A Distributed Intrusion Detection System for Computer Networks using Hybrid Model on Apache Spark Framework

MSc Research Project
Data Analytics

Harshitha Deenadayal
Student ID: x18136231

School of Computing
National College of Ireland

Supervisor: Dr. Pierpaolo Dondio

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Harshitha Deenadayal
Student ID:	x18136231
Programme:	MSc Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Pierpaolo Dondio
Submission Due Date:	12/12/2019
Project Title:	A Distributed Intrusion Detection System for Computer Networks using Hybrid Model on Apache Spark Framework
Word Count:	5328
Page Count:	22

Signature:	Harshitha Deenadayal
Date:	29th January 2020

A Distributed Intrusion Detection System for Computer Networks using Hybrid Model on Apache Spark Framework

Harshitha Deenadayal
x18136231

Abstract

Intrusion detection system (IDS) is the major aspect of security solutions for every organization relying on internet services as well as for, internet providers who offers internet services. IDSs aids proactive monitoring of the network in order to detect potential threats and protect organizations from possible data and financial loss. Due to its high importance in real world, it has been a major interest for research studies. IDSs adopts data mining techniques to analyze historical network-connection information to detect attacks. The efficiency of IDSs are measured in terms of the accuracy of prediction and its error rate. It is also important that models perform in near real-time to detect threat before the control of network resources is compromised. This creates a requirement for IDS to be built on high computing environment. This paper experiments on well-performing classification algorithms namely: support vector machine (SVM), decision tree, random forest, gradient boosting and an ensemble of all these four algorithms. Correlation-based feature engineering techniques namely ARM (Association Rule Mining) and Variable importance are applied to enhance the performance. The performance of these models are measured and compared in terms of accuracy and error rate. Additionally, this paper conducts experiments on both centralized and distributed (Apache Hadoop and Spark) environments to compare the time taken to train the data mining models for IDS. The experimental results shows that tree-based algorithms are better classifiers for IDSs and models are trained mush faster in distributed (Hadoop and Spark framework) environments in comparison to centralized.

Keywords: *Intrusion Detection system, SVM, Decision Tree, Random Forest, Gradient Boosting, Apache Spark, NSL-KDD dataset.*

1 Introduction

1.1 Background and Motivation

From the past decade, big data has become ubiquitous in all the aspects of every business organization and computer networks is no exception. Advent of smart devices and IoT has spread the web of computer networks to a great extent than ever before. These connected-networks play an important role in generating, collecting and transporting big data along with its primary purpose of sharing files and connecting devices. Such networks are enriched with an overwhelming amount of sensitive information. Due to this, they are prone to frequent attacks by people with malicious intent, commonly known as hackers, cyber criminals or intruders. Apart from traffic data, they also carry content-related information, connection-related information and timestamps. Network-related details are commonly analyzed by telecom operators with an intention to optimize the network for better availability, reliability and security to offer best customer experience which can lead to business improvement. Networks carry huge enterprise data and customer-related data and hence, network security has been one of the leading concerns of internet providers. Network-related information can be leveraged to detect abnormal activities proactively, any before potential attack on computer resources. A Intrusion detection system analyzes network related information and predicts whether a connection is normal, or an attack based on the historical data. It is also essential that IDS is built on high-computing distributed platform such as Apache Hadoop and Apache Spark, as network-related information can be in high volumes and computation speed is an essential factor along with accuracy of prediction. Although many research studies are being carried-out to predict the intrusion accurately, less importance has been given to computation speed. This paper aims to predict intrusion in the computer networks with improved accuracy along with high computational speed using Apache Spark that can aid near real-time detection of attacks in computer networks.

1.2 Research Question

"What is the improvement in performance of a predictive model build using ensemble of hyper-tuned algorithms on a Spark framework for detecting intrusion in computer networks, in terms of accuracy and computation time?"

1.3 Research Objectives

In the year 2018, 30% of the cyber-attacks experienced by companies in the U.S. are caused due to network intrusion[1]. Although, they are many software and hardware present for intrusion detection, the incidents of malware attacks have increased over years[2]. Figure 1 shows number of malware attacks between year 2015 to 2018.

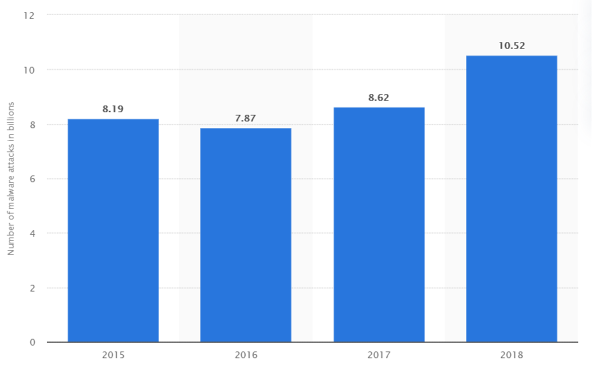


Figure 1: Incidents of Malware attacks

[1]https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/
[2]https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/

There is necessity for IDSs with high predictive ability and less false alarm rate. It is also importance to detect the threat in near real-time which requires high computing speed. This research project works towards the following objectives.

- To build intrusion detection system for computer networks using two-class classification algorithm.
- Train different models using data mining algorithms namely: Support Vector Machine, Decision Tree, Random Forest, Gradient Boosting and one hybrid model learning from all the other four models.
- Perform hyper-parameter tuning of all models to optimize its learning from train data.
- Evaluate and compare all the models for best performance with respect to its predictive ability and false alarm rate.
- Set-up a distributed environment to train the models, using Apache Hadoop and Spark for high-speed computation.
- Evaluate and compare the computation performance of centralized and distributed environments, in terms of time-taken to train the data mining models.

Rest of the the paper is as follows: Section 2 discuss related work, Section 3 explains methods adopted in this research ,Section 4 defines hardware and software requirements of the research , Section 5 illustrates the machine learning model implemented in the research experiments, Section 6 presents and discusses the results of the research experiments and final 7 concludes the research work and suggests future work.

2 Related Work

2.1 Introduction

People all over the world can connect with each other within seconds due the power of internet. Computer networks helps to connect all the smart devices in the world using internet. With its power of connecting world, internet comes with major drawback of security concerns. Computer networks are vulnerable to attacks like distributed denial of service (DDoS), access controls, data breach, data manipulation, etc which affects network availability, reliability, confidentiality and security. Cyber attacks not only lead to data and financial loss, but also cause reputational damage and potential client loss for telecom operators. In the recent times, the top kinds of network attacks are DDoS, Browser, SSL, Shellshock, Botnet, Brute Force, Backdoors, Moustafa et al. (2019). There are many instances of cyber-attacks happened in recent years that proves the impact of it. In the July of 2017, hackers of Equifax data breach stole credit card details of over 200,000 accounts. In the same year, major websites like Spotify, Twitter went unavailable due to DDoS attacks on their datacenter servers. Year 2017 also witnessed another major ransomware attack called "WannaCry" which took control of resources of over 10,000 organizations around 150 countries. Overall, second quarter of 2018 observed 50 percent increase in the DDoS attacks, Kim et al. (2019).Although, there are many reactive hardware and software solutions like anti-virus, anti-malware, firewalls developed over the years to prevent cyber-attacks, the number of incidents of network attack are still high. And current IDSs generate large number of false alarms in a day, leading to inaccuracy, Anwar et al. (2017). Pro-active or near real-time detection of threats helps to mitigate the attack prior hackers exploits the network resources. An IDS which is a combination of high accuracy predictive model and high computing framework helps to minimize the loss of intellectual properties for business organizations. Such IDS assist telecom operators and internet providers to proactively monitor and secure the network effectively.

2.2 Intrusion Detection System

An IDS is a system deployed to monitor and analyze the traffic in the computer networks for identifying potential threats in terms of CIA (Confidentiality, Integrity and Availability), Anwar et al. (2017), Inayat et al. (2016). A network-based IDS is set-up across a specific segment of a network to examine all the traffic. It only monitors for network-related information and cannot access the process running on host or the data stored within, Vijay (2017). There are two kinds of approaches for intrusion detection namely: Deterministic and Stochastic methods. In the deterministic approach, deterministic decision boundary features of connection are partitioned into 'normal' and 'attack' regions. Data mining algorithms like Support Vector Machine (SVM), Random forest, clustering and many more are adopted to determine the boundary. Whereas in Stochastic method, a probabilistic model is fitted, with respect to a reference data using Statistical Hypothesis Testing (SHT), Vijay (2017). This research experiment adopts binary/label-based IDS that labels a record as 'normal' or 'attack' based on historical data.

2.3 Data Mining techniques

Huge amount of data can be extracted, processed and analyzed by employing advanced data mining techniques, Bouzekri and Idrissi (2016). Data mining technique acts as an important aspect of IDS that looks for relevant patterns within the traffic. They assist in the threat analysis and aids system to take intelligent decision, Sangher (2019). Data mining algorithms namely random forest, bagged tress and ANN gives the best results in terms of accuracy, model training time and time taken to predict unknown instance, Sangher (2019). Other techniques like data pre-processing, feature engineering also assists in optimizing the performance of machine learning models. The most widely used feature reduction techniques are Association Rule Mining (ARM), Principal Component Analysis (PCA) and Independent Component Analysis(ICA). Out of these, ARM technique is employed for feature selection in this paper. ARM determines strongest rules between the values of the independent variables and calculates the correlation between them, Moustafa et al. (2019). ARM has been used in many IDS related studies like Nalavade and Meshram (2014), Moustafa and Slay (2015a), Moustafa and Slay (2015b) to predict threat with positive results. Decision engine is the most critical part in IDSs. Based on decision engines, IDSs are classified into six categories namely: clustering-, classification-, knowledge- , deep learning- , statistical- and combination- based as explained in Bhuyan et al. (2014), Resende and Drummond (2018) and Moustafa et al. (2018). Based on studies conducted by Horng et al. (2011),Wagner et al. (2011), Saber et al. (2017), Singh et al. (2014), Jirapummin et al. (2015), Kang et al. (2012) and Dua and Du (2016), classification techniques provides high predictive rate with low false positive rate, provided dataset is labelled correctly. Table 1 shows accuracy achieved by some of the other research studies.

Table 1: Comparison of IDS classification models on NSL-KDD dataset

Algorithm	Accuracy (in percentage)	Authors
Logistic Regression	95.1	Moustafa et al. (2019)
Naive Bayes	93.8	Moustafa et al. (2019)
Genetic	94.3	Hasan et al. (2017)
Ramp-ksvcr	98.8	Bamakan et al. (2017)
SVM	92.18	Ambwani (2003)
Random forest	92.30	Zhang et al. (2008)
FSVM	97.14	Gupta et al. (2013)

This paper conducts experiments to study the performance of variety of classification-based decision engines namely SVM, Decision tree, Random forest and Gradient boosting with parameter tuning for high predictive accuracy and less false positive rates. Then, develops an ensemble of these models to enhance the performance.

2.4 Apache Spark

Generally, IDSs needs to process huge amount of data for making intelligent decision. A distributed framework helps in storage big data effectively and perform computation efficiently in near real-time.
Apache Hadoop is a open source framework for distributed cloud computing and data storage. It is scalable and reliable framework that can handle failures at application layer. The two main components of Hadoop are Hadoop Distributed File System (HDFS) and Hadoop MapReduce, You (n.d.). HDFS is the distributed file system used by hadoop to store big data over many machines and provides high-throughput access to the data residing on clusters, Erraissi et al. (2017). MapReduce is a software that performs functions such as parallel processing and error-handling, You (n.d.). Hadoop's working is based on Master-Workers system. Master controls the global state of computation while workers execute the task directed by the master. The role of HDFS is to maintain the exchange of data among workers in shuffling manner, Aldinucci et al. (2019). Organizations like Facebook, Google, AOL and IBM has made use of Hadoop successfully for research and commercial purposes. For instance, Facebook stores internal logs and data sources on hadoop system and using it also for reporting, machine learning and other analytics tasks, You (n.d.). Apache Spark is an open-source framework for cluster computing that supports batch and real-time processing with inherent fault-tolerance and data-parallelism. Spark uses Directed Acyclic Graph (DAG) to process the data. It uses in-memory caching instead of on-disk to communicate data among processing units to enhance the performance of iterative processing, Aldinucci et al. (2019). Some of the main tasks of Spark are submission of jobs, scheduling resource, tracking and communicating between nodes, operating on data in a parallel manner, Chunduri (2019). Spark integrates with storage platform like Hadoop to access data while processing them. Additionally, Spark comes with programming interface for R and Python. The main components of spark that support machine learning are Spark MLlib, Spark GraphX, Spark SQL, Spark streaming, Chunduri (2019). This research project deploys both apache hadoop and spark for performing data mining on distributed system to measure and compare the computing performance.

3 Research Methodology

This research project follows standard CRISP-DM methodology from ideation till implementation as shown in Figure 2. CRISP-DM end-to-end data mining life-cycle completes in six stages. Following are reasons for adopting CRISP-DM methodology, Azevedo, Ana; Santos (2008). i)The defined stages are not rigid and can be customized as per data mining project. ii) All the stages are well-defined, structured and organized. iii) The discrete stages of the project can be easily explained. iv) It covers end-to-end data mining project.

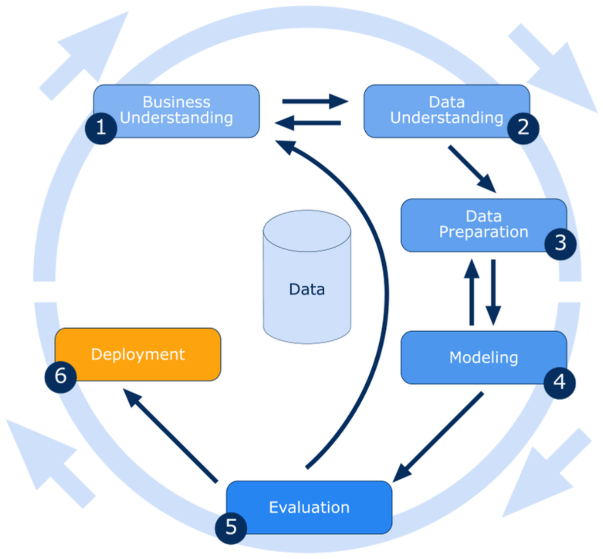


Figure 2: CRISP-DM Methodology

The six stages of CRISP-DM are Business understanding, Data understanding, Data preparation, Modelling, Evaluation and Deployment.

3.1 Business understanding

Due to the advent of digital technology, international business and trade has observed its peak and most of the transactions are happening online. It is of high importance to protect the sensitive information carried over internet to avoid any intellectual property theft or financial loss. Telecom operators and internet providers must proactively monitor the network to provide secure connections to its customers. IDS helps to identify unusual activities in the network and create alarms against them. IDSs must be able to detect alerts actively with high accuracy to take immediate action against the threats before it takes control of the network resources.

3.2 Data Understanding

Historical data essential for predictive analysis must be audited for its significance in predicting the unknown instances. A through analysis of the data features helps to understand the patterns and correlations. NSL-KDD dataset has been used for this research project to benchmark the performance of predictive models. NSL-KDD dataset is an improved version of KDD”99cup dataset which was collected from an intrusion simulated military network environment in the year 1999, Alanazi et al. (2010). This dataset does not contain redundant/duplicate records and hence produces un-biased results. It holds sufficient number of records for the train and test dataset for experimental purposes, Azevedo, Ana; Santos (2008). The NSL-KDD dataset has 41 attributes and a label assigned to each record, either 'normal' or 'attack'. The NSL-KDD dataset is downloaded from UNB (University of New Brunswick) website which is publicly made available. The dataset contains basic, content-related, host-related and time-related features of network connections. [3]

3.3 Data Preparation

Data preparation is the prerequisite task for data modelling. This involves pre-processing of independent and dependent variables of the dataset and transform it as per model's requirements. In this research experiment, feature selection is also carried out as a part of data pre-processing. Python and pyspark are the programming languages used for building models on centralized and distributed environments, respectively. With pyspark, apache spark can support Python language with machine learning libraries. For the purpose of exploratory data analysis and data modelling, the four categorical variables ('class', 'protocol type', 'service' and 'flag') are transformed into numerical values using 'LabelEncoder'. To avoid over-fitting problem, correlation-based feature reduction technique is applied, following feature selection based on variable importance graph. After these feature engineering, dimensionality of the features is reduced from 41 to 10. Two subsets of datasets are created with only independent variables in one and dependent variable in another. These subsets are further split into train (80%) and test (20%) for modelling and validation respectively.

3.4 Modelling

Data mining algorithms are applied on train dataset to learn the patterns from the historical data and build the data mining models that can predict the outcome of the unknown instances based on the learning. This paper adopts two-class classification technique to predict whether a connection on the computer network is 'normal' or 'attack'. Following data mining algorithms implemented for modelling to test and compare the performance: SVM, Decision tree, Random forest, Gradient boosting and hybrid of all four mentioned algorithms.

- **Support Vector Machine (SVM):** SVM is a data mining algorithm that plots training vectors in high-dimensional space and classifies them into different classes. It draws a hyper plane in the feature space that categories the data. This hyperplane is determined by a set of vectors from the training data. It consumes less time to build the model and hence supports near real-time detection of threats. It does not require large number of data points and dimensionality does not add additional complexity, Alanazi et al. (2010). Author Mill (2004) has employed SVM for network anomaly classification for best results.

- **Decision Tree:** Decision Tree algorithm branches classification problem into many sub-problems at every node of the tree Azevedo, Ana; Santos (2008). A node represents a condition of the attribute by which records must be classified. A decision

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

tree consists of root or parent node with high importance predictors, followed by lesser-important branch nodes and the final leaf nodes that indicates final output class, when further branching is not possible. Decision trees are considered for IDS as they work well with big dataset and provides high performance, Peddabachigari et al. (n.d.), even though it is a time-intensive processing.

- **Random Forest:** Random forest (RF) is an ensemble of decision trees which is used for both regression and classification problems. It constructs multiple decision trees during the training process to improve the accuracy and reduce the false alarm rate, Thaseen (2013), Chandra (2017). It offers low classification error in comparison to other traditional classification techniques. Its performance does not alter upon processing large datasets or removal of variables. However, due to the presence of many decision trees, the process of learning can get interrupted Chandra (2017). This project employs distributed environment to overcome interruption issue. In their paper, Farnaaz and Jabbar (2016) used random forest for classification and observed better accuracy in predicting the classes.

- **Gradient Boosting:** Gradient boosting is another ensemble machine learning algorithm that can perform both classification and regression, Alqahtani et al. (2019). It creates a model consisting of many weak learners (like decision trees). It performs loss function optimization on every weak learner, in an iterative manner. This results in a low loss function and a high prediction rate, Xia et al. (2017).
  Hybrid of techniques is widely experimented by many researchers. In Peddabachigari et al. (2007), ensemble of decision tree and random forest is applied for classification and accurate results were achieved.
  In this research, as an extension to traditional models, hyper parameters of the algorithms are tuned to its best estimators to improve the performance of the models. Additionally, hybrid of best performing models is build using 'voting classifier' technique and the improvement in the performance is compared.

3.5 Evaluation

The IDSs are evaluated by means of confusion matrix as they are classification problems. Confusion matrix provides counts of actual and predicted labels as shown in the Figure 3. For two-class detection problem, it produces a 2*2 matrix, wherein True positives (TP) and true negatives (TN) represents correctly predicted 'attack' and 'normal' instances, respectively, and in contrast, false positives (FP) and false negatives (FN) represents incorrectly predicted 'attack' and 'normal' instances, respectively, Bhuyan et al. (2014), Ahmed, Mohiuddin; Naser Mahmood, Abdun; Hu (2016), Moustafa et al. (2017).

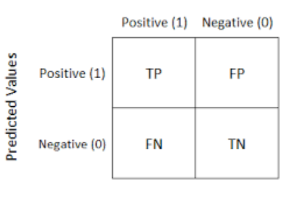


Figure 3: Confusion Matrix

Performance of the models are evaluated by means of following metrics.

Precision: Precision is the measure of correct positive prediction out of all the positive prediction made by the model, Chandra (2017) and is calculated using the equation (1).

$$Precision = TP/(TP + FP) \tag{1}$$

Recall: Also known as Sensitively, recall is the measure of correct positive predictions made by the model out of the total number of positives in the dataset, Chandra (2017) and is calculated using the equation (2).

$$Recall = TP/(TP + FN) \tag{2}$$

F1 score: F1 score is the harmonic mean of precision and recall Chandra (2017) and hence, it is a preferred metric for evaluating IDSs in cases where high false positive is not tolerable. F1 score is calculated using the equation(3).

$$F1score = 2\,(Precision * Recall)/(Precision + Recall) \tag{3}$$

Error Rate: Error rate is the number of wrong prediction divided by total number of unknown instances. It acts as a measure of quality of classifiers. It ranges between 0 and 1, 0 being best and 1 being worst. Error rate is calculated using the equation(4).

$$Errorrate = (FP + FN)/(TP + TN + FP + FN) \tag{4}$$

Mean Squared Error (MSE): Another measure of quality of classification model is MSE. It is a risk function which is the average squared difference between actual and predicted values. It is always positive between 0 and 1. Model is better when MSE is closer to zero.

Execution time: In this research experiment, the machine learning models are run on both centralized and distributed (apache hadoop and apache spark set-up) systems and the execution times are measured to analyse the computing performances.The time taken to train the models are measured in 'minutes'.

4 Design Specification

Centralized and distributed environments are set-up for the purpose of experiments on Intel(R) Core(TM) i7-5500U processor, CPU@2.4GHz, GPU: NVIDIA GeForce, RAM:16GB, Storage: 1 TB HDD and Operating system: Windows 10, 64-bit.

- Centralized environment is set-up on virtual machine with Ubuntu 64-bit OS, 2 CPU processors and 11GB RAM. Anaconda with Python version 3 is installed. Using ubuntu shell, jupyter notebook is launched on browser for writing machine learning codes in python.

- On the second part of research experiment, to check the performance variance in terms of time-taken to train the models, distributed environment is set-up using Apache Hadoop and Apache Spark on a linux virtual machine. As prerequisite steps, Java jdk version 8 and SSH (secure shell) keys are installed on Ubuntu OS. Apache Hadoop version 2.9.2 is downloaded, installed and configured. And the path is set between hadoop and java. Later, Apache spark 2.4.3 is downloaded and installed. Along with this, anaconda for python 2.7 is downloaded and installed on spark environment. Dataset is downloaded from University of New Brunswick onto the virtual machine and is copied into HDFS. Spark, hadoop's master and worker nodes are started using command line on ubuntu shell. Once all the services have started, jupyter notebook is launched on the browser using ubuntu shell.

5 Implementation

Figure 4 depicts the flow chart of research implementation, from data collection to performance visualization of all the models. Firstly, required dataset for experiments i.e, NSL-KDD dataset is downloaded from the trusted website of University of New Brunswick.

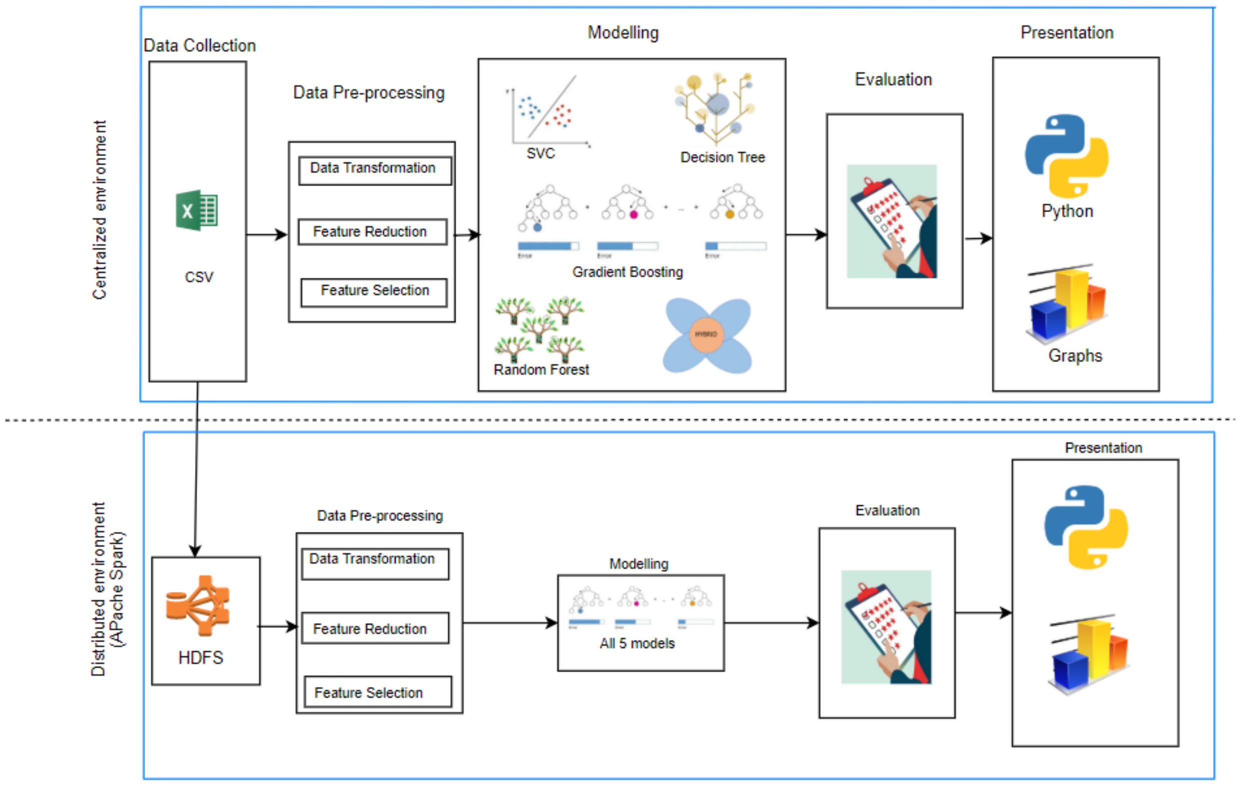


Figure 4: Flow Chart

5.1 Data Pre-processing

Following are data pre-processing steps taken prior to building classification models. These steps are mandatory to enhance the performance of models.

- **Exploratory Data Analysis (EDA):** A good data understanding can be obtained by the process of EDA. NSL-KDD dataset used for this research experiments consists of 22545 records and 42 attributes. Out of 42 attributes, 4 are categorical and rest 38 are numerical. The two-class output column is balanced. However, there exists many highly correlated independent variables as depicted in the Figure 5.

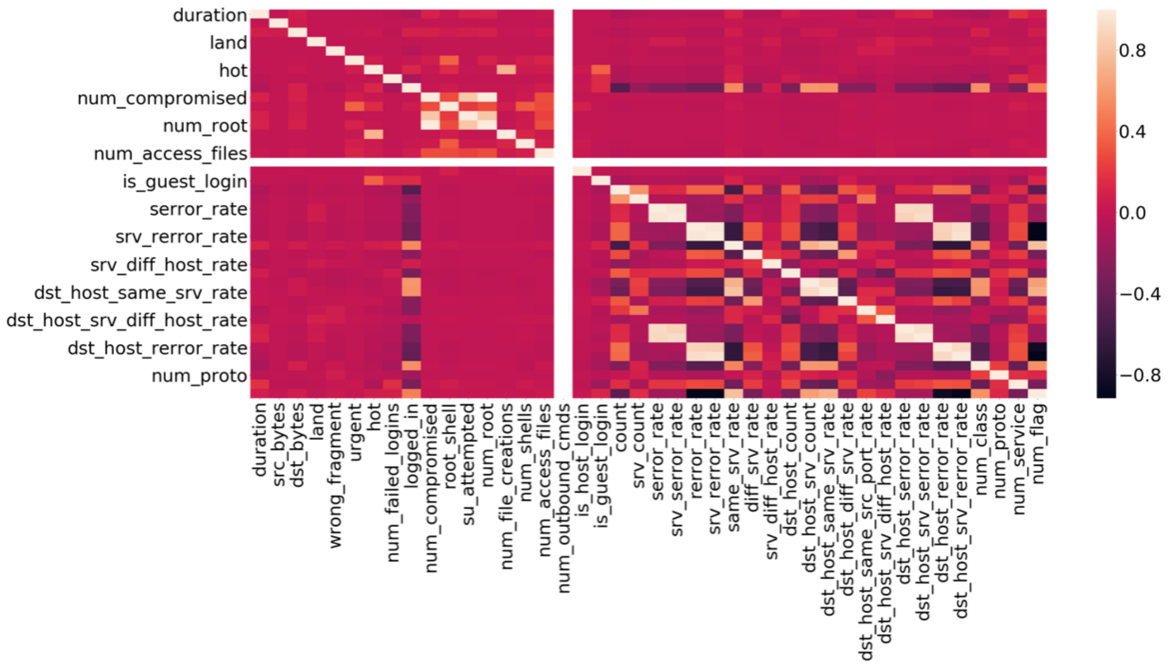


Figure 5: Correlation heatmap

- **Data transformation:** To perform classification modelling, all the attributes with datatype string are transformed to datatype float using 'LabelEncoder' function.
- **Feature reduction:** Correlation heatmap shows many highly correlated independent variables. These variables can lead to model over-fitting problem. This issue is solved by removing one of the variables from each of highly correlated pair. After feature reduction, 31 out of 41 variables are retained. Figure 6 depicts that there are no highly correlated attributes after feature reduction.

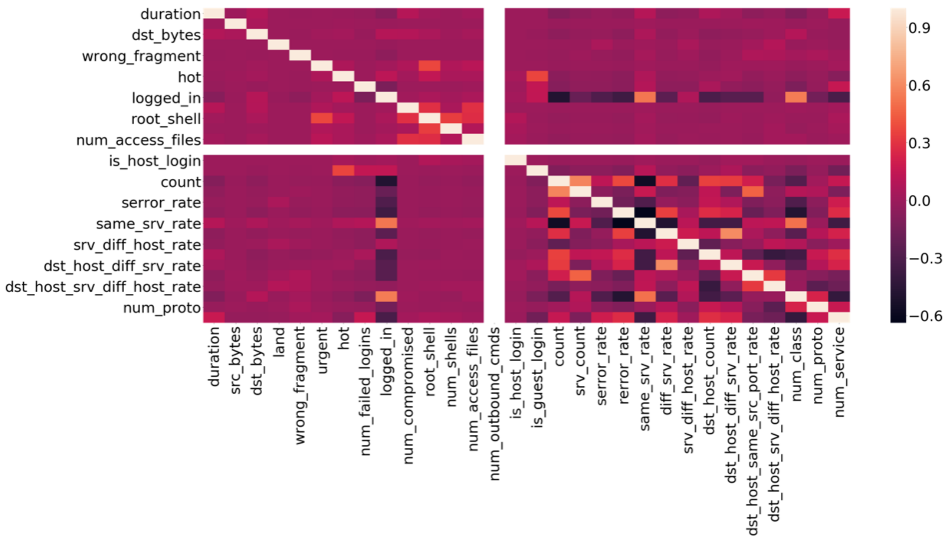


Figure 6: Correlation heat map after feature reduction

- **Feature selection:** The significance of each variables in predicting the output class is determined using random forest's 'feature importances' function. Figure 6 depicts variable importance graph. Based on each variable's importance value, 10 out of 31 attributes are selected. Dropping other lesser significant attributes helps in reducing the execution time taken to train the data mining models without any significant difference in accuracy of the model.

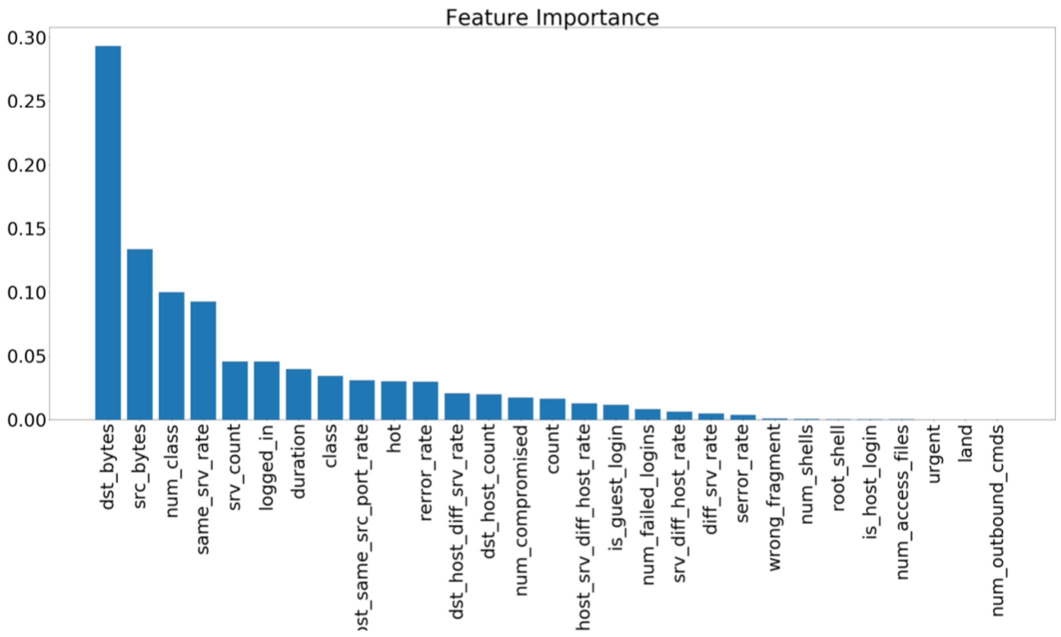


Figure 7: Variable Importance Graph

In the next step, a subset of all independent variables and a subset of dependent variable are created. These subsets are further split into train (80%) and test (20%) dataset for training and evaluation of the model, respectively.

5.2 Modelling

Different classification models are trained by tuning hyper parameters of the algorithms. The algorithms used for modelling are SVM, Decision Tree, Random Forest, Gradient Boosting and an Ensemble of these models. The models are evaluated and compared to determine the best performing model for IDS. On the second part of this research, the models are run on distributed environment (apache hadoop and spark) to compare the time-taken to train the model with centralized environment..

- **Support Vector Model:** Python's SVC() function is used to build support vector model. The hyper parameters of this function are C, kernel and gamma. C defines the error penalty. It sets the boundary between smooth decision curve and to correctly classifying the training points. High C values may cause over-fitting. Gamma is a non-linear hyper-plane parameter. Higher values of gamma will also increase over-fitting. Kernel determines the type of the space where the hyper-plane is drawn. It can be either linear (2D), rbf(non-linear) or poly(non-linear). Since the dataset is non-linear in nature, 'rbf' kernel is set. The range of hyper parameters are set and then, the function 'GridSearchCV' selects the best combination of parameters, that balances the under-fit and over-fit of the training data. The significant parameters of 'GridSearchCV' function are n-jobs, CV and refit. The number of parallel jobs running is indicated by 'n-jobs'. When 'n-jobs' is set to -1, all the available processors are utilized. 'CV' determines cross-validation splitting criteria. The 'refit' value is kept to default 'True' to indicate to fit estimator with best parameters. The model is trained with best fit hyper parameters and validated on the test dataset.

- **Decision tree:** The hyper parameter that is tuned in this algorithm is 'min-samples-split'. It determines the number of instances required to split an internal node. When set to maximum, the model fails to learn about the data. Larger values of other hyper parameters such as max-depth, min-samples-leaf and max-features may lead to under-fitting or over-fitting of the model and hence, their default values are retained. A range of 2 to 50 is assigned to min-samples-split and 'GridSearchCV' is used to choose the best fit value. The model is trained with best fit estimator and is then, validated using test dataset.

- **Random Forest:** As mentioned in the earlier section, it is an ensemble of decision trees and contains additional hyper-parameters. They are: bootstrap, max-depth, max-features, min-samples-leaf, min-samples-split and n-estimators. Random forest algorithm learns from drawing random samples from the data instances. When the bootstrap is set to true, the samples are drawn with the replacement. The number of trees in the forest is determined by n-estimators. Higher the value of n-estimators, better is the learning. However, if it is set to too high, the training process becomes noticeably slow. Hence, at not too high, it is set at a range of 100, 200, 300 and 1000. 'max-depth' indicates the depth of trees. Higher the value of max-depth, larger the number of nodes in each decision tree of the forest and hence, more learning. However, setting the value of 'max-depth' too high may also cause over-fitting issue and hence, range is set in between only 80 to 110. As mentioned earlier,

’min-samples-split’ determines the number of split at each internal node and is set to values of 8,10 and 12. ’min-samples-leaf’ is similar to ’min-samples-split’ except that it determines the split only at the leaf nodes. And it is set between 3 to 5, as higher values may cause under-fitting problems. The number of attributes to be considered prior to a split is indicated by ’max-features’ and is set to 2 and 3. A model is fitted for all the combinations of these hyper parameters with its specified range and a best fit is selected using ’GridSearchCV’. The chosen best fit trains the random forest model and it is then, validated by means of test data.

- **Gradient Boosting:** This model also learns by converting weak learners to strong learners. The three hyper parameters of this algorithm are max-features min-samples-leaf, max-depth and learning-rate. max-features, min-samples-leaf and max-depth are same as explained under random forest algorithm. And the learning-rate decides the contribution made by each tree for the learning of data. Higher value of learning-rate may cause over-fitting of the model. Graphs are plotted to determine the optimum value of each hyper-parameter for best accuracy of prediction as shown in Figure 8, Figure 9 and Figure 10

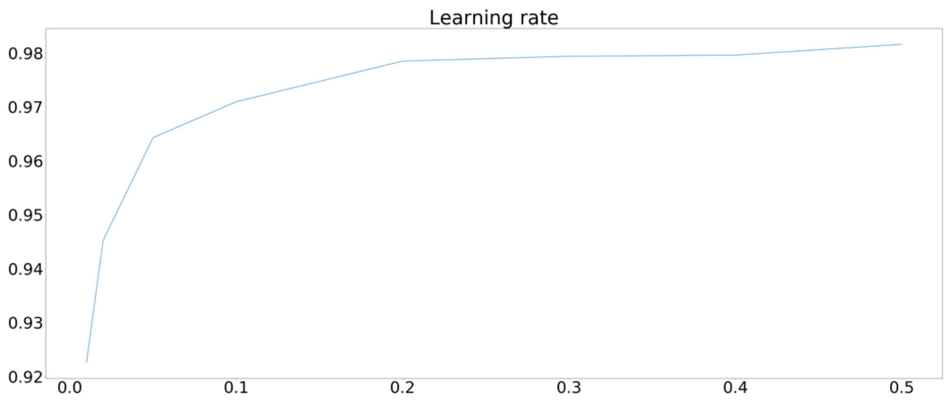


Figure 8: Accuracy Vs learning rate

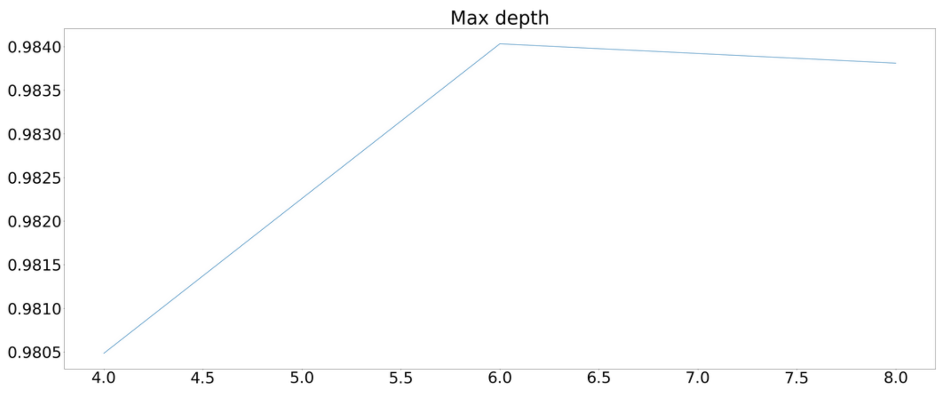


Figure 9: Accuracy Vs max depth

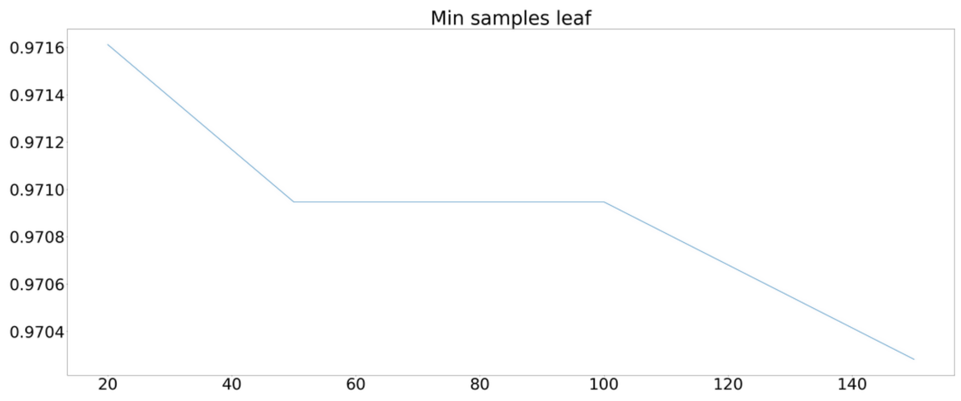


Figure 10: Accuracy Vs min samples leaf

- **Hybrid model:** A hybrid model is built that learns from all the four models, using the function 'MultiOutputClassifier' and 'VotingClassifier'. Hard voting method is adopted since it is a classification problem. Hard voting is a technique wherein, final prediction is made based on the majority vote from different classifiers. It is then cross-validated with 3 and 10 folds to ensure that there is no over-fitting problem.

A confusion matrix is drawn for all the models using the function 'confusion-matrix'. Precision, Recall, F1 score and Accuracy of the models are also calculated by means of python's pre-defined functions.

6 Evaluation

6.1 Experiment 1: Performance of data mining models

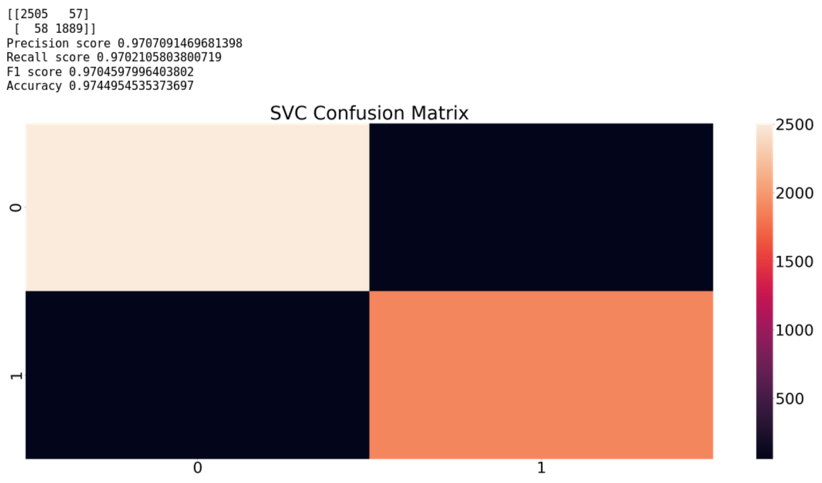
The data mining models are evaluated using performance metrics namely: Precision, Recall, F1 Score, Accuracy and Error rate. Table 2 presents the performance of various classifiers built for the experimental purpose

Table 2: Performance metrics

Models	Precision	Recall	F1 Score	Accuracy	Error rate
SVM	97.07%	97.02%	97.04%	97.44%	0.025
Decision Tree	98.07%	96.86%	97.46%	97.82%	0.021
Random forest	98.25%	96.86%	97.46%	98.42%	0.015
Gradient Boosting	98.35%	98.35%	98.17%	98.58%	0.014
Hybrid	98.70%	97.63%	98.16%	98.42%	0.015

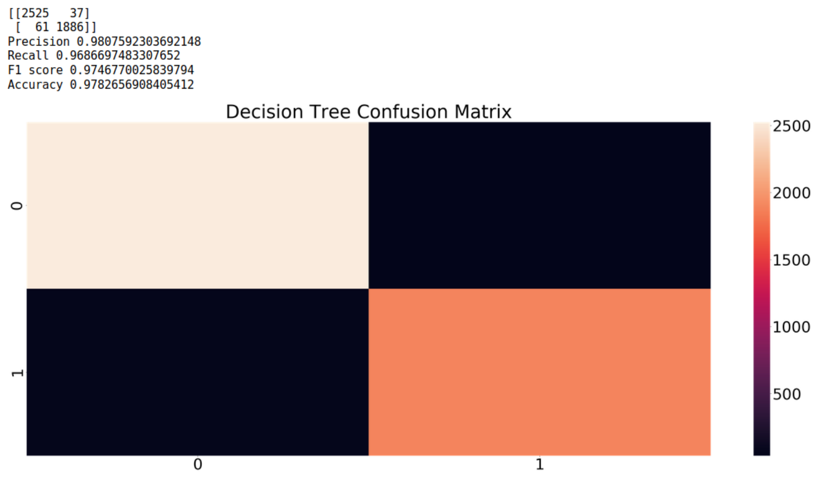
All the classifiers perform with high accuracy (above 97%). When compared with other tree-based algorithms, SVC's performance is little low, however difference is insignificant. Random forest and gradient boosting produce almost similar results. With hybrid of all four models, accuracy is slightly improved to 98.42% and Error rate is 0.015. The MSE value of hybrid model obtained is 0.015.

Figure 11, Figure 12, Figure 13, Figure 14 and Figure 15 shows the confusion matrix heat maps of SVM, decision tree, random forest, gradient boosting and hybrid models, respectively.



```
[[2505   57]
 [  58 1889]]
Precision score 0.9707091469681398
Recall score 0.9702105803800719
F1 score 0.9704597996403802
Accuracy 0.9744954535373697
```

Figure 11: Confusion matrix of SVC model



```
[[2525   37]
 [  61 1886]]
Precision 0.9807592303692148
Recall 0.9686697483307652
F1 score 0.9746770025839794
Accuracy 0.9782656908405412
```

Figure 12: Confusion matrix of Decision tree model

```
[[2528   34]
 [  37 1910]]
Precision score 0.9825102880658436
Recall score 0.9809964047252183
F1 score 0.9817527627859162
Accuracy 0.9842537147926369
```

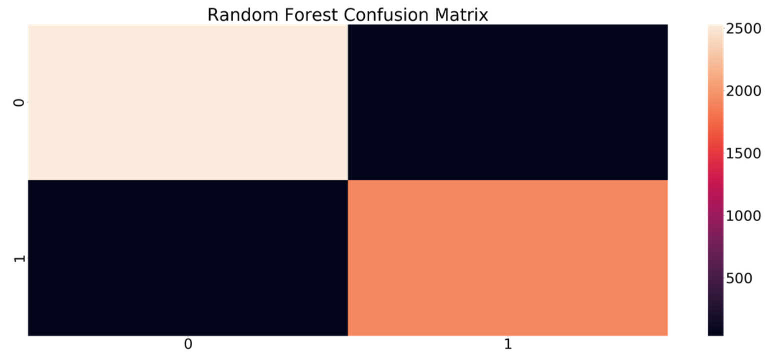


Figure 13: Confusion matrix of Random forest model

```
[[2530   32]
 [  32 1915]]
Precision score 0.9835644581407293
Recall score 0.9835644581407293
F1 score 0.9835644581407293
Accuracy 0.985806165446884
```

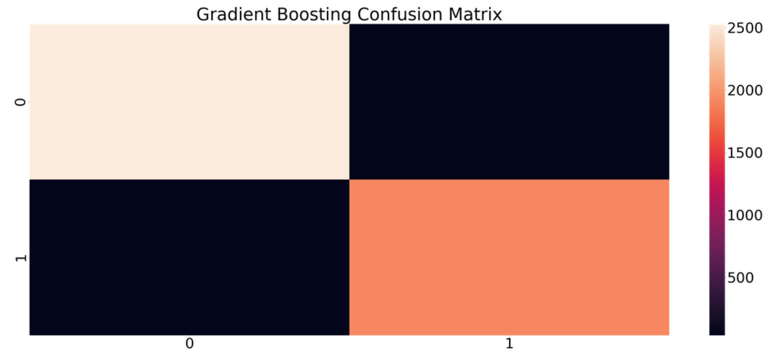


Figure 14: Confusion matrix of Gradient boosting model

```
[[2537   25]
 [  46 1901]]
Precision 0.9870197300103842
Recall 0.9763739085772984
F1 score 0.9816679576555642
Accuracy 0.9842537147926369

: Text(0.5, 1, 'Hybrid Model Confusion Matrix')
```

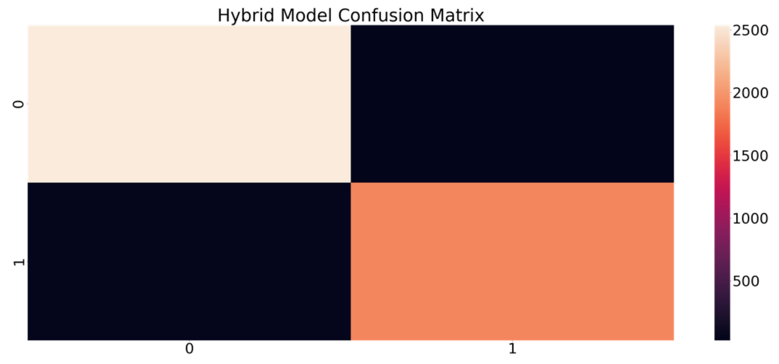


Figure 15: Confusion matrix of Hybrid model

Table 3: Accuracy after 10-fold cross-validation

Models	Validated Accuracy (k=10)
SVM	96.38%
Decision Tree	96.93%
Random forest	97.58%
Gradient Boosting	97.71%
Hybrid	97.84%

Table 3 shows the cross-validated accuracy of all the models. The values assures that models perform well without any over-fitting.

6.2 Experiment 2: Computing performance of centralized and distributed environments

Table 4 shows the time taken (in minutes) by centralized and distributed (apache hadoop and spark) environments to train the machine learning models.

Table 4: Time taken to train models

Models	Centralized Environment	Distributed Environment
SVM	382.4min	77.6min
Decision Tree	2min	1.4min
Random forest	520.6min	265.6min
Gradient Boosting	2.3min	2.3min

The models are trained much faster in distributed environment when compared to centralized. For instance, while SVM model takes 382.4min to train in centralized environment, while it only takes 77.6min in distributed environment.

6.3 Discussion

All the classification models performed well with high accuracy and low error rate. However SVC model's performance was slightly less compared to other models. In particular, all the tree-based models performs well for two-class classification problems. The difference in the performance between tree-based models is observed to insignificant. Gradient boosting algorithm's error rate is the least among all of them. Hybrid model shows some improvement in the performance. The result of cross-validated accuracies ensure that there is no over-fitting of the models. MSE values of 0.015 indicates that quality of performance of the classifiers is high. Based on the survey conducted by Moustafa et al. (2019),other algorithms like EM clustering, logistic regression and naive bayes performed with only 90.3%, 95.1% and 93.8% accuracy, respectively. ARM (correlation-based) technique for feature reduction and feature selection based on variable importance plot(using random forest technique) contributed in enhancing perform without over-fitting problems. Hence tree-based classification models, with right feature engineering methods are found to better for intrusion detection system. Another purpose of this project is compare the difference in computing-time taken by centralized and distributed environments. The results of experiments conducted clearly states that distributed environments using file storage system such as apache hadoop and computing framework such as apache spark performs with high speed, taking significantly lesser time to train the models when compared to centralized environment. Hence, distributed frameworks are the ideal environment for intrusion detection systems as it helps to identify the attacks in near real-time.

7 Conclusion and Future Work

Intrusion detection systems plays a vital role in network security. Hence, it is necessary that data mining algorithms applied in IDSs to perform with high predictive rate and low error rate. Being one of the objective, this research experiment proves that tree-based algorithms such as decision tree, random forest and gradient boosting, along with ARM-based feature engineering techniques performs better for classification problems and a hybrid of them can enhance the performance further. This research paper also conducted experiments to compare the time taken by the centralized and distributed (Apache Hadoop and Apache Spark) environments to train the models. Clearly, distributed environments performed much faster than centralized environments and hence, this research suggests tree-based algorithms and distributed environments to develop intrusion detection systems to identify the threats in computer networks. Although, the proposed IDSs on distributed environment performs at very a low latency, it still does not aid real-time detection of threats as apache spark can supports micro-batching only. To detect attacks in real-time, stream-processing engines like apache storm must be integrated in IDS. This creates a scope for future research work on IDSs.

Acknowledgement

I would like to convey my sincere thanks to my research guide Dr. Pierpaolo Dondio for sharing his valuable knowledge and timely feed-backs through out the research process. Special thanks to my family and friends for their support and encouragement.

References

Ahmed, Mohiuddin; Naser Mahmood, Abdun; Hu, J. (2016). A survey of network anomaly detection techniques.
URL: *http://dx.doi.org/10.1016/j.jnca.2015.11.016*

Alanazi, H. O., Noor, R., Zaidan, B. B. and Zaidan, A. A. (2010). Intrusion Detection System : Overview, **2**(2): 130–133.

Aldinucci, M., Drocco, M., Misale, C. and Tremblay, G. (2019). Languages and Frameworks for Big Data Analysis, pp. 1–16.

Alqahtani, M., Gumaei, A., Mathkour, H., Maher, M. and Ismail, B. (2019). A Genetic-Based Extreme Gradient Boosting Model.

Ambwani, T. (2003). Multi class support vector machine implementation to intrusion detection, *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 3, IEEE, pp. 2300–2305.

Anwar, S., Zolkipli, M. F., Anthony, B., Zain, J. M., Inayat, Z., Khan, S. and Chang, V. (2017). From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions, *Algorithms* **10**(2).

Azevedo, Ana; Santos, M. F. (2008). KDD , SEMMA AND CRISP-DM : A PARALLEL OVERVIEW Ana Azevedo and M . F . Santos, *IADIS European Conference Data Mining* pp. 182–185.
URL: *http://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf*

Bamakan, S. M. H., Wang, H. and Shi, Y. (2017). Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem, *Knowledge-Based Systems* **126**: 113–126.

Bhuyan, M. H., Bhattacharyya, D. K. and Kalita, J. K. (2014). Network Anomaly Detection : Methods , Systems and Tools, *IEEE Communications Surveys & Tutorials* **16**(1): 303–336.

Bouzekri, E. and Idrissi, E. (2016). A security approach for social networks based on honeypots, *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)* pp. 638–643.

Chandra, P. (2017). NETWORK INTRUSION DETECTION SYSTEM BASED ON MODIFIED RANDOM FOREST CLASSIFIERS FOR KDD CUP-99 AND NSL-KDD DATASET, pp. 786–791.

Chunduri, R. K. (2019). Scalable formal concept analysis algorithms for large datasets using, *Journal of Ambient Intelligence and Humanized Computing* **10**(11): 4283–4303.
URL: *http://dx.doi.org/10.1007/s12652-018-1105-8*

Dua, S. and Du, X. (2016). *Data mining and machine learning in cybersecurity*, Auerbach Publications.

Erraissi, A., Belangour, A. and Tragha, A. (2017). A Big Data Hadoop building blocks comparative study, **48**(1): 36–40.

Farnaaz, N. and Jabbar, M. A. (2016). Random Forest Modeling for Network Intrusion Detection System, *Procedia - Procedia Computer Science* **89**: 213–217.
URL: *http://dx.doi.org/10.1016/j.procs.2016.06.047*

Gupta, C., Sinhal, A. and Kamble, R. (2013). Intrusion detection based on k-means clustering and ant colony optimization: A survey, *International Journal of Computer Applications* **79**(6).

Hasan, M., Dean, T., Imam, F. T., Garcia, F., Leblanc, S. P. and Zulkernine, M. (2017). A constraint-based intrusion detection system, *Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems*, ACM, p. 12.

Horng, S.-j., Su, M.-y., Chen, Y.-h., Kao, T.-w., Chen, R.-j. and Lai, J.-l. (2011). Expert Systems with Applications A novel intrusion detection system based on hierarchical clustering and support vector machines, *Expert Systems With Applications* **38**(1): 306–313.
URL: *http://dx.doi.org/10.1016/j.eswa.2010.06.066*

Inayat, Z., Gani, A., Anuar, N. B., Khan, M. K. and Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges, *Journal of Network and Computer Applications* **62**: 53–74.
URL: *http://dx.doi.org/10.1016/j.jnca.2015.12.006*

Jirapummin, C., Wattanapongsakorn, N. and Kanthamanon, P. (2015). Hybrid Neural Networks for Intrusion Detection System, (January 2002).

Kang, I., Jeong, M. K. and Kong, D. (2012). A differentiated one-class classification method with applications to intrusion detection, *Expert Systems with Applications* **39**(4): 3899–3905.

Kim, T., Suh, S. C., Kim, H., Kim, J. and Kim, J. (2019). An Encoding Technique for CNN-based Network Anomaly Detection, *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* pp. 2960–2965.

Mill, J. (2004). Support Vector Classifiers and Network Intrusion Detection, pp. 407–410.

Moustafa, N., Creech, G. and Slay, J. (2017). Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models, *Data analytics and decision support for cybersecurity*, Springer, pp. 127–156.

Moustafa, N., Creech, G. and Slay, J. (2018). Anomaly detection system using beta mixture models and outlier detection, *Progress in Computing, Analytics and Networking*, Springer, pp. 125–135.

Moustafa, N., Hu, J. and Slay, J. (2019). A holistic review of Network Anomaly Detection Systems: A comprehensive survey, *Journal of Network and Computer Applications* **128**(April 2018): 33–55.
URL: *https://doi.org/10.1016/j.jnca.2018.12.006*

Moustafa, N. and Slay, J. (2015a). A hybrid feature selection for network intrusion detection systems : Central points, (February 2016).

Moustafa, N. and Slay, J. (2015b). The significant features of the UNSW-NB15 and the KDD99 data sets for Network Intrusion Detection Systems, *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)* pp. 25–31.

Nalavade, K. and Meshram, B. B. (2014). Mining Association Rules to Evade Network Intrusion in Network Audit Data, (2).

Peddabachigari, S., Abraham, A., Grosan, C. and Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems, **30**: 114–132.

Peddabachigari, S., Abraham, A. and Thomas, J. (n.d.). Intrusion Detection Systems Using Decision Trees and Support Vector Machines, pp. 1–16.

Resende, P. A. A. and Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems, *ACM Computing Surveys (CSUR)* **51**(3): 48.

Saber, M., El Farissi, I., Chadli, S., Emharraf, M. and Belkasmi, M. G. (2017). Performance analysis of an intrusion detection systems based of artificial neural network, *Europe and MENA Cooperation Advances in Information and Communication Technologies*, Springer, pp. 511–521.

Sangher, K. S. (2019). A Systematic Review – Intrusion Detection Algorithms Optimisation for Network Forensic Analysis and Investigation, *2019 International Conference on Automation, Computational and Technology Management (ICACTM)* pp. 132–136.

Singh, K., Chandra, S., Thakur, A. and Hota, C. (2014). Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests, *Information Sciences* **278**: 488–497.
URL: *http://dx.doi.org/10.1016/j.ins.2014.03.066*

Thaseen, S. (2013). An Analysis of Supervised Tree Based Classifiers for Intrusion Detection System, *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering* pp. 294–299.

Vijay, P. (2017). Performance Evaluation of Classification Techniques for Intrusion Detection in Noisy Datasets, (June): 1011–1016.

Wagner, C., State, R. and Engel, T. (2011). Machine Learning Approach for IP-Flow Record Anomaly Detection, pp. 28–39.

Xia, Y., Liu, C., Li, Y. and Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring, *Expert Systems With Applications* **78**: 225–241.
URL: *http://dx.doi.org/10.1016/j.eswa.2017.02.017*

You, T. (n.d.). Survey on Apache Hadoop Project.

Zhang, J., Zulkernine, M. and Haque, A. (2008). Random-forests-based network intrusion detection systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **38**(5): 649–659.