Indian Institute of Information Technology, Allahabad

6th Semester

Mini Project

Intrusion detection using Ensemble Learning

Supervised By:
Dr. O. P. Vyas

Submitted By:
IIT2016007
IIT2016038
IIT2016045
IIT2016134
IIT2016501

CERTIFICATE FROM SUPERVISOR

I do hereby recommend that the mini project report prepared under my supervision, titled **"Intrusion detection using Ensemble Learning"** be accepted in the partial fulfillment of the requirements of the completion of mid-semester of sixth semester of Bachelor of Technology in Information Technology.

Date:
Place: Allahabad

Supervisor:
.....
Dr. O. P. Vyas

Abstract

Nowadays it is very important to maintain a high level security to ensure safe and trusted communication of information between various organizations. Intrusion Detection System is a new safeguard technology for system security after traditional technologies, such as firewall, message encryption and so on. In the last three years, the networking revolution has finally come of age. More than ever before, we see that the Internet is changing computing, as we know it. The possibilities and opportunities are limitless; unfortunately, so too are the risks and chances of malicious intrusions. It is very important that the security mechanisms of a system are designed so as to prevent unauthorized access to system resources and data. However, completely preventing breaches of security appear, at present, unrealistic. We can, however, try to detect these intrusion attempts so that action may be taken to repair the damage later. This field of research is called Intrusion Detection. An intrusion detection system (IDS) is a device or software application that monitors network system activities for malicious activities or policy violations and produces reports to a Management Station. In this paper, we will look importance of IDS with different approaches. We have presented a really unique approach to solve this problem using the various machine learning algorithms, enabling us to extract the good part of various models and combining them into singular model.

Contents

1 INTRODUCTION

The project 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.

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.

The remaining report is arranged as follows: Section 1 explores ensemble learning in more detail. Section 3 defines the problem. Section 4 discusses the literature review done. Section 5 deals with the proposed methodology to solve this problem, while section 6 elicits the requirements(both hardware and software) of the model. Section 8 shows the results we achieved.

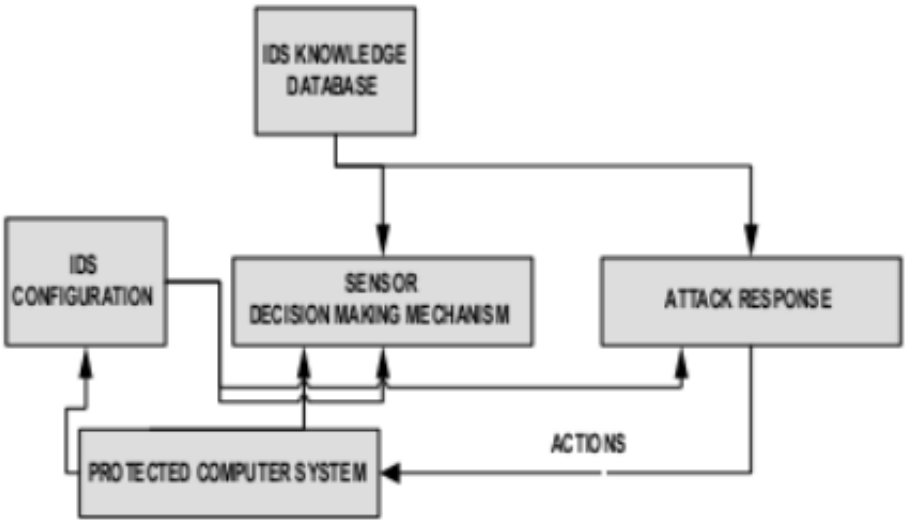


Fig. 1 Intrusion detection system

1.1 ENSEMBLE LEARNING

Ensemble learning is a machine learning paradigm where multiple learners are trained or designed to solve the same problem. In contrast to ordinary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use. Ensemble methods usually produces more accurate solutions than a single model would.

1.1.1 Reasons for using Ensemble

- **Too much or too little data** If the data is too much for a single model to learn, the data set can be partitioned into various subsets and each partition can then be used to train a separate classifier which can then be combined using an appropriate combination rule. And if the data is too little, bootstrapping can be done to train different classifiers and combining their results.
- **Divide and Conquer** If a problem is too difficult for the classifier to solve, or that the boundary separating different classes is too complex, or may the case of an outlier, an appropriate combination of linear classifiers may be used which can learn any non-linear boundary following a divide and conquer strategy by dividing data sets into easier to learn portions and giving each set to one classifier.
- **Data Fusion** If you have data from multiple sources with complementary information, merging this data together improve accuracy of the classification decision is called data fusion. And then train a separate classifier for one feature set and keep the decision weight of that classifier high for that feature.

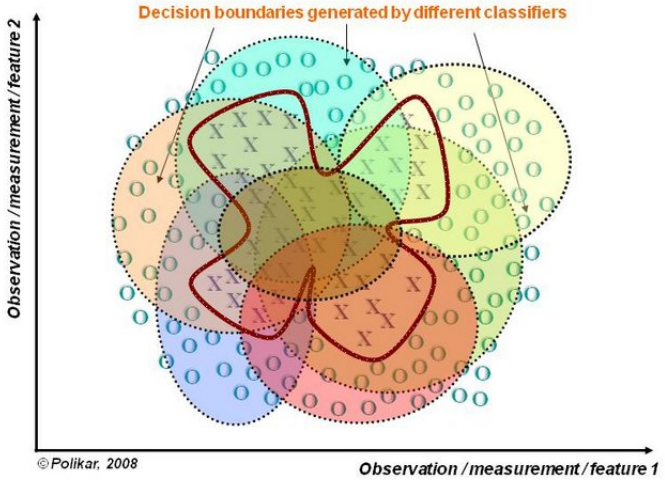


Figure 1: A combination of several circular boundaries can realize this complex boundary.

- Other Reasons

1.) Statistical reason -When the amount of training data available is too small for the learning algorithm to cover the whole search space, there may be several models which work well on training data and give almost equal accuracy and by averaging the outcomes of these models we can approximate to a point very close to the actual result.

2.) Computational Reason - Learning algorithms work in such a way that they may get stuck in a local optima, it may face computational problems in finding the best result so we can use an ensemble model where all the models have different starting points may result in a outcome better than a single learning algorithm.

3.) Representational reason - In cases when the chosen model cannot properly represent the sought decision boundary, taking their weighted sums can expand their search space.

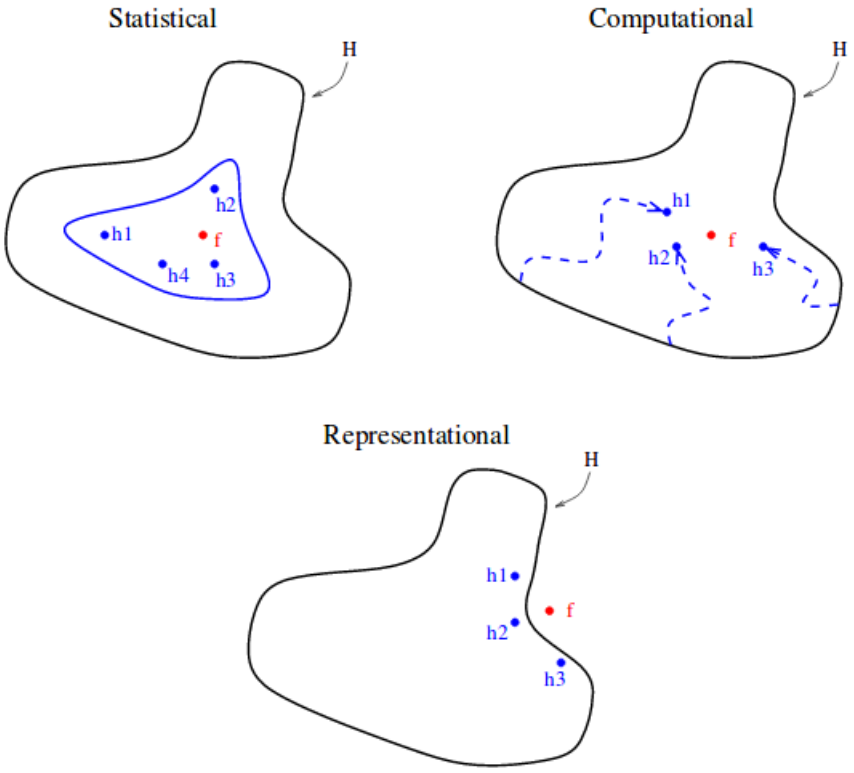


Fig. 2. Three fundamental reasons why an ensemble may work better than a single classifier

2 MOTIVATION

We are seeing a rapid upsurge in the frequency and complexity of DDOS attacks. According to statistics published in the 13th Annual Arbor Worldwide Infrastructure Security Report by netscout, there were 7.5 million DDoS attacks in 2017[1], a 20 percent increase over the last one year.Fifty-six percent of firms affected by DDoS and several other intrusion attacks experienced a financial impact between $10,000 and $100,000, almost double the proportion from 2016.

Solutions for this problem have evolved over the years. Out of the several possible solutions to this problem, one of them involves identifying the malicious packets and redirecting them to a fixed address(like that of a router). The most frequently implemented solution till date is to create firewalls. But these are complex solutions having hardware requirements as well. Thus, we see it is imperative for us to find easier solutions to these problems.

One way for identifying the malicious packets is by using machine learning, and feeding a data set into a multi-class classification model. Previous work done in this field points out to the fact that no single model has been able to give us optimal results.[2] Thus, here we try to create an ensemble of several models and achieve higher levels of accuracy and precision than the one that can be achieved by the models individually.

3 PROBLEM DEFINITION

The domain of our research is inclined towards network security, but the problem that we're trying to solve applies concepts of machine learning more specifically. We will combine models creating an ensemble model to optimize the process of classification and thus gauge better results in the end.

The problem discussed in this report is concerned with the intrusions that occur via network packets, so as to tackle this major problem of whether an incoming packet is normal or is a malicious packet. We use various machine learning models producing an end result model that is able to classify accurately and describe the type of malicious packet (DOS, U2R, R2L, PROBE) So, here we shall try and build an IDS - Intrusion detection system, an intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

To get more insight about the attacks that we shall classify, the 4 main categories of malicious packets are described below :

- **R2L** (Remote to user) Attack :- sending packets to a machine to which it doesn't have access in order to expose it's vulnerabilities.

- **DoS** (Denial of service) Attack :- makes a computing resource too busy to serve legitimate networking requests and hence denying users access to a machine.

- **U2R** (User to Root) Attack :- hacker tries to get the access rights from a normal host in order, for instance, to gain the root access to the system.

- **PROBE** :- scanning a machine to exploit it later.

The main challenge is that attackers are always keeping novelty in their tools and techniques in exploiting any kind of vulnerabilities. Hence, it is very difficult to detect all types of attacks based on single fixed solutions. For that intrusion detection system (IDS) became an essential part of network security. It is implemented to monitor network traffic in order to generate alerts when any attacks appear. IDS can be implemented to monitor network traffic of a specific device (host intrusion detection system) or to monitor all network traffics (network intrusion detection system) which is the common type used.

4 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[2] tried solving this problem using a variety of classification models, like MLP, decision trees, random forests, logistic regression. However, they concluded that t there is no single machine learning algorithm which can handle efficiently all the types of attacks. The decision table (rules base classifiers) achieved the lowest false negative value of (0.002), but it was far from the highest accuracy rate detection. On the other hand, Bayes network classifier had the highest value for correctly detecting the normal packets.

Rohit Kumar Singh Gautam[3] 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, he notes in the conclusion that there is still scope of improvements to propose systems which are able to detect all types of attacks can reduce the feature set by feature selection and Ensemble Method with the help of different classifier.

Independently, Ying Wang , Yongjun Shen and Guidong Zhang[4] tried using random tree and bayesian network for this multi class classification problem. While the results were satisfactory, they did conclude that Bayesian network presents higher accuracy for small sample classes, but have lower accuracy rate than other classifiers. On the contrary, RandomTree present good performance for big sample but not good at small sample. 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 sued to achieve far more optimal results than those that can be reached via using a single model.

5 PROPOSED METHODOLOGY

5.1 Classification Techniques for Intrusion Detection

5.1.1 Individual Classifiers

Classification techniques commonly used for classifying intrusion detection datasets are: Random Forest, Decision Trees, Naive Bayes and Multi Layer Perceptron. Most of these approaches directly apply standard methods to the publicly available intrusion detection datasets.

- ***Naive Bayes -*** Naive Bayes Classifier refers to the group of probabilistic classifiers. It implements Bayes theorem for classification problems. The first step of Naive Bayes classifier is to determine the total number of classes (outputs) and calculate the conditional probability for each dataset classes.After that the conditional probability would be calculated for each attribute. It has the ability to work with discreet and continuous attributes. Belief Network supports presenting independent conditional probability based on understanding framework. We are using Guassian Naive Bayes.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(x|c) = P(x_1|c) * P(x_2|c) * ... * P(x_n|c)P(c)$$

- ***Multi Layer Perceptron -*** In MLP, the training phase is typically implemented in a long period of time. MLP algorithm can be implemented with various transfer functions e.g. Sigmoid, Linear and Hyperbolic.We're using the softmax function for the output layer since its a multiclass classification problem. The number of outputs or expected classes and number of hidden layers are important design considerations of the MLP algorithm implementations. Features with larger weights imply greater effectiveness towards classification and vice-a-versa.

- ***Logistic Regression -*** Logistic regression is a predictive analysis used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. The dependent variable should be dichotomous in nature. There should be no outliers in the data, and no high correlations (multicollinearity) among the predictors. Multiclassification is done using the one-vs-all scheme.

$$Sigmoid(a) = \frac{1}{1 + e^{-a}}$$

- ***Decision Trees -*** Decision tree classifier is a systematic approach for multiclass classification. It poses a set of questions to the dataset (related to its attributes/features). The decision tree classification algorithm can be visualized on a binary tree. On the root and each of the internal nodes, a question is posed and the data on that node is further split into separate records that have different characteristics. The leaves of the tree refer to the classes in which the dataset is split. Its scalable for large datasets.

5.1.2 Ensemble Classifiers

- ***Random Forest -*** The main goal of this algorithm is to enhance tree classifiers based on the concept of the forest. They can be implemented to handle noise values and outliers. There is no re-modification process during the classification step. To implement this algorithm, the number of trees within the forest should be figured because each individual tree within a forest predicts the expected output. Max voting is applied on the expected output for classification.

- ***Ensemble Classifier Approach -*** Ensemble classification technique is advantageous over single classification method. It is combination of several base models and it is used for continuous learning. Ensemble classifier has better accuracy over single classification technique.
  Bagging and boosting are two of the most well-known ensemble learning methods.
  - ***Stacking -*** Stacking uses predictions from multiple models (for example decision tree, KNN or SVM) to build a new model.
  - ***Bagging -*** Bagging performs random sampling(with replacement) to train different classifiers.
  - ***Boosting -*** Boosting performs sampling based on a distribution that is continuously updated to increase the chances of sampling instances that are often mis-classified.

Technique	Pros	Cons
Bagging	Stable against noise	Needs many comparable classifiers
Boosting	Improves margins	Unstable against noise

5.2 Designing the proposed scheme for Intrusion Detection

The methodology of designing the proposed scheme is divided into three phases: Normalization, Feature Selection and Ensembling.

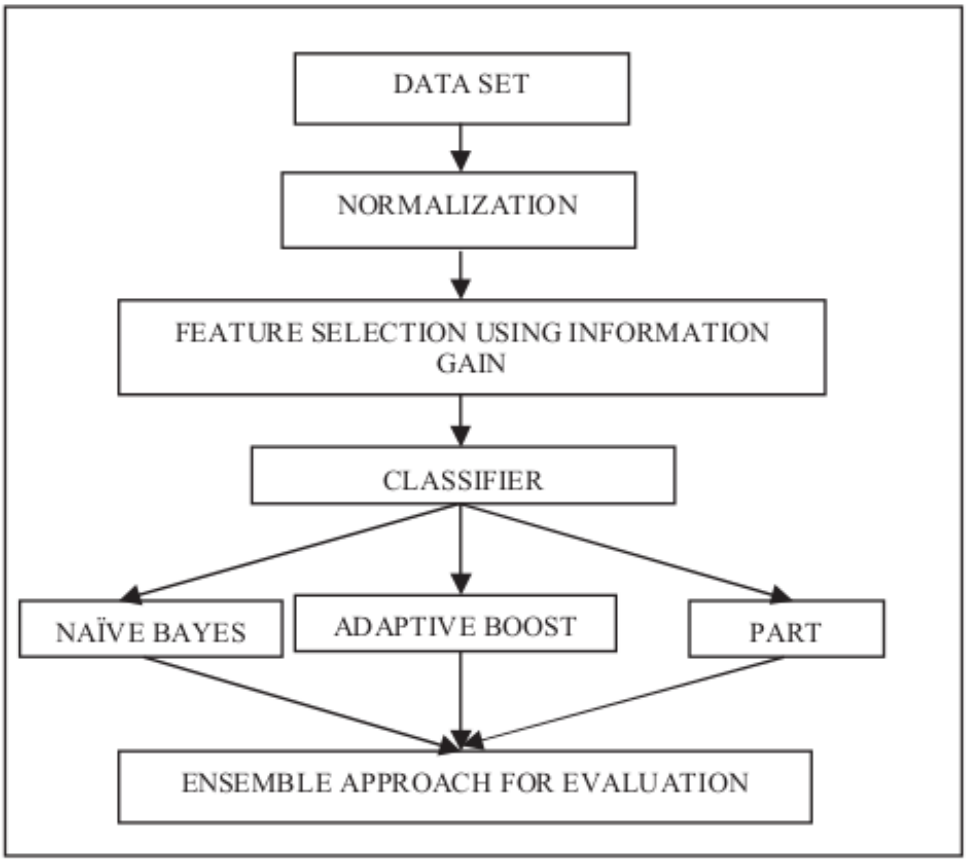


Figure 1. General Methodology

- ***Normalization -*** In the primary analysis, the frameworks are prepared utilizing all the 41 features. This helps to reduce the dimensionality of dataset.
- ***Feature Selection -*** The Filter Method uses statistical approaches to provide a ranking to the features. After providing the ranking, user can choose best k features for the experiment (top 24 or 41) according to their requirement.
- ***Ensemble Approach -*** Ensemble Approach is performed in the third analysis by utilizing to choose some best components as opposed to utilizing all the 41 features. They are then tried with Naïve Bayes, stacking and Adaptive boost and the outcomes analysed.

6 REQUIREMENTS

6.1 Dataset

Data set KDDcup 99 is used to carry out the experiment which was derived from the 1998 DARPA Intrusion detection Evaluation program. It is the most widespread dataset collected over a period of nine weeks for a LAN simulating a typical U.S. Air Force LAN. The dataset contains a collection of simulated raw TCP dump data, where, multiple intrusions attacks was introduced and widely used in the research community. From seven weeks of network traffic,four gigabytes of compressed binary TCP dump training data was processed into five million connection records. Similarly,two weeks of test data yielded about two million connection records. The dataset contains 4,898,430 labeled and 311,029 unlabeled connection records. The labeled connection records consist of 41 attributes.

In network data of KDD99 dataset, each instance represents feature values of a class, where each class is categorized either normal or attack. The classes in dataset are characterized into one normal class and four main intrusion classes: Denial of Service (DoS), Probe, User-to-Root (U2R), Remote-to-Login(R2L). Smurf attacks are in majority.

- ***DoS Attacks-*** Use of resources or services is denied to authorized users.
- ***Probe Attacks-*** Information about the system is exposed to unauthorized entities.
- ***User to Remote Attacks-*** Access to account type of administrator is gained by unauthorized entities.
- ***Remote to Local Attacks-*** Access to host gained by unauthorized entities.

6.2 Hardware Requirements

The project required a huge dataset to be trained (800 MB), first on various individual models, and then on ensemble models. Since its highly computationally heavy, a normal desktop machine wasn't enough. Most of the computation involved is of same type and required presence of a dedicated CPU boost the overall throughput of the experiment.The hardware requirements of the project are:

- A GPU machine
- A capable workstation

6.3 Software Requirements

The project is built on python and requires a number of libraries to handle tasks from data manipulation,data analysis etc.Since this project is based on Python, it is platform independent and so it only requires the following dependencies irrespective of the underlying operating system.

- ***Python 3 :*** It is the primary development language used to develop the project.
- ***Libraries Required :*** There are a number of libraries which are required to perform various computations. Some of the libraries are:
  - ***numPy :*** Library to adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
  - ***pandas :*** A data analysis toolkit used to provide real world data manipulation and analysis in Python.
  - ***sckit-learn :*** To use various machine learning regression models in the project.
  - ***matplotlib :*** To use the plotting function in the experimental analysis.

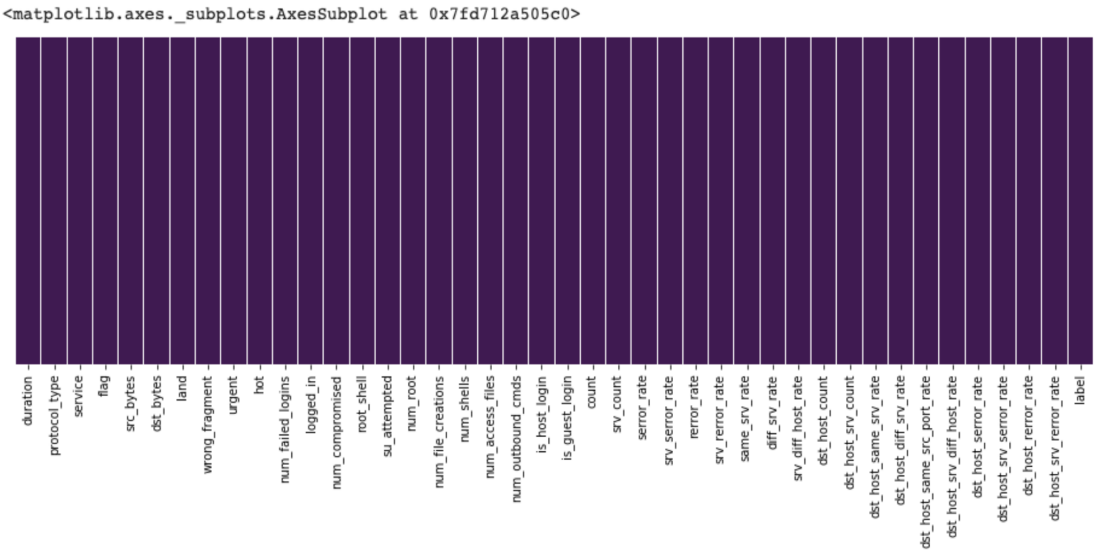
7 IMPLEMENTATION PLAN

7.1 Steps to implementation

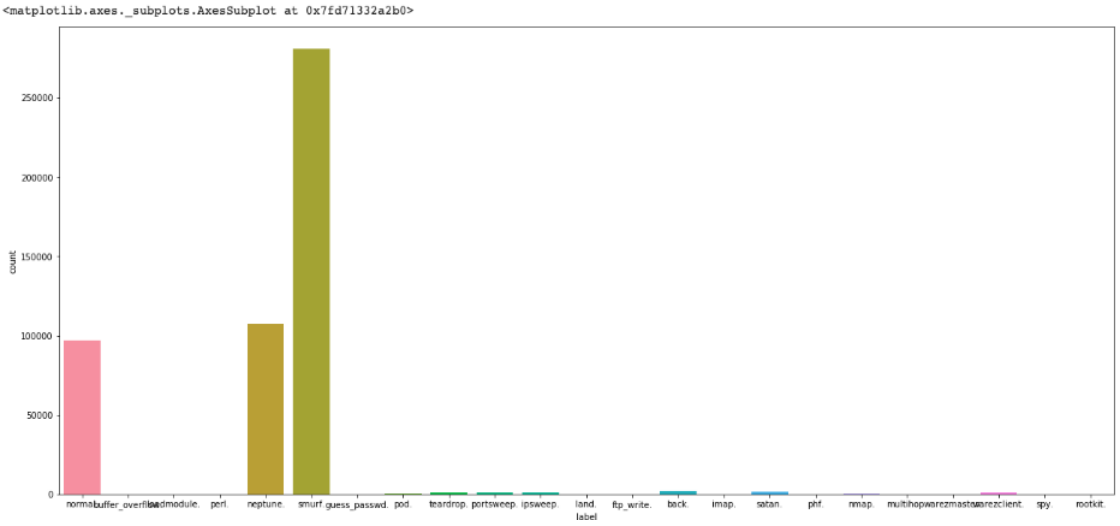
- ***Data Acquisition -*** The implementation of our project begins with searching for a suitable data-set that would suit the purpose of our research. We found the KDD99 (Knowledge Discovery in Databases) dataset to be the most suitable with about 5 million entries covering nearly every real-life scenario.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

- ***Dataset Cleaning -*** 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). 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 that is 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.



So, once the check for missing values is done, we can go onto analyzing the class frequency by plotting the number of time the particular label existed in the dataset.



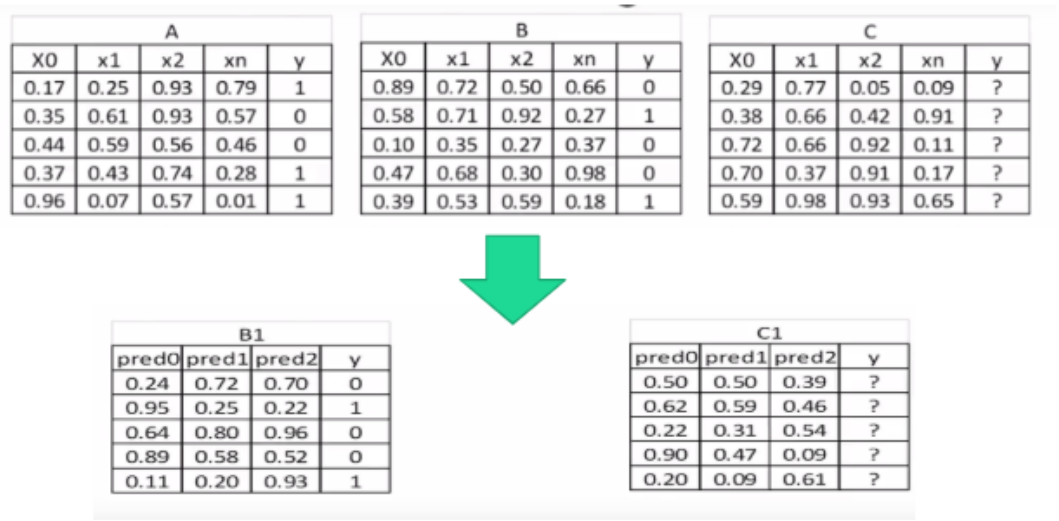
Using this analysis, we can conclude that majorly the attacks we're smurf, neptune or it was a normal packet and rest are present minutely.

- ***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.

- ***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 combination 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 create the ensemble, we trained and tested all the individual models, these 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.

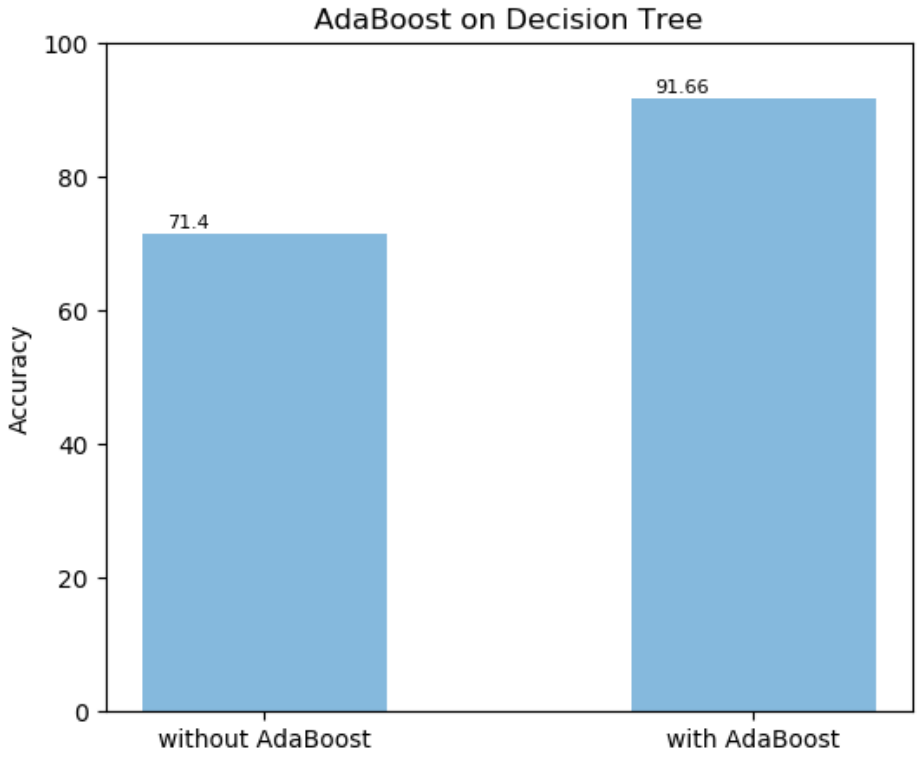


  Here, **A** is **training set**, **B** is **validation set** and **C** is **testing set** and **B1** contains the results of each model on the validation set stacked as columns. Similarly, the same stack of models is tested on the test set **C** and the corresponding output from each model is stacked 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**.

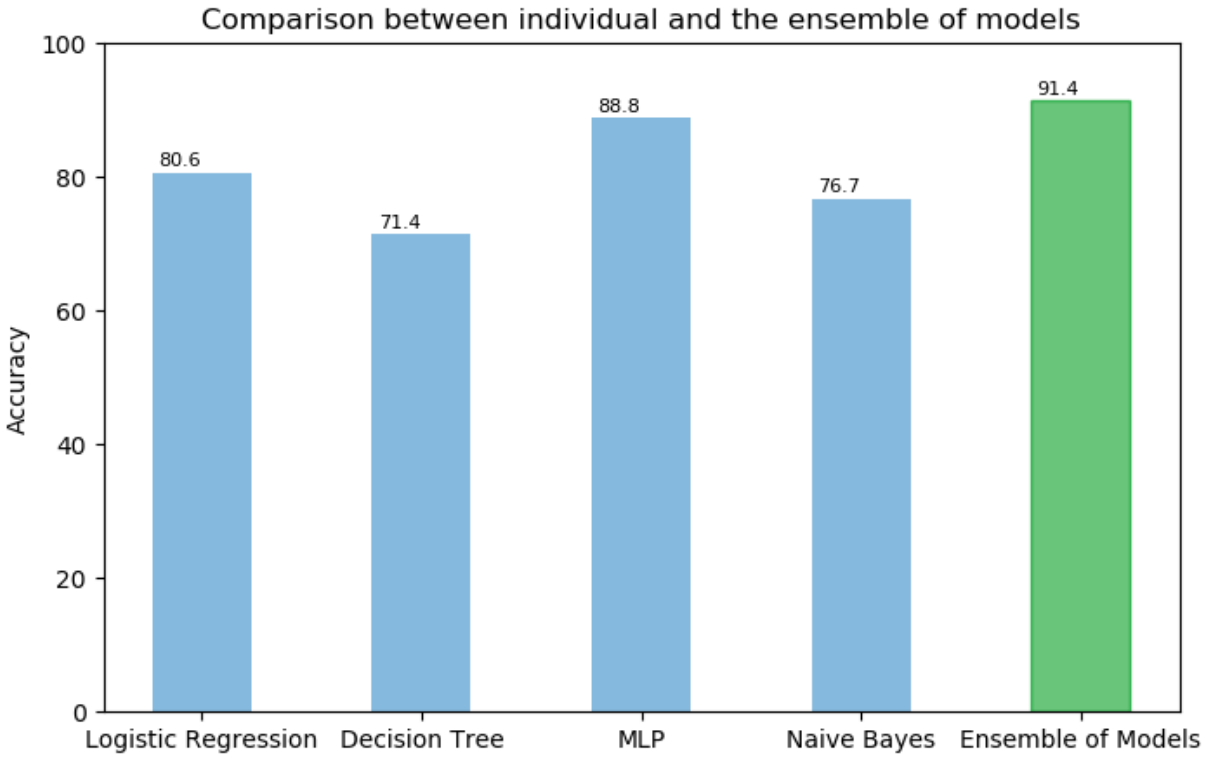
Stacking resulted in an accuracy of **91.4%** as it can be seen the comparison chart above, which is a significant improvement in the accuracy, that is from an average of **80%** using the individual models. We can see there was an overall improvement of **10%** using the ensemble model instead of using the individual model which is great! Becuase 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.



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

8 Observations



To conclude with the report, 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.

9 CONCLUSION

In this report, several experiments were performed and tested to evaluate the efficiency and the performance of the following machine learning classifiers: Random Forest, Decision Table, MLP, Naive Bayes. All the tests were based on the KDD intrusion detection dataset. To conclude we can say that using the ensembling model on the datatest has proven to have worked quite well with an overall accuracy on this test data of **91.4%** with an overall improvement of about **10%** when compared of classic individual classifiers.

10 FUTURE SCOPE

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] https://www.netscout.com/news/press-release/complexity-ddos-attacks.

[2] Research on intrusion detection model using ensemble learning methods. 2016.

[3] Mohammad Almseidin. Evaluation of machine learning algorithms for intrusion detection system. 2018.

[4] ACM Computing Surveys 50. A survey on ensemble learning for data stream classification. 2016.

[5] Rohit Kumar Singh Gautam . An ensemble approach for intrusion detection system using machine learning algorithms. 2018.

[6] UK Springer-Verlag London. Mcs '00 proceedings of the first international workshop on multiple classifier systems. 2018.

[7] Xiaolin Li Xiaoyong Yuan, Chuanhuang Li. Identifying ddos attack via deep learning.

[8] IYongjun Shen Guidong Zhang Ying Wang Guidong Zhang. Research on intrusion detection model using ensemble learning methods. 2017.