Ayodeji Ireti Fasiku, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Performance Analysis of an Intrusion Detection System Based on Big Data Analytics and Ensemble Techniques

Ayodeji Ireti Fasiku[1], Oghenerukevwe Oyinloye[2]
[1]Department of Computer Engineering, Ekiti State University, Ado – Ekiti, Ekiti State, Nigeria
[2]FutelSecure Inc., New Brunswick, Canada.

Abstract- Datasets encompass a wide range of network activities and intrusion patterns. The traditional intrusion detection systems (IDS) are struggling to provide all-round protection to the network but unable to analyze the new volumes of data and the velocity of today's networks. This research leverages the capabilities of big data analytics to process and analyze large-scale datasets collected from network traffic logs. Feature engineering and selection techniques were applied to extract relevant features that capture the distinguishing characteristics of normal and intrusive activities. Each model in the ensemble is trained independently using a subset of the data, utilizing their unique algorithms and strengths. The proposed system employs a range of machine learning models including Support Vector Machines (SVM), Decision Trees, Naive Bayes, k-Nearest Neighbors (KNN), Random Forest, Neural Networks, and two ensemble techniques, Bagging Ensemble, and XGBoosting. A comprehensive comparative analysis of these models were conducted to evaluate their performance in detecting intrusions accurately and efficiently. Hence, a comparative analysis was carried out to evaluate the performance of each model individually and as part of the ensemble. Performance metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are employed to assess the effectiveness of the models in identifying intrusions and minimizing false positives. The research contributes to the field of intrusion detection by providing insights into the performance of different machine learning models when applied to big data analytics and ensemble techniques. The comparative analysis aids in selecting the most effective models for building robust IDS solutions, improving network security, and safeguarding critical information assets against emerging cyber threats.

Keywords: Decision Trees, Naive Bayes, Random Forest, Neural Networks, Bagging Ensemble.

I. INTRODUCTION

The developmental revolution experience in computer technology, besides the widespread acceptance of the internet has grown to a level to make the world becoming a global village. Hence, easy access to different computers at a press of button resulted to security threat to computer users. With the ever-increasing sophistication of cyber threats, the development of robust and effective Intrusion Detection Systems (IDS) is crucial to protect computer users on the internet and local network from malicious activities [1-3]. The traditional IDS approaches often face challenges in handling large volumes of data and accurately detecting intricate intrusion patterns [4,5]. Hence, to address these limitations, IDSs have received a lot of interest in the research domain [6], using big data analytics [7-9] and ensemble techniques [10-13] to enhance IDS performance and accuracy.

The use of big data in IDS has become increasingly important as the volume of data generated by modern networks continues to grow [7,9,18]. Big data analytics provides a powerful framework for processing and analyzing massive amounts of data, enabling the identification of hidden patterns and anomalies [17-19]. Leveraging big data analytics in the context of intrusion detection allows for a more comprehensive understanding of network behaviour and facilitates the detection of previously unseen intrusion patterns [8]. Big data technologies, such as Spark, allow IDS to process vast amounts of network traffic data in real time, making it possible to detect threats that would otherwise go unnoticed [14,16,19]. By analyzing large amounts of data, IDS

can identify patterns and correlations that might indicate an intrusion, and then take appropriate action to prevent or mitigate the attack [7,9].

Ensemble learning is a machine learning technique that combine multiple individual models to form a single, more powerful model [10,12]. The idea behind ensemble technique, involves combining the predictions of multiple base models to create a more robust and accurate system than the predictions of any single model [10,13,15]. By leveraging the diverse capabilities and perspectives of individual machine learning models, ensemble techniques enhance the overall detection performance of IDS [16-17]. Models such as Support Vector Machines (SVM), Decision Trees, and Naive Bayes, k-Nearest Neighbors (KNN), Random Forest, Neural Networks, Bagging Ensemble, and XGBoosting [10,12-16] are commonly employed in ensemble-based IDS due to their distinct strengths and abilities to capture various types of intrusion patterns.

Ensemble techniques refer to the use of multiple models or algorithms to make predictions or decisions. An ensemble architecture where n number of weak learners are combined to form a strong learner is shown in Fig. 1.1. The weak learners are also called the base learners (learner 1 to learner n) which are usually generated from base learning algorithms that may be decision tree, neural network or any kind of learning algorithms [20, 21]. By combining each prediction or an output of individual models, ensemble techniques aim to improve the overall performance, accuracy, and robustness of the model compared to using a single model.

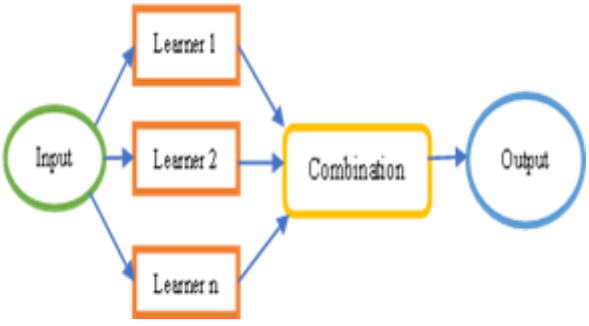

Fig. 1.1. Ensemble Architecture

This paper, presents a comprehensive study on the development of an IDS that integrates big data analytics [7-9] and ensemble techniques [10-13]. Hence, uses these approaches to enhance the accuracy, efficiency, and robustness of intrusion detection. In this research, a comparative analysis of multiple machines learning models, including SVM, Decision Trees, Naive Bayes, kNN, Random Forest, Neural Networks, Bagging Ensemble, and XGBoosting, were conducted [10, 12-16]. To evaluate the performance of these models, we utilized simulated attack scenarios comprising both normal and attack instances.

These datasets encompass a wide range of network activities and intrusion patterns. Feature engineering and selection techniques were applied to extract relevant features that effectively capture the characteristics of intrusions. Through rigorous experimentation and evaluation, we assessed the performance of each model individually and compared them with the ensemble approach, employing established performance metrics such as accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC) [12-14].

The results of our comparative analysis provide valuable insights into the strengths and weaknesses of each individual model and highlight the advantages of utilizing ensemble techniques in IDS. By leveraging big data analytics and ensemble approaches, our proposed IDS demonstrates superior performance that accurately detect intrusions while minimizing false positives. The findings of this study contribute to the field of intrusion detection, showcasing the potential of a big data analytics and ensemble techniques improving IDS performance and addressing the challenges posed that evolve cyber threats.

The paper organization is presented as follows. Section II describes the related work, reviewing several ML-based IDSs and Section III describes the proposed methodology and introduces the two ensemble techniques, Bagging and Boosting, and ML model used in the experiment. The proposed design architecture is given in Section IV. The

experimental results and discussion of the proposed intrusion detection are given in Section V. Section VI is dedicated to the conclusion of the paper.

II. RELATED WORK

In literature, different current state of the art solutions for the prevention and detection of cyber threat have been proposed. Many studies were engaged to improve the performance of the IDS system in different ways. Due to increase in the number of cyber-attacks, researchers are working daily on how to improve computer system security. This section discusses some related work done in IDS. Many IDSs have been proposed using ensemble method to improve accuracy for classification. Kumari and Mehta [20] proposed an ensemble-based model for intrusion detection using multiple ML techniques of classification such as DT, J48 and SVM. Particle swarm optimization was used for selecting nine most relevant and important features in KDD99 dataset of intrusion detection. Proposed model's results produced higher accuracy of 90% with low FAR 0.9%.

A performance analysis of multiple classical ML algorithms on several ID-based datasets for detecting attack traffic was proposed by Kilincer et al. [21]. After normalization of datasets (CICIDS2018, UNSW-NB15, ISCX2012, NSLKDD and CIDDS001), three ML techniques such as SVM, KNN and DT were deployed. DT outperforms other classifiers by producing detecting accuracy rate between 99 and 100% for all datasets. Another study of building an IDS using classification technique RF on NSL-KDD dataset was proposed by Fitni and Ramli [22]. Tree depth value was calculated by considering entropy score and Gini-index as z-score. Boruta technique was used for selecting 34 important features from dataset. The proposed model [22] produced 99% accuracy for detecting attacks.

The Jiang et al. [23] proposed an application layer IDS to detect DDoS attacks (ALDDoS) that utilized 30.8K of the CICIDS2017 dataset. The result obtained with their proposed system is able to detect DDoS attacks with an accuracy rate of 99.8%. However, the proposed IDS framework achieved accuracy up to 99.99% to detect DDoS attacks. Furthermore, the proposed IDS targets all the attack families existing in the dataset. Mokbal et al. [24] proposes a robust and effective intrusion detection framework based on the ensemble learning technique using eXtreme Gradient Boosting (XGBoost) and an embedded feature selection method. The proposed IDS framework successfully exceeded several evaluations on a big test dataset over both multi and binary classification. The authors limited the research on CICIDS2017 dataset only and did not include big data analytics tool that would have enhance the research.

Abdulhammed et al. [25] proposed IDS that employed two features with dimensionality reduction methods, they are Principle Component Analysis (PCA) and Auto-Encoder (AE). They applied different machine algorithms such as RF, Quadratic Discriminant Analysis (QDA), Bayesian Network, and Linear Discriminant Analysis (LDA) individually for the detection task. The RF achieved the best results on the CICIDS2017 dataset with accuracy, FP rate, and detection rate score of 99.50% 0.2% 98.5 respectively for binary-class with 59 features using PCA and the accuracy of 99.60% for multi-class. However, the proposed IDS results achieved an accuracy, FP rate, and detection rate score of 99.86%, 0.2%, and 99.75% for binary-class (abnormal) with 50 features and an accuracy score of 99.90% for multi-class.

Vijayanan et al. [26] proposed an IDS using multiple-SVM detectors trained on features selected by genetic algorithm. The proposed method trained each detector to detect a specific attack utilizing selected features by the GA. However, only a small part of CICIDS2017 dataset samples is utilized to estimate the IDS, and the results achieved for accuracy, precision, detection rate, and specificity score are 99.39%, 99.43%, 96.04%, and 99.67%, respectively. However, the proposed IDS results achieved accuracy, precision, detection rate, and specificity score of 99.90%, 99.90%, 99.97%, and 99.90%, respectively. Furthermore, almost all the

samples included in the CICIDS2017 dataset are used.

Abbas et al. [27] proposes an ensemble-based intrusion detection model. In the proposed model, logistic regression, naive Bayes, and decision tree have been deployed with voting classifier used to analyze the model's performance with some prominent existing state-of-the-art techniques. The results illustrate significant improvement in terms of accuracy as compared to existing models in terms of both binary and multi-class classification scenarios. The authors limited the research on CICIDS2017 dataset only and did not include big data analytics tool that would have enhance the research.

Moreover, Ustebay et al. [28] proposed DoS-IDS to detect denial of service. The Fisher Score method is utilized for feature selection. The Nearest Neighbor (KNN), Support Vector Machine (SVM), and Decision Tree (DT) algorithms are applied individually to use as the classifiers. The accuracy results achieved with SVM, KNN, and DT algorithms were 99.7%, 57.76%, and 99%, respectively. In contrast, the proposed IDS targets all the attack families existing in CICIDS2017, further, as shown within confusion matrix results, the proposed work achieved 100% results for DoS and also100% for DDoS attacks.

Aksu et al. [29] proposed IDS for Distributed Denial-Of-Service (DDoS) using 70% of the DDoS dataset, which belongs to CICIDS2017. The ten important features were selected using Recursive Feature Elimination (RFE) and utilizing Random Forest (RF) and Multilayer Perceptron (MLP) for detection tasks. Their results obtained on binary classification with accuracy and Receiver Operating Characteristic (ROC) score of 91% and 97%. However, our proposed achievements on binary classification with accuracy and ROC score of 99.86% and 99.72%. Furthermore, our IDS applied to all attacks embedding in the dataset.

III. METHODOLOGY

Developing an Intrusion Detection System (IDS) using big data analytics and an ensemble learning approach is a sophisticated and effective way to enhance network security. This approach combines the power of big data processing and multiple machine learning models to improve the accuracy and robustness of an intrusion detection system. In an intrusion detection system, different methodologies have been deployed. Developing an IDS using Big Data Analytics and Ensemble Techniques involves selecting appropriate algorithms to detect and classify intrusions effectively [1,3,4]. It is an iterative process that involves continuous monitoring, adaptation, and improvement. Regular models updates, perfect feature selections, and response mechanisms are essential to keep the system effective against emerging threats [5]. The following outline the methodological structural approaches employ for developing the proposed IDS using Big Data Analytics and an Ensemble Learning Techniques [8,10,11].

A. Data Sources and Collection

The sources of data to be used for analysis would be identify. Collect diverse and representative dataset that includes both normal and malicious activities. Hence, preprocess the data to handle missing values, outliers, and ensure it is suitability for machine learning. Also, determine the extent of data collection in term of its historical data, real-time data streams, or both.

B. Data Preprocessing

The processing dataset is been clean to care for missing values, outliers, inconsistencies and noise. Hence, apply standardize features to ensure data consistency by extracting relevant features from raw data, encode, perform exploratory data analysis to gain insights into the data distribution.

C. Feature Selection/Engineering

The data are explored and design with relevant features that can capture patterns indicative of intrusions. Using domain-specific knowledge to enhance feature selection.

D. Ensemble Techniques and Big Data Analytics Tools Utilization

Ensemble techniques involves combination of multiple machines learning models, boosting methods, bagging, or hybrid approaches such as Random Forest, AdaBoost, or Gradient Boosting to either combine the predictions of multiple base models suitable for the proposed design. Hence, define the specific big data analytics tools and technologies to be used such as Hadoop, Spark and how they will be integrated into the system.

E. Algorithm Selection

In chosen an ensemble learning algorithms suitable for intrusion detection (e.g., Random Forest, AdaBoost, Gradient Boosting). Hence, implement parallelized versions of these algorithms to leverage the big data infrastructure.

F. Model Training

In training the proposed data collated, firstly we split the dataset into training and testing sets. Train individual base models on different subsets of the training data. Hence, train the ensemble model on the predictions of these base models. The data trained would be experiment with various algorithms and compare their performances.

G. Bagging

Bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. Bagging technique [32] builds multiple classifiers based on a number of bootstrap samples from a given dataset. The label outputs of these classifiers are then decided by majority vote. The training and testing operation of Bagging algorithm is shown in Algorithm I.

Algorithm I

Input: Given a labeled dataset D

Training

- Choose the number of bootstrap samples n and the base classifier C;
- Create n new training datasets D1, D2, ..., Dn by sampling with replacement;
- Train the base classifier on each Di to build n classifiers C1, C2, ..., Cn;

Testing

- For each object x in the testing dataset, classify x by all classifiers C1, C2, ..., Cn;
- Decide the label assigned to x by majority voting: the label is the class with the largest number of votes;
- Repeat with all the data points in the testing set and return the outputs;

H. XGBoost

XGBoost is known for its scalability, efficiency, and ability to handle large-scale datasets. It also provides various hyper parameters that can be tuned to optimize the model's performance, such as learning rate, tree depth, subsampling ratios, and regularization parameters.

$$\widehat{y_j} = \sum_m^M f_m(x_j),\ f_m \in \mathcal{F}$$

Where m is the number of trees, f is the functional space of F, F is the set of possible CARTs. The objective function for the above model is given by:

$$obj(\theta) = \sum_n^m l(y_n, \hat{y}_n) + \sum_k^K \Omega(f_k)$$

Where the first term is the loss function and the second is the regularization parameter.

The boosting technique used in this paper is based on AdaBoost algorithm, which uses distributed weights to calculate and decide the label outputs. The training and testing operation of AdaBoost algorithm used n the proposed work is shown in Algorithm II.

Algorithm II

Input: Given a labeled dataset **D** with **N** instances

Training

(1) Choose the base classifier **C**;

(2) Set the initial weights $w_{1i} \in [0,1]$, $sum_{i=1}^{N} w_{1i} = 1$. Usually $w_{1i} = \frac{1}{N}$;

(3) For $k = 1 \rightarrow n$, create a training sample $\mathbf{D_k}$ from **D** using the distribution w_k;

(4) Train the base classifier **C** on $\mathbf{D_k}$ to build the classifier $\mathbf{C_k}$;

(5) Calculate the ensemble error $\epsilon_k = \sum_{j=1}^{N} w_{ki}$ if $\mathbf{C_k}$ missclassifies the **i**_th data point in **D**;

(6) If $\epsilon_k \in (0, 0.5)$, calculate $\beta_k = \frac{\epsilon_k}{1-\epsilon_k}$, and update the next weight

$$w_{k+1,i} = \begin{cases} w_{ki} \times \beta_k, & \text{if } \mathbf{C_k} \text{ classifies correctly the } \mathbf{i}\text{_th data point,} \\ w_{ki} & \text{otherwise} \end{cases} \quad (1)$$

(7) $w_{k+1,i}$ is normalized in order to be a distribution;

(8) For other values of ϵ_k, set $w_{ki} = \frac{1}{N}$ and continue;

(9) Return the set of classifiers $\mathbf{C_1}, \mathbf{C_2}, ..., \mathbf{C_n}$ and $\beta_1, \beta_2, ..., \beta_n$;

Testing

(1) For each object **x** in the testing dataset, classify **x** by all classifiers $\mathbf{C_1}, \mathbf{C_2}, ..., \mathbf{C_n}$;
(2) For each class **y** assigned to **x** by $\mathbf{C_k}$, calculate $\mu_y(x) = \sum_{C_k(x)=y} \ln(\frac{1}{\beta_k})$. The class with the maximum $\mu_y(x)$ is decided as the label of **x**;
(3) Repeat with all the data points in the testing set and return the outputs;

Voting

The voting classifier takes majority voting based on weights applied to the class or class probabilities and assigns a class label to a record based on a majority vote. In this case, the class that received the highest number of votes $N_c(y_t^n)$ will be chosen. represents the classifier. Here we predict the class label via majority voting of each classifier.

$\hat{y} = argmax\ (N_c(y_t^1), N_c(y_t^2), \ldots\ N_c(y_t^n))$

Machine Learning Models

The following machine-learning models will be employed for the proposed intrusion detection system:

Support Vector Machines (SVM): SVM is a powerful classification algorithm that aims to find the optimal hyperplane that separates different classes. A support vector machine represents a data set as points in space divided into categories by a distinct gap or line that stretches as far as feasible. The additional data points are now mapped into the same area and categorized into one of many categories based on which side of the line or separation they land on as shown in Fig. 3.1.

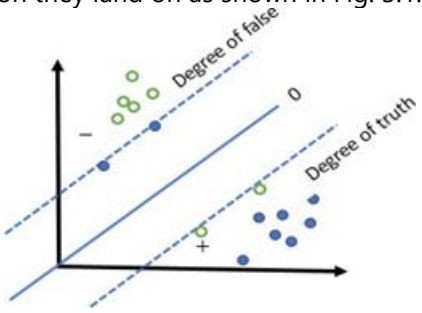

Fig. 3.1. Graph showing hyperplane

The equation for the linear hyperplane can be written as:

wTx + b = 0

The vector W represents the normal vector to the hyperplane. i.e. the direction perpendicular to the hyperplane. The parameter b in the equation represents the offset or distance of the hyperplane from the origin along the normal vector w.

Decision Tree: It can handle both numerical and categorical data, decision trees may be used for both classification and regression. As the tree grows, it splits down the data set into smaller and smaller sections or nodes. A decision tree's output contains decision and leaf nodes, each of which has two or more branches and reflects a choice. The root node is the highest node that corresponds to the best predictor. To build our classification tree we use Entropy (H)

H(S)= - $\sum_i^C PxlogPx$

S - Set of all instances

N - Number of distinct class values Px - Event probability

C – Number of classes

K-nearest neighbors (KNN): The K-nearest neighbors (KNN) method predicts which cluster a new data point will fall into using the approach of "feature similarity" or "nearest neighbors." Manhattan distance and Euclidean distance are the most commonly used to measure the closeness of the new data instance with the training data instances. In this work, we used the Euclidean distance measure for the k-NN algorithm. Euclidean distance d between data point x and data point y are calculated as follows:

$$d(x, y) = \sqrt{(\sum_{i=1}^{N} (xi - yi)^2)}$$

d is the distance between two points (x and y)

Store all the training data set P in pairs in the training phase as follows:

P = (yi , ci), i = 1, ..., n

where in the training dataset, yi is a training pattern, n is the number of training patterns and ci is its corresponding class.

Naive Bayes: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle. Every pair of features being classified is independent of each other. The assumptions made

by Naive Bayes are not generally correct in real-world situations. In- fact, the independence assumption is never correct but often works well in practice. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: The probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: The probability of the hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Random Forest Classifier: Random Forest is a popular ensemble learning algorithm that combines the predictions of multiple decision trees to make more accurate and robust predictions. It is a variant of the bagging ensemble technique applied to decision trees. If a dataset T contains examples from n classes, Gini index, Gini (T) is defined as:

Gini (T) = $1 - \sum_{j=1}^{n}(p_j)^2$

Where pj is the relative frequency of class j in T

K. **Performance Metrics**

The ensemble model performance would be evaluated using metrics like accuracy, precision, recall, and F1 score. Also, cross-validation was employed to ensure robustness of the model and putting into consideration the trade-offs, strengths, and weaknesses of each algorithm. The performance evaluation will be carried out based on the following metrics and presented follows:

Accuracy is the percentage of the correctly classified records out of the overall total records;

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

Sensitivity (recall) is the ratio of actually positive outcome predicted to the total number of the positive outcome that could have been predicted;

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

Precision is the proportion of the actually positive classifications out of all those classified positive;

$$\text{Precision} = \frac{TP}{TP+FP}$$

Where:

- **False negative (FN):** an outcome where the model incorrectly predicts negative class,
- **True Negative (TN):** an outcome where the model correctly predicts negative class,
- **True Positive (TP):** it means that the outcome where the model correctly predicts a positive class,
- **False Positive (FP):** an outcome where the model incorrectly predicts a positive class.

IV. PROPOSED DESIGN ARCHITECTURAL MODEL

An intrusion detection architectural model is presented in Fig. 4.1. The major components of the proposed system architecture are the dataset, Data pre-processing, feature selection, machine learning techniques, and ensemble techniques. The dataset used for training the model was obtained from Kaggle.com; CIC-DDoS2019, KDD-CUP 99, and CSE-CIC-IDS 2018. The dataset is split into training and testing datasets. The training dataset is used for model the training and parameter tuning, while the testing dataset is used to evaluate the final performance of the model.

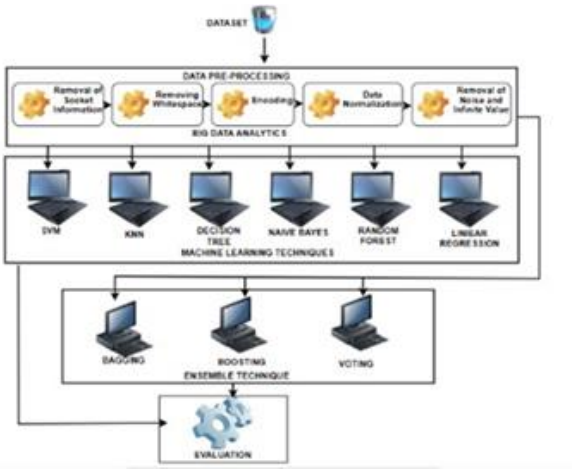

Fig. 4.1: Model Architecture

Preprocessing activities such as Removal of Socket Information, Removing Whitespace, Encoding, Data Normalization and Removal of Noise and Infinite Value will be performed on the dataset. These steps help in preparing the data for training the model. Support Vector Machine (SVM), Neural Network (NN), Random Forest (RF), Naïve Bayes (NB) Linear Regression (LR), and K- Nearest Neighbor which were Ensemble using Bagging, Boosting and Voting are the machine learning and ensemble techniques that will be employed for the model.

The proposed algorithm is trained on the preprocessed training dataset. After training, the results of individual techniques are evaluated to ascertain the technique that perform better. Finally, once the meta model is trained, it can be used to make predictions on new and unseen data. The testing dataset is passed through the trained model to obtain their predictions.

A. Proposed Intrusion Detection System

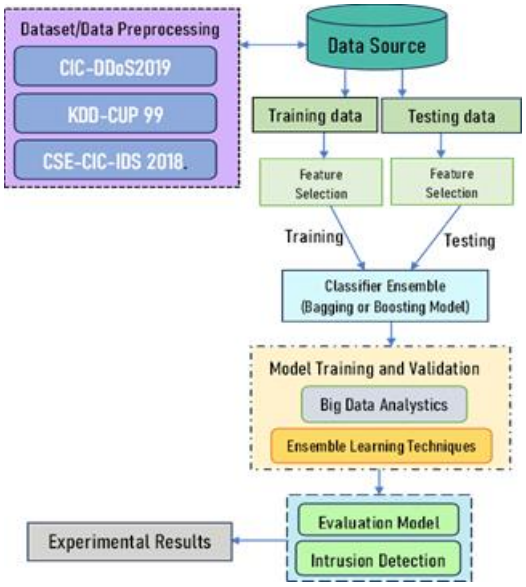

Fig. 4.2: Proposed Model Block Diagram

In this paper, the proposed Intrusion Detection System (IDS) model using Big Data Analytics and Ensemble Learning approach, employed three diverse datasets from Kaggle.com (CIC-DDoS2019, KDD-CUP 99, and CSE-CIC-IDS 2018). The proposed model block diagram is shown in Fig. 4. The proposed model was trained and tested with the preprocessed datasets to produce results for evaluation of the proposed system. In the proposed method, both the training and testing dataset were applied feature selection in data preprocessing to remove irrelevant features. The proposed model was implemented using pythons.

The proposed model block diagram for development of Intrusion Detection using Big Data Analytics and Ensemble learning techniques is shown in Fig. 4.2. The major components used in the proposed system design are the dataset. The following described the procedure deployed for the proposed experiment.

B. Dataset

The dataset used for training and evaluation are the CIC-DDoS2019, CSE-CIC-IDS 2018 and KDD-CUP 99dataset. It is a publicly available network intrusion detection dataset collected in a controlled environment. The dataset contains various features related to network traffic, including protocol types, service types, source and destination IP addresses, port numbers, and payload attributes. The dataset consists of both numerical and categorical features, with the target variable indicating the attack category.

C. Data Preprocessing

Data preprocessing is a crucial step in building an effective intrusion detection system (IDS) using the CIC-DDoS2019, CSE-CIC-IDS 2018 and KDD-CUP 99 dataset. This paper provides a comprehensive overview of the data preprocessing techniques employed to ensure the compatibility and reliability of the dataset for training and evaluation.

D. Data Cleaning

The initial step in the data preprocessing stage is to perform data cleaning to handle any missing or inconsistent values. Each feature is examined for missing values, and appropriate strategies are applied, such as imputation or removal of incomplete records. Any inconsistent or erroneous values are identified and resolved through techniques like data validation and outlier detection.

E. Feature Selection and Transformation

Feature selection plays a crucial role in data preprocessing, as it helps identify the most relevant features for training the IDS models. Prior domain knowledge and feature importance techniques (e.g., correlation analysis, feature ranking) are utilized to select the most informative features. Redundant or irrelevant features are eliminated to improve the model's efficiency and prevent overfitting. Feature transformation techniques, such as normalization or scaling, are applied to ensure that all features are on a similar scale and have a comparable impact on the models.

F. Handling Categorical Data

The CIC-DDoS2019, CSE-CIC-IDS 2018 and KDD-CUP
99 dataset contains categorical features, which need to be appropriately handled to make them compatible with machine learning algorithms. The Label Encoder class from the Scikit-learn library is utilized to convert the categorical values into numerical labels.

G. Encoding Target Variable

The target variable, representing the attack category, is also encoded to facilitate model training and evaluation. Similar to the categorical features, the Label Encoder class is used to encode the attack categories into numerical labels. This step ensures consistency in the representation of the target variable across the models and allows for accurate evaluation of the IDS performance.

H. Dataset Splitting

After preprocessing the dataset, it is split into training and testing sets to evaluate the IDS models. A commonly used split ratio of 80:20 is applied, where 80% of the data is used for training the models, and 20% is reserved for evaluating their performance.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The experiments were performed using python 3.11.7 software environment. The Python was installed on Intel CORE i7-6700HQ CPU @ 2.60GHz, with 64-bit Windows 10 operating system and has 8GB RAM and 1TB hard disk memory. The Python was installed over the Windows 10 operating system, which is on the laptop. Initially, the input data is pre-processed with data cleaning and standardization. In data cleaning, the corrupted, incorrectly formatted, incomplete, incorrect, or duplicate data are eliminated within the dataset. Then the data is converted into a simplified format to simplify the intrusion detection process.

The features are extracted with 5-fold cross validation of the random forest algorithm. On the other hand, data is grouped into clusters, and each instance is assigned a unique cluster ID. The extracted features and the cluster ID is given to the input of the ensemble classifier. Hence, the parameters are selected for enhancing the intrusion detection performance. By using this parameter optimal level of intrusion detection was attained with the proposed big data analytics and ensemble techniques.

A. Dataset Description

This research make use of three diverse datasets, our research aims to provide a comprehensive evaluation of the developed Intrusion Detection System, enabling it to detect and respond effectively to various network intrusions in real- world scenarios.

The datasets used are as follows:

- **CSE-CIC-IDS 2018:** The CSE-CIC-IDS 2018 dataset is a comprehensive collection of network traffic data, encompassing various types of intrusions and normal activities. It was created for the purpose of developing and evaluating Intrusion Detection Systems. The dataset includes different attack scenarios, such as Denial of Service (DoS), Port Scan, and Brute-Force attacks, among others, making it an ideal resource for assessing the IDS model's ability to handle diverse intrusion attempts. This data was compiled on February Fourteenth, 2018. This particular dataset has a majority of normal traffic in comparison to actual attacks.
- **DDoS2019:** The DDoS2019 dataset contains network traffic data related to Distributed Denial

of Service (DDoS) attacks that occurred in the year 2019. It comprises a wide range of network activities and anomalies caused by DDoS attacks. The dataset provides valuable insights into the characteristics of DDoS attacks, enabling the IDS model to recognize and respond effectively to such threats.

- **KDD-CUP 99:** The KDD-CUP 99 dataset is a widely used benchmark dataset for intrusion detection, created by MIT Lincoln Laboratory. It consists of network traffic data that spans multiple attack types and normal activities. The dataset was generated based on the DARPA Intrusion Detection Evaluation Program, making it a suitable choice for evaluating the IDS model's performance against various types of intrusions.

B. Feature Extraction

Feature extraction is a critical preprocessing step in machine learning, aimed at reducing the dimensionality of the dataset while retaining essential information. The feature extraction process was conducted on the three datasets: CSE-CIC-IDS2018, CIC-DDoS2019, and KDDCup99, each initially containing a large number of rows and features. The main objective was to extract a reduced set of 18 important features from each dataset. For the CSE- CIC-IDS2018 dataset, which contained 1,252,846 rows and 78 features, 18 significant features were identified and retained as shown in Fig.

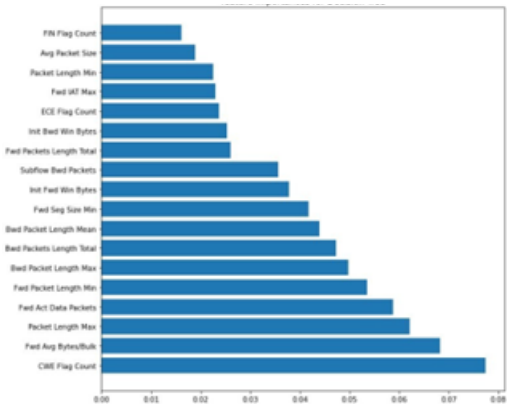

Fig. 5.1: Extracted Features for CSE-CIC-IDS2018 Dataset

5.1. Similarly, from the CIC-DDoS2019 dataset, comprising 125,170 rows and 78 features, 18 important features were also extracted as shown in Fig. 5.2. Additionally, the KDDCup99 dataset, with 494,020 rows and 42 features, was subjected to the feature extraction process, resulting in 18 relevant features being retained as shown in Fig. 4.3. These 18 features were carefully selected based on their ability to capture relevant patterns and relationships in the data, while also minimizing noise and redundant information.

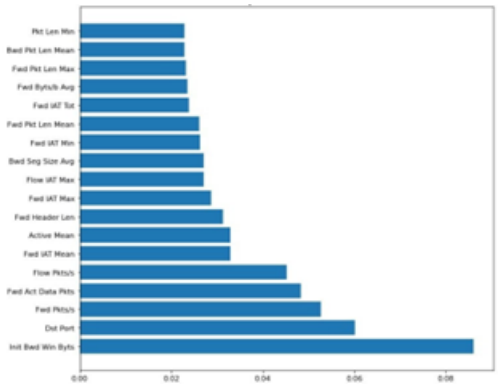

Fig. 5.2: Extracted Features for CIC-DDOS2019 Dataset

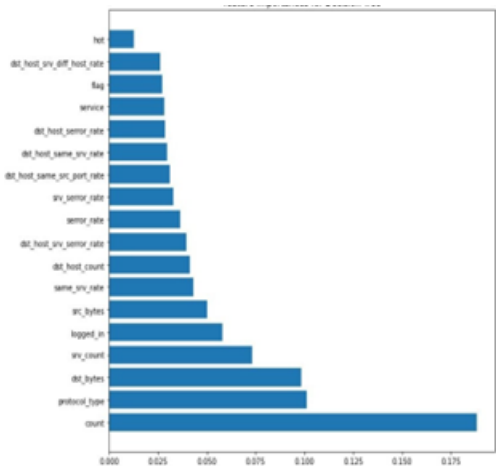

Fig.5.3. Extracted Features for KDDCU99 Dataset

C. Performance Analysis of Binary Classification Results

The performance of evaluation of the binary classification was done using individual dataset classifiers as follows:

1. Results of Binary Classification with CSE-CIC-IDS2018 Dataset

In the case of the CSE-CIC-IDS2018 dataset the

individual classifiers, Logistic Regression, KNearest Neighbors, Naïve Bayes, and Linear SVC demonstrated reasonably good results. The results on Table 5.1 show that K-Nearest Neighbors exhibited the highest accuracy (99.67%), precision (99.60%), and recall (99.73%). On the other hand, Naïve Bayes has the worst result with an accuracy of 90.87%, precision of 85.13%, and recall of 98.79%.

Table 5.1: Results of Binary Classification with CSE-CIC-IDS2018 Dataset

Classification with CIC-DDOS2019 Dataset	Accuracy	Precision	Recall
Logistic Regression	99.55	99.79	99.50
KNeighbors	99.79	99.92	99.75
GaussianNB	98.59	98.42	99.37
SVM	99.57	99.87	99.45
DecisionTree	99.97	100.00	99.96
RandomForest	99.95	99.87	99.92
Bagging	99.92	100.00	99.87
xgboosting	99.97	100.00	99.96
Voting	99.65	99.96	99.50

Table 5.2: Results of Binary Classification with CIC-DDOS2019 Dataset

Classification CSE-CIC-IDS2018	Accuracy	Precision	Recall
Logistic Regression	97.47	95.24	99.87
KNeighbors	99.67	99.60	99.73
GaussianNB	90.87	85.13	98.79
SVM	97.87	95.98	99.87
DecisionTree	99.93	100.00	99.87
RandomForest	99.93	100.00	99.87
Bagging	99.93	100.00	99.87
xgboosting	99.93	100.00	99.87
Voting	99.07	98.28	99.87

However, the ensemble learning methods proved to be more effective. Decision Tree, Random Forest, Bagging Ensemble, and XGBoost achieved exceptional performance, with accuracy, precision, and recall values of 99.93%, and 100% 99.87% respectively. These ensemble methods outperformed the individual classifiers significantly, showcasing their ability to effectively detect intrusions. The Voting Ensemble, while slightly less accurate (99.07%), still exhibited commendable precision (98.28%) and recall (99.87%). In overall, the ensemble learning approach demonstrated its superiority in handling the complexity of the CSE-CIC-IDS2018 dataset. Decision Tree, Random Forest, Bagging Ensemble, and XGBoost emerged as particularly powerful models for intrusion detection, boasting near-perfect accuracy, precision, and recall scores.

2. Results of Binary Classification with CIC-DDOS2019 Dataset

The results of the Intrusion Detection System (IDS) on the CIC-DDOS19 dataset indicating the performance of various classifiers and ensemble methods in detecting Distributed Denial of Service (DDoS) attacks is shown in Table 5.2. The evaluation metrics used for each model are accuracy, precision, and recall, which are measures of the model's ability to correctly classify normal and malicious network traffic.

On the Individual Classifiers, the Logistic Regression achieved high accuracy (99.55%), Precision (99.79%), and recall (99.50%), demonstrating its effectiveness in distinguishing between normal and DDoS traffic. K-Nearest Neighbors outperformed Logistic Regression with even higher accuracy (99.79%), precision (99.92%), and recall (99.75%), making it a robust classifier for identifying DDoS attacks. GaussianNB achieved respectable accuracy (98.59%), precision (98.42%), and recall (99.37%). Although not as high as the other classifiers, it still demonstrated a good ability to detect DDoS attacks. Linear SVC showed similar performance to Logistic Regression, with high accuracy (99.57%), precision (99.87%), and recall (99.45%).

The Ensemble Learning Classifiers using Decision Tree and Random Forest achieved nearly perfect accuracy (99.97% and 99.95%, respectively), precision (100.00%), and recall (99.96%). These

ensemble methods excelled in detecting DDoS attacks with minimal false positives or negatives. Bagging

Ensemble also performed impressively, with high accuracy (99.92%), precision (100.00%), and recall (99.87%). While the Voting Ensemble achieved lower accuracy (99.65%) compared to other ensemble methods but still exhibited good precision (99.96%) and recall (99.50%). The XGBoost demonstrated outstanding performance with nearly perfect accuracy (99.97%), precision (100.00%), and recall
(99.96%).

In overall, the results suggest that the ensemble learning methods, particularly Decision Tree Classifier, Random Forest, Bagging Ensemble, and XGBoost, are highly effective in detecting DDoS attacks on the CIC-DDOS19 dataset. These models achieved remarkable accuracy, precision, and recall, making them suitable techniques for building robust and accurate IDS solutions for DDoS attack detection in real-world cybersecurity applications.

3. Results of Binary Classification with KDDCU99 Dataset

The results of the Intrusion Detection System (IDS) on the KDD Cup 1999 (KDDCU99) dataset shown in Table 5.3 describe the performance of various classifiers and ensemble methods in detecting network intrusions. The evaluation metrics used are accuracy, precision, and recall, which provide insights into the models' ability to correctly classify normal and malicious network traffic.

Table 5.3: Results of Binary Classification with KDDCU99 Dataset

Classification with KDDCU99 Dataset	Accuracy	Precision	Recall
Logistic Regression	99.47	99.65	99.30
KNeighbors	99.75	99.89	99.61
GaussianNB	96.02	97.22	94.72
SVM	99.39	99.72	99.05
DecisionTree	99.93	99.96	99.89
RandomForest	99.96	100.00	99.93
Bagging	99.95	100.00	99.89
xgboosting	99.98	100.00	99.96
Voting	99.70	99.82	99.58

On the individual classifiers, the Logistic Regression achieved high accuracy (99.47%), precision (99.65%), and recall (99.30%). It demonstrated effective differentiation between normal and intrusive network traffic. K-Nearest Neighbors outperformed Logistic Regression with even higher accuracy (99.75%), precision (99.89%), and recall (99.61%), making it a strong classifier for detecting network intrusions. Naïve Bayes attained accuracy (96.02%), precision (97.22%), and recall (94.72%). While it achieved slightly lower values than the other classifiers, it could detect intrusions well. Linear SVC achieved high accuracy (99.39%), precision (99.72%), and recall (99.05%), indicating its competence in distinguishing between normal and intrusive network activities.

The ensemble learning classifiers exhibited remarkable performance. Decision Tree, Random Forest, Bagging Ensemble, and XGBoost demonstrated near-perfect accuracy, precision, and recall scores. Decision Tree achieved high accuracy (99.93%), precision (99.96%), and recall (99.89%). Random Forest achieved even higher accuracy (99.96%) with perfect precision (100.00%) and high recall (99.93%). Bagging Ensemble achieved high accuracy (99.95%), precision (100.00%), and recall (99.89%), showcasing its effectiveness in detecting network intrusions. XGBoost demonstrated outstanding performance with the highest accuracy (99.98%), perfect precision (100.00%), and high recall (99.96%). The Voting Ensemble achieved good accuracy (99.70%), precision (99.82%), and recall (99.58%). Although slightly lower than the individual ensemble methods, it still provided strong overall detection capability.

D. Comparison of Accuracy of Multiple Classifiers Results of the Three Datasets

The multi-classification performance of evaluation of the was done using the three proposed datasets (CSE-CICIDS2018, CIC-DDOS2019, and KDDCU99).
Table 5.4 shown the performance of the three datasets on the same classifier are compared and the strengths of each dataset lie in the effectiveness of XGBoost in detecting intrusions, while the

weaknesses lie in the limitations of Naïve Bayes in handling complex patterns. The three intrusion detection results on the CIC-DDOS2019, CSE-CIC-IDS2018, and KDDCU99 datasets demonstrate variations in classifier performances, strengths, and weaknesses.

Table 5.4: Comparison of Accuracy of Multiple Classifiers using Three Datasets (CSE-CICIDS2018,

	CSE-CIC-IDS2018	CIC-DDOS2019	KDDCU99
Logistic Regression	97.47	99.55	99.47
KNeighbors	99.67	99.79	99.75
GaussianNB	90.87	98.59	96.02
SVM	97.87	99.57	99.39
DecisionTree	99.93	99.97	99.93
RandomForest	99.93	99.95	99.96
Bagging	99.93	99.92	99.95
xgboosting	99.93	99.97	99.98
Voting	99.07	99.65	99.70

CIC-DDOS2019: The highest accuracy was achieved by XGBoost (99.97%), showcasing its robustness in detecting DDoS attacks. The ensemble methods (Random Forest, Bagging Ensemble, and Voting Ensemble) also displayed high accuracy, indicating the benefits of combining multiple classifiers. Naïve Bayes had the lowest accuracy (98.59%), suggesting limitations in handling complex patterns in the dataset.

CSE-CIC-IDS2018: KNN achieved the highest accuracy (99.67%), showcasing its competence in identifying network intrusions. Decision Tree, Random Forest, and Bagging Ensemble also demonstrated high accuracy, indicating their suitability for intrusion detection. Naïve Bayes had the lowest accuracy (90.87%), suggesting limitations in handling the dataset's complexity.

KDDCU99: XGBoost achieved the highest accuracy (99.98%), showcasing its effectiveness in detecting network intrusions. Random Forest, Bagging Ensemble, and KNN also demonstrated high accuracy, indicating their suitability for intrusion detection tasks. Naïve Bayes had the lowest accuracy (96.02%), suggesting limitations in handling the dataset's complexity.

In overall, XGBoost consistently achieved the highest accuracy across all three datasets, making it a strong technique for intrusion detection in various scenarios. The Ensemble methods, Random Forest, Bagging Ensemble, and Voting Ensemble, also demonstrated high accuracy on multiple datasets. Naïve Bayes consistently had the lowest accuracy, indicating that it may not be the best choice for complex intrusion detection tasks.

E. Analyses of Individual Model Performance on the CIC-DDoS2019 dataset using Confusion Matrix

With respect to the confusion matrices Shown in Fig. 5.4, Logistic Regression achieved a high number of true positives, correctly classifying 1369 instances as DDoS attacks. However, it also had a relatively low number of false positives (5) and false negatives (12), indicating that it occasionally misclassified normal traffic as attacks and vice versa. Nonetheless, the model's overall accuracy remains strong due to its high true positive rate.

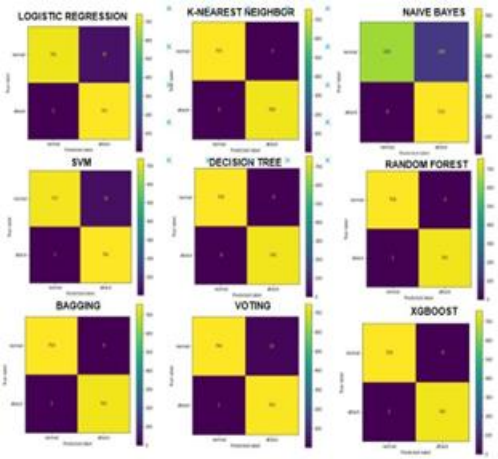

Fig. 5.4: Confusion Matrix of Individual Model on the CIC-DDoS2019 Dataset

K-Nearest Neighbor (KNN) performed well with a substantial number of true positives (1336). However, it had a higher number of false positives and false negatives compared to Logistic Regression. The higher false positive rate indicates

that KNN occasionally misclassifies normal traffic as DDoS attacks, which might be due to the sensitivity of the k-nearest neighbors' approach to the dataset's distribution. Naïve Bayes achieved a high number of true positives and true negatives, with a very low number of false positives (3) and false negatives (13). This indicates that the model has a good balance between precision and recall, effectively classifying DDoS attacks and normal traffic.

Support Vector Machine (SVM) demonstrated excellent performance with no false positives and only two false negatives. It accurately identified both DDoS attacks and normal traffic, making it a robust model for DDoS detection. Similar to SVM, Decision Tree achieved perfect classification, with no false positives and only two false negatives. Decision Trees are known for their ability to capture complex decision boundaries, which likely contributed to their outstanding performance.
Random Forest, being an ensemble of Decision Trees, also achieved perfect classifTreesn. It combines multiple Decision Trees predictions to improve accuracy and robustness, making it highly effective in DDoS attack detection. Similar to Random Forest, Bagging also achieved perfect classification. It leverages bootstrap aggregating to reduce variance and improve accuracy, which is evident in its excellent performance.

Voting achieved a high number of true positives but had a few false positives and false negatives. The presence of false positives and false negatives suggests that Voting might not always reach a unanimous decision when combining the predictions of its constituent models. XGBoost exhibited a high true positive rate and a very low false negative rate, making it particularly effective in identifying DDoS attacks. It achieved the lowest number of false negatives among all models, highlighting its strength in detecting positive instances.

In general, all of the individual models performed impressively on the CIC-DDoS2019 dataset as shown in Fig. 5.4. The Fig. showcasing their suitability for DDoS attack detection. SVM, Decision Tree, Random Forest, and Bagging achieved perfect classification, indicating their robustness and accuracy. Naïve Bayes, Logistic Regression, and KNN also exhibited competitive performance, with Naïve Bayes achieving a good balance between precision and recall. Voting and XGBoost emerged as the top-performing models, demonstrating high true positive rates and low false negatives, making them excellent choices for accurate DDoS attack detection in real-world scenarios. Comparative analysis of the training and test accuracy of all the on CIC-DDoS2019 dataset is shown in Fig. 5.5.

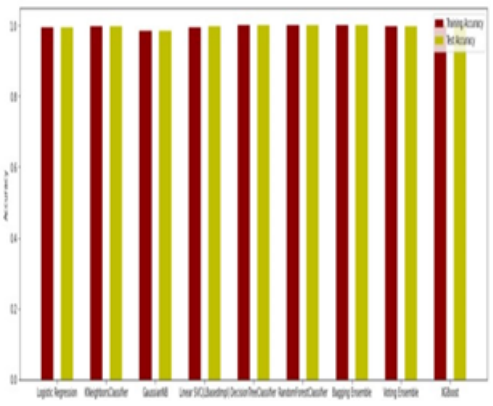

Fig. 5.5: Comparative Analysis of Training and Test Accuracy on CIC-DDoS2019 Dataset

F. Analyses of Individual Model Performance on the CSE-CIC-IDS2018 Dataset Using Confusion Matrix

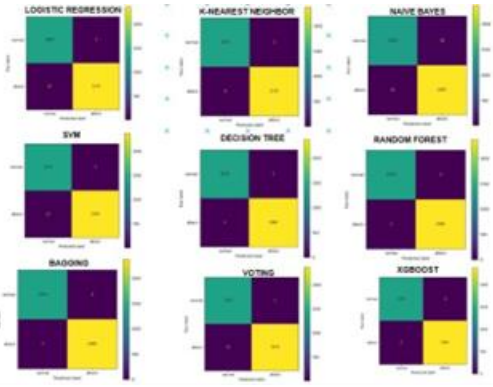

Fig. 5.6: Confusion Matrix of Individual Model on the CSE-CIC-IDS2018 Dataset

Fig. 5.6 shown the confusion matrices performance evaluation of each model on the CSE-CICIDS2018 dataset. Logistic Regression achieved high true positive and true negative rates, indicating its ability

to correctly classify both normal and attack traffic instances. With only a small number of false positives and false negatives, this model demonstrated good overall accuracy and precision.

KNN showed identical results to Logistic Regression, correctly classifying both positive and negative instances. The model's similarity to Logistic Regression suggests that the choice of k in the k- Nearest Neighbors algorithm might not significantly impact performance on this dataset. Naïve Bayes demonstrated a higher false positive rate compared to Logistic Regression and KNN. It misclassified 128 instances of normal traffic as attacks, leading to a decrease in precision. However, the model still achieved a relatively high true positive rate, indicating its effectiveness in identifying attack instances.

SVM achieved a relatively higher false positive rate compared to Logistic Regression and KNN, but it still demonstrated good performance in detecting attacks. The model's high true positive and true negative rates make it a reliable option for attack detection. Decision Tree achieved perfect classification, with no false positives or false negatives. This indicates that the model was able to perfectly separate attack and normal traffic instances, making it highly accurate on this dataset. Random Forest also performed exceptionally well, with only one false negative. The ensemble model's aggregation of multiple Decision Trees likely improved generalization and robustness, resulting in high accuracy. Bagging achieved nearly perfect classification, with only one false negative. The model's robustness and low variance contributed to its high accuracy.

Voting performed well, with a slightly higher false positive rate compared to the ensemble models. However, it still achieved high true positive and true negative rates, making it effective in attack detection. XGBoost exhibited similar results to Random Forest and Bagging, with only one false negative. It demonstrated high accuracy and robustness in detecting attacks.

In general, all individual models showed good performance on the CSE-CIC-IDS2018 dataset. Decision Tree, Random Forest, Bagging, and XGBoost achieved perfect or near-perfect classification, indicating their superior accuracy and robustness. Logistic Regression and KNN demonstrated consistent and accurate results, performing similarly to the ensemble models. Naïve Bayes and SVM achieved competitive performance but with slightly higher false positive rates. Voting also performed well, but its false positive rate was higher compared to the top-performing models.

The ensemble models (Decision Tree, Random Forest, Bagging, and XGBoost) outperformed individual algorithms in terms of accuracy and robustness. The choice of the best model should consider specific requirements, such as interpretability, computational resources, and the desired balance between false positives and false negatives. Decision Tree and XGBoost stand out as top performers for their perfect classification, while Random Forest and Bagging offer a trade-off between interpretability and accuracy through ensemble aggregation. Ultimately, the model selection should be tailored to the specific needs and constraints of the application scenario. Comparative analysis of the training and test accuracy of all the classifiers on CSE-CIC-IDS2018 dataset is shown in Fig. 5.7.

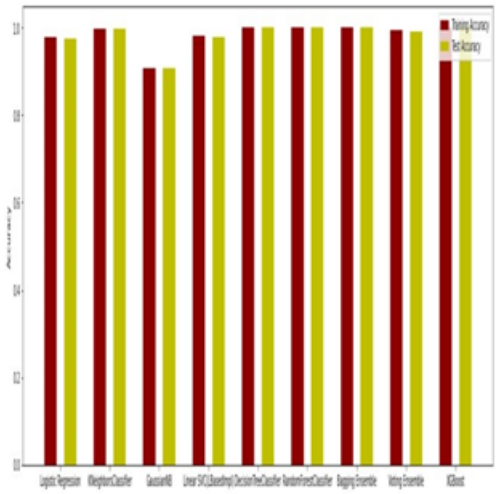

Fig. 5.7: Comparative Analysis of Training and Test Accuracy on CSE-CIC-IDS2018 Dataset

G. Analyses of Individual Model Performance on the KDDCU99 dataset using Confusion Matrix

In analyzing the performance of individual models on the KDDCUP99 dataset based on the confusion matrices is shown in Fig. 5.8. Logistic Regression achieved a high true positive rate, correctly classifying 2850 instances as positive (anomalous) out of 2870 actual positives. The model had a low false positive rate (10), indicating that it misclassified only 10 normal instances as anomalous. With a few false negatives (20), while KNN also performed well with a high true positive rate and a low false positive rate. It correctly identified 2857 instances as positive, with only 3 false positives. However, the model had a few false negatives (11), suggesting that it missed detecting a small number of anomalous instances.

Naïve Bayes exhibited a higher false positive rate (77) compared to Logistic regression and KNN models. Although it correctly identified 2783 anomalous instances, it misclassified 77 normal instances as anomalies. The model also had a higher false negative rate (150), indicating it missed some actual anomalous instances. SVM achieved a good true positive rate, accurately classifying 2853 instances as positive. However, it had a higher false positive rate (7) than some other models, indicating it misclassified 7 normal instances as anomalies. The model also had a higher false negative rate (29), suggesting it missed some actual anomalies.

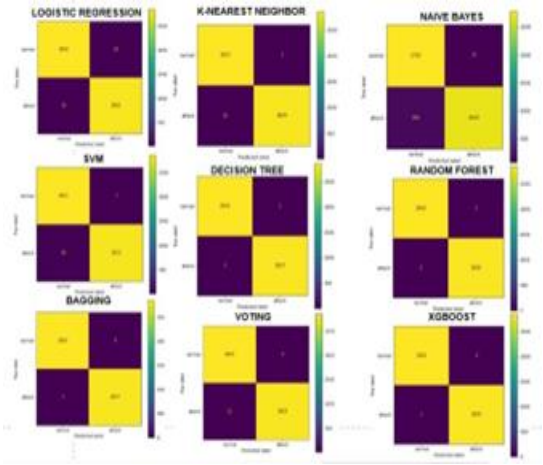

Fig. 5.8: Confusion Matrix of Individual Model on the KDDCU99 Dataset

Decision Tree performed impressively with only one false positive and three false negatives. It achieved near-perfect classification, showcasing its effectiveness in distinguishing normal and anomalous traffic. Random Forest, Bagging, and XGBoost achieved a perfect true positive rate and no false positives. However, Bagging had a slightly higher false negative rate (3) compared to Randon Forest with two and XGBoost with one false negative rate. Nevertheless, the models still demonstrated high accuracy and robustness. Voting achieved a high true positive rate, correctly identifying 2855 anomalous instances. It had a few false positives (5) and false negatives (12), showing slightly lower performance compared to some ensemble models.

All models demonstrated good performance on the KDDCup99 dataset, with some models achieving near-perfect classification. Decision Tree, Random Forest, Bagging, and XGBoost stood out for their superior ability to distinguish between normal and anomalous traffic, while Logistic Regression and KNN also showed commendable results. However, Naïve Bayes and SVM exhibited slightly higher false positive and false negative rates. Comparative analysis of the training and test accuracy of all the on KDDCup99 dataset is shown in Fig. 5.9.

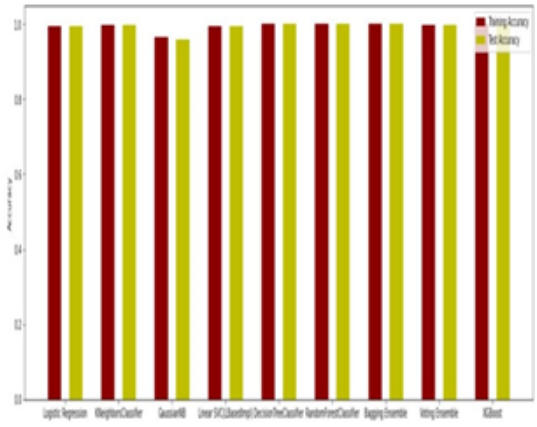

Fig. 5.9: Comparative Analysis of Training and Test Accuracy on KDDCU99 Dataset

VI. CONCLUSION

In this paper, application of big data analytics and ensemble techniques for network intrusion

detection were explored. The comprehensive evaluation of machine learning models' performance on the CSE- CIC-IDS2018, CICDDoS2019, and KDDCup99 datasets brings significant advancements to the field of network security. Particularly, the XGBoost exhibits exceptional accuracy across all three datasets, achieving 99.93%, 99.97%, and 99.98% accuracy in CSE-CIC-IDS2018, CIC-DDoS2019, and KDDCup99, respectively.

This remarkable consistency establishes XGBoost as a robust technique for intrusion detection in various scenarios. However, Naïve Bayes consistently displays the lowest accuracy across CSE-CIC-IDS2018, achieving 90.87%, and CIC-DDoS2019 and KDDCup99, achieving 98.59% and 96.02%, respectively. These results suggest that Naïve Bayes might not be the most suitable choice for complex intrusion detection tasks, indicating the need for caution in its application.

The study significantly contributes to our understanding of model effectiveness and the advantages of ensemble techniques in intrusion detection. By providing a comparative analysis of model performance, the research equips researchers and practitioners with valuable guidance for selecting appropriate models and designing robust intrusion detection systems. Moreover, the findings hold the potential to advance the state of-the-art in network security, fostering further research and innovation in the field.

REFERENCES

1. Luo B. and Xia J. (2014). "A Novel Intrusion Detection System Based on Feature Generation with Visualization Strategy," Expert Systems with Applications, vol. 41, no. 9, (pp. 4139-4147).
2. Marir N., Wang H., Feng G., Li B., and Jia M. (2018). Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark," IEEE Access, vol.6, (pp. 59657-59671).
3. Carley, K.M. (2020). Social cybersecurity: an emerging science. Computational and mathematical organization theory 26(4), (pp. 365–381).
4. Wang H., Gu J., and Wang S. (2017). An Effective Intrusion Detection Framework Based on SVM with Feature Augmentation, Knowledge-Based Systems, vol. 136, (pp. 130-139).
5. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. (2020). Machine learning-based iot-botnet attack detection with sequential architecture. Sensors 20(16), 4372.
6. Liao H., Lin C., Lin Y., and Tung K. (2013). Intrusion Detection System: A Comprehensive Review, Journal of Network and Computer Applications, vol. 36, no. 1, (pp. 16-24).
7. Dias, L.F. and Correia, M. (2020). Big data analytics for intrusion detection: an overview. Handbook of research on machine and deep learning applications for cyber security, (pp.292-316).
8. Faker, O.; Dogdu, E.: Intrusion detection using big data and deep learning techniques (2019). In: Proceedings of the 2019 ACM Southeast Conference. ser. ACM SE '19. New York, NY, USA: Association for Computing Machinery, (pp. 86-93).
9. Moustafa, N.; Creech, G.; Slay, J. (2017). Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models. In Data Analytics and Decision Support for Cybersecurity; Springer: Cham, Switzerland, 2017; (pp. 127–156).
10. Kamel Y., Jemili F., and Meddeb R. (2022). Ensemble learning based big data classification for intrusion detection. In Intelligent Systems Design and Applications: 19th International Conference on Intelligent Systems Design and Applications (ISDA 2022). Springer.
11. Krishnaveni, S., Sivamohan, S., and Sridhar, S.S. (2021). Enceinte feature selection and classi¦cation through ensemble method for network intrusion detection on cloud computing. Cluster Comput 24, (pp. 1761–1779), https://doi.org/10.1007/s10586-020- 03222-y.
12. Mhawi, D.N.; Aldallal, A.; and Hassan, S. (2022). Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems. Symmetry 2022,

14, 1461. https://doi.org/10.3390/sym14071461

13. Fitni, Q.R.S. and Ramli, K. (2020). Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), (pp. 118–124).
14. Pokharel, P.; Pokhrel, R. and Sigdel, S.(2020). Intrusion detection system based on hybrid classifier and user profile enhancement techniques. Int. Workshop Big Data Inf. Secure. 2020, (pp. 137–144).
15. Marir N., Wang H., Feng G., Li B., and Jia M. (2028). "Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark," IEEE Access, vol. 6, (pp. 59657-59671).
16. Mirza, A.H (2018). Computer network intrusion detection using various classifiers and ensemble learning. In: 2018 26th Signal processing and communications applications conference (SIU), (pp. 1– 4).
17. Rajagopal, S.; Kundapur, P.P. Hareesha, K.S. (2020). A stacking ensemble for network intrusion detection using heterogeneous datasets. Secur. Commun. Netw. 2020, 4586875.
18. El Arass, M., Souissi, N. (2019). Smart siem: From big data logs and events to smart data alerts. Int. J. Innov. Technol. Explore. Eng 8(8), (pp. 3186–3191).
19. Yang, J., Xiang, F., Li, R., Zhang, L., Yang, X., Jiang, S., Zhang, H., Wang, D. and Liu, X., (2022). Intelligent bridge management via big data knowledge engineering. Automation in Construction, 135, p.104118.
20. Kumari, A. and Mehta, A.K., (2020); A hybrid intrusion detection system based on decision tree and support vector machine. In 2020 IEEE 5th International conference on computing communication and automation (ICCCA) (pp. 396-400).
21. Kilincer, I.F., Ertam, F. and Sengur, A., (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks, 188, p.107840.
22. Fitni, Q.R.S. and Ramli, K., (2020). Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT) (pp. 118-124). IEEE.
23. Jiang J., Yu Q., Yu M., Li G., Chen J., Liu K., and Huang W. (2018). ALDD: A Hybrid Traffic-User Behavior Detection Method for Application Layer DDoS, in Proceedings of 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, New York, (pp. 1565-1569).
24. Mokbal, F.M., Wang, D., Osman, M., Yang, P. and Alsamhi, S.H., 2022. An efficient intrusion detection framework based on embedding feature selection and ensemble learning technique. Int. Arab J. Inf. Technol., 19(2), pp.237-248.
25. Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M. and Abuzneid, A., (2019). Features dimensionality reduction approaches for machine learning based network intrusion detection. Electronics, 8(3), p.322.
26. Vijayanan R., Devaraj D., and Kannapiran B. (2018). Intrusion Detection System for Wireless Mesh Network Using Multiple Support Vector Machine Classifiers with Genetic-AlgorithmBased Feature Selection, Computers and Security, vol. 77, (pp. 304- 314).
27. Abbas, A., Khan, M.A., Latif, S., Ajaz, M., Shah, A.A. and Ahmad, J., 2021. A new ensemble-based intrusion detection system for internet of things. Arabian Journal for Science and Engineering, (pp.1-15).
28. Ustebay S., Turgut Z., and Aydin M. (2019). Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier, in Proceedings of International Congress on Big Data, Deep

Learning and Fighting Cyber Terrorism, Ankara (pp. 71-76).

29. Aksu, D., Üstebay, S., Aydin, M.A. and Atmaca, T. (2018). Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In Computer and Information Sciences: 32nd International Symposium, ISCIS 2018, Held at the 24th IFIP World Computer Congress, Poznan, Poland, Sept. 20-21, Springer International Publishing, (pp. 141-149).
30. Singh R., Kumar H., and Singla R., (2015). An Intrusion Detection System Using Network Traffic Profiling and Online Sequential Extreme Learning Machine, Expert Systems with Applications, vol. 42, no. 22, (pp. 8609-8624).
31. Tabash M., Allah M., and Tawfik B., (2020). Intrusion Detection Model Using Naive Bayes and Deep Learning Technique, The International Arab Journal of Information Technology, vol. 17, no. 2, (pp. 215- 224).
32. Abiodun, O.I.; Abiodun, E.O.; Alawida, M.; Alkhawaldeh, R.S.; Arshad, H., (2021). A review on the security of the internet of things: challenges and solutions. Wireless Pers. Commun. https://doi.org/10.1007/s11277-021-08348-9.