Intrusion Detection and Attack Classification using an Ensemble Approach

Gagan Ganpathy[1], Niharika Shrivastava[2], Prof. O.P. Vyas[3], Mantek Singh[4], Ridam Arora[5], Saurabh Mishra[6]

[1-6]Indian Institute of Information Technology, Allahabad, India
[3]Professor, Dept. of Information Technology, Indian Institute of Information Technology, Allahabad, India

---***---

Abstract - *The challenges to ensure safe and trusted communication of information between various organizations have increased multifold in recent past. Intrusion Detection Systems such as firewall, message encryption and other approaches are being employed with partial success, however the risks and chances of malicious intrusions are still posing a threat. We are proposing to make use of recent advancements in the field of machine learning to develop an intrusion detection system. In our work, the machine learning classifiers namely, random forest, decision table, multi-layer perceptron and naive bayes were used in an ensemble model showing a significant improvement in the overall accuracy. The proposed approach was implemented using a bench-marking dataset from KDDCup.*

Key Words: Ensemble learning, machine learning, intrusion detection

1. INTRODUCTION

The paper aims at finding a novel approach to identifying and classifying malicious packets found during an intrusion attack. It is not easy to identify an attack by just looking at certain attributes related to a network packet. Therefore, we try to use the KDD Dataset, which contains a lot of data samples along with 41 features that classifies the packets into 23 major and 4 broad categories (DDoS, R2U, U2R and Probe Attacks). While some work has been done on using machine learning to classify packets, we try using ensemble learning to combine several models. An intrusion detection system (IDS) [Fig. 1] is a system for monitoring traffic(network traffic) and identifying suspicious activity and issuing alerts when such activity is discovered. In this paper, the importance of an IDS with a variety of different approaches is discussed. The problem certainly is no easy task. We need to train a variety of models, and then take a weighted mean of their outputs before reaching a conclusion. The steps would include training various models independent of each other on the KDD dataset, and then stacking them together(called model stacking) to give us an ensemble model.

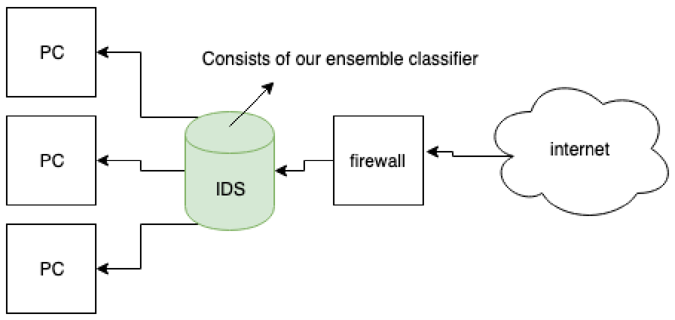


Fig-1: Network Architecture

2. LITERATURE REVIEW

Classification of packets into malicious and non malicious(and if it is malicious, the type of intrusion attack it represents) via machine learning has been a problem being researched upon in the past 2-3 years. As a result of the research carried out on this problem, various approaches with their pros and cons came to the foreground. Mohammad Almseidin and his team at University of Miskolc [4] tried solving this problem using a variety of classification models, like MLP, decision trees, random forests, logistic regression. However, the conclusion that there is no one fits all model was reached i.e. no individual machine learning paradigm could handle all the attack types effectively. Decision table classifier was able to obtain the lowest false negative value of about 0.002, but it was not even close to the highest accuracy rate detection. On the flip side, the Bayesian network classifier detected the normal packets most efficiently. Rohit Kumar Singh Gautam [8] in his research used an ensemble model by training 3 distinct models, the naive bayes, PART and Adaptive Boost. While the results of the ensemble model were better than those achieved by any of the models individually. Independently, Ying Wang, Yongjun Shen and Guidong Zhang [11] tried using random trees and bayesian classifiers for tackling the intrusion detection problem. While the results were satisfactory, they did conclude that the Bayesian classifier scored the highest amongst all other classifiers but for a smaller set of classes. But when tested for a much larger set of classes it performed poorly with a really low accuracy rate compared to other classifiers. And on the flip side, random trees did present a really promising score for large samples of the dataset but did not perform so well for smaller samples. This further gives impetus to our thought that an ensemble of these various models, which can combine the pros of various different types of models can be used to achieve far more optimal results than those that can be reached via using a single model. The remaining sections of the paper are arranged as follows: Section 3 discusses the work done previously in the field of intrusion detection systems. Section 4 lists the features of the dataset used. Section 5 deals with the proposed methodology to solve this problem and Section 6 shows the results we achieved.

3. RELATED WORK

Since, this work is primarily based on machine learning and creating an ensemble of the machine learning models on the KDDCup Dataset, we shall briefly review the most recent literature and the work done.

1.1 KDD Dataset

The research proposed by the authors in [7] have individually trained and evaluated their machine learning classifiers (eg: naive bayes, multi-layer perceptron, decision table) on the KDD Dataset. Each model in [7] uses 49596 instances of the KDDCup99 dataset (which is just 10% of the entire KDD dataset, in the paper we have built a model that is able to produce benchmark results using the complete dataset having 4,898,430 instances without the problem of over-fitting which seemed to be the case in [7] using a smaller version of the dataset) to implement their models. The authors in [1] used a Support Vector Machine classifier to detect intrusions using MATLAB using the KDD dataset as the benchmark dataset and found the SVM classifier to not be a suitable algorithm for such large datasets considering the long time it takes to train. [5]

The authors in [3], claim that neural networks are not suitable for U2R and R2L attacks, but recorded decent accuracy scores for PROBE and DOS attack packets [9]. A study in similar lines to [3] implemented several other algorithms involving neural networks, tweaking certain parts of the neural network classification process and implementing a hybrid for perceptron back propagation algorithm and gauged high accuracy rates.

In order to achieve low computation times, authors in [10] made sure to select the most significant attributes to design the intrusion detection model with trying to achieve high accuracy at the same time. To keep the false positive rate to a minimum in [10] they implemented a detection system based on neural networks [6] and extended classifier system [2]. Whereas, in [11] an information gain algorithm was found to be the most effective.

But, all the previous works work with optimizations on single classifiers, this paper takes an ensemble approach combining several individual models, being able to collectively learn from the features that each base model presents.

4. DATASET DESCRIPTION

This is the data set that was used for The Third International Knowledge Discovery and Data Mining Tools Competition. The competition was organized in conjunction with KDD99 the Fifth International Conference on Knowledge Discovery and Data Mining. The problem statement involved building a network intrusion detector, a model capable of identifying the bad connections (referred to as intrusions or attacks), and normal connections. This database contains a wide variety of intrusions simulated in a military network environment. The dataset contains 4,898,430 labeled and 311,029 unlabeled connection records. The labeled connection records consist of 41 attributes.

In the dataset used for the purposes of this project, each instance represents the attribute values of a class, and each class is categorized as either normal or bad (i.e. an intrusion or attack). The classes are divided into one normal and four main intrusion classes: Probe, User-to-Root (or U2R), Remote-to-Login (or R2L) and Denial of Service (DoS). The majority of the instances belong to the Smurf category (a type of DoS attack)

Denial of Service Attack - In computing, a DoS attack is a cyber-attack in which a machine or network resource is made unavailable to its intended authorized users by disrupting services of a host connected to the Internet.
Probe Attacks - System Information gets exposed to unauthorized entities.
U2R(User to Remote) Attacks - Unauthorized people get access to the administrator account type.
R2L(Remote to Local) Attacks - Unauthorized entities gain access to the host.

5. IMPLEMENTATION PLAN

5.1 Data Acquisition

The implementation of our project begins with searching for a suitable data-set that would suit the purpose of our research. We found the KDD99 (Knowledge Discovery in Databases) dataset to be the most suitable with about 5 million entries covering nearly every real-life scenario. The KDD dataset was found to be a benchmark dataset towards our domain of research with 41 features.

5.2 Preprocessing

The data-set contained some cells having null values, which may result in the model being trained imperfectly and result in a low accuracy rate. If a column consisted of mostly null values, we dropped the entire column and in cases where only a few of them were missing, we filled in with mean value calculated over the dataset for that feature. We also encountered categorical data which was important for our learning algorithms. But various ML algorithms don't handle categorical data (Naive Bayes).

Categories of Attack	Attack name	Number of instances
DOS	SMURF	2807886
	NEPTUNE	1072017
	Back	2203
	POD	264
	Teardrop	979
U2R	Buffer overflow	30
	Load Module	9
	PERL	3
	Rootkit	10
R2L	FTP Write	8
	Guess Passwd	53
	IMAP	12
	MulitHop	7
	PHF	4
	SPY	2
	Warez client	1020
	Warez Master	20
PROBE	IPSWEEP	12481
	NMAP	2316
	PORTSWEEP	10413
	SATAN	15892
normal		972781

Fig-2: Distribution of the dataset

For such features we decide to convert them into numerical columns using sci-kit learn's get_dummies() function. We then scaled the features as per our algorithms (e.g. MLP is sensitive to scaling). We performed a set of exploratory analysis on the data set found trying to find if the dataset has missing values. Apart from just finding if the data has missing values we also need to analyze the type of classes and the frequency of each class in the training sample, this will give us a good idea about how normalized the dataset is in terms of the distribution of the classes. How do we check for missing values in the entire dataset, since the dataset is huge, we can simply plot the heatmap representing if each entry in the dataset is null or is filled. As it can be seen in the picture below, the heatmap represents every entry in the dataset and tells us if the entry is null or not. Since, every entry in the heatmap is solid purple in colour which suggests every entry is filled and there are no null values / missing values in the dataset. Also, since there are no missing values we can go ahead and analyse what type of classes exist in the dataset and view its distribution in the dataset. The image below shows the colour filled columns every column in the dataset.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd712a505c0>
```

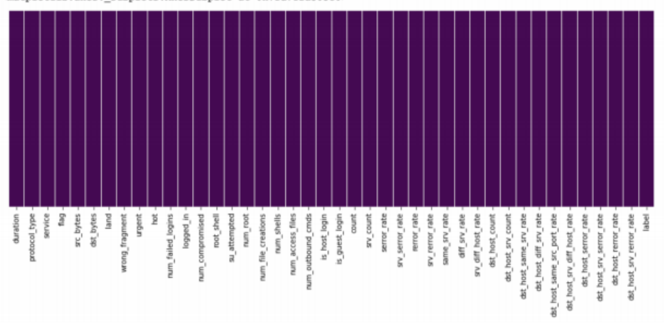


Fig-3: Heatmap of null / non-null values

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd71332a2b0>
```

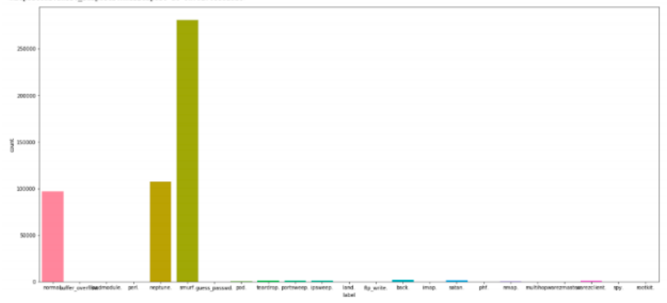


Fig-4: Histogram showing the frequency of instances in the Dataset

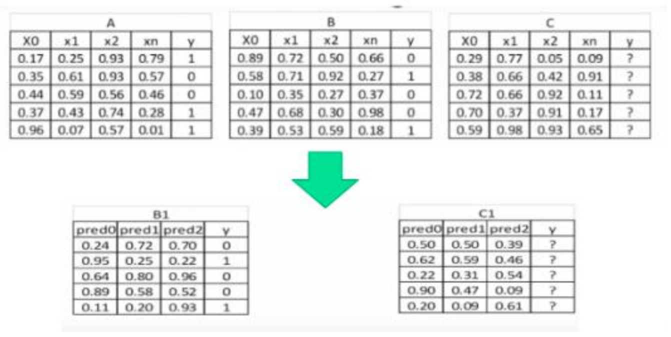
So, once the check for missing values is done, we can go onto analyzing the class frequency by plotting the number of times the particular label existed in the dataset. Using this analysis, we can conclude that majorly the attacks we're smurf, neptune or it was a normal packet and rest are present minutely.

5.3 Model Selection

Before we go on to create an ensemble of various models, we select a few models that can be used to classify the data we have, we selected about 5 models - Decision Trees, Random forests, Naive Bayes, Logistic Regression and Multi-layered perceptron. Initially, we train the dataset on the models individually to gauge their performance and accuracy.

5.4 Ensembling

Once we have tested our training data on the individual models, we go on and implement an ensemble of all the different models and gauge it's accuracy and compare it with the individual classifiers. We then try different combinations of models and create an ensemble with the selected models, in the end we just select the one which is the most efficient computationally and most accurate. To perform ensembling we used two of the basic techniques, Stacking and adaboosting. But, before we created the ensemble, we trained and tested all the individual models, including - Logistic Regression, Decision Tree, MLP, Naive Bayes. The following image shows the accuracies gauged. We created a stacking model, which consists of 2 layers the first layer consists of 3 models - MLP, Decision Tree and Naive Bayes Once, this layer is trained on the validation set the outputs are stored as a set of columns forming a n column table with n being the number of classifiers in the first level. In the second level, we have used a random forest classifier which works as an aggregate model learning from the outputs of the first level models. Image below depicts how exactly the stacking is carried out.



A

X0	x1	x2	xn	y
0.17	0.25	0.93	0.79	1
0.35	0.61	0.93	0.57	0
0.44	0.59	0.56	0.46	0
0.37	0.43	0.74	0.28	1
0.96	0.07	0.57	0.01	1

B

X0	x1	x2	xn	y
0.89	0.72	0.50	0.66	0
0.58	0.71	0.92	0.27	1
0.10	0.35	0.27	0.37	0
0.47	0.68	0.30	0.98	0
0.39	0.53	0.59	0.18	1

C

X0	x1	x2	xn	y
0.29	0.77	0.05	0.09	?
0.38	0.66	0.42	0.91	?
0.72	0.66	0.92	0.11	?
0.70	0.37	0.91	0.17	?
0.59	0.98	0.93	0.65	?

B1

pred0	pred1	pred2	y
0.24	0.72	0.70	0
0.95	0.25	0.22	1
0.64	0.80	0.96	0
0.89	0.58	0.52	0
0.11	0.20	0.93	1

C1

pred0	pred1	pred2	y
0.50	0.50	0.39	?
0.62	0.59	0.46	?
0.22	0.31	0.54	?
0.90	0.47	0.09	?
0.20	0.09	0.61	?

Fig-5: Displays the process of ensembling

Here, A is a training set, B is a validation set and C is a testing set and B1 contains the results of each model on the validation set stacked as columns. Similarly, the same stack of models is tested on the test set C and the corresponding output from each model is stacked column-wise to form the table C1 whose values will be predicted once the model on the second level (Random Forest) is trained on the table B1. Stacking resulted in an accuracy of 91.4% as it can be seen in the comparison chart above, which is a significant improvement in the accuracy, that is from an average of 80% using the individual models. We can see there was an overall improvement of 10% using the ensemble model instead of using the individual model which is great! Because it accounts to around 50,000 more entries being classified correctly from the test set. Apart from using the stacking model, we also tried using the AdaBoost classifier on a decision tree which also showed a significant improvement in its accuracy as it can be seen from the image below.

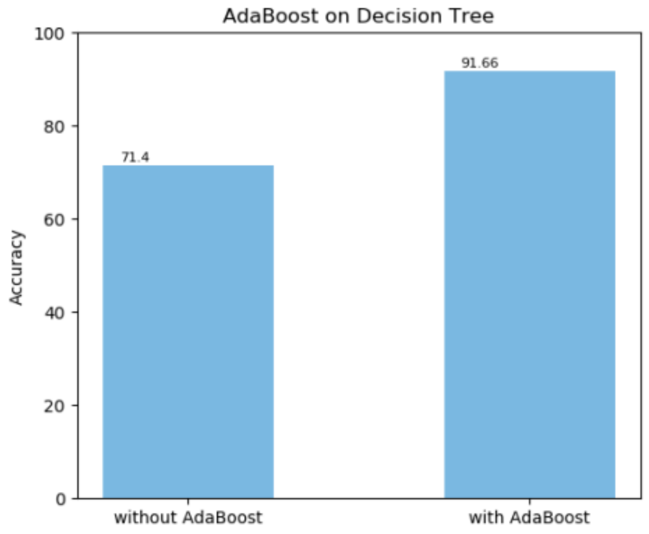


Fig-6: AdaBoost on Decision Trees

We can see there was an overall improvement of 20% using the ensemble model instead of using the individual model which is quite significant. Because it accounts to around 100,000 more entries being classified correctly from the test set.

5.5 Performance Metrics

Precision, Recall and Accuracy are the 3 main performance metrics use widely to evaluate machine learning models:

Precision:

Precision answers the question of what proportion of the positive identifications was actually correct. Hence, it can be written as the fraction number of correctly identified positive class tuples to the total number of positive class tuples present and can be mathematically written as follows:

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

Recall:

Recall gives the proportion of the actual positive tuples that were identified correctly. It can be mathematically written as follows:

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

Accuracy:

Accuracy is a metric used to evaluate classification models, it is measured as the fraction of the predictions that the model got right. The formal definition for accuracy is as follows:

$$Accuracy = \frac{Number of correct predictions}{Total number of predictions}$$

In case of binary classification, accuracy can also be defined in terms of positives and negatives as follows:

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

6. OBSERVATIONS

We can analyze the overall comparison chart comparing the accuracy of the individual models and the stacked ensemble model. We can see there is an overall increase in the accuracy when compared to all the individual models by about 10% meaning which 50,000 more test entries were classified correctly using the ensemble model. The improvement is quite significant meaning the project's aim of improving the accuracy using ensembling techniques was done successfully.

Table -1: Average Accuracy rate

Classifier	***Accuracy (%)***
Logistic Regression	80.6
Decision Tree	71.4
Multi-layer perceptron	88.8
Naive Bayes	76.7
Ensemble of Models	**91.4**

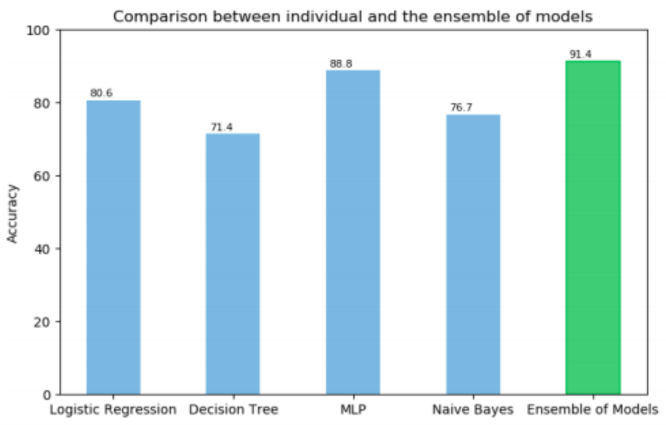


Fig-7: Final Comparison of accuracies

7. CONCLUSION AND FUTURE SCOPE

The following machine learning algorithms - Random Forest, Decision Trees, Multi-Layered Perceptron and Naive Bayes were tested and evaluated for their performance on the KDDCup99 Dataset and so to conclude we can say that our ensemble model has proven to have worked quite well with an overall accuracy on the test data of 91.4% with an overall improvement of about 10% when compared to classic individual classifiers.

To make the project commercially viable we shall implement the model into a network analysis software such as wireshark, allowing us to be able to first store all the present data of packets in the network traffic, dynamically update the dataset and then run the ensemble model on this updated data to classify the incoming packets. Doing this will enable us to classify network packets in real-time and prevent any type of
malicious attacks such as a DoS attack.

REFERENCES

[1] ACM Computing Surveys 50. "A Survey on EnsembleLearning for Data Stream Classification". In: (2016).

[2] "Applying artificial neural network and extended classi-fier system for network intrusion detection". In: (2013).

[3] F Haddadi, S Khanchi, M. Shetabi, V Derhami. "In-trusion detection and attack classification using feed-forward neural network". In: (2010).

[4] "Hide: a hierarchical network intrusion detection sys-tem using statistical preprocessing and neural network classification". In: (2001).

[5] Deokjai Choi Huy Anh Nguyen. "Application of data mining to network intrusion detection: classifier selec- tion model". In: (2008).

[6] "Intrusion detection and attack classification using feed- forward neural network". In: (2010).

[7] Mohammad Almseidin. "Evaluation of Machine Learn-ing Algorithms for Intrusion Detection System". In: (2018).

[8] Rohit Kumar Singh Gautam. "An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms". In: (2018).

[9] UK Springer-Verlag London. "MCS '00 Proceedings of the First International Workshop on Multiple Classifier Systems". In: (2018).

[10] Xiaolin Li Xiaoyong Yuan Chuanhuang Li. "Identifying DDoS Attack via Deep Learning".

[11] Guidong Zhang Ying Wang Guidong Zhang IYongjun Shen. "Research on Intrusion Detection Model using ensemble learning methods". In: (2017).