmission are mainly implemented by Kafka. This later is used because of its power to stream a huge amount of data in real time. Kafka mainly includes a producer and a consumer as shown in Fig. 2. The input application report messages through the producer API, and the output application subscribes to the message through consumer API.

To enhance detection efficiency, this proposed architecture makes use of the Apache Spark framework, as an open source powerful, scalable and fast distributed data processing engine in big data. Apache Spark provides a large number of libraries, such as MLlib and Spark Streaming. These tools allow Spark to support not only Streaming data processing but also machine learning algorithms. The current popular streaming frameworks mainly include Samza, Strom and Spark. Moreover, the use of Apache Spark framework is proved with the performance comparison of the literature [17].

1) The proposed algorithm : A joint algorithm is proposed for all layers to identify intruder flows. First, for each captured packet, the algorithm checks whether the incoming packet belongs to an already registered flow (lines 1-8 of Algorithm 1). A flow is defined as a five-tuples: IP source, IP destination, source port, and destination port (scr_{IP}, dst_{IP}, scr_{Port}, dst_{Port}, respectively). If the packet does not belong to a registered flow, then it is registered as a new distinct flow based on its scr_{IP}, dst_{IP}, scr_{Port}, dst_{Port} and a new sequence file is created for this flow by putting packet information into the first line (line 3 of Algorithm 1). On the other hand, if the packet belongs to a registered flow, then the packet information is simply sent to the particular sequence file corresponding to the registered flow (lines 6 and 7 of Algorithm 1). When the duration threshold deviates, the sequence file is sent to one of the preprocessing nodes to compute the flow parameters. The preprocessing nodes are equipped with a parameter calculation code to measure the feature values (lines 11 and 14 of Algorithm 1). In order to consider the big data environment, the code can be run in parallel by taking the sequence file as input in the distributed processing. Finally, the calculated feature values are sent to the decision server, which is equipped with several machine learning classifiers to decide whether the flow is an intrusion or normal flow based on its extracted features (line 17 of Algorithm 1). The complete system's flow is depicted in Fig. 3.

A. Data Preprocessing

The preprocessing step is critical for preparing data for feature extraction in machine learning. The real dataset used in

TABLE I. SUMMARY AND COMPARISON OF RELATED WORKS

Ref	Data streams	ML integration	ML algorithm used	Dataset description	Dataset used	Metrics comparison	Evaluation metrics
[8]		✓	DNN, SVM, RF DT and NB	✓	synchrophasor dataset.	✓	accuracy, recall, false rate, specificity
[9]		✓	SVM, RF DT and NB	✓	UNSW-NB15 dataset	✓	accuracy, recall, specificity
[10]	✓	✓	RF	✓	CICIDS2017open dataset	✓	precision, recall, F1-score
[11]		✓	LR, SVM, RF GBDT and NB	✓	DARPA's KDD'99, NSL-KDD	✓	accuracy, sensitivity, and specificity
[13]		✓	DNN	✓	DARPA's KDD'99, NSL-KDD		accuracy
[15]		✓	SVM and DT	✓	UNB ISCX 2012 dataset	✓ ✓	accuracy, recall, f-measure sensitivity, precision
[16]			CNN and stacked ensemble		NSL-KDD dataset	*	*
[17]		✓	RF, DT and NB	✓	BoT-IoT dataset		F-Measure (f1)
[18]	✓	✓	RF and DT	✓	CICIDS2017open dataset	✓	accuracy, precision, recall and F1-score
[19]		✓	CNN and LSTM	✓	NSL-KDD dataset	✓	accuracy and false alarm rate
[20]	✓	✓	Deep Learning	✓	MISP	✓	accuracy, precision recall and F1-score
[21]		✓	K-means	✓	Customize dataset		Anomaly rate
Our approach	✓	✓	K-means and KNN	✓	*Ibn Zohr* university Dataset	✓	accuracy, precision and recall

✓: Approach has this feature, Approach has not this feature, *: Information not available.

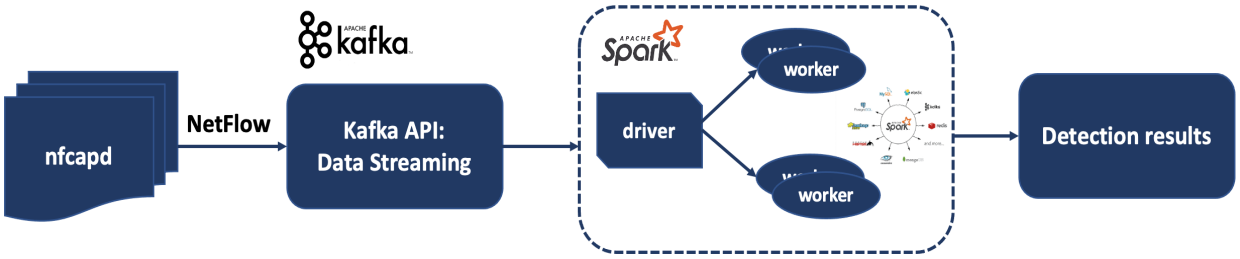

Fig. 1. The proposed approach architecture.

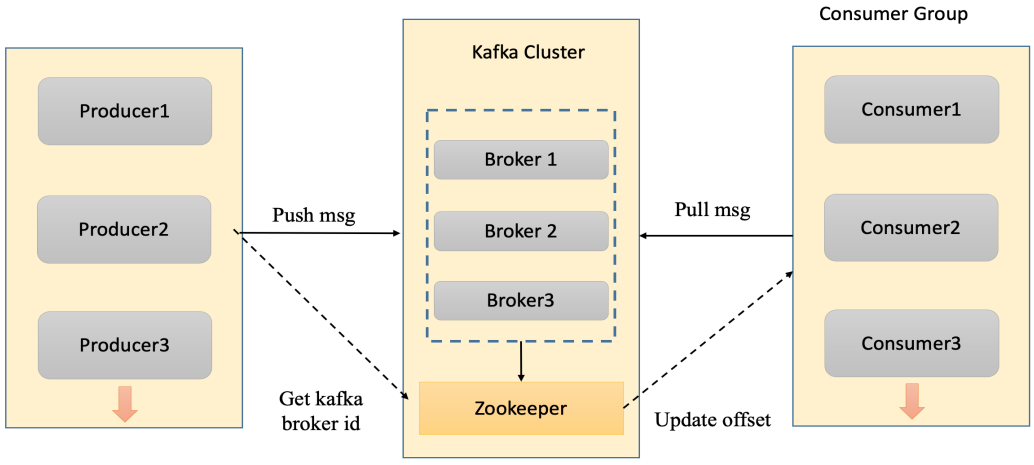

Fig. 2. Kafka architecture-Kafka Cluster.

this analysis includes both numerical and symbolic representations, with 46 numerical attributes and two symbolic attributes. To enable the use of symbolic attributes in the subsequent analysis, they are converted to numerical values, ensuring that all features are represented on the same scale. [21] emphasized that data normalization is an essential preprocessing step to eliminate the dimensionality effect and improve the accuracy of the model. StandardScaler is a widely used normalization method that scales input features to have a mean of 0 and a standard deviation of 1, which helps to remove biases caused by different units or scales of measurement as given in Eq.1. For numerical attributes, each value is normalized over the

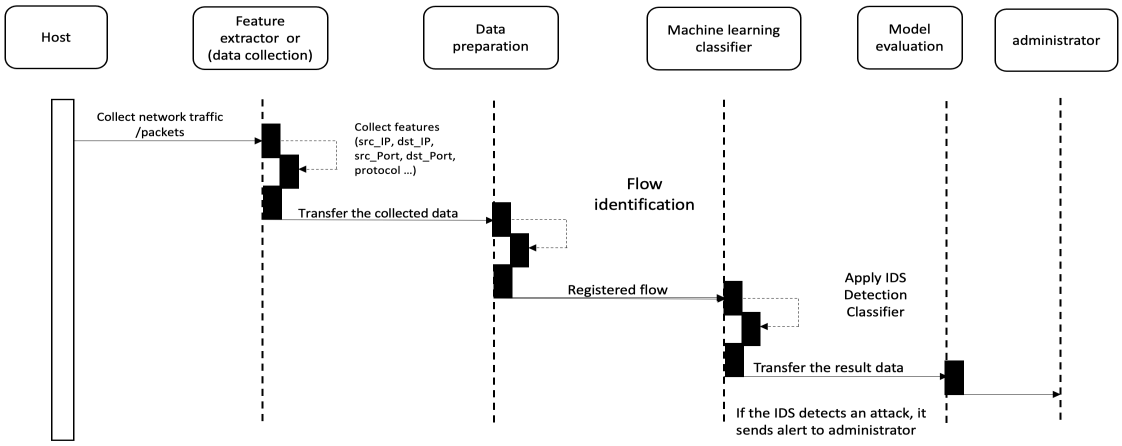

Fig. 3. The proposed algorithm sequence diagram.

Algorithm 1 Anomaly detection Algorithm

Input: continuous real-time network traffic/packets
Output: result, intrusions flows/normal flows.

for each incoming packet **do**
  if flow already register? = No **then**
    $flow_{list}$+=$new_{flow}(pktscr_{IP}, pktdst_{IP}, pktsrc_{Port}, pkdst_{port})$
    add packet parameters(new flown, new sequence file)
    return to next incoming packet
  else ▷ flow already register? = Yes
    add packet parameters (registered flow, sequence file)
  end if
  if flow duration $<$ time threshold **then**
    return to next incoming packet
  else ▷ flow duration $>$ time threshold
    send sequence file to reprocessing node
    for each sequence file **do**
      calculate flow parameter features
      send feature values to detection server
    end for
    result = ML classifier (parameters values)
    return next packet
  end if
end for

range [0,1] based on its deviation from the feature's mean and standard deviation. By standardizing the data in this way, the effect of dimensionality can be reduced, leading to more accurate and reliable analysis of the data [22].

$$X' = \frac{X - \mu}{\sigma} \tag{1}$$

Where X is a feature vector containing the original values, μ is the mean of the feature vector X, σ is the standard deviation of the feature vector X and X' is the scaled feature vector, with mean 0 and variance 1.

B. Clustering

K-means clustering is a widely utilized unsupervised learning algorithm in data analysis and machine learning [22]. K-means partitions the dataset into K non-overlapping clusters based on their feature values, with objects that belong to the same cluster having similar feature values [21]. The K-means algorithm consists of four main steps: first, it initializes the number of clusters K and the initial centroids for each cluster. Next, it iterates over all objects in the dataset and computes the distances between them and the centroids of each cluster. Based on this, it assigns the objects to the nearest cluster centroid. In the third step, it recalculates the centroids for each cluster based on the objects that are currently assigned to it. Finally, the algorithm repeats the second and third steps until the centroids of all clusters no longer change.

In order to determine the distance or similarity between two objects, it is necessary to use a distance function. The most commonly utilized distance metric is the Euclidean distance, which is defined as the square root of the sum of the squared differences between corresponding features in two input vectors, as illustrated in Equation 2. However, given that various features are often measured using different scales or metrics, it is crucial to normalize them before applying the distance function. An alternative to the Euclidean distance is the Mahalanobis distance function. This distance metric incorporates statistical correlations between different features by utilizing the inverse covariance matrix, as shown in Equation 3. However, the Mahalanobis distance can be computationally intensive, particularly for high-dimensional feature vectors. The Euclidean distance was utilized for the initial evaluation of the proposed anomaly detection method because the database was pre-normalized.

$$d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{2}$$

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)} \tag{3}$$

The K-means clustering algorithm is utilized for preprocessing training datasets that may consist of both normal and anomalous traffic. This is achieved without the need for prior labeling. The proposed approach is based on the assumption that normal and anomalous traffic can be distinguished by creating separate clusters in the feature space. The K-means clustering algorithm divides the training data into K clusters, without distinguishing between clusters that reflect time intervals of normal or anomalous traffic. For instance, an anomalous

cluster may be identified through a higher average number of packets. Clusters that are closely located to each other may arise due to either an unsuitable selection of K or the homogenous nature of the training data, where either there is no anomalous traffic or the anomalous traffic is similar to the normal traffic. A fundamental challenge of the K-means clustering method is to determine the appropriate number of K-clusters. Our study addresses this issue by concentrating on the evaluation criteria for optimizing clusters. Selecting the optimal value of k for K-means clustering is a crucial step in data analysis. This paper used the Silhouette approach to identify the optimal k value for *Ibn Zohr* dataset. The Silhouette method computes the silhouette coefficient for all instances across a range of k values. Equation 4 used to calculate the silhouette coefficient for each instance.

Silhouette coefficient:

$$\frac{b-a}{max(a,b)} \tag{4}$$

C. Anomaly Detection and Attack Classification

Clustering-based methods are commonly used to detect anomalies by analyzing the relationship between data instances and clusters. The existence of a large distance between an instance and the clusters can be used to identify an anomaly. The proposed approach utilizes the K-Nearest Neighbors (KNN) algorithm with cross-validation techniques to detect network traffic anomalies.

KNN [23] algorithm is founded on the principle that objects or instances which are similar tend to be situated in close proximity. One commonly used distance metric in KNN is the Euclidean distance method. After computing the distance between the current data point and the query data point, the KNN algorithm sorts the distances and their corresponding indices and stores them in a collection. The classification of instances in the KNN algorithm involves taking the mode of the first K labels from the sorted collection. Although KNN is a non-parametric algorithm that is simple and easy to understand, it can become slow when processing large datasets and may struggle to make accurate predictions in high-dimensional datasets. Furthermore, the process of selecting an appropriate value for K in KNN can be a significant challenge. The KNN algorithm can be implemented in six steps

- Selecting the number of neighbors (K);
- Computing the Euclidean distance;
- Selecting the K nearest neighbors;
- Counting the number of data points in each category among the K nearest neighbors;
- Assigning the new data point to the category with the highest number of neighbors, and
- Creating the K-NN classification model for future predictions.

The selection of an appropriate value for the hyperparameter K is crucial in achieving high accuracy when using the KNN algorithm. In this study, cross-validation was used to assess the performance of different K values on the test dataset. Cross-validation methodology involves segmenting the dataset into multiple subsets, and subsequently training the KNN model iteratively on one subset while testing it on the remaining subsets. This methodology permits the estimation of the KNN model's performance on new data, and facilitates the identification of the optimal K value that yields the best results. The use of cross-validation ensures that the KNN model neither overfits nor underfits the data, and guarantees optimal performance on previously unseen data. The dataset was partitioned into training and testing sets, with 70% of the data designated for training and 30% for testing. The training dataset was then passed through the KNN classifier for classification.

D. Decision Making

To evaluate the effectiveness and efficiency of the suggested method in detecting anomalies, a comparison was performed against well-known classification techniques such as Random Forest (RF), Support Vector Machines (SVM), Naive Bayes (NB), and Gradient Boosting (GB). The comparison enabled the determination of the relative performance of the proposed approach and the identification of its strengths and weaknesses. Additionally, experiments were conducted using various evaluation metrics to provide a comprehensive assessment of the model's performance. The aim was to demonstrate the superiority of the proposed method in accurately detecting anomalies in the *Ibn Zohr* university network. The results of the comparison are presented in the following section.

1) Random forest [3]: is an influential machine learning technique that combines decision trees with ensemble learning. The algorithm employs multiple decision trees that are trained on different subsets of the training data and features. Each tree independently provides a prediction, and the final output is the average prediction of all the trees. The algorithm works by partitioning the feature space into smaller subsets recursively, utilizing various metrics such as Gini impurity or entropy. The decision trees in Random Forest are created using a randomized feature selection process that reduces model variance and overfitting. The algorithm's benefits include its high accuracy, noise robustness, and capacity to handle high-dimensional data. However, the algorithm's computational cost can be significant, and its interpretability may be compromised due to its ensemble nature.

2) Support vector machine [4]: is a renowned machine learning algorithm that is widely used in both classification and regression tasks. The method works by creating a hyperplane that separates classes and maximizes the margin between them. This optimal hyperplane is identified by finding a subset of training data points called support vectors that lie nearest to the hyperplane. To handle non-linear decision boundaries, SVM transforms the input data into a higher-dimensional space using a kernel function, such as linear, polynomial, or Gaussian. The choice of kernel function depends on the nature of the problem.

3) Naive bayes [4] : is a well-known algorithm utilized in classification tasks. It is based on Bayes' theorem, which states that the probability of a hypothesis given the data is proportional to the product of the prior probability of the hypothesis and the likelihood of the data given the hypothesis. Naive Bayes is considered "naive" because it assumes that all features are independent of one another, simplifying the

computations but possibly leading to suboptimal performance in cases where the features are highly correlated.

4) Gradient boosting [5]: Gradient Boosting [5] is a machine learning algorithm commonly utilized in regression and classification tasks. The approach involves the combination of multiple weak learners, generally decision trees, to form a strong learner. Learners are added incrementally, with each new learner attempting to correct the errors of the preceding one. Gradient Boosting uses the gradient descent optimization algorithm to find optimal weak learner parameters. It achieves this by iteratively adjusting the parameters of the learners to minimize a loss function, such as mean squared error or log loss. The gradient of the loss function is calculated with respect to the output of the preceding learner, which is utilized to train the next learner. Gradient Boosting is highly customizable, with hyper parameters like the number of learners, learning rate, and maximum depth of trees, which can be tuned.

IV. Experimental Result

In this section, the dataset and the performance metrics used for the comparison are described in order to conduct the different experiments carried out. Then, the results that were obtained by using the proposed hybrid method in the big data ecosystem are presented and discussed. Apache Spark and Python language are used. All experiments are conducted using a machine with 1.5 GHz Intel Corei7 CPU@, 16 GB RAM, and with Ubuntu 20.04 trusty installed.

A. Dataset

In order to verify the effectiveness of the proposed approach a real dataset is used collected from digital data related to network traffic of *Ibn Zohr* university. The dataset was collected in the presidency of *Ibn Zohr* which consists of several faculties. The total size of the dataset is up to one million records. Data are sent mainly by network devices installed by *nfcapd* which is a *netflow* capture daemon of the *nfdump* tools. *nfcapd* reads *netflow* data from the network and stores it into files. The output file is automatically rotated and renamed every 5 minutes.

B. Performance Metrics

To evaluate the performance of an anomaly detection algorithm, four fundamental metrics can be used: true positive (TP), which represents the number of correctly identified attacks, true negative (TN), which represents the number of accurately identified normal connections, false positive (FP), which denotes the number of normal connections wrongly identified as attacks, and false negative (FN), which signifies the number of attack connections erroneously identified as normal [24]. Subsequently, the proposed approach is evaluated using the following metrics:

1) Accuracy: provides a measure of how well the classification model is able to accurately classify data instances, regardless of whether they are classified as positive or negative. It is calculated as the proportion of the total number of correctly classified records in a dataset over all the rows in that dataset.

$$Accuracy = \frac{TP + Tn}{TP + Tn + Fp + Fn} \tag{5}$$

2) Precision: measures how many of the positive predictions made by the model are actually correct. It is calculated as the ratio of true positive predictions to the total number of positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

3) Recall: measures how many of the positive instances in the dataset were correctly identified by the model. It is calculated as the ratio of true positive predictions to the total number of actual positive instances in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

C. Implementation and Experimental Results

1) Selecting the optimal number of clusters: Let us recall that data instances are grouped using K-means clustering technique. We have conducted a graph of Silhouette score vs K which represent an effective way to visualize and select the optimal number of clusters (K). Cluster cohesion and separation are measured using the Silhouette score. Cluster cohesion refers to how closely related the data points are within the same cluster, while cluster separation refers to how well-separated the clusters are from each other. A higher Silhouette score indicates a better clustering outcome (i.e. it implies that the data points are more closely related to their own cluster and less closely related to neighboring clusters). The ideal K number for finding abnormalities in *Ibn Zohr* data is thought to be the K value that corresponds to the highest Silhouette score. The cluster value K of 2 exhibits the highest Silhouette score, making it the best K value for anomaly identification, as shown in Fig. 4.

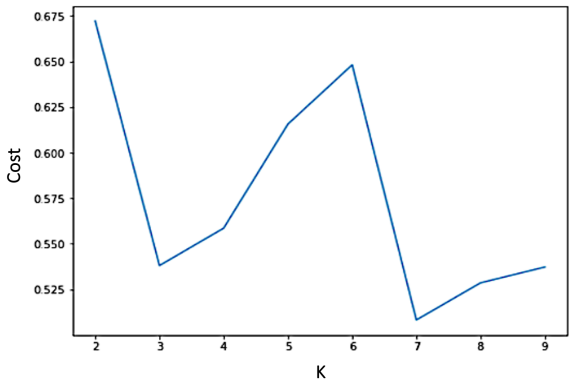

Fig. 4. Evolution of K value vs. silhouette score.

2) Apply k-means clustering Using optimal cluster value of K: To identify anomalies in each cluster, KNN with cross-validation is used The ideal number of folds for k-fold cross-validation is determined by comparing the average performance metrics obtained for different values of k. The best k value is determined by the greatest average accuracy. The proposed approach first determines the optimal number of folds for cross-validation, then evaluates the performance of the KNN model on new, unseen data using cross-validation techniques such as k-fold cross-validation. The average distance

to the k nearest neighbors for each point is then calculated using Euclidean distance to determine the boundary between different classes and identify the most relevant. Finally, the plot method is applied to visualize the average distance to the k nearest neighbors for each point in the *Ibn Zohr* dataset.

Fig. 5 shows the results of the anomaly detection algorithm conducted on the *Ibn Zohr* dataset. It is shown that the algorithm was able to successfully separate the anomaly instances from the normal ones. Anomalies are presented by the points that are far from the group of points in the feature space.

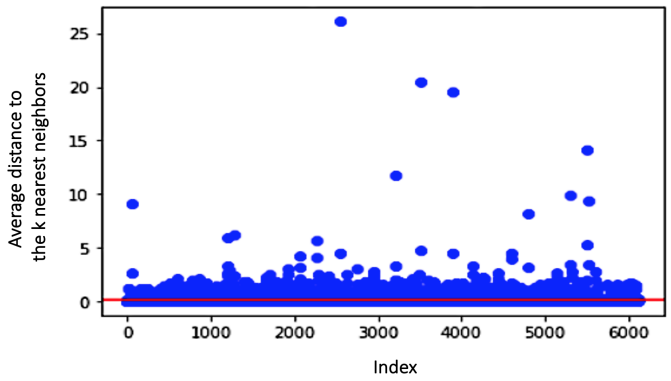

Fig. 5. Anomaly detection using KNN.

The proportion of anomalies identified by the model for various cluster values, as well as the non-linear connection between the cluster value and the number of observed anomalies, are depicted in Fig. 6. Anomalies, especially, tend to form their own clusters as the cluster value increases and may be incorrectly identified as normal data instances. Additionally, normal cases may form clusters, leading to some of them being wrongly classified as anomalies based on cluster threshold values.

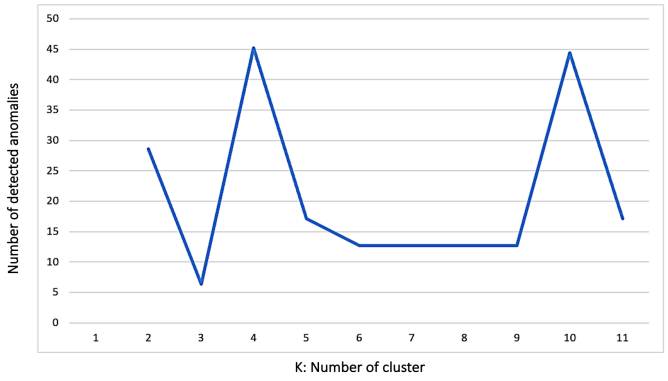

Fig. 6. The number of detected anomalies Vs. cluster number.

3) Effectiveness analysis of the proposed anomaly detection model using optimal K: In order to evaluate the effectiveness and efficiency of the suggested method in detecting anomalies, a comparison is performed against well-known classification techniques, such as Random Forest (RF), Support Vector Machines (SVM), Naive Bayes (NB), and Gradient Boosting (GB). The experimental data have been split into 70% as training data and 30% as test data. The evaluation metrics include accuracy, precision, and recall.

The results of the comparison using *Ibn Zohr* data are presented in Table II. Our approach is found to outperform the other models with consistently higher accuracy score. In contrast, Naive Bayes is found to achieve a lower accuracy score compared to the other evaluated methods.

TABLE II. COMPARISON OF THE APPROACH WITH THE OTHERS

Method	Accuracy (%)	Precision (%)	Recall (%)
RF	99.88	99.66	99.46
SVM	99.81	99.33	99.30
NB	95.04	73.77	100
GB	99.89	99.73	99.5
Our approach	99.94	99.92	99.5

4) Effectiveness analysis of the anomaly detection model for different cluster values: The performance of clustering-based anomaly detection methods can be significantly influenced by the choice of the number of clusters, represented by K. Therefore, identifying the optimal cluster value is crucial for achieving optimal results. Additionally, the choice of cluster value has a significant impact on the classification of anomalies in *Ibn Zohr* dataset.

A comparison of various cluster values ranging from 2 to 11 was conducted, and the results are presented in Table III. Our model achieved the highest classification accuracy of 99.94% for $K = 2$, while GB had the highest accuracy value of 99.75% for both $K = 2$ and $K = 4$. RF had the best accuracy value of 99.61% for $K = 4$, while SVM achieved the best accuracy value of 99.49% $K = 3$. NB achieved the best accuracy value of 97.66% for both $K = 2$ and $K = 4$.

TABLE III. ACCURACY COMPARISON FOR DIFFERENT k VALUES

Cluster Value	RF	SVM	NB	GB	Our approach
2	99.49	99.38	97.66	99.75	99.94
3	99.57	99.49	93.69	99.66	99.80
4	99.61	99.36	97.66	99.72	99.75
5	99.44	99.35	96.48	99.48	99.71
6	99.35	99.17	90.54	99.30	99.70
7	99.26	99.23	94.29	99.38	99.67
8	99.12	97.94	83.38	99.21	99.53
9	99.15	98.98	87.04	99.30	99.75
10	99.36	99.33	82.68	99.51	99.78
11	99.25	98.90	78.24	99.35	99.70

Based on the above analysis, it can be concluded that the optimal cluster values are $K = 3$ and $K = 4$, as they yielded better performance across most classifiers. However, since the accuracy value never exceeded 99.75%, the effectiveness of the model in terms of the trade-off between accuracy and computation cost is demonstrated for the optimal cluster value of $K = 2$.

V. DISCUSSION

The proposed anomaly detection model is based on the K-means clustering algorithm [21]. To determine the most suitable number of clusters, the Silhouette method was employed, which yielded $K = 2$ as the optimal value. Following this direction, the K-nearest neighbors (KNN) algorithm is applied in each cluster to identify anomalies in the whole dataset.

To ensure the reliability of the proposed model, cross-validation was implemented, and the optimal number of folds was determined using the K-fold method [25][26]. By selecting the most suitable value of K through K-fold cross-validation, overfitting was reduced, and the accuracy of the model was improved. Furthermore, to obtain the best possible results, an investigation is conducted of the data behavior for the global cluster, varying the number of clusters from $K = 2$ to $K = 11$. The findings indicated that the combination of the K-means clustering algorithm, KNN anomaly detection technique, and cross-validation was a remarkably successful method for identifying anomalies in the *Ibn Zohr* dataset. However, increasing the cluster value could result in anomalies themselves forming their own clusters, which could lead to some anomalies being classified as normal data instances.

Although better accuracy was achieved for the proposed anomaly detection model for $K = 2$, 3, 9, and 10, with the best accuracy being 0.994, a trade-off between accuracy and computation cost was observed. The evaluation of the K-means-based anomaly detection model showed that it was effective for the optimal cluster value of $K = 2$. However, the model's accuracy was found to be sensitive to the balance between the number of anomalies detected and the classification accuracy. Increasing the cluster value K could result in poor anomaly detection, as illustrated in Table III. Despite the use of the *Ibn Zohr* dataset to evaluate the effectiveness of the model, the methodology could be extended to other domains, such as IoT analytics and cybersecurity, and could prove useful for rule-based analysis based on specific datasets.

VI. Conclusion

The growth of digital technology and the Internet has led to a significant increase in data creation and consumption in universities and research institutions. Consequently, processing the resulting network traffic data has become a complex and challenging task, which increases the likelihood of intrusions and anomalies. Therefore, the primary objective of this paper is to address the issue of scalable network intrusion detection in a big data environment. The proposed hybrid model was validated using a real dataset from *Ibn Zohr* university. The evaluation results indicate that the suggested approach is efficient in detecting various anomalies. In addition, the effectiveness of the proposed model was verified by comparing its performance with other well-known models using different metrics.

The suggested framework's efficacy was evaluated using Apache Spark, a large data processing tool, and machine learning algorithms. The suggested model's performance was further examined using a hybrid machine learning method that blends k-means and KNN to identify anomalies.

Future work aims to integrate real-time anomaly identification to enable quick response and mitigation of any security breaches, thereby safeguarding all network traffic and protecting the privacy of all users on the *Ibn Zohr* university network.

Acknowledgment

The authors would like to express their gratitude to *Ibn Zohr* Presidency for providing their genuine dataset to assess the effectiveness of the proposed approach in this study.

References

[1] Connolly, Lena Y., and David S. Wall. "The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures." Computers & Security 87 (2019): 101568.

[2] Kristen, Erwin, et al. "Security assessment of agriculture iot (Aiot) applications." Applied Sciences 11.13 (2021): 5841.

[3] Reis, Itamar, Dalya Baron, and Sahar Shahaf. "Probabilistic random forest: A machine learning algorithm for noisy data sets." The Astronomical Journal 157.1 (2018): 16.

[4] Mandal, Jyotsna Kumar, and Debika Bhattacharya. Emerging Technology in Modelling and Graphics. Springer Singapore, 2020.

[5] Sandhu, Amandeep Kaur, and Ranbir Singh Batth. "Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm." Software: Practice and Experience 51.4 (2021): 735-747.

[6] Aljanabi, Mohammad, Mohd Arfian Ismail, and Ahmed Hussein Ali. "Intrusion detection systems, issues, challenges, and needs." International Journal of Computational Intelligence Systems 14.1 (2021): 560-571.

[7] Quincozes, Silvio E., et al. "A survey on intrusion detection and prevention systems in digital substations." Computer Networks 184 (2021): 107679.

[8] Vimalkumar, K., and N. Radhika. "A big data framework for intrusion detection in smart grids using apache spark." 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, 2017.

[9] Belouch, Mustapha, Salah El Hadaj, and Mohamed Idhammad. "Performance evaluation of intrusion detection based on machine learning using Apache Spark." Procedia Computer Science 127 (2018): 1-6.

[10] Zhang, Hao, et al. "Real-time distributed-random-forest-based network intrusion detection system using Apache spark." 2018 IEEE 37th international performance computing and communications conference (IPCCC). IEEE, 2018.

[11] Gupta, Govind P., and Manish Kulariya. "A framework for fast and efficient cyber security network intrusion detection using apache spark." Procedia Computer Science 93 (2016): 824-831.

[12] Azeroual, Otmane, and Anastasija Nikiforova. "Apache spark and mllib-based intrusion detection system or how the big data technologies can secure the data." Information 13.2 (2022): 58.

[13] Rawat, Shisrut, et al. "Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network." Internet Technology Letters 5.1 (2022): e232.

[14] Alferaidi, Ali, et al. "Distributed deep CNN-LSTM model for intrusion detection method in IoT-based vehicles." Mathematical Problems in Engineering 2022 (2022).

[15] Mighan, Soosan Naderi, and Mohsen Kahani. "A novel scalable intrusion detection system based on deep learning." International Journal of Information Security 20 (2021): 387-403.

[16] Chen, Chen, et al. "FCNN-SE: An Intrusion Detection Model Based on a Fusion CNN and Stacked Ensemble." Applied Sciences 12.17 (2022): 8601.

[17] Abushwereb, Mohamed. An accurate IoT intrusion detection framework using Apache Spark. Diss. Princess Sumaya University for Technology (Jordan), 2020.

[18] Seo, Wooseok, and Wooguil Pak. "Real-time network intrusion prevention system based on hybrid machine learning." IEEE Access 9 (2021): 46386-46397.

[19] Alferaidi, Ali, et al. "Distributed deep CNN-LSTM model for intrusion detection method in IoT-based vehicles." Mathematical Problems in Engineering 2022 (2022).

[20] Fotiadou, Konstantina, et al. "Incidents information sharing platform for distributed attack detection." IEEE Open Journal of the Communications Society 1 (2020): 593-605.

[21] Sinaga, Kristina P., and Miin-Shen Yang. "Unsupervised K-means clustering algorithm." IEEE access 8 (2020): 80716-80727.

[22] Goldstein, Markus, and Seiichi Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data." PloS one 11.4 (2016): e0152173.

[23] Imandoust, Sadegh Bafandeh, and Mohammad Bolandraftar. "Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background." International journal of engineering research and applications 3.5 (2013): 605-610.

[24] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." AI 2006: Advances in Artificial Intelligence: 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings 19. Springer Berlin Heidelberg, 2006.

[25] Pugazhenthi, A., and Lakshmi Sutha Kumar. "Selection of optimal number of clusters and centroids for k-means and fuzzy c-means clustering: A review." 2020 5th International conference on computing, communication and security (ICCCS). IEEE, 2020.

[26] De Bruin, Sytze, et al. "Dealing with clustered samples for assessing map accuracy by cross-validation." Ecological Informatics 69 (2022): 101665.