INTERNATIONAL JOURNAL OF LATEST TECHNOLOGY IN ENGINEERING, MANAGEMENT & APPLIED SCIENCE (IJLTEMAS)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIII, Issue X, October 2024

Harnessing Machine Learning for Adaptive Signature-Based Network Intrusion Detection: A Simulation-Driven Approach

***[1]Peter Paul Issah, [2]Ransford Ganyo**

[1]*Department of Computer Science, Kwame Nkrumah University of Science and Technology, Ghana*

[2]*Department of Mathematics, University of Cape Coast, Ghana*

DOI : https://doi.org/10.51583/IJLTEMAS.2024.131022

Received: 02 November 2024; Accepted: 07 November 2024; Published: 17 November 2024

Abstract: Network security is essential for data sharing on the internet. Traditional methods such as firewalls cannot detect fragmented packets and are often outmaneuvered by increasingly sophisticated attackers, resulting in productivity losses, financial damage, and reputational harm. This study investigates the use of machine learning (ML) models in developing effective intrusion detection systems (IDS) using signature-based methods. The research leverages the UNSW-NB15 dataset and compares four ML algorithms: K-Nearest Neighbor (KNN), Random Forest (RF), Bayesian Network (Bayes Net), and Decision Tree (J48), with feature reduction applied using Principal Component Analysis (PCA) to improve efficiency. The models were built and evaluated using the WEKA platform, with 10-fold cross-validation applied to assess accuracy, precision, recall, and F-measure. Results show that J48 significantly outperforms the other algorithms in terms of overall accuracy, while Bayes Net produces the least accurate results. These findings underscore the efficacy of J48 and Random Forest in signature-based IDS for network security.

Keywords: Machine Learning, Intrusion Detection System, Signature-based IDS, UNSW-NB15, J48, Random Forest

I. Introduction

The widespread adoption of the internet has revolutionized communication, commerce, and the global economy. However, the rapid digital transformation has also escalated the prevalence and sophistication of cybercrimes. With the rise of online banking, e-commerce, and cloud computing, businesses and individuals alike face significant risks of data theft, financial fraud, and service disruptions. A successful cyberattack can lead to catastrophic consequences, including economic loss, reputational damage, and legal liabilities. As a result, securing networks and safeguarding sensitive data has become a paramount concern for organizations worldwide.

Traditional cyber security defenses, such as firewalls, encryption protocols, and access control mechanisms, have long been the first line of defense against cyber threats. While these methods provide critical protection, they are increasingly inadequate in the face of evolving threats. As cybercriminals continually develop new strategies to bypass defenses, particularly through sophisticated intrusion techniques, it becomes clear that reactive measures alone are insufficient [1]. Firewalls, for example, may fail to detect fragmented packets or detect anomalies hidden within seemingly legitimate traffic. This creates the need for more proactive solutions like Intrusion Detection Systems (IDS).

Intrusion Detection Systems (IDS) play a vital role in modern cyber security architectures. IDS monitor network traffic, identify suspicious activities, and alert administrators when potential intrusions are detected [2]. By providing real-time monitoring, IDS can help mitigate the damage caused by cyber attacks. IDS are generally classified into two types: signature-based IDS and anomaly-based IDS. Signature-based IDS rely on predefined patterns or attack signatures, while anomaly-based IDS detect deviations from established norms in network traffic. While anomaly-based systems are effective at detecting unknown threats, signature-based systems are highly accurate in identifying known attacks [3].

Despite the advancements in IDS technology, cyberattacks continue to become more complex. Modern attacks, such as advanced persistent threats (APT), zero-day exploits, and distributed denial-of-service (DDoS) attacks, challenge traditional IDS methods, necessitating more sophisticated detection techniques [4]. Machine Learning (ML) has emerged as a promising solution for enhancing the effectiveness of IDS by allowing systems to learn from historical data and improve their detection capabilities over time.

Problem Statement

Cyber security remains a significant global challenge, and the complexity of modern networks and the volume of traffic make it difficult to effectively monitor and protect systems from intrusions. The limitations of traditional signature-based IDS include their inability to detect new or variant attacks not previously cataloged [5]. As attackers frequently modify their methods, signature-based systems require constant updates to remain effective. Anomaly-based systems, while capable of detecting novel threats, often suffer from high false-positive rates. Thus, there is a need for a more balanced approach—one that can provide high accuracy without being overwhelmed by false alerts.

Machine learning techniques have been successfully applied to IDS to address these challenges by automating the detection of both known and novel intrusions. However, the optimal ML algorithm for signature-based IDS has yet to be determined. Different algorithms—such as Decision Trees, Random Forests, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN)—have been employed to enhance IDS performance, each with varying levels of success depending on the dataset and context [6]. Therefore, a comprehensive comparison of ML algorithms is needed to identify the most effective approach for improving signature-based IDS.

Intrusion Detection with Machine Learning

In recent years, the application of ML techniques to IDS has gained traction due to their ability to analyze vast amounts of network data and identify complex patterns indicative of malicious activity [7]. ML models can be trained on historical attack data to detect known threats, while also generalizing from this data to identify potential new threats. Key ML techniques used in IDS include decision trees, random forests, KNN, support vector machines, and neural networks. These algorithms enable IDS to adapt to changing attack patterns and improve over time without relying solely on static attack signatures.

Among these techniques, Decision Trees (e.g., J48) and Random Forests are widely favored for their interpretability and high accuracy in classification tasks, making them particularly useful for signature-based IDS [8]. Bayesian Networks provide a probabilistic framework that can model the uncertainty and interdependencies between various network events, while K-Nearest Neighbors (KNN) offers a straightforward, instance-based approach to classification. Principal Component Analysis (PCA) is often employed as a preprocessing step to reduce the dimensionality of network traffic data, improving the efficiency and speed of ML models without sacrificing accuracy [9].

II. Methodology

Dataset Selection

This study employs the UNSW-NB15 dataset for both training and testing. The UNSW-NB15 dataset is widely recognized for its realistic representation of contemporary network traffic and attack behaviors, making it ideal for evaluating network intrusion detection systems. Developed at the Australian Centre for Cyber Security, this dataset integrates a mixture of real-world network traffic and synthetic attack behaviors, allowing for robust testing of machine learning algorithms in intrusion detection scenarios [10]. Compared to earlier datasets like KDDCup99, the UNSW-NB15 dataset offers a more accurate reflection of modern network environments. [6] noted that the KDDCup99 dataset, while pioneering, is outdated and lacks the complexity of contemporary cyber threats, such as advanced persistent threats (APT) and more sophisticated denial-of-service (DoS) attacks. UNSW-NB15 addresses these limitations by capturing detailed traffic features—44 attributes in total—that span various attack types, including Fuzzers, Backdoors, DoS, Exploits, Reconnaissance, Shellcode, and Worms.

The UNSW-NB15 dataset’s attributes are divided into flow-based, basic, content-based, and time-based categories, providing a rich set of features for machine learning models. This variety of features allows for a comprehensive analysis of network traffic, distinguishing between normal and malicious behavior. Given its extensive scope and accuracy, the dataset has been widely adopted in recent intrusion detection research. Several other datasets, such as NSL-KDD, CAIDA, and WSN-DS have also been employed in IDS studies [12]. However, UNSW-NB15 is preferred for its robustness in capturing contemporary attack behaviors, making it a more suitable choice for evaluating machine learning-based intrusion detection systems.

By using UNSW-NB15, this study ensures that the machine learning models are tested against realistic traffic patterns and modern intrusion techniques, providing valuable insights into the performance of various algorithms under real-world conditions.

Software Tools and Hardware Platform

The Weka Workbench/Software

A collection of machine learning algorithms for data mining tasks. Weka stands for Waikato Environment for Knowledge Analysis, and it was developed at the University of Waikato in New Zealand. The algorithms can be used directly on a dataset or used from your own Java code. Weka offers tools and functions for pre-processing data, feature selection, regression, clustering, association rules, classification, and data visualization. Use the following URL to obtain the UNSW NB15 dataset from GitHub in ARFF format: https://github.com/InitRoot/UNSW NB15/blob/master/UNSW NB15.zip.

Using Weka, certain benefits are achieved:

- Definition of the class attributes that categorize the collection of instances into the correct classes.
- Investigation into any imbalances in the chosen data collection and possible solutions.
- Utilizing a classifier method to aid in learning.
- Choosing a testing strategy to gauge how well the chosen algorithm will function.

Hardware Platform

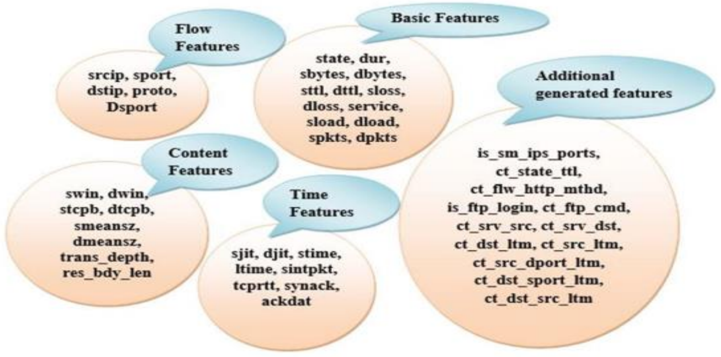
This experiment will be carried out on a Windows 11 laptop with an Intel(R) Core (TM) i7 processor running at 3.20GHz and 32GB of installed primary memory.

Data Information

Network intrusion dataset UNSW-NB15 is sometimes referred to as UNSQ-NB15. It was developed by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to provide a hybrid of real-world, contemporary normal behaviors and synthetic, modern attack behaviors [10]. The UNSW NB15 dataset consists of 175341 unique data sets with 44 unique attributes. 175,341 records make up the training set, while 82,332 records from the attack and normal types make up the testing set.

Flow features, basic features, content features, temporal features, and further generated features are the several categories into which the features of the UNSW NB15 dataset are divided. The dataset is shown in broad strokes in the Figure below. Backdoors, Denial of Service (DoS), Fuzzers, Worms, Reconnaissance, Generic, Exploits, Analysis, and Shellcode are the nine different assaults in this dataset.

Figure 1: Description of the features of the UNSW NB15 dataset [13].


Flow Features
srcip, sport, dstip, proto, Dsport
Basic Features
state, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, service, sload, dload, spkts, dpkts
Additional generated features
is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm
Content Features
swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len
Time Features
sjit, djit, stime, ltime, sintpkt, tcprtt, synack, ackdat

Data Preprocessing

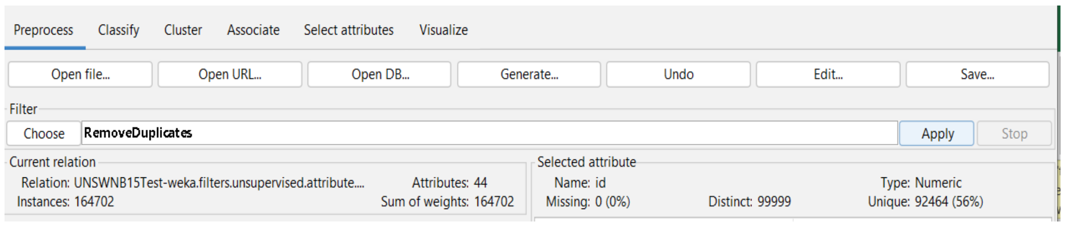
This step is the next in the process of detecting intrusions and any harmful activity. This stage includes data cleansing, randomization, and feature/dimensionality reduction. Building a machine learning model requires the preparation of the data. This is because a machine learning method's performance is influenced by how effectively the dataset is prepared and structured [14]. The obtained dataset (the UNSW NB15 dataset) has undergone preprocessing to ensure that the data is clean. Normalization, randomization, and dimension reduction using Principal Component Analysis (PCA) are all carried out as part of the dataset's preprocessing.

Data Cleaning (Standardize, Randomize, Normalize)

Data cleaning entails locating and improving, replacing, or eliminating irrelevant, false, and incomplete data. You will find numerous functions to utilize in cleaning the dataset if you select unsupervised in the Filter section of the Preprocess tab. As noted in the section, this project used a few of them.

1. Mean values are used to fill in any missing data. Choose ‘Replace Missing Values’ from the unsupervised folder of the Filter section under the Preprocess tab.
2. Remove Duplicates
3. The mean values are substituted for some infinity values in the two columns Flow Bytes and Flow Pkts /s.
4. The Time Stamp columns should be removed because they serve no purpose.

Figure 2: The picture above shows duplicate values been removed through the data cleaning process



Filter: RemoveDuplicates

Current relation
Relation: UNSWNB15Test-weka.filters.unsupervised.attribute.... Attributes: 44
Instances: 164702 Sum of weights: 164702

Selected attribute
Name: id Type: Numeric
Missing: 0 (0%) Distinct: 99999 Unique: 92464 (56%)

Data Standardization: By setting the mean and standard deviation to 0 and 1, respectively, data standardization is employed to guarantee that the feature value ranges are homogeneous. Values above the mean change from negative to positive, and vice versa.

Data normalization involves scaling and shifting real-valued numerical properties or values into the range between 0 and 1 [15].

Data randomization is a widely used method that can improve the accuracy of research data and reduce fraud.

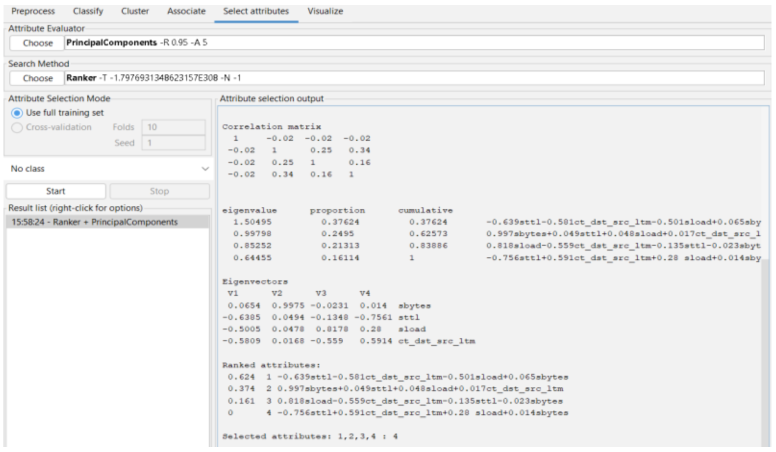
Dimensionality Reduction using Principal Component Analysis

The feature extraction stage is another name for this phase. The features that will feed into the machine learning models for network intrusion are selected at this point. [16] used a general feature and a subset of features as the foundation for the feature selection. The roles chosen to establish the identification of the feature, including whether it was common, distinctive, labeled, etc.

By splitting up a big set of associated features into a smaller set of less correlated features called principal components, principal component analysis (PCA) is used to reduce the dimensionality of datasets while retaining the majority of the data from the original dataset. Because smaller datasets are simpler to process, dimensionality reduction trades off accuracy for simplicity. The accuracy of a classification model may decline with a reduced number of features. Thus, the project used data cleansing techniques, but it was careful to avoid abusing their predicted accuracies by using them excessively.

The following steps comprise PCA. The first step is to standardize the values of the dataset's features using the WEKA platform's default standardization. In WEKA, PCA is accessible from the preprocess panel's unsupervised folder. PCA employs a rank filter that ranks the features from highest to lowest. To choose features independently of any predictor, the filter approach depends on the general properties of the training data [17]. Using Weka, the filter approach was applied to the dataset.

Figure 3: PCA feature reduction in WEKA.



Attribute Evaluator: PrincipalComponents -R 0.95 -A 5
Search Method: Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode: Use full training set; Cross-validation Folds 10 Seed 1
No class
Result list (right-click for options): 15:58:24 - Ranker + PrincipalComponents

Attribute selection output

```
Correlation matrix
  1      -0.02  -0.02  -0.02
 -0.02   1       0.25   0.34
 -0.02   0.25   1       0.16
 -0.02   0.34   0.16   1


eigenvalue      proportion      cumulative
  1.50495         0.37624         0.37624       -0.639sttl-0.581ct_dst_src_ltm-0.501sload+0.065sby
  0.99798         0.2495          0.62573       0.997sbytes+0.049sttl+0.048sload+0.017ct_dst_src_l
  0.85252         0.21313         0.83886       0.818sload-0.559ct_dst_src_ltm-0.135sttl-0.023sbyt
  0.64455         0.16114         1             -0.756sttl+0.591ct_dst_src_ltm+0.28 sload+0.014sby

Eigenvectors
 V1       V2       V3       V4
 0.0654  0.9975 -0.0231  0.014  sbytes
-0.6385  0.0494 -0.1348 -0.7561 sttl
-0.5005  0.0478  0.8178  0.28   sload
-0.5809  0.0168 -0.559   0.5914 ct_dst_src_ltm

Ranked attributes:
 0.624  1 -0.639sttl-0.581ct_dst_src_ltm-0.501sload+0.065sbytes
 0.374  2 0.997sbytes+0.049sttl+0.048sload+0.017ct_dst_src_ltm
 0.161  3 0.818sload-0.559ct_dst_src_ltm-0.135sttl-0.023sbytes
 0      4 -0.756sttl+0.591ct_dst_src_ltm+0.28 sload+0.014sbytes

Selected attributes: 1,2,3,4 : 4
```

Source: Experimental Analysis

Model Building and Testing

The UNSW NB15 dataset is used in this step to train and test the models, and only a small number of features have been chosen through dimensionality reduction and data cleaning. The performance of the classifiers is evaluated during training and testing using ten-fold cross-validation. The results were also evaluated and contrasted with the cross-validation using an 80:20 split. In this project's Weka Workbench, the UNSW-NB15 dataset is uploaded in order to develop models for classification modeling issues and to ascertain the accessibility of various attacks. The network dataset was then examined for various irregularities. Using performance evaluation metrics, it was also determined how effective these algorithms were.

Machine Learning

Machine learning (ML), a subfield of artificial intelligence (AI), enables learning without explicit programming. Its foundations are in computational mathematics, data mining, and statistics. A dataset is used to train machine learning (ML) algorithms, which then use those patterns to predict the future. IDSs frequently make use of ML. ML algorithms can be broken down into three groups: reinforcement learning, unsupervised learning, and supervised learning. Only supervised learning techniques are used in this research project. In the project, the terms "classifiers," "techniques," and "algorithms" are all used synonymously to denote the same thing.

Unsupervised learning is a machine learning technique that identifies hidden patterns in unlabeled datasets, reducing dimensionality and grouping issues, and is used in Hidden Markov models and K-means clustering. Reinforcement learning enables models to learn from their experiences and behaviors, similar to how humans learn from their environment. It's commonly used in resource management, video games, and business simulation. Supervised learning is a machine learning method that creates models to categorize unobserved data using explained patterns. It's the most commonly used method and is influenced by data quality and model configuration parameters. It's used in tasks like task-oriented learning, spam classification, face recognition, and advertisement popularity.

For the purpose of intrusion detection, the UNSW-NB15 dataset was analyzed using supervised machine learning techniques, including K-Nearest Neighbors (KNN), Decision Tree (J48), Random Forest (RF), and BayesNet. This study chose KNN as a non-parametric model due to its ability to provide non-linear solutions and outperform linear regression when signal-to-noise ratio is high. Neural networks require more hyperparameter adjustment and more training data, while Random Forest is less expensive and doesn't require a GPU for training, making it more efficient. The J48 method is superior for categorical and continuous data analysis, providing more precise findings than logistic regression. Decision Trees segment regions into smaller portions, addressing issues like missing values, pruning, and error rates. J48 is useful for predicting data structure and explaining patterns, unlike neural networks that are black boxes. Bayesian networks infer causal links.

K-Nearest Neighbor (KNN) Algorithm

The K-Nearest Neighbor method is one of the algorithms used to detect intrusions (KNN). The K-nearest Neighbor Classification Algorithm is a theoretically developed, extremely simple data mining algorithm. The KNN ML method takes into account the nearest neighbors. KNN can be used to solve problems involving classification and regression. Since the entire dataset is used during the testing stage, it is sometimes referred to as instance-based learning or lazy learning. The learning and classification algorithm KNN, is sample-based. The k-nearest neighbor classification algorithm assumes that a sample belongs to a category if the majority of its K-nearest neighbor samples also fall into that category [19]. The closest criterion might be the Euclidean distance of the feature vector, and the term ‘nearest neighbor’ refers to a single or multiple feature vector that is used to describe the sample.

As a result, testing is expensive and slow. KNN makes use of neighbors' majority votes and feature similarity. The prediction result received the most votes among the classes. KNN has drawbacks including high storage needs, computational complexity, and inability to handle missing values.

Random Forest (RF) Algorithm

An ML classifier called Random Forest (RF) uses numerous decision trees on various dataset subsets and averages the outcomes to increase prediction accuracy. A condition is compared with one or more properties of the input data at each treenode. To get the findings, RF gathers predictions from various decision trees rather than relying solely on one. Each tree casts a vote for a specific class, and the class receiving the most votes is the projected class. On unbalanced datasets, RF performs admirably, and there aren't many classification mistakes.

J48 Decision Tree Algorithm

Based on different attribute values of the available data, a predictive machine learning model known as the J48 Decision tree selects the target value of a new sample. In a decision tree, the internal nodes stand in for the distinguishing characteristics, the branches between the nodes show potential values for these characteristics in the observed samples, and the terminal nodes show the dependent variable's final value (classification). To categorize a new object using the J48 technique, a decision tree based on the attribute values of the available training data must first be built. Thus, as soon as it makes touch with a collection of items, the characteristic that most obviously sets the other cases apart is detected (the training set). Since it can correctly identify the data, it is said that this characteristic offers the greatest information gain. Now, out of all the potential values for this feature, if there is one for which there is no ambiguity, meaning that all data instances falling under its type have the same value for the goal variable, then we end that branch and assign it the target value.

Bayesian Network (Bayes Net) Algorithm

A probabilistic model called a Bayesian Network (Bayes Net) uses a graphical layout to resolve difficult issues. It offers information on a field where each node corresponds to a collection of random variables and each edge to a statistical link between those variables. Additionally, each node's related random variables are connected to a conditional probability distribution. Conditions or states might be part of the random variable. Probabilistic causal links are represented using the Bayes Net system.

System Implementation

The UNSW-NB15 dataset is used as the working dataset in the Network Intrusion Detection System project, which seek to assist in the detection of intrusion in a network. The Weka machine learning workbench is used to train four models: K-Nearest Neighbor (KNN), Random Forest (RF), J48, and Bayes Network. These models are developed and tested using a contemporary network dataset. The choice of the Weka Simulation tool to be utilized for this purpose is the first stage in the modeling of the network intrusion detection system models.

Performance Evaluation Metrics

The performance of the machine learning algorithms in the detection of anomalies is assessed using fundamental performance scoring criteria, such as precision and recall. ML models operate on the tenet of helpful feedback. Models are created, reviewed using metrics, and adjusted as needed to increase performance. Regression, classification, clustering, and ranking all use different metrics. The performance criteria that were utilized in this study to rate the ML classifiers for intrusion detection are described below. The confusion matrix's elements are used to evaluate the evaluation metrics. The confusion matrix is a N * N matrix, where N is the total number of classes.

The **precision ratio** is the sum of the true positives and false positives.

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

where false positive (FP) is the number of times benign conditions were mistakenly classified as attacks and true positive (TP) is the number of attacks that were accurately identified as attacks.

Recall is the proportion of true positive responses to the total of false negative and true positive responses.

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

where false negative (FN) is the number of incorrect classifications of an attack as benign.

The percentage of incidents in a dataset that were correctly classified as an attack or benign is known as **accuracy**.

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

where true negative (TN) is the number of correct classifications of benign as benign.

The recall and precision are combined to form the weighted harmonic mean, or **F-Measure.**

$$F - Measure = \frac{2TP}{2TP + FP + FN}$$

The amount of time needed to train and test the classification model is called **execution time.**

Accuracy, precision, recall, and F-measure are all reported as percentages in the performance evaluation.

Classification Model

Simulating and Modelling ML Algorithms in Weka

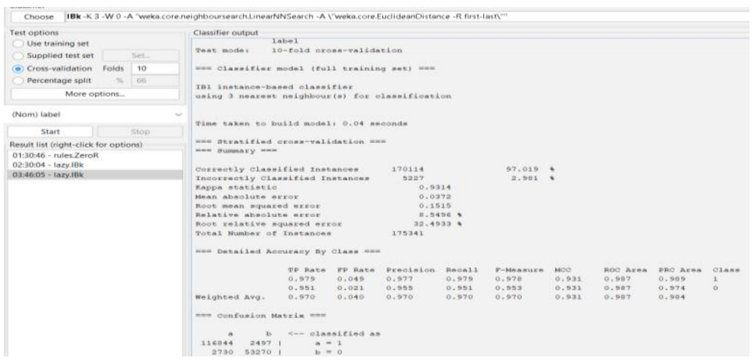
Here, we employ the classification algorithms in Weka; cross-validation with ten folds is the test option applied to all techniques. It repeats the entire procedure ten times, calculates the rule using the entire dataset, and outputs the outcome. The Zero R classifier serves as the baseline for four classification techniques, forecasting the predominant category. It provides a reliable performance estimate and has a 68.06 percent accuracy rate by class. The true baseline was tested using the UNSW-NB15 dataset, with accurate classifications of 68.06 percent and incorrect classifications at 31.94 percent.

Model Built with the K-Nearest Neighbour Algorithm (Lazy)

In the Weka tool, the KNN method is referred to as IBK, which stands for instance-based. As a result, it is known as the instance-based model, where k is the number of knowledge instances. It constructs the model in 0.03 seconds using the stratified 10-fold cross-validation technique.

KNN had a K value of 3, correctly classifying instances (170114) at 97.02 percent. The time required to develop the model grows as the number of k-values climbs as well. More accurately classified instances (accuracy), better TP, FP, and Recall results from higher k-values, but lower Precision. According to interpretation, accuracy in each class is 97.02 percent. The table below shows the recorded data for the confusion matrix, True Positive, False Positive, Precision, Recall, F-Measure, MCC, ROC Area, PRC Area, and Class Rates.

Figure 6: Performance of KNN on the UNSW_NB15 dataset



IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Cross-validation Folds 10

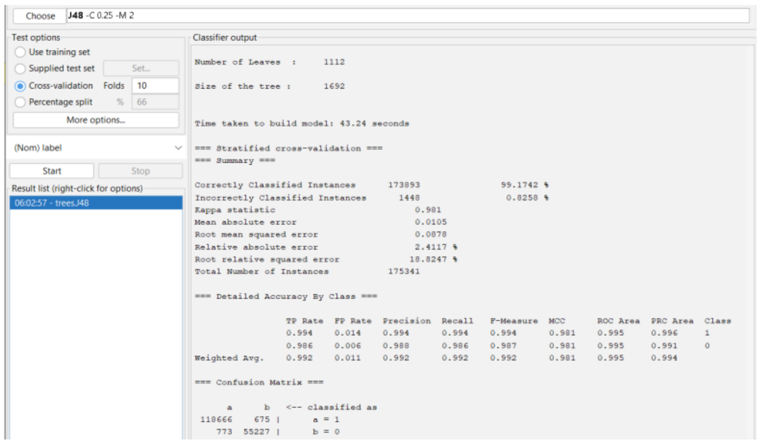
(Nom) label

```
              label
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification


Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      170114              97.019 %
Incorrectly Classified Instances      5227               2.981 %
Kappa statistic                          0.9314
Mean absolute error                      0.0372
Root mean squared error                  0.1515
Relative absolute error                  8.5496 %
Root relative squared error             32.4933 %
Total Number of Instances           175341

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.979    0.049    0.977      0.979    0.978      0.931    0.987     0.989     1
                 0.951    0.021    0.955      0.951    0.953      0.931    0.987     0.974     0
Weighted Avg.    0.970    0.040    0.970      0.970    0.970      0.931    0.987     0.984

=== Confusion Matrix ===

      a      b   <-- classified as
 116844   2497 |     a = 1
   2730  53270 |     b = 0
```

k-value =3

Model Built with The J48 Algorithm (Trees)

Figure 7: Performance of J48 on the UNSW_NB15 dataset



J48 -C 0.25 -M 2

Cross-validation Folds 10

(Nom) label

06:02:57 - trees.J48

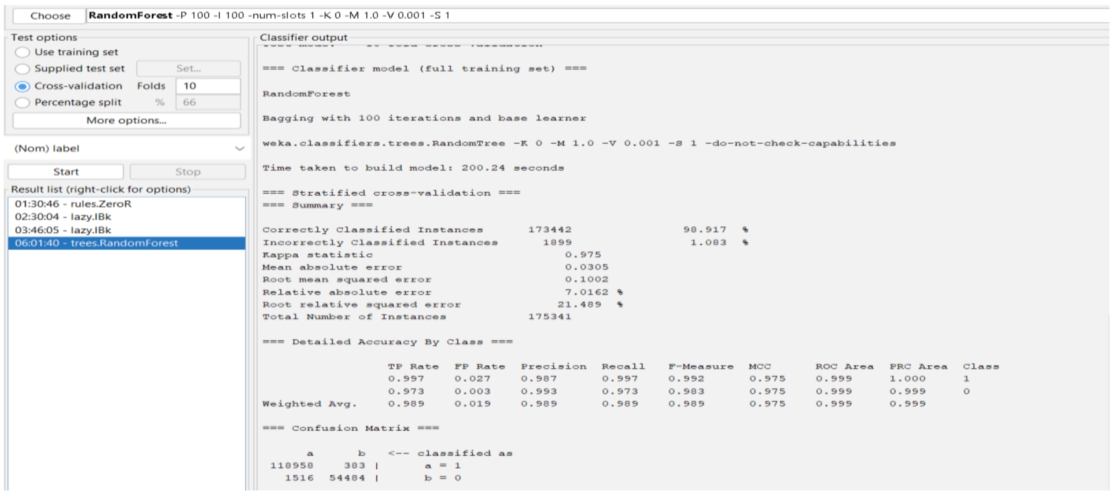
```
Number of Leaves  :     1112

Size of the tree :      1692


Time taken to build model: 43.24 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      173893               99.1742 %
Incorrectly Classified Instances      1448                0.8258 %
Kappa statistic                          0.981
Mean absolute error                      0.0105
Root mean squared error                  0.0878
Relative absolute error                  2.4117 %
Root relative squared error             18.8247 %
Total Number of Instances           175341

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.994    0.014    0.994      0.994    0.994      0.981    0.995     0.996     1
                 0.986    0.006    0.988      0.986    0.987      0.981    0.995     0.991     0
Weighted Avg.    0.992    0.011    0.992      0.992    0.992      0.981    0.995     0.994

=== Confusion Matrix ===

      a      b   <-- classified as
 118666    675 |      a = 1
    773  55227 |      b = 0
```

The decision tree algorithm took 43.24 seconds to generate the j48 classifier model. It categorized occurrences properly in 173893 cases with a 99.17 percent accuracy rate and inaccurately in 1448 cases with a 0.83 percent accuracy rate. This might also mean that the accuracy for each class is 99.17 percent. As illustrated in the diagram, the confusion matrix, True positive, False positive, Precision, Recall, F-Measure, ROC Area, PRC Area, and Class Rates were all reported.

Model Built with The Random Forest Algorithm (Trees)

Building the model for Random Forest bagging in Weka takes 200.24 seconds with 100 iterations and a base learner. 1899 cases were wrongly classified at 1.08 percent, while successfully classified instances (173442) had 98.92 percent. As illustrated in the diagram below, the confusion matrix, True positive, False positive, Precision, Recall, F-Measure, ROC Area, PRC Area, and Class Rates were all reported.

Figure 8: Performance of RF on the UNSW NB15 dataset



RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Cross-validation Folds 10

(Nom) label

01:30:46 - rules.ZeroR
02:30:04 - lazy.IBk
03:46:05 - lazy.IBk
06:01:40 - trees.RandomForest

```
=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 200.24 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      173442               98.917  %
Incorrectly Classified Instances      1899                1.083  %
Kappa statistic                          0.975
Mean absolute error                      0.0305
Root mean squared error                  0.1002
Relative absolute error                  7.0162 %
Root relative squared error             21.489  %
Total Number of Instances           175341

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.997    0.027    0.987      0.997    0.992      0.975    0.999     1.000     1
                 0.973    0.003    0.993      0.973    0.983      0.975    0.999     0.999     0
Weighted Avg.    0.989    0.019    0.989      0.989    0.989      0.975    0.999     0.999

=== Confusion Matrix ===

      a      b   <-- classified as
 118958    383 |      a = 1
   1516  54484 |      b = 0
```

Model Built With the Bayes network Algorithm (Bayes)

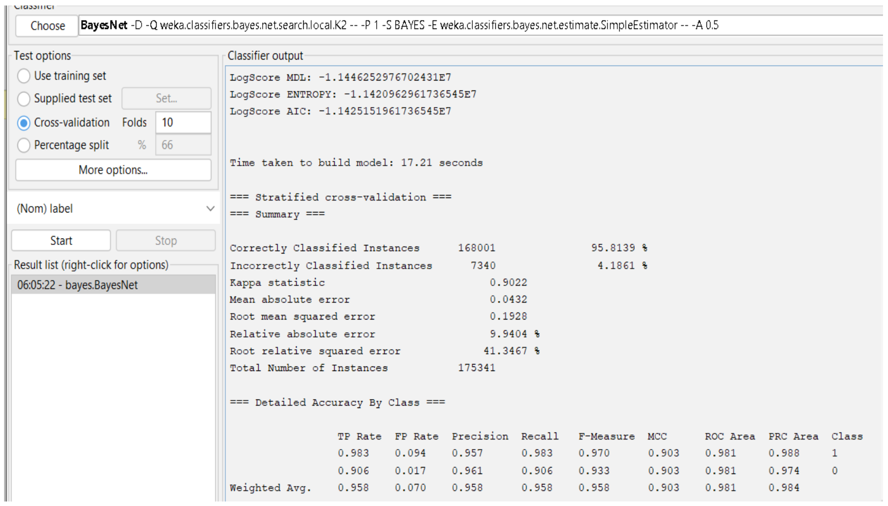
Bayes Net using the same stratified cross-validation spent 17.21 seconds in building the model. It has correctly classified instances (168001) at 95.81% and incorrectly classified instances (7340) at 4.19%. This can also be interpreted that accuracy by class is 95.81%. The confusion matrix,

True positive, False positive, Precision, Recall, F-Measure, ROC Area, PRC Area, and Class rates were recorded as shown in the diagram below.

Model Built with The Bayes Network Algorithm (Bayes)

Using the same stratified cross-validation, BayesNet built the model in 17.21 seconds. Instances correctly classified (168001) have a 95.81 percent accuracy rate, whereas those mistakenly classified (7340) have a 4.19 percent accuracy rate. This can also be read as 95.81 percent accuracy by class. As illustrated in the diagram below, the confusion matrix, True positive, False positive, Precision, Recall, F-Measure, ROC Area, PRC Area, and Class Rates were all reported.

Figure 9: Performance of Bayes Net on the UNSW_NB15 dataset



Choose: BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5

Test options: Use training set / Supplied test set / [✓ Cross-validation] Folds 10 / Percentage split % 66

(Nom) label

Result list (right-click for options): 06:05:22 - bayes.BayesNet

Classifier output

```
LogScore MDL: -1.1446252976702431E7
LogScore ENTROPY: -1.1420962961736545E7
LogScore AIC: -1.1425151961736545E7


Time taken to build model: 17.21 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      168001               95.8139 %
Incorrectly Classified Instances      7340                4.1861 %
Kappa statistic                          0.9022
Mean absolute error                      0.0432
Root mean squared error                  0.1928
Relative absolute error                  9.9404 %
Root relative squared error             41.3467 %
Total Number of Instances           175341

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.983    0.094    0.957      0.983    0.970      0.903    0.981     0.988     1
                 0.906    0.017    0.961      0.906    0.933      0.903    0.981     0.974     0
Weighted Avg.    0.958    0.070    0.958      0.958    0.958      0.903    0.981     0.984
```

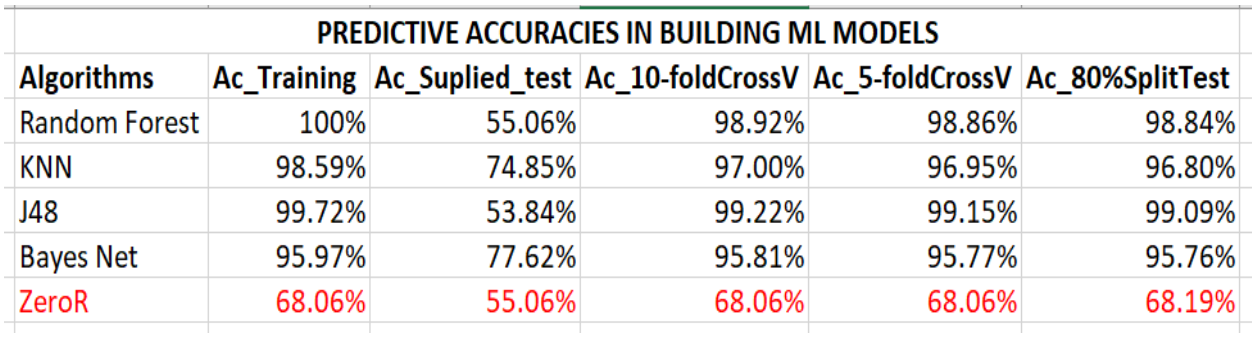
Source: Experimental Analysis 2022

Cross-Validation

The preferred approach used for this project is cross-validation. It is a system for determining how a factual dissection will ultimately add up to a separate dataset. It is primarily used in situations where the goal is known, and one wants to make an educated guess about how precisely a prophetic model would function in real-world applications. It is typical to use 10-fold cross-validation. In stratified K-fold cross-validation, the folds are selected so that the mean response value is essentially the same across all of the folds.

III. Results and Discussions

Figure 11: Predictive Accuracies of the ML Models



PREDICTIVE ACCURACIES IN BUILDING ML MODELS

Algorithms	Ac_Training	Ac_Suplied_test	Ac_10-foldCrossV	Ac_5-foldCrossV	Ac_80%SplitTest
Random Forest	100%	55.06%	98.92%	98.86%	98.84%
KNN	98.59%	74.85%	97.00%	96.95%	96.80%
J48	99.72%	53.84%	99.22%	99.15%	99.09%
Bayes Net	95.97%	77.62%	95.81%	95.77%	95.76%
ZeroR	68.06%	55.06%	68.06%	68.06%	68.19%

Source: Experimental Analysis

The table above shows the accuracy results for the training set, supplied test set, 10-fold cross-validation, 5-fold cross-validation, and 80 percent split test sets. Although technically not a component of the models created, ZeroR was simply utilized as a baseline benchmark and has been demonstrated to produce workable results with accuracies above 50%. All algorithms in the training test performed well, which aids in the development of more precise predictive models. To provide a conclusive, objective performance measure of the entire model construction process, the Supplied test set option was also available. Compared to training and other methods, its results were noticeably less impressive, but it nevertheless served its intended role in the model-building process.

Additionally, the 10-fold and 5-fold validation processes were used in the cross-validation process. It is clear from both perspectives that the model produces significantly more accurate results when it is repeated ten times rather than five. Actually, only 10-fold was used in the project; however, 5-fold was added for comparisons solely to drive a point home.

It was also noted that the 80/20 split, which represents the training set and test set, did not perform poorly at all. Only a few decimals separate 10-fold cross-validation from the 80% mark. It was also clear that although the 5-fold cross-validation yielded lesser results than the 10-fold, it still outperformed the 80 percent split.

As a result, it is possible to conclude that the percentage split does not produce as accurate predictive models as the 10-fold cross-validation. As a result, it is frequently used throughout this model building.

Table 4: Correct and Incorrectly Classified Instances

Technique		Results (Accuracy)	
		Correctly Classified Instances	Incorrectly Classified Instances
1.	K-Nearest Neighbour (KNN)	97.02%	2.98%
2.	Random Forest (RF)	98.91%	1.08%
3.	BaiyesNet	95.81%	4.19%
4.	J48	99.17%	0.83%

Source: Experimental Analysis

J48 correctly identified the majority of occurrences, while RF, the KNN, and BayesNet correctly identified the minority. Here, it is concluded that using 10-fold cross-validation, J48 and RF outperformed the other classifiers in terms of accuracy, precision, recall, and F-measure. However, the sluggish KNN was the fastest in terms of execution time or model building time, clocking in at 0.04 s, while RF was the slowest, taking 200.24 s.

Confusion Matrix

An instrument for visualizing supervised learning is a confusion matrix (in unsupervised learning it is called a matching matrix). An overview of the number of instances a categorization model properly or erroneously predicted.

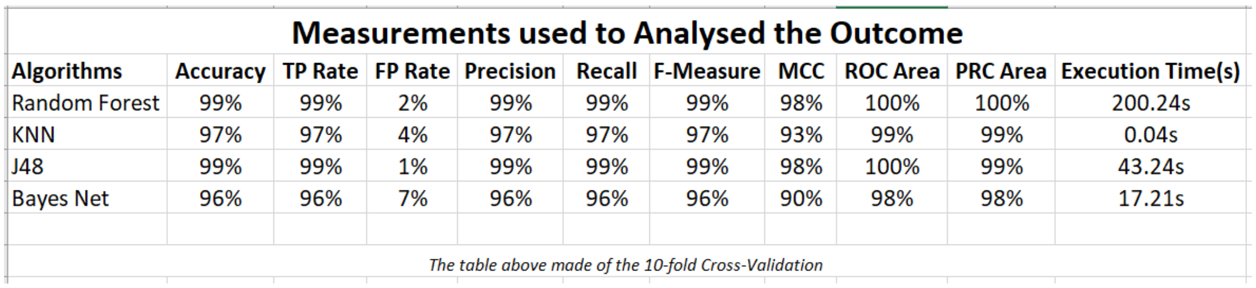
Overall Evaluation

According to the analysis of the results in the table below, the network intrusion model with the J48 algorithm outperformed all the other models with a 99.17 percent predictive accuracy, followed by the Random Forest, the KNN algorithm, the BayesNet, and the Logistic Regression with 98.91 percent, 97.02 percent, 95.81 percent, and 93.76 percent, respectively. Another inference that can be made from the findings of the models is that they are suitable for use in any intrusion detection techniques. This is supported by the fact that the True positive, precision recall, f-measure, ROC area, and PRC area all produced good positive results.

Additionally, the MCC's possible values range from -1 to +1. An ideal model has a score of +1, whereas an imperfect model has a score of -1. In addition, when percentages are converted to decimals, all of the MCC values in the table have positive values, which is another argument in favor of calling these models better. One of the main benefits of MCC is its quality because it makes it simple to interpret. Instead, the Matthews correlation coefficient (MCC) is a more dependable statistical measure that only yields a high score if the prediction performed well in each of the four categories of the confusion matrix (true positives, false negatives, true negatives, and false positives), proportionally to the size of the dataset's positive and negative elements.

Another inference from the results is that RF, one of the better at detecting accuracy, takes a long time to create its model, taking 200.24 seconds, whereas KNN generated its model in 0.04 seconds. The least accurate prediction model (BayesNet) ran in 17.21 seconds, whereas J48 executed in a respectable 43.24 seconds.

Figure 12: Performance of the classifiers under cross-validation



Measurements used to Analysed the Outcome

Algorithms	Accuracy	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Execution Time(s)
Random Forest	99%	99%	2%	99%	99%	99%	98%	100%	100%	200.24s
KNN	97%	97%	4%	97%	97%	97%	93%	99%	99%	0.04s
J48	99%	99%	1%	99%	99%	99%	98%	100%	99%	43.24s
Bayes Net	96%	96%	7%	96%	96%	96%	90%	98%	98%	17.21s

The table above made of the 10-fold Cross-Validation

Source: Experimental Analysis

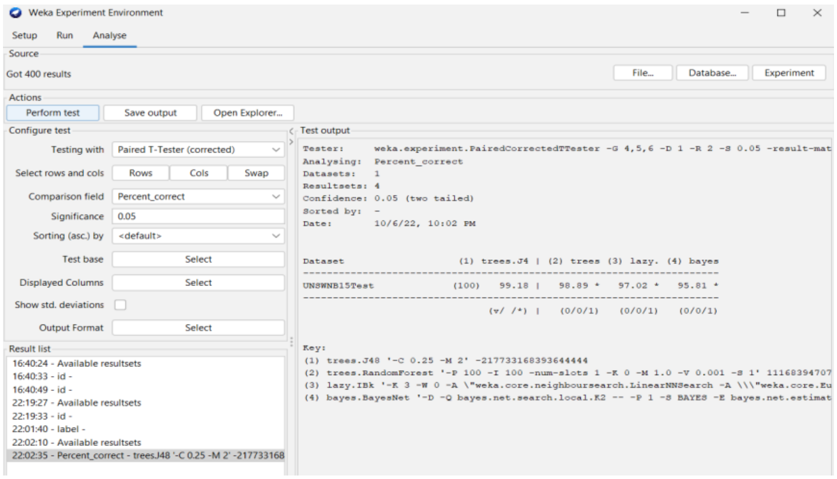
Comparing Classifiers Using the Weka Experimenter

The Weka ML tool comprises five applications, as was already described earlier, however only the Explorer was used throughout the project to generate models. The inability to make two or more classifiers run simultaneously is one drawback of using the explorer. The use of two datasets simultaneously is also prohibited.

Another program that could be able to fix these explorers' flaws is the Weka Experimenter. While the experimenter can use as many classifiers or datasets as necessary, the explorer is limited to evaluating models separately. In addition to the dataset that was used, the four ML classifiers were therefore loaded into the experimenter in this stage.

The classifiers are loaded, run against the dataset, and then analyzed. Additionally provided below is the performance evaluation table for the classifiers. As with the explorer, there is no need to assess each classifier's performance separately.

Figure 14: The experimenter results evaluation table



Got 400 results

Testing with: Paired T-Tester (corrected)
Comparison field: Percent_correct
Significance: 0.05
Sorting (asc.) by: <default>
Show std. deviations: ☐

```
16:40:24 - Available resultsets
16:40:33 - id -
16:40:49 - id -
22:19:27 - Available resultsets
22:19:33 - id -
22:01:40 - label -
22:02:10 - Available resultsets
22:02:35 - Percent_correct - trees.J48 '-C 0.25 -M 2' -217733168
```

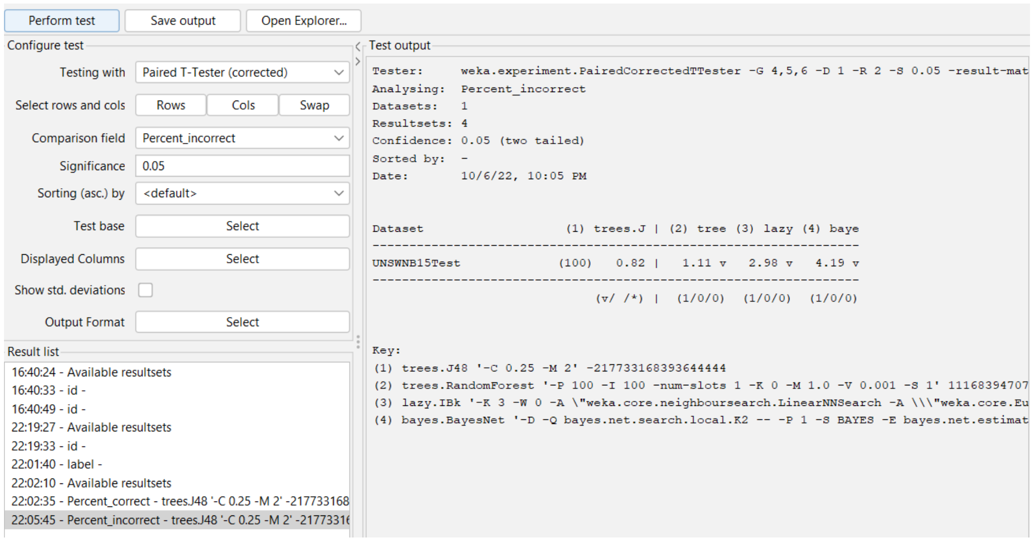
```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-mat
Analysing:  Percent_correct
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       10/6/22, 10:02 PM


Dataset                   (1) trees.J4 | (2) trees (3) lazy. (4) bayes
----------------------------------------------------------------------
UNSWNB15Test              (100)   99.18 |   98.89 *   97.02 *   95.81 *
----------------------------------------------------------------------
                                (v/ /*) |   (0/0/1)   (0/0/1)   (0/0/1)


Key:
(1) trees.J48 '-C 0.25 -M 2' -217733168393644444
(2) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 11168394707
(3) lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.Eu
(4) bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimat
```

Source: Experimental Analysis

The Weka Experimenter allows users to contrast the results of various classifiers on a single or a collection of datasets. According to the aforementioned data, the J48 model had a forecast accuracy of 99.18 percent, followed by RF with 98.89 percent, KNN with 97.02 percent, and BayesNet with a final accuracy of 95.81 percent. The data were evaluated using the 10-fold cross-validation method to test all classifiers.

By choosing either precision, recall, f-measure, or any other evaluation metric under consideration, the Experimenter comparison field allows for the creation of several comparisons. Accuracy (percent correct) was chosen for this project's comparisons, as can be seen above. Also compared were occurrences that were incorrectly classified.

Figure 15: Percentage of incorrectly classified results.



Testing with: Paired T-Tester (corrected)
Comparison field: Percent_incorrect
Significance: 0.05
Sorting (asc.) by: <default>
Show std. deviations: ☐

```
16:40:24 - Available resultsets
16:40:33 - id -
16:40:49 - id -
22:19:27 - Available resultsets
22:19:33 - id -
22:01:40 - label -
22:02:10 - Available resultsets
22:02:35 - Percent_correct - trees.J48 '-C 0.25 -M 2' -217733168
22:05:45 - Percent_incorrect - trees.J48 '-C 0.25 -M 2' -2177331
```

```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-mat
Analysing:  Percent_incorrect
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       10/6/22, 10:05 PM


Dataset                   (1) trees.J | (2) tree (3) lazy (4) baye
------------------------------------------------------------------
UNSWNB15Test              (100)   0.82 |   1.11 v   2.98 v   4.19 v
------------------------------------------------------------------
                               (v/ /*) |  (1/0/0)  (1/0/0)  (1/0/0)


Key:
(1) trees.J48 '-C 0.25 -M 2' -217733168393644444
(2) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 11168394707
(3) lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.Eu
(4) bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimat
```

Source: Experimental Analysis

Utilizing the experimenter is intended to produce valid comparisons between the models developed.

IV. Conclusion

This study evaluated and discussed models for network intrusion detection. The implementation of the intrusion detection models makes use of supervised machine learning techniques. The J48 Decision Tree, the Random Forest (RF), the Bayesian Network algorithm, and the K-Nearest Neighbor (KNN) approaches were used in this paper. In the K-Nearest Neighbor model, different values of K (1 and 3) were tried, and the best results were chosen. Using the UNSW NB15 dataset, the performances of the proposed models were compared to one another. Two distinct sets of the UNSW NB15 dataset were created. Eighty percent of the dataset was used in one set for training, while twenty percent of the dataset was used in the other set for model testing.

The implementation of the models made use of forty-four (44) features from the dataset. The nine categories of network intrusions in the UNSW NB15 dataset are Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. The classifiers were trained and tested using the percentage split (80:20), five-fold, and ten-fold cross-validation. The 10-fold cross-validation produced the model with the highest prediction accuracy, which is why it is preferred in this research. Accuracy, precision, recall, and F-measure were employed as performance indicators for evaluation. The network intrusion detection models were created using the Weka Machine Learning workbench.

According to the findings obtained, J48 performed the best and had an average execution time. The execution time for RF was the longest. KNN's execution time was the quickest, and its performance results were marginally worse than those of J48 and RF. In conclusion, J48 offers the optimum balance between execution speed and performance.

Research Limitations

The project has limitations, even though this study presented and examined network intrusion detection models utilizing machine learning techniques such as KNN, J48, RF, and Bayes Net. Based on research done by [13] features in the UNSW NB15 dataset were chosen. Without examining how the chosen features may impact the developed models' accuracy of detection and recall, they were used. Since smaller datasets are simpler to analyze, feature reduction poses a compromise between accuracy and simplicity, the research used relatively little feature selection (dimensionality reduction).

Future Works

Future studies may expand the model construction to take into account additional techniques like neural networks and support vector machines. Additionally, a distinct dataset can be utilized to train and test the models, ensuring that they can recognize intrusions.

Conflicts of Interest

The authors declare that there are no conflicts of interest on the manuscript.

References

1. W. Steingartner, D. Galinec, and A. Kozina, “Threat defense: Cyber deception approach and education for resilience in hybrid threats model,” Symmetry (Basel)., vol. 13, no. 4, pp. 1–25, 2021, doi: 10.3390/sym13040597.
2. A. V. Jatti and V. J. K. K. Sonti, “Intrusion Detection Systems: A Review,” Restaur. Bus., vol. 118, no. 7, pp. 50–58, 2019, doi: 10.26643/rb.v118i7.7246.
3. P. Panagiotou, N. Mengidis, T. Tsikrika, S. Vrochidis, and I. Kompatsiaris, “Host-based Intrusion Detection Using Signature-based and AI-driven Anomaly Detection Methods,” Inf. Secur. An Int. J., vol. 50, no. x, pp. 37–48, 2021, doi: 10.11610/isij.5016.
4. J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, “A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms,” IEEE Access, vol. 11, no. October 2023, pp. 121118–121141, 2023, doi: 10.1109/ACCESS.2023.3328351.
5. B. Lampe and W. Meng, “Intrusion Detection in the Automotive Domain: A Comprehensive Review,” IEEE Commun. Surv. Tutorials, vol. 25, no. 4, pp. 2356–2426, 2023, doi: 10.1109/COMST.2023.3309864.
6. T. U. Sheikh, H. Rahman, H. S. Al-Qahtani, T. Kumar Hazra, and N. U. Sheikh, “Countermeasure of Attack Vectors using Signature-Based IDS in IoT Environments,” 2019 IEEE 10th Annu. Inf. Technol. Electron. Mob. Commun. Conf. IEMCON 2019, pp. 1130–1136, 2019, doi: 10.1109/IEMCON.2019.8936231.
7. Stefanos Kiourkoulis, “DDoS Dataset - Use of machine learning to analyse intrusion detection performance,” Lulea Univ. Technol., p. 81, 2020, [Online]. Available: https://www.kaggle.com/devendra416/ddos-datasets/data
8. M. Zakariah, S. A. AlQahtani, and M. S. Al-Rakhami, “Machine Learning-Based Adaptive Synthetic Sampling Technique for Intrusion Detection,” Appl. Sci., vol. 13, no. 11, 2023, doi: 10.3390/app13116504.
9. J. P. Bharadiya, “A Tutorial on Principal Component Analysis for Dimensionality Reduction in Machine Learning,” Int. J. Innov. Res. Sci. Eng. Technol., vol. 8, no. 5, pp. 2028–2032, 2023, doi: 10.5281/zenodo.8002436.
10. N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” 2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc., pp. 1–6, 2015, doi: 10.1109/MilCIS.2015.7348942.
11. I. Almomani and M. Alenezi, “Efficient Denial of Service Attacks Detection in Wireless Sensor Networks,” J. Inf. Sci. Eng., vol. 34, no. 4, pp. 977–1000, 2018, doi: 10.6688/JISE.201807_34(4).0011.
12. V. Kumar, A. K. Das, and D. Sinha, “UIDS: a unified intrusion detection system for IoT environment,” Evol. Intell., no. 0123456789, 2019, doi: 10.1007/s12065-019-00291-w.
13. U. Matthew, J. Kazaure, and N. Okafor, “Contemporary Development in E-Learning Education, Cloud Computing Technology & Internet of Things,” EAI Endorsed Trans. Cloud Syst., vol. 7, no. 20, p. 169173, 2021, doi: 10.4108/eai.31-3-2021.169173.
14. D. Singh and B. Singh, “Investigating the impact of data normalization on classification performance,” Appl. Soft Comput., vol. 97, no. xxxx, p. 105524, 2020, doi: 10.1016/j.asoc.2019.105524.

15. S. H. Huang, “Supervised feature selection: A tutorial,” Artif. Intell. Res., vol. 4, no. 2, 2015, doi: 10.5430/air.v4n2p22.
16. C. C. Aggarwal, “Educational and software resources for data classification,” Data Classif. Algorithms Appl., pp. 657–665, 2014, doi: 10.1201/b17320.
17. A. M. Rahmani et al., “Machine learning (Ml) in medicine: Review, applications, and challenges,” Mathematics, vol. 9, no. 22, pp. 1–52, 2021, doi: 10.3390/math9222970.
18. Rashmi Agrawal, “K-Nearest Neighbor for Uncertain Data,” Int. J. Comput. Appl., vol. 105, no. 11, pp. 13–16, 2014.