Intrusion Detection with Ensemble of Classifiers

Akhil Veluvolu
Computer Science
Missouri S&T
Rolla, United States
avk2q@mst.edu

Sai Tigmanya K
Computer Science
Missouri S&T
Rolla, United States
sk786@mst.edu

Lakshmi Tanuja Tammireddy
Computer Science
Missouri S&T
Rolla, United States
lttk4@mst.edu

Yashwanth Reddy Gudipally
Computer Science
Missouri S&T
Rolla, United States
yg2bf@mst.edu

***Abstract*—With the increase in connectivity between computers, networks are exposed to various kinds of security threats. An intrusion detection system (IDS) helps in identifying security breaches such as unauthorized access, policy violations, or malicious activities. Accuracy of Intrusion Detection is vital to assure minimal false alerts. Intrusion Detection System (IDS) is the most commonly used technique to protect the integrity and accessibility of assets in the network. Even today IDS is an open problem where a lot of research work is going on to achieve better performance. Few possibilities for low efficacy in IDS are First, imbalanced data. Second, a single classifier may not classify all kinds of attacks. In this paper, we proposed a new intrusion detection system using Synthetic Minority Oversampling Technique (SMOTE) and ensemble learning techniques. In our ensemble approach, we used Random Forest (RF), XGBoost and Extra Trees (ET) algorithms. For attack classification, we used max voting strategy. Experimental results show that our proposed ensemble of models achieve the performance of Convolution Neural Network (CNN)**

***Index Terms*—Machine Learning,Intrusion Detection, Ensemble of Classifiers**

I. Introduction

In today's fast-emerging world, internet-connected devices play a vital role. Data is stored in these devices and transferred using the internet. Any malicious activity such as unauthorized access into these internet-connected devices is called intrusion and such intrusions can cause confidentiality or integrity violations. There are different types of attacks that happen on a computer network. These attacks can be categorized into R2L, U2R, Probe and DoS.

As technology advances, these attacks threaten the confidentiality, integrity, and availability of cybersystems. Intrusion Detection System (IDS) is, therefore, necessary to safeguard the systems from various attacks. IDS is a software application or a device that helps in monitoring a network or systems for malicious activity, policy violations and unauthorized access. These intrusion detection systems can also be built using Machine learning techniques.

Machine learning is one of the branches of Artificial Intelligence, where algorithms are studied to train the system to make decisions and identify patterns with less human intervention. It is used to build and automate analytical models. Widely used machine learning techniques are Supervised and Unsupervised learning.

In this paper, we implemented IDS using an ensemble of models to detect and classify the attacks. The rest of the paper is organized as follows: Section 2 is about the related works that were previously done, Section 3 is about data pre-processing that we have done on the dataset, Section 4 describes an ensemble of classifiers and then Section 5 describes our experimental results and then we sum it up with a conclusion in Section 6.

II. Related work

As an important tool to ensure cybersecurity in computer-based systems, IDS has always been an area of interest for researchers. Although there were numerous solutions that were proposed to enhance the performance of IDS, we only considered the work related to Intrusion Detection with machine learning including Neural Networks.

A single ML model might not yield the desired accuracy as it may not identify the boundaries between the class due to the imbalanced nature of the data set. Whereas, CNN is made of several layers like convolutional, pooling, and fully-connected (FC) layers which increases the training time and computational complexity. By combing several machine learning models we were able to achieve similar accuracy to that of CNN with lesser computations. Ensemble learning is a machine learning technique that reduces false-positive rates by combining several base models and produce more accurate solutions when compared to a single model.

III. Data Preprocessing

In Machine Learning, the quality of the model depends on the quality of data. Data pre-processing is the phase where our data is transformed or encoded to a level where our machine learning algorithm understands easily. First, we label the instances under Normal, R2L, U2R, DoS and Probe-based on attack types. Second, we convert continuous variables to categorical variables using equal-width binning and equal frequency binning. Third, we encode all categorical features in the NSL-KDD dataset using Label Encoder between 0 and n - 1 (n stands for the number of unique elements in that column). Finally, we apply an oversampling technique called

SMOTE.

Normalization is a commonly used technique in pre-processing that changes the numeric values of a column in the data set to a common scale, without disturbing the differences in the range of values.

Binning is a process in which numerical values are placed into bins. Dividing the data set into k bins of equal width is called equal width binning. Whereas dividing the data set into k bins, where each bin has an equal number of frequencies is called equal frequency binning.

SMOTE (Synthetic Minority Over-sampling Technique) is a statistical technique used to increase the number of instances of the minority class in a balanced way by combining the features of the minority class with its nearest neighbors without copying the existing minority instances. It doesn't change the count of the majority class. This technique will only increase the features available for minority classes and makes the samples more general.

IV. Ensemble of Classifiers

In Machine learning, Ensemble models are used to combine the verdict from multiple learning algorithms to obtain better predictive performance. Ensemble models help in minimizing the errors in learning models due to noise, bias, variance and improves the stability and accuracy of machine learning models. There is no fixed combination of models to do this, we have to try different models according to the specified problem. Tried various combinations using the below models and used the "Majority voting" strategy to predict the class.

Majority voting: In every model, a prediction is made for each test instance and the one that receives more than half of the votes is the final output. If no prediction receives more than half of the votes, we may conclude that a stable prediction could not be made by the ensemble method. In such cases, we select from the best model.

A. Random Forest

Random forests also known as random decision forests contain several decision trees and each tree has different hyperparameters and different subsets of data. It uses the bagging (bootstrap aggregation) method for building the model. Each decision tree predicts an output and using max voting policy we decide the final output. This is a flexible algorithm that produces great results without tuning the parameters and requires less configuration to generate a reasonable prediction.

The majority output from many uncorrelated trees will be a better prediction when compared to a single one because each tree's output error is minimized by the other tree's output. The main idea behind random forest is that each tree must have low correlation when compared to other trees.

B. KNN

K – Nearest Neighbors (KNN) works by considering similar things are in close proximity. The 'K' in KNN stands for the number of nearest neighbors that participate in the voting process. The distance between two points is measured using Euclidean distance or Manhattan distance. The model's accuracy depends on the accuracy of the data given to it along with the selection of optimal K value. When the value of K is small, the model could be too sensitive to noise and could be overfitting and when the K value is high, the model could be generalized and result in underfitting, which could be computationally expensive. The optimal value of K can be found using the GridSearchCV function that allows us to check multiple values for K. KNN is a well-preferred algorithm because of its easy-to-use nature and low calculation time that is required to run the algorithm.

C. Decision Trees

A decision tree is a classification algorithm that classifies data using series of rules. The major building blocks of decision trees are splitting, pruning, and tree selection. Splitting means partitioning the dataset into subsets that contain instances of a particular class. The various types of splitting techniques are Gini Impurity, Information gain, and Chi-Square techniques. Pruning reduces the tree size by removing the leaf nodes under certain branch nodes. Pruning helps in avoiding overfitting by keeping the model simpler. Tree selection is the process of identifying the smallest tree that best fits our data.

Gini impurity: Gini stands for purity and works only for categorical targets. Gini impurity calculates the measure of impurity of a split. The feature with the lowest impurity will be the best to split the current node. The node with uniform class distribution will have high impurity and the node with all instances that belongs to the same class will have low impurity.

$$\text{Gini Impurity} = 1 - \Sigma_{i=1}^{n} P_i^2$$

Information Gain (IG): IG measures the reduction of entropy by splitting a dataset according to a value of the random variable. Information Gain is used to split the nodes when the target class is categorical. Information gain is high when the entropy is less and vice versa.

Chi-Square: Same as Gini Index, Chi-Square also works only for categorical targets. Chi-square test compares the data collected and the data predicted, then evaluates the inherent variability. It is calculated using the below formula.

$$x^2 = \Sigma_i \frac{(O_i - E_i)^2}{E_i}$$

X^2 = Chi-square value,
O_i = Observed frequency,
E_i = Expected frequency.

When the value of Chi-Square is high, the purity of the nodes are also high after the split.

D. Extra Trees

Extra trees are also known as Extremely randomized trees are similar to the random forest and combine the decisions from various unpruned decision trees. Unlike bagging in a random forest, extra trees pick each decision tree from the whole dataset. The split point is selected at random and will randomly sample the attributes at each split. This random nature of the algorithm helps in the creation of de-correlated decision trees. The three main parameters which help in increasing the accuracy of extra trees are the minimum number of samples required at each split, the number of input features that should be randomly selected, and the number of decision trees in the model.

E. XGBoost

XGBoost also known as the eXtreme Gradient Boosting algorithm which is an ensemble machine learning algorithm that uses a gradient boosting framework based on decision trees. It is based on function approximations which are done by optimizing some particular loss functions and by applying regularization techniques. This algorithm can be used for solving various prediction problems such as user-defined problems, regression, classification, and ranking. XGBoost uses gradient descent architecture with algorithmic enhancement to boost weak learners. It is well known for tree pruning using depth-first search, efficiency in handling missing data, and avoiding over-fitting of data. This algorithm builds a sequential tree and uses max depth as a parameter for pruning trees. During each iteration, the model chooses a different fold as cross-validation data. In practice, the algorithm starts with a root node that contains all the data and then iterates over all the values of each feature, and then evaluates the split loss reduction using the below formula. If the gain for the best split is negative then we stop growing the branch.

V. Experiment and Results

*A. **Dataset Description***

NSL-KDD dataset that was introduced to handle intrinsic difficulties of KDD cup 1999 dataset[5], which has a huge number of duplicate records. Even though it is old and does not constitute actual networks, it is constantly used to compare network intrusion detection models. Many scientists/researchers use the NSL-KDD dataset as the standard dataset.

NSL-KDD dataset includes 2 sub folders: KDD Train that has 125,973 samples and KDD Test that has 22,554 samples. Due to an imbalance in the decision attributes, it becomes difficult for the machine learning model to predict the class label. So, the records in the dataset were classified into 5 types according to their characteristics as shown in table 3.

Category	Attacks
DoS	smurf, teardrop, back, land, neptune, pod
Probe	ipsweep, nmap, portsweep, satan
R2L	warezmaster, warezclient, spy, phf, multihop, imap, guess_passwd, ftp_write
U2R	rootkit, perl, loadmodule, buffer_overflow
Normal	normal

Table 3. Categories of attacks

Denial of Service (DoS): This attack absorbs more memory or computing resources so that the system cannot access multiple requests[1].

Probe: This attack collects data about probable vulnerabilities of the target system that can be used to set up attacks[1].

Remote to Local (R2L): Attacker does not have access to the victim system. So, it tries to get access as a user of that system[1].

User to Root (U2R): By this, attackers access the system as a user and utilize some vulnerability to obtain root access[1].

*B. **Evaluation Metrics***

Accuracy is considered as one of the key performance measures for intrusion detection. The formal definition of accuracy is the ratio of the number of correct predictions our model makes to the total number of predictions. Apart from accuracy, we also consider the detection rate and false-positive rates.

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

TP = True positives,
FP = False positives,
FN = False negatives,
TN = True negatives.

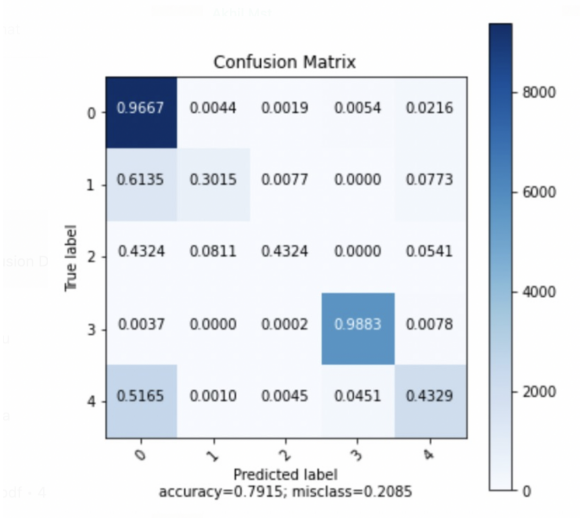

Fig. 1. RF, XGB, DT

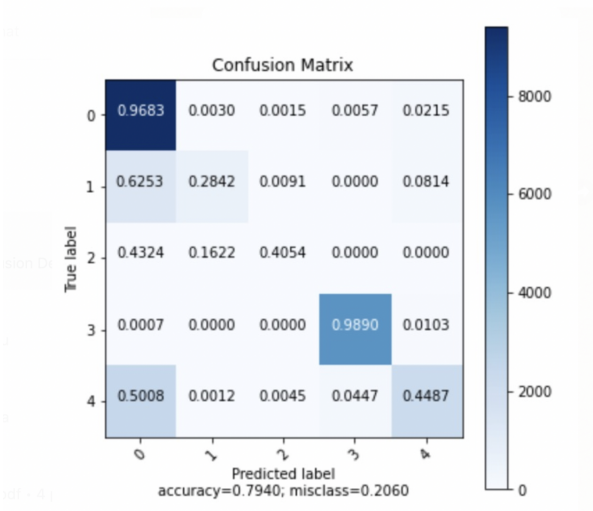

Fig. 2. RF, XGB, ET

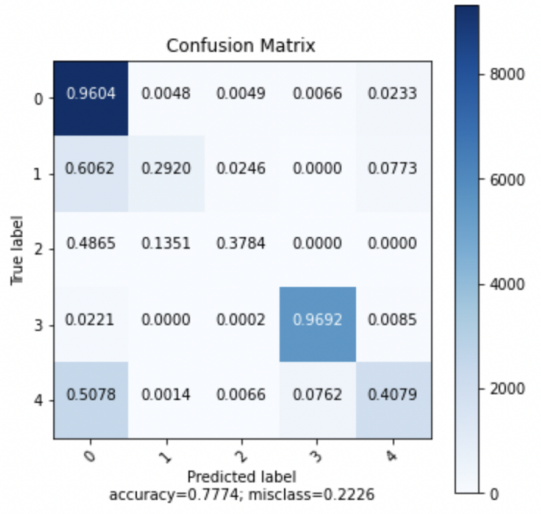

Fig. 3. RF, KNN, ET

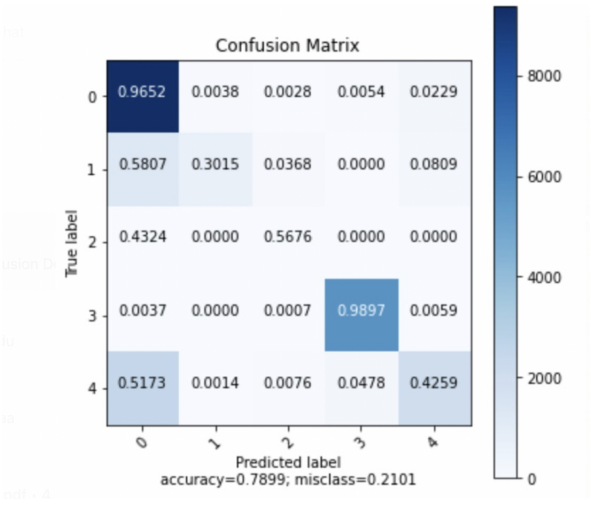

Fig. 4. RF, DT, ET

C. *Experiment*

Using the NSL-KDD dataset, we implemented our experiment in two different ways. First, by splitting KDD train data set into train and test with a split ratio of 0.2. Second, by using KDD Train as training set and KDD Test as testing set.

Implemented multiple combinations of algorithms and the below four combinations of ensemble models are best among them. The results obtained when implemented with validation data and KDD test data are shown in table 2. The four combinations are as follows:

1. Random forest, K - Nearest Neighbours, Extra trees.
2. Random forest, XGBoost, Extra trees.
3. Random forest, Decision trees, Extra trees.
4. Random forest, XGBoost, Decision trees.

D. *Results*

The accuracy achieved by individual models are shown in the table below.

	RF	KNN	XGBoost	ET	DT	AdaBoost
Validation Data	99.87	99.24	99.71	99.83	99.78	83
KDD Test	74.74	73.85	74.46	74.38	75.52	56.37

Table 1. Individual Model Accuracy

The table below shows the results obtained when implemented using ensemble models.

The best accuracy achieved with KDD Test data is while using the combination of Random forest, XGBoost, Extra trees which is close to the accuracy obtained using Convolutional neural networks (80 %)[1].

	RF, KNN, ET	RF, XGB, ET	RF, DT, ET	RF, XGB, DT
Validation data	99.7	99.5	99.81	99.71
KDD Test	77.67	79.4	78.15	79.15

Table 2. Ensemble Model accuracy

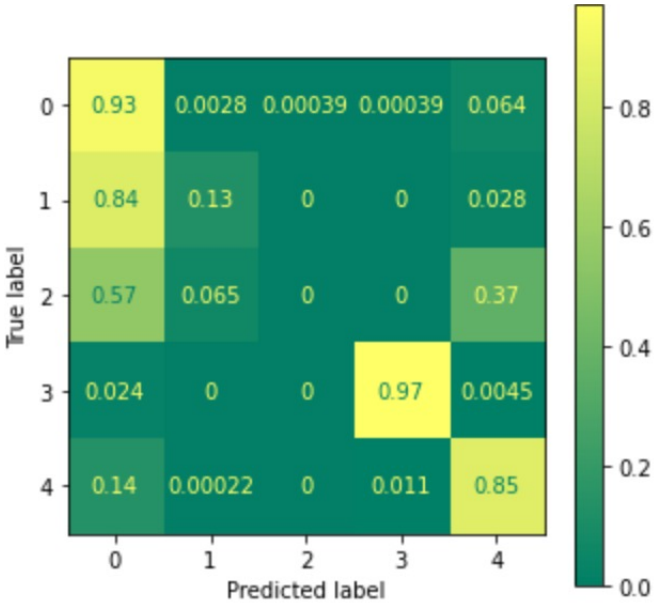

Fig. 5. RF, DT, ET

From Table 1 and Table 2, we can observe that there is a huge variation in accuracy between the KDD-Train validation set and the KDD-Test set. We tried reverse engineering to figure out whether our model is over-fitting or KDD-Test contains complex patterns. To prove this hypothesis we have trained a Random forest model with 80 % of the KDD-Test set and achieved an accuracy of 99 %. We have used the same model on the KDD-Train set and achieved 93 % accuracy. From this, we can conclude that KDD-Test contains patterns that are complex for the models trained on the KDD-Train set.

VI. Conclusion

We analyzed various Machine Learning algorithms on the NSL-KDD dataset. To date many machine learning and deep learning models have been applied, out of all CNN achieves the state of the art accuracy of 80.13%[1] on the KDD Test set. In this paper, we proposed an intrusion detection framework, which is based on SMOTE and an ensemble of classifiers. The experimental results show an accuracy of 79.4%on the KDD Test set which is almost close to CNN. Based on the comparative results our proposed system is as powerful as CNN. In the future, we plan to increase the accuracy of R2L, U2R, and Probe of the KDD Test set by removing redundant features using feature selection techniques.

VII. References

[1] Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks. Yalei Ding, Yuqing Zhai. https://dl.acm.org/doi/10.1145/3297156.3297230

[2]Building an Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier. Yuyang Zhoua,b,c, Guang Chenga,b,c,, Shanqing Jianga,d and Mian Daia,b,c

[3]Anomaly-based network intrusion detection: Techniques, systems and challenges. P. Garcı a-Teodoroa,*, J. Dı az-Verdejoa, G. Macia -Ferna ndeza, E. Va zquez

[4]MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM. Anish Halimaa A, Dr. K.Sundarakantham

[5]https://www.unb.ca/cic/datasets/nsl.html