Comparative Study of Supervised Learning Algorithms in Intrusion Detection

Susmitha reddy Arikatla, 2151229
Siva Subrahmanya Prasanna Kumar Bojja, 2156567
Akshay Singh Thakur, 2145566
Subhash Mekapati 2156465

***Abstract*— In the field of network security, Intrusion detection is a practical approach for dealing with many issues. This project presents a comparative study of supervised learning algorithms that can be used for intrusion detection. The algorithms Decision tree, Random Forest, Support Vector Machine, Naive Bayes, XGBoost, and Artificial Neural Networks are employed. A hybrid data set generated in the Australian Centre for Cyber Security's Cyber Range Lab is used to train the algorithm. The data is then well processed using techniques like Principal Component Analysis and Label Encoding and is then used to train and validate the algorithms. The algorithms are compared using the metrics Accuracy and Execution time to find the best algorithm that suits the dataset to predict the attacks.**

Keywords—Intrusion Detection, Supervised learning, Machine learning, Cyber security, Random Forest, Decision Tree, Support Vector Machine, CNN

I. Introduction

Intrusion detection is a system that detects attacks or malicious activity by analyzing the traffic and sending an alert to the user. The way Intrusion Detection works is that it runs through a network normalization process from which they learn themselves what the normal functions of the network are like and detect any abnormal activity. Machine learning is a great way to perform intrusion detection. Many classification models in machine learning are widely used for intrusion detection, such as SVM, Random Forest, and Logistic Regression [2]. The algorithms mentioned are Supervised learning algorithms that classify the attacks when the dataset is a labeled dataset that consists of the features (independent variables) and labeled predictors (dependent variable). From a cybersecurity point of view, we use a labeled dataset with attack types as predictors and train the model based on that dataset.

A. Dataset

The data set used for this project was created by the PerfectStorm tool in the Australian Centre for Cyber Security's (ACCS) Cyber Range Lab [1]. It is a hybrid of real modern normal activities and synthetic contemporary attack behaviors. 100 GB of raw traffic is captured using the Tcpdump tool (e.g., Pcap files). Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms are among the nine types of attacks in this dataset. To generate a total of 49 features with the class label, the Argus and Bro-IDS tools are used, and twelve algorithms are developed. A training set and a testing set are created from this dataset, named UNSWNB15 training-set.csv and UNSWNB15 testing-set.csv, respectively. The training set contains 175,341 records, while the testing set contains 82,332 records of various types, including attack and normal.

B. Approach

The approach we employed to analyze and make valuable insights from the data to achieve the goal of making a perfect machine learning model for intrusion detection can be seen in figure 1.

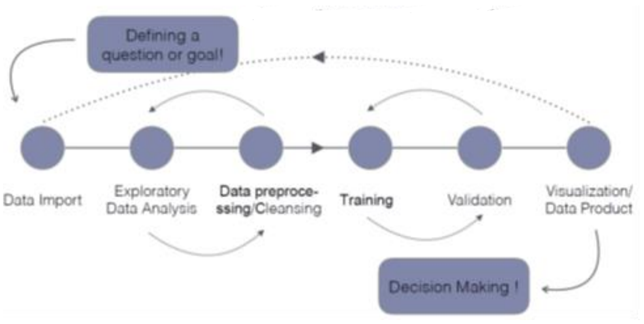
Defining a question or goal!
Data Import
Exploratory Data Analysis
Data preprocessing/Cleansing
Training
Validation
Visualization/ Data Product
Decision Making !

Figure 1. Stages to implement Data Analysis

1. In the first stage, the data set available in variable formats are imported and converted into a DataFrame that can be analyzed.
2. Exploratory data analysis on the dataset is performed to get fundamental insights from the data, such as getting to know the datatypes, spotting any outliers, getting to know the basic structure of the dataset, such as the number of rows and columns, and basic statistical inferences such as the correlation between the variables.
3. In the third stage, data preprocessing is performed to improve the quality of both data and prediction. At this stage, usually, Duplicate rows and blank rows in the dataset are removed, and the null or empty values in the dataset are replaced with the mean values of the entire dataset. The categorical or the data having object as datatype are converted to numerical datatype using One Hot label Encoder. Also, the Principal Component Analysis technique is used for dimensionality reduction since there are some unwanted columns that consume the storage and code execution time and might also result in a less accurate model due to their less correlation with the dependent variable. The data is also split into training and validation data sets, preparing it for the next stage.
4. Training is the most important stage in the process but, surprisingly, is a short one. Suitable Machine learning algorithms are chosen, and the training data set is used to fit the algorithms.
5. For validation, the fit algorithm is used to predict the values for the test data set, and the results are compared with the original to calculate the metrics like accuracy.

6. In the final stage, the results obtained are insights gained are presented in a visual form for quick and better understanding.

II. Algorithms

A. Decision trees

These are supervised ML algorithms that can be used for both classification and regression problems. Though these trees are not mighty, sophisticated, they are the most comprehensive and interpretable. They give better results when used as ensemble algorithms such as Random Forest and Extra Trees. Decision trees are greedy and top-down. The decision trees are highly susceptible to overfitting. A decision tree usually begins with a single node and branches out into different outcomes. Each of those outcomes leads to new nodes, leading to new possibilities. The tree branches represent the decision rule, and each leaf node represents the prediction outcome. The splitting process could go until each training data is in its own box, which would make the model suitable only for the trained dataset and will not be generalized. To make a better model without overfitting, we would simplify the tree with fewer splits and try to apply minimum training data in each box.

B. *Random Forest*

Random Forest is a robust machine learning algorithm that can be used for both classification and regression problems. Random Forest is an ensemble algorithm (machine learning algorithm which uses more than one machine learning model to improve the stability and accuracy of the model) of two or more decision trees, or it can also be thought of as a forest of decision trees. In the Random Forest model, the dataset is divided into several subsets in order to limit the depth of the decision tree using the Bootstrap Aggregating method, which is also known as bagging. In the Bootstrap Aggregating method, the dataset is divided into subsets with replacements. The prediction of a random forest tree is the average prediction value of all the decision trees used to create a Random Forest. The random forest model reduces the problem of overfitting in decision trees and reduces the variance, thereby increasing the model's accuracy. Random Forest can automatically handle the missing values and is robust to outliers. Since the Random Forest uses the rule-based approach rather than the distance-based approach, no feature scaling such as Standardization or Normalization is required.

C. *Support Vector Machine (SVM)*

Support Vector Machine is a supervised ML algorithm that can be used for both classification and regression problems. The objective of the SVM is to minimize the cost function and maximize the margin between the support vectors through a hyperplane. Therefore, a hyperplane that can be of any dimension and linear or non-linear is used in SVM to differentiate between two or more classes for classification.

D. *Naïve Bayes*

Naive Bayes is a supervised machine learning algorithm consists probabilistic classifiers which works by Bayes' theorem with strong independence between the features. We can achieve high accuracy levels with kernel density estimation. It is a group of algorithms where all of them share a common principle where every pair of features which is being classified is independent of each other

E. *XGBoost*

Gradient boosting is another type of ensemble supervised ML algorithm used for both classification and regression problems. It is a much more generalized form of Gradient Boosting. It uses $\ell1$ and $\ell2$ regularization, which improves model generalization and overfitting reduction. Where $\ell1$ and $\ell2$ are loss functions. This algorithm is usually faster than Gradient Boosting because of the parallelization of tree construction. It can handle missing values within a data set; therefore, the data preparation is not as time-consuming. When there is a more significant number of training samples, ideally, greater than 1000 training samples and more minor 100 features, or when the number of features < number of training samples and when there is a mixture of categorical and numeric features or just numeric features, XGBoost is ideal to use

F. *Artificial Neural Networks (ANN)*

An Artificial Neural Network is an ML algorithm based on the model of a human neuron, which can be applied for both classification and regression problems. ANN consists of three layers. They are the input layer for feeding raw information into the network, the hidden layer (which can be multiple) to determine the activity of each hidden unit, and the output layer to get the output which depends on the activities of the hidden layers and connection between hidden layers and output layers. The model requires a large diversity of learning for real-world operation. Neural networks work even if a few units fail to respond to the network but implement extensive and effective software neural networks. One downside of ANN is that more processing and storage resources need to be committed.

G. *Data Import*

As mentioned earlier, the data from the source is provided as two data sets—one dedicated to training the algorithm and one for testing it. The train data set contained 82332 rows and 45 columns, also called features, and the test data set had 175341 rows and 45 features. They are read using the simple pandas read_csv command.

H. *Exploratory Data Analysis*

In this stage, the data is explored to understand the features and find the underlying trends. The primary functions '.describe()', '.head( )' are used to observe the samples of both the test and train data sets. It was observed that there are four features, 'attack_cat', 'proto', 'service', and 'state', that are formatted as categorical values that should be pre-processed. It was also observed that each record is categorized either as a normal record or an attack, and there are nine diverse types of attacks in the data. They are Generic, Exploits, Fuzzers, Dos, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms.

I. *Data Preprocessing*

First, the train and test data sets are checked for null values using the command '.isnull()'. Surprisingly, there are no records with incomplete data in the data sets. Since the categorical values cannot be used for data analysis, the four features mentioned in the above sections are converted into numerical values using the ‘preprocessing.LabelEncoder()’. In the datasets obtained from the source, the training data set contained fewer records than the testing/validation data set. We combined the data and later split it into a known train-test ratio to use a much more common approach. In the dataset, the value to be predicted using the machine learning algorithms (target dataset) is ‘label’ from the features. This value is dropped from the dataset and is stored in the ‘Y’ variable. The rest of the features are stored in the ‘X’ variable. To minimize the bias that can occur because of the difference in the proportionality of the data range, we normalized the data using the standard scalar function.

To reduce the features not required for intrusion detection, we used the concept of Principle Component Analysis. As we know from the previous data, the variance for 10 PCA components is 87%, the variance for 25 PCA components is 95%, and the variance for 30 components is 99%; it is advisable to convert data to less than 30 components. So we reduced the features in the train data from 45 to three levels, 30, 25, and 10, to observe the run time and accuracy of various ML algorithms. Since the data sets for testing and training are given separately, we just separated and value to be predicted which is the ‘label feature’, and saved it in different variables. Finally, the datasets are X_train, X_test, Y_train, and Y_test. These data sets are then used to trian and validate the algorithms in the next stage.

J. *Training and Validation:*

1) *Decision Tree:*

To implement the algorithm on the data DecisionTreeClassifier is imported from the sci-kit learn library. The classifier then fits on the X_train and Y_train data sets to train the algorithm. To perform validation, the accuracy of the algorithm prediction is measured. For this, the trained algorithm is first used to predict for the X_test dataset, and the prediction values are stored in the 'Y_pred' variable. To measure the accuracy of the algorithm accuracy_score function is imported from the 'sklearn.metrics library'. This function is then used to compare the 'Y_test' and 'Y_pred' giving the accuracy score. One more metric that we used to measure the algorithm's performance is Time complexity. This is obtained by calculating the difference between the algorithm's start time and end time. For 30 PCA components, the algorithm took approximately 5061 seconds, giving a prediction accuracy of 80.39 percent. The results for 25 and 10 PCA components can be seen in Section IV.

2) *Random Forest:*

To implement this algorithm, RandomForestClassifier is imported from the Sklearn.ensemble library. Similar to the Decision Tree algorithm, the Random Forest is trained on X_train and Y_train data sets. The accuracy and execution time of the algorithm are also measured using the same functions. For 30 PCA components, the algorithm took approximately 50.02 seconds giving a prediction accuracy of 84.71 percent. The results for 25 and 10 PCA components can be seen in Section IV.

3) *Support Vector Machine:*

The SVM classifier is imported from the Sklearn library and is trained on the X_train and Y_train datasets. The accuracy and execution time are measured using the same functions. For 30 PCA components, the algorithm took approximately 218.26 seconds, giving a prediction accuracy of 78.62 percent. The results for 25 and 10 PCA components can be seen in Section IV.

4) *Naïve Bayes:*

The GaussianNB classifier is imported from the sklearn.naive_bayes library and is trained on the X_train and Y_train datasets. The accuracy and execution time are measured using the same functions. For 30 PCA components, the algorithm took a quick 0.15 seconds giving a prediction accuracy of 69.65 percent. The results for 25 and 10 PCA components can be seen in Section IV.

5) *XGBoost:*

From the xgboost library, the XGBClassifier is imported and is trained on the X_train and Y_train datasets. The accuracy and execution time are measured using the same functions. For 30 PCA components, the algorithm took 17.95 seconds, giving a prediction accuracy of 84.63 percent. The results for 25 and 10 PCA components can be seen in Section IV.

6) *Artificial Neural Network:*

Unlike the algorithms above, ANN is a complex algorithm. To train the algorithm we added 1 input layer, 2 hidden layers and one output layer. For the first input layer we used ‘Dense’ class which is a famous class in tensor flow, and we used 6 output dimensions as there is no rule to decide the neurons in the hidden layer. The input dimensions in this layer are equal to the number of PCA components. We also used ‘relu’ activation function. For the second hidden layer we used 9 output dimensions (units). In the output layer we used only one output dimensions are the dependent variables is in binary form. The process of training is performed in 2 steps. The first one is compiling the ANN and the second stage is to fit the ANN to the training set. The Compile used is a method of Tensor flow and the optimizer used is ‘adam’. This optimizer in general performs Stochastic gradient descent and updates the weight during training to reduce the loss. We fit the data on X_train and Y_train data sets with a batch_size of 10 and nb_epoch =10 to improve the accuracy over time. For 30 PCA components, the algorithm took 10 iterations ie., 60 seconds to give a prediction accuracy of 81.13 percent. The results for 25 and 10 PCA components can be seen in Section IV.

III. Results and Conclusion

The above algorithms are trained with 30, 25, and 10 PCA components. The results obtained in terms of accuracy and execution time are presented in table below for comparison. can be observed that Both Decision Tree and RandomForest

Table 1. Accuracy & Execution time for the algorithms at 30, 25, 10 features

Algorithm	PCA (n=30)		PCA (n=25)		PCA (n=10)	
	Execution Time (sec)	Accuracy (%)	Execution Time (sec)	Accuracy (%)	Execution Time (sec)	Accuracy (%)
Decision Tree	5.61	80.39	9.77	80.47	1.32	78.67
Random Forest	50.02	84.71	50.33	82.43	27.20	76.99
SVM	218.26	78.62	182.08	77.99	299.46	61.98
XGBoost	17.95	84.63	18.05	85.10	7.73	80.72
Naive Bayes	0.15	69.65	0.17	74.82	0.08	82.63
ANN	11.00	99.98	10.00	99.98	41.00	98.01

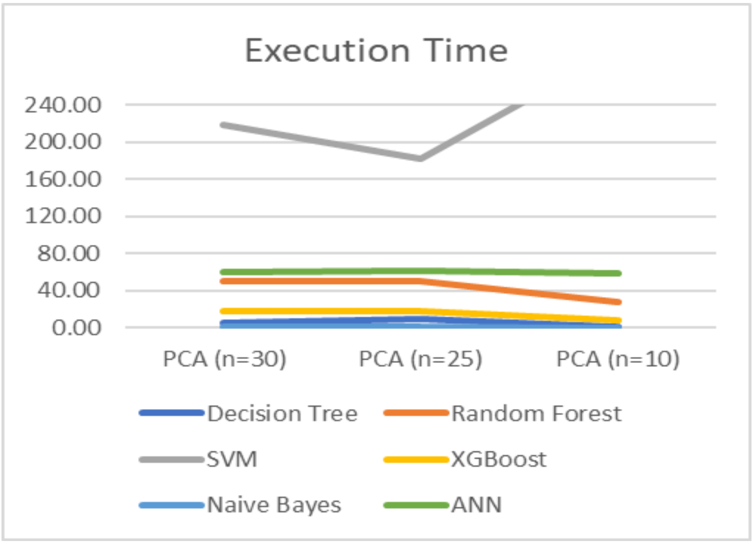

Figure 2. Execution time for the algorithms at 30, 25, 10 features

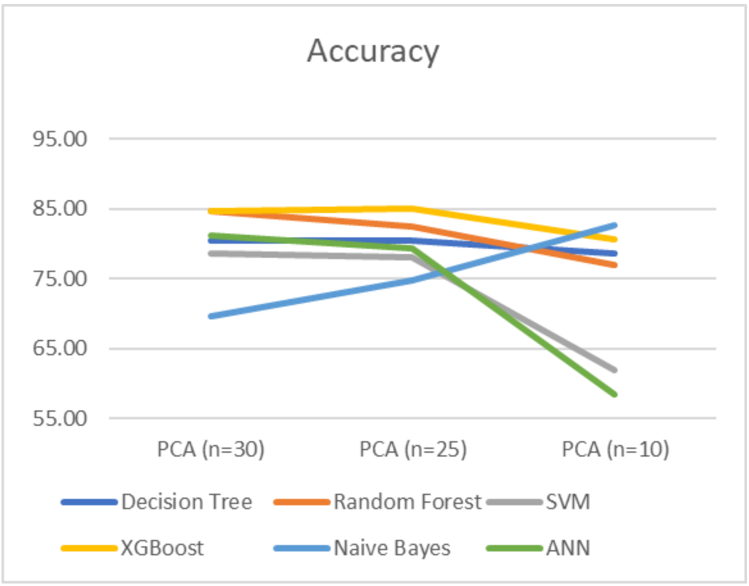

Figure 3. Accuracy for the algorithms at 30, 25, 10 features

algorithms followed an Identical trend. With the decrease in PCA components, the accuracy decreased significantly, but the execution time only increased for 25 PCA components. In the case of SVM, the execution time increased, and accuracy decreased significantly with the decrease in PCA components. The accuracy with ANN also followed a negative trend with the decrease in PCA components but the time remained constant. To get a better picture of the trend in metrics the trend of accuracy and execution time can be seen in Figures 2 and 3. It can be seen in figure 2 that SVM has the highest execution time in all the cases while Naïve Bayes has the lowest. Figure 3 is more crucial as accuracy is the most important metric to decide the suitability of the algorithm. It can be observed that XGBosst performed the best in most cases followed by the Random Forest algorithm. The Decision Tree performed well when the features are fewer but were inaccurate for the higher number of features. From this information, it can be concluded that XGBoost is the most suitable algorithm for this scenario with the highest accuracy of 85.10 %- and 18.05-seconds execution time.

REFERENCES

[1] M. W. David, “Unsw_nb15,” *Kaggle*, 29-Jan-2019. [Online]. Available: https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15. [Accessed: 29-Apr-2022].

[2] F. Gharibian and A. A. Ghorbani, "Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection," Fifth Annual Conference on Communication Networks and Services Research (CNSR '07), 2007, pp. 350-358, doi: 10.1109/CNSR.2007.22.