sustainability

MDPI

Article

Real-Time DDoS Attack Detection System Using Big Data Approach

Mazhar Javed Awan [1,*], Umar Farooq [1], Hafiz Muhammad Aqeel Babar [1], Awais Yasin [2], Haitham Nobanee [3,4,5,*], Muzammil Hussain [6], Owais Hakeem [6] and Azlan Mohd Zain [7]

1 Department of Software Engineering, University of Management and Technology, Lahore 54770, Pakistan; f2019114002@umt.edu.pk (U.F.); f2019114021@umt.edu.pk (H.M.A.B.)
2 Department of Computer Engineering, National University of Technology, Islamabad 44000, Pakistan; awaisyasin@nutech.edu.pk
3 College of Business, Abu Dhabi University, Abu Dhabi 59911, United Arab Emirates
4 Oxford Centre for Islamic Studies, University of Oxford, Marston Rd, Headington, Oxford OX3 0EE, UK
5 Faculty of Humanities & Social Sciences, University of Liverpool, 12 Abercromby Square, Liverpool L69 7WZ, UK
6 Department of Computer Science, University of Management and Technology, Lahore 54770, Pakistan; muzammil.hussain@umt.edu.pk (M.H.); owais.hakeem@umt.edu.pk (O.H.)
7 UTM Big Data Centre, School of Computing, Universiti Teknologi Malaysia, Skudai Johor 81310, Malaysia; azlanmz@utm.my
* Correspondence: mazhar.awan@umt.edu.pk (M.J.A.); nobanee@gmail.com (H.N.)

Abstract: Currently, the Distributed Denial of Service (DDoS) attack has become rampant, and shows up in various shapes and patterns, therefore it is not easy to detect and solve with previous solutions. Classification algorithms have been used in many studies and have aimed to detect and solve the DDoS attack. DDoS attacks are performed easily by using the weaknesses of networks and by generating requests for services for software. Real-time detection of DDoS attacks is difficult to detect and mitigate, but this solution holds significant value as these attacks can cause big issues. This paper addresses the prediction of application layer DDoS attacks in real-time with different machine learning models. We applied the two machine learning approaches Random Forest (RF) and Multi-Layer Perceptron (MLP) through the Scikit ML library and big data framework Spark ML library for the detection of Denial of Service (DoS) attacks. In addition to the detection of DoS attacks, we optimized the performance of the models by minimizing the prediction time as compared with other existing approaches using big data framework (Spark ML). We achieved a mean accuracy of 99.5% of the models both with and without big data approaches. However, in training and testing time, the big data approach outperforms the non-big data approach due to that the Spark computations in memory are in a distributed manner. The minimum average training and testing time in minutes was 14.08 and 0.04, respectively. Using a big data tool (Apache Spark), the maximum intermediate training and testing time in minutes was 34.11 and 0.46, respectively, using a non-big data approach. We also achieved these results using the big data approach. We can detect an attack in real-time in few milliseconds.

Keywords: DoS attack; Apache Spark; big data; privacy; sustainability; machine learning; real-time; DDoS detection

Citation: Awan, M.J.; Farooq, U.; Babar, H.M.A.; Yasin, A.; Nobanee, H.; Hussain, M.; Hakeem, O.; Zain, A.M. Real-Time DDoS Attack Detection System Using Big Data Approach. *Sustainability* **2021**, *13*, 10743. https://doi.org/10.3390/su131910743

Academic Editor: Young-Gab Kim

Received: 5 August 2021
Accepted: 24 September 2021
Published: 27 September 2021

1. Introduction

The data stored on the internet is growing day by day particularly for threats that target sensitive or crucial data, and this has raised many security issues, such as malicious intrusions [1]. Targeted attacks and threats such as malware and botnets cause great damage to the community in different factors, such as financial loss or health loss. Significant research has been conducted in which researchers have proposed different Intrusion Detection Systems (IDS) to mitigate the risk of malicious intrusion attacks [2]. Now,

the data can be extracted easily from information retrieval models as well as information extraction of any kind [3,4]. Traditional intrusion detection techniques can only work best on slow-speed data or small data. Currently, they are inefficient on big data and are incapable of handling high-speed data, so new methods need to be adapted to work on large data to detect any signs of intrusion. The security and privacy of data is now the big challenge in the life of big data, particularly in network attacks [5]. One major attack is the DDoS attack. DDoS attacks are cyber-attacks on specific servers or network with the intended purpose of disrupting that network or server's normal operation [6]. Real-time detection of DDoS attacks is not easy to detect and mitigate, but this solution holds great value as attacks can cause big issues [7]. First, we need to define intrusion, intrusion detection, and intrusion detection systems.

According to the National Institute of Standards and Technology (NIST), intrusion is defined as bypassing the security mechanism or violating the security policies Confidentiality, Integrity and Availability (CIA) in computer networks [8].

- Intrusion detection (ID) is divided into two processes: monitoring the intrusions and analyzing a network's events to seek any malicious packets or a source in the computer network or a computer system.
- Intrusion Detection System (IDS) detects an event as an intrusion when a noticeably different event occurs from a legitimate or authorized event [9].

Machine learning (ML) is the study that is continuously being enhanced via training and the exploitation of information. It is considered a component of artificial intelligence. There are different kinds of learnings based on the information available, such as supervised, semi-supervised and unsupervised learning. ML has a good results through the use of bimodal approaches through big data and deep learning models. ML techniques are utilized in a broad range of applications, such as in healthcare, for predicting COVID19, Osteoporosis and Schistosomiasis [10–15].

The recognition and detection performance in terms of security evaluation remain an issue, such as with DDoS attacks and blockchain. Classification algorithms have been used in many studies and aimed to detect and solve the DDoS attack. DDoS attacks can be performed easily by using the weaknesses of networks and by generating requests for services for software [16,17]. The elapsed time during any kind of real-time detection of DDoS attacks is also a challenge, and as attacks are not easy to detect and mitigate, this solution holds great value, as these attacks can cause big issues [18]. However, the existing approaches have many problems in detecting the DDoS attacks, such as computation costs while detecting, and the inability to deal with large requests passing through the network towards the server. Classification algorithms classify the packets for the identification of DDoS from the normal packets [19].

In recent years, many studies have applied the big data framework Spark ML to achieve better results, but they have not calculated the running time after applying Spark, as in our approach [20–25]. Moreover, machine learning algorithms, together with the big data approach, can solve many complex problems and find many hidden patterns in the context [26]. Big data promotes environment sustainability to solve very complex problems in real-time on social media using big data approaches, applications such fake profile detection or supply chain management with blockchain technology [27,28]. Apache Spark has evolved into an illegitimate platform for big data analysis. It is a computational framework for a high-performance cluster featuring Java, R language-integrated APIs, and Python. As a free software product that is quickly evolving, with a rising number of participants from university and business, the whole production and studies underlying Apache Spark, particularly by those that are novices in this domain, has proven difficult for academics. This article presents a practical assessment of extensive data analysis using Apache Spark. The major elements, concepts, and capabilities of Apache Spark are addressed in this analysis. In particular, this article illustrates the architecture and implementation of Apache Spark large-scale database techniques and pathways for master learning, diagram research, and data aggregation [29,30].

Two models were selected for real-time detection: Random Forest (RF) and Multi-Layer Perceptron (MLP). Both classifiers were evaluated using the Scikit ML libraries as a non-big data approach and Spark ML libraries as big data approach. In terms of accuracy, we achieved similar mean accuracy in both of the models. Still, in terms of training time and testing time, the big data approach outperforms the non-big data approach because Spark computations in memory take place in a distributed manner. The minimum average training and testing time in minutes was 14.08 and 0.04, respectively. Using a big data tool (Apache Spark), the maximum average training and testing time in minutes was 34.11 and 0.46, respectively, using a non-big data approach. We also achieved these results using the big data approach. We can detect an attack in real-time in few milliseconds.

The following are the major contributions of our study:

- As far as we know, there is no study such as this, which has compared accuracies as well as execution time with machine learning and Apache Spark ML on Distributed Denial of Service (DDoS).
- We detected DoS attacks in an efficient manner with and without the use of big data machine learning approaches in Random Forest and Multi-Layer Perceptron models.
- In addition to the detection of DDoS attack, we have optimized the performance of the models by minimizing the execution time as compared with other existing approaches using the big data framework.

The document is organized as follows: Section 2 describes the related work that has been done and reviews the current studies related to machine learning approaches in detecting network attacks; Section 3 presents a methodology and dataset we used for the experiments; Section 4 shows the experiments we performed using both techniques (big data and non-big data); Section 5 discusses the results obtained from both methods of detecting DDoS attacks in real-time; finally, in Section 6, the conclusions and future work are presented.

2. Related Work

Zhang et al. [31] proposed a framework to detect intrusion using a Random Forest algorithm using big data Apache Spark. Random Forest algorithm was implemented in Apache Spark to detect intrusion in the high-speed network data, and their efficiency and accuracy were then evaluated and compared to existing systems. An issue with this approach is that they evaluated few classification algorithms in their research. They implemented an RF algorithm in Apache Spark's design and evaluated their results compared with other fast speed data network models. Their proposed model achieved higher accuracy and can detect intrusion in a shorter amount time. The results showed that their model could detect intrusion in 0.01 s compared with other existing models. A simple RF model can detect an attack in 1.10 s.

Wang et al. [32] proposed a model that improves data learning with fewer resources and taking less time. They proposed a model "SP-PCA-SVM", which combines Support Vector Machine (SVM) with Parallel Principal Component Analysis (PCA) using Apache Spark tool to reduce training time and learning efficiency. The results showed that the accuracy rate is somehow reduced, reduces classification time (training time), and increases model learning efficiency. They proposed a novel model (SP-PCA-SVM), which significantly reduced training time and increased model learning efficiency. PCA was used to reduce the data set, and the SVM algorithm is used in parallel. Their results showed that the proposed model (SP-PCA-SVM) time was reduced from 46.4 s to 35.8 s, while Parallel SVM and accuracy were reduced from 93.56% to 93.40%; however; the training time for the classifier can be further optimized.

Zekri et al. [33] focused on how Distributed Denial of Service affects cloud performance by utilizing network resources. The attack techniques implemented and evaluated different ML algorithms in a cloud computing environment. The authors presented a DDoS protection design, and the algorithms they implemented and evaluated in cloud environments were Naive Bayes, Decision Tree (C4.5), and K-Means (KM). The results

showed that their accuracy and detection time of algorithms Naive Bayes, Decision Tree (C4.5), K-Means were 91.4% in 1.25 s, 98.8% in 0.58 s, and 95.9% in 1.12 s, respectively. This approach can work on real-time anomaly detection and mitigation techniques and other security challenges. The drawback of this approach is that they evaluated only a few models to detect DoS attacks.

Halimaa and Sundarakantham [34] implemented different machine learning techniques and evaluated their accuracies. Authors implemented Support Vector Machine (SVM) and Naive Bayes (NB) and evaluated their performance to detect intrusion using the KDD Cup data set. The authors in this paper performed a detailed comparative analysis for detecting intrusions in the network. The results showed that SVM achieved an accuracy of 93.95%, whereas NB achieved an accuracy of 56.54%.

A Raman et al. [35] study proposed a novel support vector machine (SVM) with Hyper-graph based Genetic Algorithm (HG-GA) to increase the efficiency of intrusion detection and reduce the false alarm rate. The authors evaluated this approach only on a few models. The results showed that the proposed approach increased the accuracy with the feature selection technique and decreased the false alarm rate. The proposed model achieved an accuracy of 96.2% when using with feature selection technique and 95.8% when using it without feature selection technique. Additionally, the false alarm rate scored 0.83 compared to the highest 1.74.

Wang et al. [36] proposed a new framework, "LMDRT-SVM", which is based on SVM (Support Vector Machine) that increases model accuracy and detection rate with augmented features. The NSL-KDD dataset has been used for evaluating model performance by applying the SVM model. The results showed that overall, it improved model performance and achieved accuracy, detection rate, and false alarm rate of 99.31%, 99.20%, and 0.60 on the NSL-KDD data set, respectively, and 99.3%, 99.94%, and 0.10 on the KDD data set, respectively.

Teng et al. [37] proposed a novel approach for securing the network from intrusions with the help of machine learning models. They performed experiments on the KDD Cup data set. The authors proposed an adaptive intrusion detection technique, Collaborative and Adaptive Intrusion Detection Model (CAIDM), which is based on various machine learning algorithms such as Decision Tree (DT) and Support Vector Machine (SVM). In addition, they used a tool, Environments classes, agents, roles, groups, and objects (E-CARGO), for modeling purposes. The results showed that the accuracy, error rate, and training time were significantly improved. Compared with the single type SVM accuracy of the proposed model (CAIDM), accuracy of the model was increased to 89.02% from 81.716%, and the error rate was reduced from 19.39% to 12.19% in CAIDM. However, the training time of the proposed model was also reduced from 29.730 min to 7.247 min.

Ahmad et al. [38] evaluated various machine learning models for detecting intrusions in the network system. Authors evaluated the SVM, RF, and Extreme Learning Machine (ELM) models for accuracy, recall, and precision. The train test split was used for the training and testing purpose at 80% for training and 20% for testing. The results showed that ELM outperforms other approaches with higher accuracy and precision of 95.5% on full sample data, and recall is slightly lower compared with SVM and RF.

Yin et al. [39] proposed a deep learning approach for IDS using recurrent neural networks. The authors used various machine learning models for performing their experiments on the benchmark data sets. Their experiment results showed that the Recurrent Neural Network Intrusion Detection System (RNN-IDS) outperforms other classification models and achieved high accuracy than other machine learning models. The proposed model achieved high accuracy in multiclass and binary classification methods. The results showed that the performance of the "RNN-IDS" model for binary and five category classifications achieved a detection rate of 83.28%.

Li and Yan [40] proposed IDS based on Apache Spark using "MSMOTR" and Adaboost models. The experiment was performed on the KDD99 dataset. The results showed that the proposed approach could reduce the system's error rate and processing time and improves

the accuracy rate. The results showed that the traditional model achieved accuracy, error rate, and processing time of 84.28, 17.2, and 18.26 s. The model achieved accuracy, error rate, and processing time of 86.32, 13.6, and 15.24 s, respectively.

Aftab et al. [41] applied deep learning models using the big data approach for the detection of the disease Leukemia. The library they used was Spark BigDL and their results showed that the proposed model was able to identify four types of Leukemia with an accuracy of 0.9733% on training data and 0.9478% on test data.

Al-Qatf et al. [42] implemented an anomaly-based network intrusion detection system. The proposed model can work on big data and achieves scores in minimum time. The proposed model used the big data tools Apache Spark and Hadoop. The authors encapsulated the approaches using data mining methods and machine learning for cyber analytics for IDS. The results showed that the system managed the large-scale network packet analysis quickly using Apache Hadoop and Spark.

Kato and Klyuev [43] studied the major problems of network intrusion detection using machine learning algorithms. The authors focused on designing a practically intelligent intrusion detection system having defined accuracy and a low false-positive rate. The results showed that the proposed approach could develop the intelligent IDS and achieved an accuracy of 86.2%, a false-positive rate of 13%, and a true negative rate of 87%.

Marir et al. [44] proposed a novel distributed approach (distributed deep method) to detect abnormal behavior in large-scale networks. The proposed model combined a deep neural network with multi-layer machine learning ensemble models for detecting abnormalities. The authors used the hidden layer model (SVM) with ensemble technique to enhance the prediction performance by reducing the number of features. Apache Spark was used to train and test the model and deep belief network to reduce the number of features. The results showed that the proposed model outperforms existing approaches.

Kim et al. [45] emphasized the IDS model by using an enthusiastic learning approach. The proposed model was based on "Long Short-Term Memory architecture." The experiment was performed on the KDD cup data set. The proposed model used a soft computing technique to create utilities in the intrusion detection field for learning and flexible capabilities. The proposed model was trained on ten dummy data sets and evaluated its performance. The results showed that the proposed approach achieved an average accuracy of 98.7% and an error rate of 11.3%.

Jha et al. [46] used the approach of data analysis to get such helpful information. Many big data research architecture structures for clustering algorithms have been created to expand big data research. They showed a maintainable cluster with progressive improvement in this paper. The Scalable Random Sampling with Iterative Optimization Fuzzy c-Means algorithm (SRSIO-FCM) approach was used to handle huge data collecting challenges in an Apache Spark cluster fuzzy c-means. SRSIO-performance FCM's was evaluated regarding the suggested expandable implementation of the Apache Spark cluster's Literal Fuzzy c-Means (LFCM) and Random Sampling plus Extension Fuzzy c-Means. Based on the duration and storage difficulty, run-time and clustering suggest that SRSIO-FCM can be run in significantly less time without clustering performance compromise.

Saravanan [47] presented a classification algorithm that works on network security data for intrusion detection. Multiple classification algorithms were implemented and evaluated using the big data tool Apache Spark and training time and testing time were measured. However, the authors evaluated few classification algorithms. As compared to the existing systems, they found better results with a good false-positive ratio. Better results were found for assessing the classification algorithm in Apache Spark on network security data than in existing systems. The accuracy of algorithms Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM) and SVM with Stochastic Gradient Descent (SGD) were 96.8%, 93.9%, 92.8%, and 91.1%.

Syed et al. [48] applied an ML-based application layer DoS attack detection framework for the detection of DoS attack on Message Queuing Telemetry Transport (MQTT) data communication protocol. The ML models applied for the detection of attack were Decision

Trees (C4.5), Multi-Layer Perceptron (MLP) and Average One-Dependence Estimator (AODE). The results showed that the C45 outperforms other with an accuracy of 99.09% than the AODE and MLP with accuracy of 99.00% and 95.93%, respectively, using limited features. The recall of C45 achieves a high score of 99.1% with respect to AODE and MLP, with 99% and 95.9%, respectively.

Priya et al. [49] proposed an ML-based model for the detection of DDoS attacks. The authors applied three different machine learning models: K-Nearest Neighbors (KNN), Random Forest (RF) and Naive Bayes (NB) classifier. The proposed approach can detect any type of DDoS attack in the network. The results of the proposed approach showed that the model can detect attacks with an average accuracy of 98.5%.

Ujjan et al. [50] proposed entropy-based DoS detection to identify features of traffic DoS by combining two entropies through Stacked auto encoder (SAE) and CNN. The CPU utilization was much higher and time-consuming. The accuracies of the models were 94% and 93%, respectively.

Gadze et al. [51] proposed deep learning models to detect and mitigate the risk of DDoS attacks that target the centralized controller in Software Defined Network (SDN) through Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). The accuracies of the models were lower. LSTM and CNN were 89.63% and 66%, respectively, when data spitted was in 70/30 ratio. However, in the case of LSTM model to detect TCP, UDP and ICMP, DDoS was the most time-consuming among the 10 attempts.

Ahuja et al. [52] proposed a hybrid ML model Support Vector Classifier with Random Forest (SVC-RF) for the classification of traffic as BENIGN or DDoS. The authors extracted the number of features from the original dataset and created a new dataset called SDN dataset with novel features. The proposed model results showed that the classifier SVC-RF can successfully classify the traffic with an accuracy of 98.8% using SDN dataset.

Wang et al. [53] introduced a new deep learning model based on an improved deep belief network (DBN), which works more efficiently to detect intrusions in the networks. They used Kernel-based Extreme Learning Machine (KELM) in DBN by replacing the Back Propagation algorithm. Their proposed model works efficiently as compared to existing approaches in a neural network. They evaluated different classification algorithms and measured their accuracies. The results showed that the "DBN-KELM" algorithm achieved an accuracy of 93.5%, and "DBN-EGWO-KELM" achieved an accuracy of 98.60%.

Dehkordi et al. [54] proposed a method to detect DDoS attack in Software Defined Network (SDN) using different machine learning models. The proposed method contained three main sections: (1) collect; (2) entropy-based; and (3) classification. The proposed method was applied on three different datasets. The results showed that by applying the ML models Logistic algorithms, J48 algorithm, BayesNet algorithm, Random Tree algorithm and REPTree algorithm, the average accuracy achieved was 99.62% 99.87%, 99.33%, 99.8% and 99.88%, respectively by application on an ISCX-SlowDDos2016 dataset.

3. Material and Methodology

3.1. Dataset

We used the application layer DDoS dataset available on Kaggle. The dataset belongs to a large dataset category and consists of around 0.9 million records with 77 features columns and a target column. Mainly it consists of three labels: (1) DDoS slow loris; (2) DDoS Hulk; and (3) BENIGN [55].

3.2. Our Approach and Data Pre-Processing

Figure 1 shows the system block diagram that applies classification algorithms after pre-processing to detect the DDoS attack. There are four main components involved: (1) data acquisition, (2) data pre-processing, (3) classification machine model, and (4) evaluation to produce the output of whether the attack is DDoS.

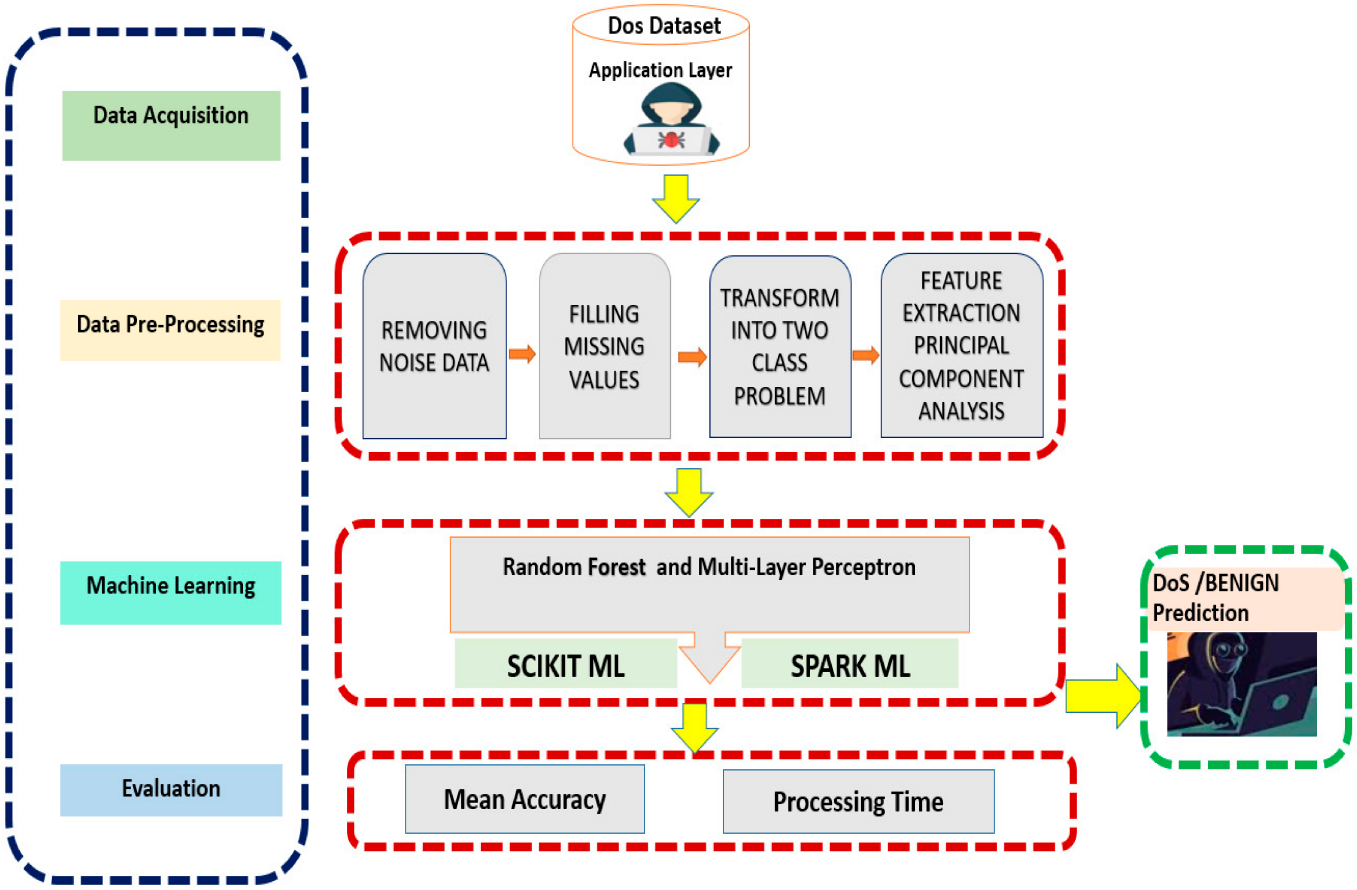

Figure 1. System Block Diagram to predict DDoS Attacks.

The first component (data acquisition) of the proposed model highlights the dataset that is used for the pre-processing phase of the model. In the second component of the model (pre-processing), we pre-processed the data before applying any machine learning model. First, noise filtering is performed on the dataset. Noise filtering is a collection of procedures used to reduce noise from data collected on development. In the next step, missing values are handled utilizing various policies, such as ignoring data with missing entries, replacing data with a universal, consistent method and explicitly filling in missing attributes depending on your area of expertise. In the third step, we transformed the 3-class problem into 2-class problems. We considered "DDoS slow loris" and "DDoS Hulk" as a single class and "BENIGN" as another class. In the last step, we reduced the number of features from 77 to 25 features. There were 77 features in the dataset, so Principal Component Analysis (PCA) technique was used to reduce the number of features and left only those most suitable features for predicting the model. Principal Component Analysis (PCA) tool is used for dimensionality reduction in both approaches with and without the big data approach. Scikit PCA is used for the non-big data approach, and Spark PCA is used for the big data approach. In the final step, all string type data was transformed into float type. Input data for Apache Spark ML models should be in vector form, so the string values of the data were transformed into numerical to create the dense vectors. The string indexer library was used to index the target column for Spark ML and Standard Scaler library for Scikit ML models.

In the third component of our proposed model, we applied two machine learning models, RF and MLP, for the training and testing of the data. Both models were applied with and without big data approach. We used Scikit libraries for the modeling purpose and considered it as non-big data approach, and for the big data approach we applied Apache

Spark libraries Spark ML for the training and testing of the dataset. Finally, in the last steps we measured the evaluation matrices of all the approaches and compared the results.

3.3. Classification Machine Learning Models

We applied two machine learning models for our approach in the case of with and without big data machine learning.

3.3.1. Random Forest

Stochastic classification systems are part of the wide range of outfit approaches to learning. They are easy to construct and quick to operate, and in several fields, they have been found quite beneficial. In the learning phase, multiple "basic" decision trees and the maximum vote (modal) in them at the categorization phase are the fundamental premise behind the Random Forest technique. This voting approach has, amongst other advantages, a correction to the excess dataset for the undesired characteristic of decision bodies. Random forests utilize the broad process-based packaging for individual trees in the composition during the learning phase. Periodically the packing chooses and matches a unique item to update the exercise set. Unlike any cutting, every tree is cultivated. Research has used RF for the construction of spatial distribution of the population density in a region [56].

The Equation (1) of Gini Importance, assuming only two child nodes is as follows:

$$mx_y = wtI_y - wt_{left(y)}I_{left(y)-}wt_{right(y)}I_{right(y)} \quad (1)$$

where in Equation (1) the mx_y is importance of node and I is Impurity and wt is weight of left and right with number of samples

Figure 2 shows the Random Forest classifier diagram of how it works. Several tresses are used for the predictions, and the voting approach is used for the target output. The random character of tree construction is an intrinsic outcome of that.

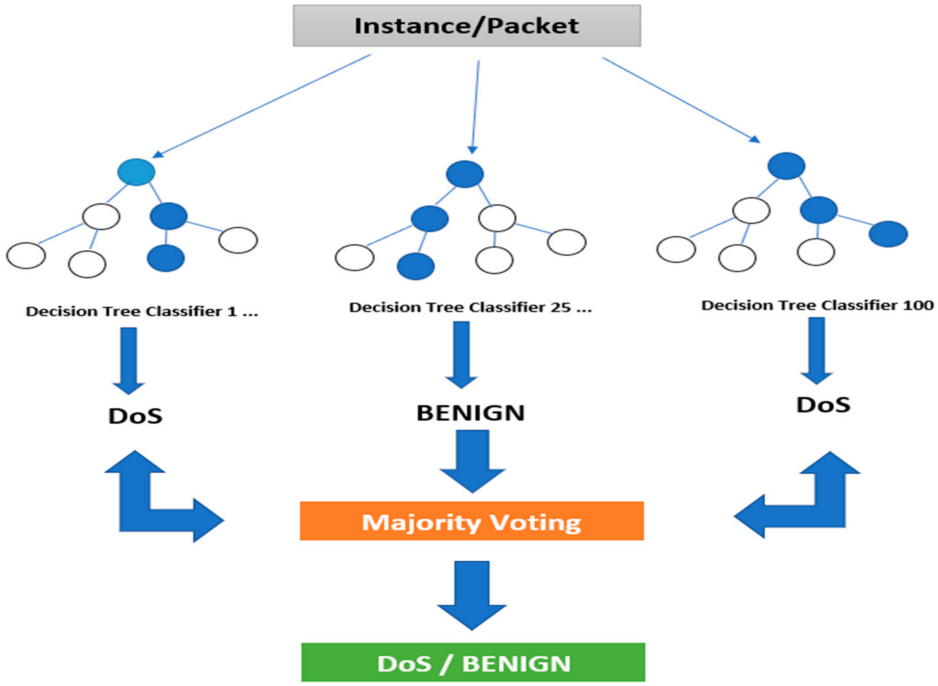

Figure 2. Random Forests classifier.

The number of trees in the whole set is a flexible variable that the out-of-bag mistake quickly discovers. As with Naive Bayes—and neighboring methods close to it—Random Forest is very common due to its clarity and strong results. In contrast to the two previous

techniques, however, Random Forest is not predictable in terms of the architecture of the resulting framework produced.

3.3.2. Multi-Layer Perceptron

One of the main frequent algorithms in the domain of neural learning is a Multi-Layer Perceptron (MLP). A brain network called 'vanilla' is sometimes referred to as MLP, and the intricate patterns of the present era are easier. However, it has also paved the way for increasingly sophisticated convolutional neural networks [15,57].

The MLP is a neural feedback system, meaning the input layer is used to send knowledge towards the output layer. Weights are allocated to the links across the layers. The gravity indicates the significance of a link. The MLP is used for various activities such as inventory evaluation. A Multi-Layer Perception is comprised of interlinked cells that transmit knowledge to each other, like the human brain.

The MLP updated the weight when it occurred error during misclassification. The Equation (2) shows the weight update through old weight plus learning rate (lr) multiplied with expected value (y) and predicted value (x) during forward and backward processes.

$$\text{wt updation} = \text{old wt} + \text{lr} * (\text{y} - \text{x}) * \text{a} \quad (2)$$

Figure 3 shows the Multi-Layer Perceptron diagram including inputs, hidden layer and output layer. A value is allocated to every cell. The system can be split into three top layers: (1) input Layer, the earliest communication layer that receives an intake to create an outcome; (2) hidden layer(s), where at least one hidden layer is a critical system, and they input data, calculations and processes to create everything useful; and (3) output layer, where there is a significant output in the neurons in this layer.

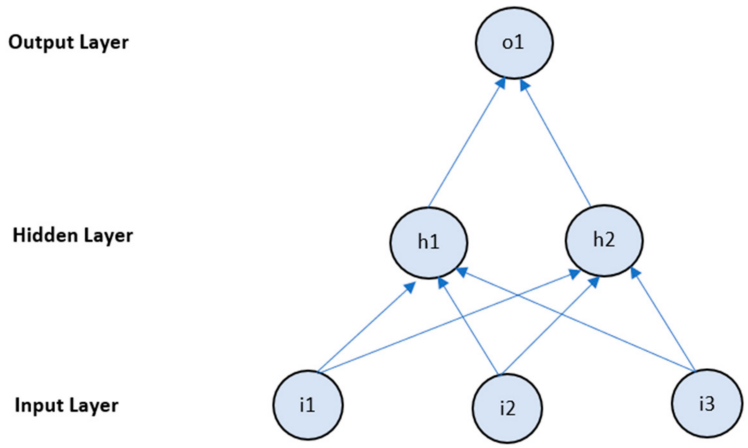

Figure 3. Multi-Layer Perceptron classifier.

4. Experiments and Results

4.1. Experiment Setup

Random Forest and Multi-Layer Perceptron were used to detect an attack in real-time and evaluate the performance with and without the big data approach. We used Apache Spark, a big data tool distributed framework, to speed up the computational and time-consuming tasks. Spark ML libraries were used to evaluate the performance with the big data approach and Scikit ML on Google Colab [58] libraries for the non-big data approach. The system specifications of the server we used for both approaches are the same. We used Databricks Community Editions for experiments with available memory of 15.3 GB and two cores with a single DB [59]. Spark v3.1.1 was used for Spark ML Libraries, and no worker nodes were used. The availability zone of the server was US-West-2C.

4.2. Experimental Parameters

We used the Cross-Validation technique, and started with the 100 trees at the beginning up to 500 trees with a step of 100 and with other default settings. There were two minimum samples used to split the internal node of a tree and one minimum number of samples were used to be at a leaf node. For measuring the quality of the split criteria, "Gini" was selected. Moreover, the parameter was set to auto for the RF to consider the number of features for the split. About 70% of the data was used for training, and 30% was used for testing.

The variables used for performing the experiment with Multi-Layer Perceptron classifier were as follows. We used Cross-Validation technique, and started with the 100 iterations at the beginning up to 500 iterations with a step of 100 and with default other settings. The Adam optimizer was utilized for the weight optimization of the model and Rectified Linear Unit (ReLu) was used as activation function. We had also shuffle samples for each iteration and the value of exponential decay rate was set to 0.9. Here also 70% of the data was used for training and the remaining 30% was used for testing.

4.3. Evaluation Results

We measured results through accuracy, precision, recall, F1 score and confusion matrix of our both models.

Figure 4 shows the accuracy comparison of both models, Random Forest classifier (RF) and Multi-Layer Perceptron (MLP) for both approaches, big data and non-big Data.

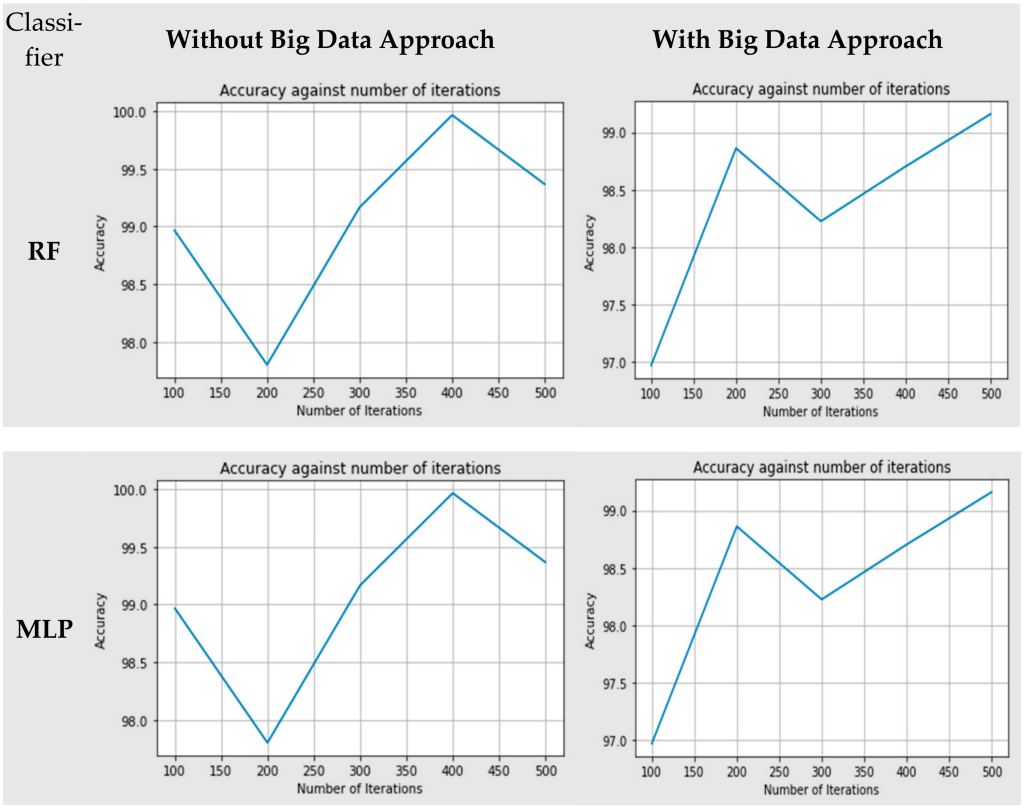

Figure 4. Accuracy comparison of both models (RF and MLP) for both approaches (big data and non-big data).

The RF classifier was used, and this decision was based on the majority of the voting. When the model was trained with 100 trees, we achieved the minimum accuracy of about 99.868%, and the maximum accuracy achieved was 99.989% when the model was trained with the 400 trees for non-big data approach. On the other hand, for big data approach the model was trained with 200 trees, and we achieved the minimum accuracy of about 99.045%, and maximum accuracy achieved was 99.895% when the model was trained

with the 400 trees. The MLP classifier was used with a minimum of 100 iterations up to a maximum of 500 iterations. When the model was trained with 200 iterations, we achieved the minimum accuracy of about 97.5%, and maximum accuracy achieved was 99.9% when the model was trained with the 400 iterations for non-big data approach. On the other hand, for the big data approach, the model was trained with 100 iterations, and we achieved the minimum accuracy of about 96.9%. Maximum accuracy achieved was 99.5% when the model was trained with the 500 iterations.

Table 1 shows the evaluation matrix (accuracy, precision, recall, and F1 score) of both models, Random Forest classifier (RF) and Multi-Layer Perceptron (MLP), for both big data and non-big data approaches. When the RF model was evaluated the precision of the model was 99.97%, whereas recall was 99.98%. The false-positive rate was 0.004, and the false-negative rate was 0.002 for non-big data approach. When the big data approach was used the precision of the model was 99.94%, whereas recall was 99.97%. The false-positive rate was 0.007 and false-negative rate was 0.003.

Table 1. Evaluation matrix with and without big data approaches.

Evaluation Metrics	Random Forest Classifier		Multi-Layer Perceptron	
	Without Big Data	**With Big Data**	**Without Big Data**	**With Big Data**
Accuracy	99.97%	99.95%	99.96%	99.95%
Precision	99.97%	99.94%	99.96%	99.94%
Sensitivity	99.98%	99.97%	99.97%	99.97%
F1 Score	99.97%	99.95%	99.97%	99.96%
Matthews Correlation Coefficient	99.94%	99.90%	99.93%	99.91%
False-positive Rate	0.04%	0.07%	0.05%	0.07%
False Discovery Rate	0.03%	0.06%	0.04%	0.06%
False-Negative Rate	0.02%	0.03%	0.03%	0.03%
Specificity	99.96%	99.93%	99.95%	99.93%
Negative Predictive Value	99.97%	99.96%	99.97%	99.97%

When the MLP model was evaluated, the precision of the model was 99.96%, whereas recall was 99.97%. The false-positive rate was 0.005 and false-negative rate was 0.003 for the non-big data approach. When the big data approach was used, the precision of the model was 99.94%, whereas recall was 99.97%. The false-positive rate was 0.007 and false-negative rate was 0.003.

4.4. Execuation Time

Our main aim was to calculate the execution time of our all four approaches.

Figure 5 shows the running time comparison of both Random Forest classifier (RF) and Multi-Layer Perceptron (MLP) for both approaches, big data and non-big data.

When the RF model was trained with 100 trees, we achieved the minimum testing time of about 0.2 min, and the maximum testing time took around 0.8 min when the model was trained with the 400 trees for non-big data approach. On the other hand, for the big data approach the model was trained with 100 trees, and we achieved the minimum testing time of about 0.064 min. For the maximum testing time it took around 0.148 min when the model was trained with the 300 trees.

When the MLP model was trained with 300 trees, we achieved the minimum testing time of about 0.065 min, and the maximum testing time took around 0.168 min when the model was trained with the 500 iterations for non-big data approach. On the other hand, for the big data approach, when the model was trained with 500 iterations, we achieved the minimum testing time of about 0.023 min, and the maximum testing time took around 0.073 min when the model was trained with the 100 iterations.

Figure 5. Running time comparison of both models (RF and MLP) for both approaches (big data and non-big data).

5. Discussion

In terms of accuracy, we achieved a similar mean accuracy of the approaches, but in terms of training time and testing time the big data approach outperforms the non-big data approach due to the fact that Spark does all computations in memory in a distributed manner. In addition, the Random Forest classifier outperforms MLP, having a higher mean accuracy. In Figure 6, a comparison of average accuracies is presented for both the approaches.

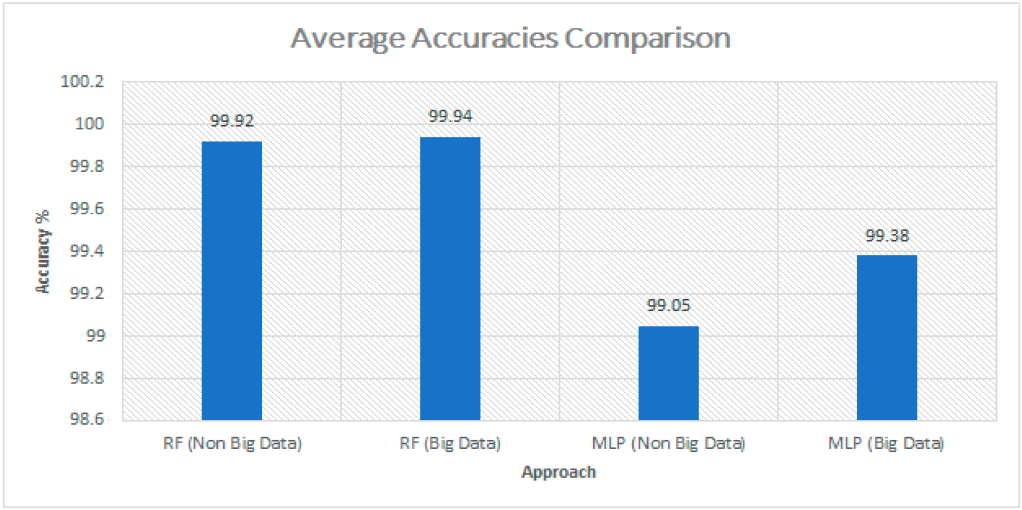

Figure 6. Average accuracies comparison between models using big data approach and non-big data approach.

It is clearly seen that the average accuracy of using big data approach or non-big data approach was somehow similar, and Figure 7 shows the average training time of the models using the Scikit Libraries and Apache Spark ML Libraries. The RF models took more time on training as it uses the ensemble technique which uses various decision tress to be trained.

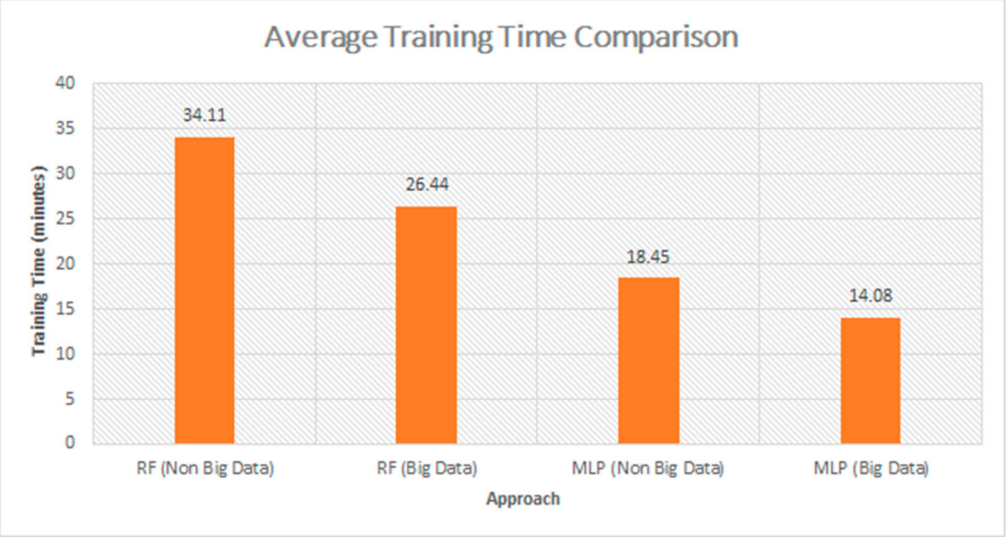

Figure 7. Average training time comparison between models using big data approach and non-big data approach.

Figure 8 shows the average testing time of the models using the Scikit Libraries and Apache Spark ML Libraries. It can clearly be seen that by using that big data approach, we have reduced the prediction time of the model to about 75% with almost similar prediction accuracy. The MLP algorithm achieves a minimum prediction time of about a few milliseconds for predicting an attack.

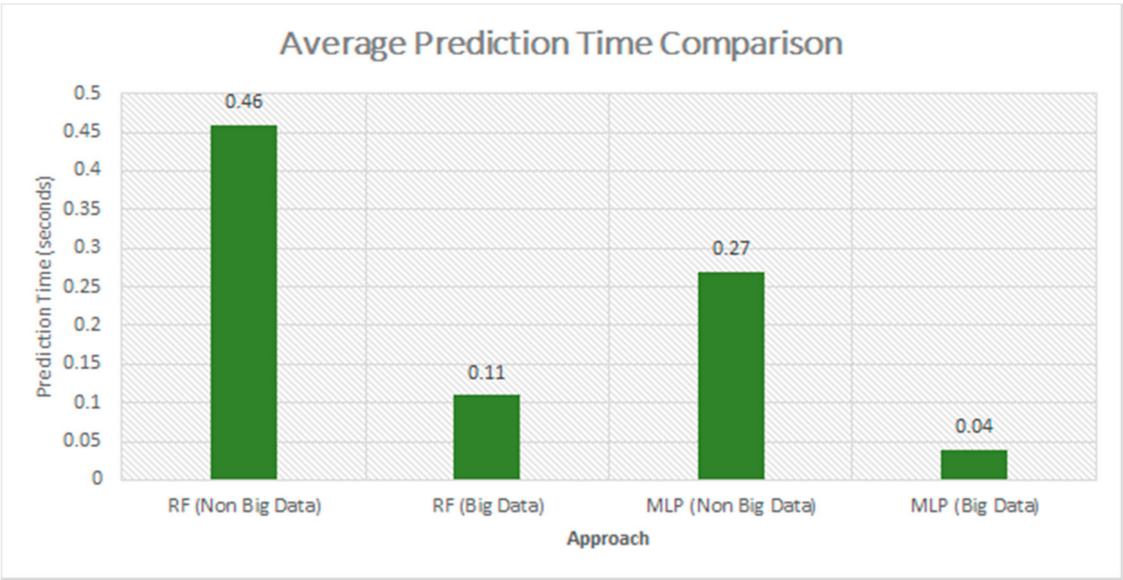

Figure 8. Average testing time comparison between models using big data approach and non-big data approach.

Table 2 shows the recent studies related to DDOS through various machine learning, deep learning and big data approaches with accuracies and execution time.

Table 2. State-of-the-art comparison work with our approaches in terms of accuracy and execution time.

Studies and Year	Model	Accuracy%	Execution Time (s)	Big Data Framework
Saravanan [47], 2020	Logistic Regression	93.90	0.48	-
	Decision Tree	96.80	1.02	-
	SVM	92.80	2.28	-
	SVM with SGD	91.10	2.06	-
Ujjan, Pervez, Dahal, Khan, Khattak and Hayat [50], 2021	SAE	94%	25% CPU used	-
	CNN	93%		-
Zhang, Dai, Li and Zhang [31], 2018	Random Forest	97.4%	1.10	Spark, (IDS detection)
Wang, Xiao and Long [32], 2017	PCA-SVM	86.31	-	Spark, (IDS detection)
	Parallel SVM	87.72	-	
	SP-PCA-SVM	90.24	-	
Zekri, El Kafhali, Aboutabit and Saadi [33], 2017	Naive Bayes	91.40	1.25	-
	Decision Tree	98.80	0.58	-
	K-Means	95.90	1.12	-
Halimaa and Sundarakantham [34], 2019	SVM	93.95	-	-
	Naive Bayes	56.54	-	-
Wang, Zeng, Liu and Li [53], 2021	DBN-KELM	93.50	-	-
	DBN-EGWO-KELM	98.60	-	-
Gadze, Bamfo-Asante, Agyemang, Nunoo-Mensah and Opare [51], 2020	Long Short-Term Memory	89.63	12.83	-
	Convolutional Neural Network	66.00	-	-
Ahuja, Singal, Mukhopadhyay and Kumar [52], 2021	Support Vector Classifier with Random Forest	98.80	-	-
Syed, Baig, Ibrahim and Valli [48], 2020	Decision Trees	99.09	-	-
	Multi-Layer Perceptron	99.00	-	-
	Average One-Dependence Estimator	95.93	-	-
Dehkordi, Soltanaghaei and Boroujeni [54], 2021	Logistic algorithms	99.62	1.19	-
	Bayes-Net algorithm	99.87	1.17	-
	Random Tree algorithm	99.33	1.17	-
	REPTree algorithm	99.80	1.16	-
	K-Nearest Neighbors	99.88	1.15	-
Priya, Sivaram, Yuvaraj and Jayanthiladevi [49], 2020	Naive Bayes	98.50	-	-
Our Approach-I With and without big data	Random Forest	**99.92**	**0.46**	Without Big data
		99.94	**0.11**	Spark ML
Our Approach-II With and without Big data	Multi-Layer Perceptron	**99.05**	**0.27**	Without Big data
		99.38	**0.04**	Spark ML

It can clearly be seen above in Table 2 that our approach using Random Forest and Multi-Layer Perceptron and big data framework Spark achieved better accuracies and processing time overall. The minimum accuracy in the above studies achieved was 56.64% in [34] using Naive Bayes classifier with the non-big data approach. By comparing with our approach, our minimum accuracy achieved was 99.05% with MLP classifier using the non-big data approach. The maximum accuracy achieved was 99.88% in [54] with REPTree model but without the big data approach, whereas compared with our approach, the maximum accuracy achieved was 99.94% with RF classifier using big data approach. The minimum processing time achieved was 0.48 s with Logistic Regression classifier in other studies. Using our proposed model, the minimum processing time achieved was 0.04 s using MLP classifier with big-data approach. Our proposed models achieved high accuracy and low processing time as compared with other existing models.

However, there are some limitations of our study, First, we used only two machine learning models. Second, the dataset has only two classes.

6. Conclusions

With the growing presence of data on the internet, new opportunities for threats to target sensitive data have been raised many security issues, such as malicious intrusions.

One major type of attack is the DDoS attack. Traditional intrusion detection techniques can only work best on slow-speed data or small data. Still, they are inefficient on big data and are incapable of handling high-speed data, so new methods adapted to work on large data to detect any signs of intrusion are needed. In this paper, we predicted DDoS attacks in real-time with different machine learning models using a big data approach. We used a distributed system, Apache Spark, and a classification algorithm to enhance the algorithms' execution. Additionally, we compared the results of the big data approach and how it outperforms the non-big data approach. Apache Spark is a big data tool to detect an attack in real-time with Spark ML libraries. We applied the two machine learning approaches, Random Forest (RF) and Multi-Layer Perceptron (MLP), through the Scikit ML library and big data framework Spark-ML library for the detection of DoS attack. In addition to the detection of DoS attacks we have optimized the performance of the models by minimizing the prediction time as compared with other existing approaches using big data framework. We achieved a similar mean accuracy in the models used, but in terms of training time and testing time big data approach outperforms the non-big data approach due to the fact that Spark performs computations in memory in a distributed manner. The minimum average training and testing time in minutes was 14.08 and 0.04, respectively, by using the big data tool (Apache Spark), and maximum average training and testing time in minutes was 34.11 and 0.46, respectively, by using the non-big data approach. Using the big data approach, we were able to detect an attack in real-time in a few milliseconds.

In the future, we will evaluate Apache Spark with other big data tools in terms of accuracy, training time and testing time of the machine learning models. We could also train different models and combine them with deep learning approaches using neural networks for predicting real-time results from convolutional neural network architectures [60–62]. Moreover, we could also apply datasets by recent work on big deep Learning (Big DL) framework [63].

Author Contributions: All authors contributed equally to this work; conceptualization, M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; methodology, M.J.A., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; software, M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; validation, M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; investigation, M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; resources; M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; data curation, M.J.A., A.Y., H.N., U.F., O.H., M.H., A.M.Z. and H.M.A.B.; writing—original draft preparation, M.J.A., A.Y., H.N., U.F. and H.M.A.B.; writing—review and editing, M.J.A., H.N., U.F., O.H. and H.M.A.B.; visualization, M.J.A., A.Y., H.N., U.F., O.H. and H.M.A.B.; project administration M.J.A., A.Y., H.N., O.H., M.H. and A.M.Z.; funding acquisition, M.J.A., A.Y., H.N.; A.M.Z. and H.M.A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: It is stated that dataset was publicly available on the Kaggle website https://www.kaggle.com/wardac/applicationlayer-ddos-dataset (accessed date 7 November 2019).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript.

AODE	Average One-Dependence Estimator
CAIDM	Collaborative and Adaptive Intrusion Detection Model
CIA	Confidentiality, integrity and availability
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DBN-EGWO-KELM	Deep Belief Network—Enhanced Grey Wolf Optimizer—Kernel-based Extreme Learning Machine
DDoS	Distributed Denial of Service
DoS	Denial of Service
DR	Diabetic retinopathy
DT	Decision Tree
E-CARGO	Environments classes, agents, roles, groups, and objects
EGWO	Enhanced Grey Wolf Optimizer
ELM	Extreme Learning Machine
FCM	Fuzzy C-Means
HG-GA	Hyper-graph based Genetic Algorithm
ID	Intrusion Detection
IDS	Intrusion Detection System
KELM	Kernel-based Extreme Learning Machine
KM	K-Means
KNN	K-Nearest Neighbors
LFCM	Literal Fuzzy c-Means
LMDRT	Logarithm Marginal Density Ratios Transformation
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine learning
MLP	Multi-Layer Perceptron
MQTT	Message Queuing Telemetry Transport
NB	Naïve Bayes
NIST	National Institute of Standards and Technology
NSL	Network Security Laboratory
PCA	Parallel Principal Component Analysis
PSO and KNN	Particle Swarm Optimization and K-Nearest Neighbors
RF	Random Forest
RNN	Recurrent Neural Network
RNN-IDS	Recurrent Neural Network Intrusion Detection System
SDN	Software Defined Network
SGD	Stochastic Gradient Descent
SRSIO-FCM	The Scalable Random Sampling with Iterative Optimization Fuzzy c-Means algorithm
SVC	Support Vector Classifier
SVC-RF	Support Vector Classifier with Random Forest
SVM	Support Vector Machine

References

1. Munoz-Arcentales, A.; López-Pernas, S.; Pozo, A.; Alonso, Á.; Salvachúa, J.; Huecas, G. Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE. *Sustainability* **2020**, *12*, 3885. [CrossRef]
2. Song, J.; Lee, Y.; Choi, J.-W.; Gil, J.-M.; Han, J.; Choi, S.-S. Practical In-Depth Analysis of IDS Alerts for Tracing and Identifying Potential Attackers on Darknet. *Sustainability* **2017**, *9*, 262. [CrossRef]
3. Rehma, A.A.; Awan, M.J.; Butt, I. Comparison and Evaluation of Information Retrieval Models. *VFAST Trans. Softw. Eng.* **2018**, *6*, 7–14.
4. Alam, T.M.; Awan, M.J. Domain analysis of information extraction techniques. *Int. J. Multidiscip. Sci. Eng.* **2018**, *9*, 1–9.
5. Koo, J.; Kang, G.; Kim, Y.-G. Security and Privacy in Big Data Life Cycle: A Survey and Open Challenges. *Sustainability* **2020**, *12*, 10571. [CrossRef]
6. Privalov, A.; Lukicheva, V.; Kotenko, I.; Saenko, I. Method of Early Detection of Cyber-Attacks on Telecommunication Networks Based on Traffic Analysis by Extreme Filtering. *Energies* **2019**, *12*, 4768. [CrossRef]

7. Nishanth, N.; Mujeeb, A. Modeling and detection of flooding-based denial-of-service attack in wireless ad hoc network using Bayesian inference. *IEEE Syst. J.* **2020**, *15*, 17–26. [CrossRef]
8. Scarfone, K.; Mell, P. Guide to intrusion detection and prevention systems (idps). *NIST Spec. Publ.* **2007**, *800*, 94.
9. Mukherjee, B.; Heberlein, L.T.; Levitt, K.N. Network intrusion detection. *IEEE Netw.* **1994**, *8*, 26–41. [CrossRef]
10. Gupta, M.; Jain, R.; Arora, S.; Gupta, A.; Awan, M.J.; Chaudhary, G.; Nobanee, H. AI-enabled COVID-9 Outbreak Analysis and Prediction: Indian States vs. Union Territories. *Comput. Mater. Contin.* **2021**, *67*, 933–950. [CrossRef]
11. Anam, M.; Ponnusamy, V.; Hussain, M.; Nadeem, M.W.; Javed, M.; Guan Goh, H.; Qadeer, S. Osteoporosis Prediction for Trabecular Bone using Machine Learning: A Review. *Comput. Mater. Contin.* **2021**, *67*, 89–105. [CrossRef]
12. Ali, Y.; Farooq, A.; Alam, T.M.; Farooq, M.S.; Awan, M.J.; Baig, T.I. Detection of Schistosomiasis Factors Using Association Rule Mining. *IEEE Access* **2019**, *7*, 186108–186114. [CrossRef]
13. Javed, R.; Saba, T.; Humdullah, S.; Jamail, N.S.M.; Awan, M.J. An Efficient Pattern Recognition Based Method for Drug-Drug Interaction Diagnosis. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 221–226.
14. Nagi, A.T.; Awan, M.J.; Javed, R.; Ayesha, N. A Comparison of Two-Stage Classifier Algorithm with Ensemble Techniques On Detection of Diabetic Retinopathy. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 212–215.
15. Abdullah, A.; Awan, M.; Shehzad, M.; Ashraf, M. Fake News Classification Bimodal using Convolutional Neural Network and Long Short-Term Memory. *Int. J. Emerg. Technol. Learn.* **2020**, *11*, 209–212.
16. Polat, H.; Polat, O.; Cetin, A. Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models. *Sustainability* **2020**, *12*, 1035. [CrossRef]
17. Ochôa, I.S.; Leithardt, V.R.Q.; Calbusch, L.; Santana, J.F.D.P.; Parreira, W.D.; Seman, L.O.; Zeferino, C.A. Performance and Security Evaluation on a Blockchain Architecture for License Plate Recognition Systems. *Appl. Sci.* **2021**, *11*, 1255. [CrossRef]
18. Dos Anjos, J.C.S.; Gross, J.L.G.; Matteussi, K.J.; González, G.V.; Leithardt, V.R.Q.; Geyer, C.F.R. An Algorithm to Minimize Energy Consumption and Elapsed Time for IoT Workloads in a Hybrid Architecture. *Sensors* **2021**, *21*, 2914. [CrossRef]
19. Ganguly, S.; Garofalakis, M.; Rastogi, R.; Sabnani, K. Streaming algorithms for robust, real-time detection of ddos attacks. In Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07), Toronto, ON, Canada, 25–27 June 2007; p. 4.
20. Awan, M.J.; Rahim, M.S.M.; Nobanee, H.; Yasin, A.; Khalaf, O.I.; Ishfaq, U. A Big Data Approach to Black Friday Sales. *Intell. Autom. Soft Comput.* **2021**, *27*, 785–797. [CrossRef]
21. Awan, M.J.; Rahim, M.S.M.; Nobanee, H.; Munawar, A.; Yasin, A.; Azlanmz, A.M.Z. Social Media and Stock Market Prediction: A Big Data Approach. *Comput. Mater. Contin.* **2021**, *67*, 2569–2583. [CrossRef]
22. Ahmed, H.M.; Javed Awan, M.; Khan, N.S.; Yasin, A.; Shehzad, H.M.F. Sentiment Analysis of Online Food Reviews using Big Data Analytics. *Elem. Educ. Online* **2021**, *20*, 827–836.
23. Awan, M.J.; Khan, R.A.; Nobanee, H.; Yasin, A.; Anwar, S.M.; Naseem, U.; Singh, V.P. A Recommendation Engine for Predicting Movie Ratings Using a Big Data Approach. *Electronics* **2021**, *10*, 1215. [CrossRef]
24. Awan, M.J.; Gilani, S.A.H.; Ramzan, H.; Nobanee, H.; Yasin, A.; Zain, A.M.; Javed, R. Cricket Match Analytics Using the Big Data Approach. *Electronics* **2021**, *10*, 2350. [CrossRef]
25. Khalil, A.; Awan, M.J.; Yasin, A.; Singh, V.P.; Shehzad, H.M.F. Flight Web Searches Analytics through Big Data. *Int. J. Comput. Appl. Technol.* **2021**, in press.
26. Zhou, L.; Pan, S.; Wang, J.; Vasilakos, A.V. Machine learning on big data: Opportunities and challenges. *Neurocomputing* **2017**, *237*, 350–361. [CrossRef]
27. Park, K.O. A study on sustainable usage intention of blockchain in the big data era: Logistics and supply chain management companies. *Sustainability* **2020**, *12*, 10670. [CrossRef]
28. Awan, M.J.; Khan, M.A.; Ansari, Z.K.; Yasin, A.; Shehzad, H.M.F. Fake Profile Recognition using Big Data Analytics in Social Media Platforms. *Int. J. Comput. Appl. Technol.* **2021**, in press.
29. Kshetri, N.; Torres, D.C.R.; Besada, H.; Ochoa, M.A.M. Big Data as a Tool to Monitor and Deter Environmental Offenders in the Global South: A Multiple Case Study. *Sustainability* **2020**, *12*, 10436. [CrossRef]
30. Awan, M.J.; Yasin, A.; Nobanee, H.; Ali, A.A.; Shahzad, Z.; Nabeel, M.; Zain, A.M.; Shahzad, H.M.F. Fake News Data Exploration and Analytics. *Electronics* **2021**, *10*, 2326. [CrossRef]
31. Zhang, H.; Dai, S.; Li, Y.; Zhang, W. Real-time distributed-random-forest-based network intrusion detection system using Apache spark. In Proceedings of the 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), Orlando, FL, USA, 17–19 November 2018; pp. 1–7.
32. Wang, H.; Xiao, Y.; Long, Y. Research of intrusion detection algorithm based on parallel SVM on spark. In Proceedings of the 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), Macau, China, 21–23 July 2017; pp. 153–156.
33. Zekri, M.; El Kafhali, S.; Aboutabit, N.; Saadi, Y. DDoS attack detection using machine learning techniques in cloud computing environments. In Proceedings of the 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), Rabat, Morocco, 24–26 October 2017; pp. 1–7.

34. Halimaa, A.; Sundarakantham, K. Machine learning based intrusion detection system. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 916–920.
35. Raman, M.G.; Somu, N.; Kirthivasan, K.; Liscano, R.; Sriram, V.S.S. An efficient intrusion detection system based on hypergraph-Genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowl.-Based Syst.* **2017**, *134*, 1–12. [CrossRef]
36. Wang, H.; Gu, J.; Wang, S. An effective intrusion detection framework based on SVM with feature augmentation. *Knowl.-Based Syst.* **2017**, *136*, 130–139. [CrossRef]
37. Teng, S.; Wu, N.; Zhu, H.; Teng, L.; Zhang, W. SVM-DT-based adaptive and collaborative intrusion detection. *IEEE/CAA J. Autom. Sin.* **2017**, *5*, 108–118. [CrossRef]
38. Ahmad, I.; Basheri, M.; Iqbal, M.J.; Rahim, A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access* **2018**, *6*, 33789–33795. [CrossRef]
39. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
40. Li, Z.; Yan, G. A Spark Platform-Based Intrusion Detection System by Combining MSMOTE and Improved Adaboost Algorithms. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; pp. 1046–1049.
41. Aftab, M.O.; Awan, M.J.; Khalid, S.; Javed, R.; Shabir, H. Executing Spark BigDL for Leukemia Detection from Microscopic Images using Transfer Learning. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 216–220.
42. Al-Qatf, M.; Lasheng, Y.; Al-Habib, M.; Al-Sabahi, K. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access* **2018**, *6*, 52843–52856. [CrossRef]
43. Kato, K.; Klyuev, V. Development of a network intrusion detection system using Apache Hadoop and Spark. In Proceedings of the 2017 IEEE Conference on Dependable and Secure Computing, Taipei, Taiwan, 7–10 August 2017; pp. 416–423.
44. Marir, N.; Wang, H.; Feng, G.; Li, B.; Jia, M. Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark. *IEEE Access* **2018**, *6*, 59657–59671. [CrossRef]
45. Kim, J.; Kim, J.; Thu, H.L.T.; Kim, H. Long short term memory recurrent neural network classifier for intrusion detection. In Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 15–17 February 2016; pp. 1–5.
46. Jha, P.; Tiwari, A.; Bharill, N.; Ratnaparkhe, M.; Nagendra, N.; Mounika, M. Fuzzy-Based Kernelized Clustering Algorithms for Handling Big Data Using Apache Spark. In *Proceedings of 6th International Conference on Harmony Search, Soft Computing and Applications, ICHSA 2020, Advances in Intelligent Systems and Computing*; Nigdeli, S.M., Kim, J.H., Bekdaş, G., Yadav, A., Eds.; Springer: Singapore, 2021; Volume 1275. [CrossRef]
47. Saravanan, S. Performance evaluation of classification algorithms in the design of Apache Spark based intrusion detection system. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020; pp. 443–447.
48. Syed, N.F.; Baig, Z.; Ibrahim, A.; Valli, C. Denial of service attack detection through machine learning for the IoT. *J. Inf. Telecommun.* **2020**, *4*, 482–503. [CrossRef]
49. Priya, S.S.; Sivaram, M.; Yuvaraj, D.; Jayanthiladevi, A. Machine learning based DDoS detection. In Proceedings of the 2020 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 12–14 March 2020; pp. 234–237.
50. Ujjan, R.M.A.; Pervez, Z.; Dahal, K.; Khan, W.A.; Khattak, A.M.; Hayat, B. Entropy Based Features Distribution for Anti-DDoS Model in SDN. *Sustainability* **2021**, *13*, 1522. [CrossRef]
51. Gadze, J.D.; Bamfo-Asante, A.A.; Agyemang, J.O.; Nunoo-Mensah, H.; Opare, K.A.-B. An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers. *Technologies* **2021**, *9*, 14. [CrossRef]
52. Ahuja, N.; Singal, G.; Mukhopadhyay, D.; Kumar, N. Automated DDOS attack detection in software defined networking. *J. Netw. Comput. Appl.* **2021**, *187*, 103108. [CrossRef]
53. Wang, Z.; Zeng, Y.; Liu, Y.; Li, D. Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection. *IEEE Access* **2021**, *9*, 16062–16091. [CrossRef]
54. Dehkordi, A.B.; Soltanaghaei, M.; Boroujeni, F.Z. The DDoS attacks detection through machine learning and statistical methods in SDN. *J. Supercomput.* **2021**, *77*, 2383–2415. [CrossRef]
55. Warda. Application-Layer DDoS Dataset. Available online: https://www.kaggle.com/wardac/applicationlayer-ddos-dataset (accessed on 7 November 2019).
56. Wang, F.; Lu, W.; Zheng, J.; Li, S.; Zhang, X. Spatially explicit mapping of historical population density with random forest regression: A case study of Gansu Province, China, in 1820 and 2000. *Sustainability* **2020**, *12*, 1231. [CrossRef]
57. Awan, M.J.; Raza, A.; Yasin, A.; Shehzad, H.M.F.; Butt, I. The Customized Convolutional Neural Network of Face Emotion Expression Classification. *Ann. Rom. Soc. Cell Biol.* **2021**, *25*, 5296–5304.
58. Awan, M.J. Acceleration of Knee MRI Cancellous bone Classification on Google Colaboratory using Convolutional Neural Network. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 83–88. [CrossRef]
59. Salloum, S.; Dautov, R.; Chen, X.; Peng, P.X.; Huang, J.Z. Big data analytics on Apache Spark. *Int. J. Data Sci. Anal.* **2016**, *1*, 145–164. [CrossRef]

60. Mujahid, A.; Awan, M.J.; Yasin, A.; Mohammed, M.A.; Damaševičius, R.; Maskeliūnas, R.; Abdulkareem, K.H. Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model. *Appl. Sci.* **2021**, *11*, 4164. [CrossRef]
61. Mubashar, R.; Awan, M.J.; Ahsan, M.; Yasin, A.; Singh, V.P. Efficient Residential Load Forecasting using Deep Learning Approach. *Int. J. Comput. Appl. Technol.* **2021**, in press.
62. Awan, M.J.; Rahim, M.S.M.; Salim, N.; Mohammed, M.A.; Garcia-Zapirain, B.; Abdulkareem, K.H. Efficient Detection of Knee Anterior Cruciate Ligament from Magnetic Resonance Imaging Using Deep Learning Approach. *Diagnostics* **2021**, *11*, 105. [CrossRef] [PubMed]
63. Awan, M.J.; Bilal, M.H.; Yasin, A.; Nobanee, H.; Khan, N.S.; Zain, A.M. Detection of COVID-19 in Chest X-ray Images: A Big Data Enabled Deep Learning Approach. *Int. J. Environ. Res. Public Health* **2021**, *18*, 10147. [CrossRef]