INFORMATION SECURITY MANAGEMENT

(CSE3502) – PROJECT COMPONENT

Final Report

Submitted by

TEAM MEMBERS:

Malla Jyotsna Sree Mahima-18BCE0912

Adarsh Reddy-18BCI0196

Preetham Lekkala-18BCE0854

Submitted To

PROF. Ramani S

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

NETWORK INTRUSION DETECTION USING MACHINE LEARNING

Abstract

There are a lot of companies and individuals that suffer from numerous security problems without ever actually knowing it happened to them. The worse part is that when these security problems go unsolved, they generally create a lot of other openings for attackers to breach the security infrastructure to steal the data and generally destroy a lot.

There are a lot of small businesses that usually don't have the complete knowledge of all of theirs IT assets that are tied to their network. This is a very big problem. T,he data cited by Harvard Business Review, "60% of all the attacks were carried out by the insiders. Maybe because of an honest mistake like accidentally sending info to the wrong email or losing work credentials, or even a friend posting your personal information online. To be save one must at least know that they are being attacked so that they will at least get a chance to prepare for the fight either by taking a professional help or do it yourself. To do this we will try to detect the intruders by implementing a network intrusion detection system which predict possible hostile environments by using neural networks with the help of various machine learning algorithms and selecting the most accurate one to prepare the IDS tool.

Many researches have proposed a lot of machine learning models for intrusion detection to reduce the false positive cases and help in creating an accurate IDS. But, to deal with the Big Data that we have, the traditional machine learning models are taking a long time in learning and classifying the input data. With the help of Big Data techniques and machine learning models for IDS can solve many challenges such as speed and computational time and develop accurate IDS that can help a lot of small and big companies. we are comparing the performance of various Machine learning models to get the most efficient one to use it for the ids tool. The result of the experiment showed that model random forest and KNN clarification models has high performance, reduces the training time and is efficient for Big Data.

Data Set:

https://www.kaggle.com/sampadab17/network-intrusion-detection?select=Train_data.csv

1. Problem Statement

The Internet is getting unsafe as technology is getting better. differentiation of the normal activities and intrusions in the wide internet network is very difficult and very time-consuming. This problem can be solved by an analyst who can review the big data that is large and wide to find the sequences and patterns of intrusion on the internet network. We need a way that can detect network intrusions

with the help of past network intrusions. In this project, we are going to test the accuracy of predictions using various classification algorithms and make a tool with the best one to find intrusion characteristics. We are expecting our algorithm to determine intrusion characteristics then recommend appropriate prevention measures.

2. Introduction

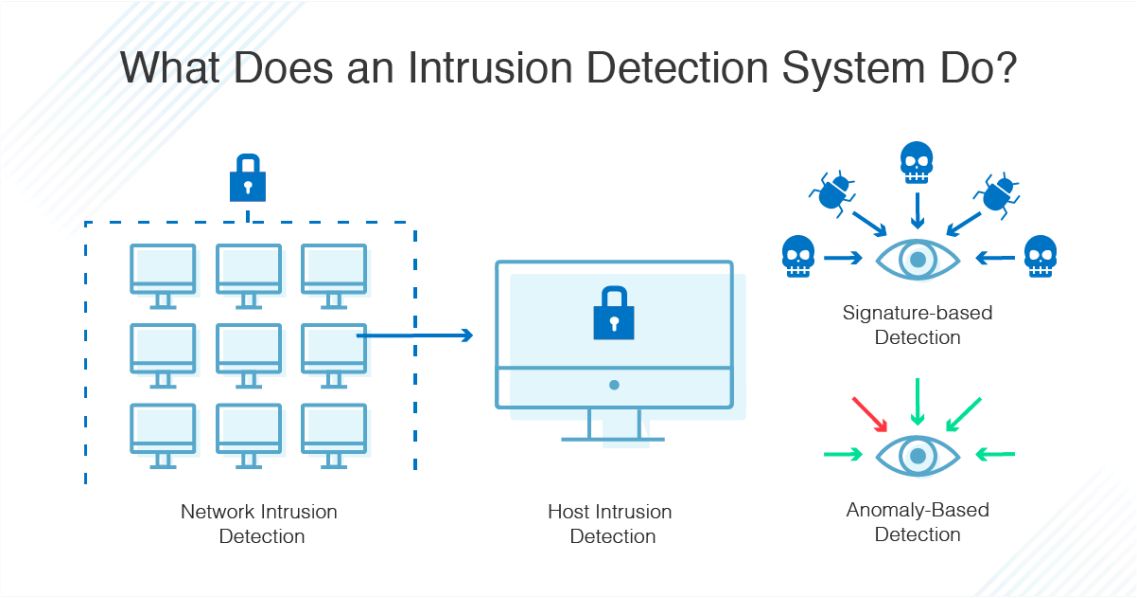
Intrusion detection system is a system which is used to find the intruders before the attack so the attack can be stopped, and the damage can be reduced. Intrusion detection systems monitors events that are occurring in a computer system and networks by analyzing the patterns of the intrusions. IDS examine the host and networks to spot potential intrusions. We have tried to detect the intruders by implementing the network intrusion detection system using neural network with the help of Python and classification algorithms. Some data preprocessing will be used in initial stages of implementation for data cleaning for better accuracy and less memory usage. This system can is used to detect malicious activities or something unusual in the network and can also be used to send signal to the user about the activities.

2.1. Importance of Network Intrusion prediction

Attackers take multiple approaches when trying to hack a system or network. With the help of the network intruder detection software, we can understand the kinds of attacks which can be helpful while setting up an effective prevention system. In most cases of intrusions, the attack involves the flooding or overloading concepts of the network. The intruders general gather the information about the network to attack it from a weak side or try inserting malicious information into the network to spread and gain access from inside. so, it’s really important to keep the detection tools active, so you can prevent the vulnerabilities.

Intrusion Detection System Using Data Mining Technique, we were able to classify the attack types as: -

- Denial of service attack (Dos)
- Probing Attack (Probe)
- User to Root Attack (U2R)
- Remote to Local Attack (R2L)



2.2.
Evasion Techniques:

- **Fragmentation:** Sending the fragmented parcels can permit the assailant to remain under the radar, so by also bypassing the identification framework's capacity to identify the assaulter's signature.

- **Avoiding defaults:** A port utilized by a protocol does not always gives the indication the protocol that's being transported. If an attacker reconfigures it to use a different port, the IDS can not be able to detect the trojan.

- **Coordinated, low-bandwidth attacks:** coordinating the scan among numerous attackers and allocating various ports or hosts to different attackers is an effective evasion technique. This makes IDS difficult to make sense of the captured packets and deduce that the network scan is in the progress.

- **Address spoofing:** attackers can escape the source of the attack by using a insecure or incorrectly configured servers to create an attack. the source Will be spoofed and bounced by a server which make the ids to detect difficult.

2.3. Detection Method of IDS:

• Signature-based Method:

Signature-based intrusion detection system detects the attacks on the bases of the past patterns such as number of bytes the network traffic. It also helps to detect on the basis of the already happened malicious instruction sequences that is used by the malware. The detected patterns by the IDS are

known as the signatures. Signature-based IDS are known t detect to the attacks whose pattern already exists in database. but, it will be quite difficult to detect new malware attacks as their signature are not known.

• Anomaly-based Method:

Anomaly-based intrusion detection system is used to detect the unknown malware attacks as the new malware are developing rapidly in recent years. The anomaly-based IDS use machine learning to create an activity model the new attacks are compared with that model and will be declared as suspicious or not. Machine learning method has a better generalization in comparison to that of signature-based IDS.

3. Literature Review

1. Intrusion detection system modelling based on neural networks and fuzzy logic.

Midzic, A., Z. Avdagic, and S. Omanovic. "Intrusion detection system modeling based on neural networks and fuzzy logic." In *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*, pp. 189-194. IEEE, 2016.

Methodology: Modern network IDS (Intrusion Detection Systems) and IPS (Intrusion Prevention Systems) increasingly use DNN in classification of network traffic process. In this paper they have used the concept of deep learning and explained the implementation of artificial neural network which will form a network of nodes that are connected with edges. The nodes represents individual attribute in the beginning(level 0) and, combination of attributed in all other levels. The ANN model is implemented with many learning rate. On continuously increasing learning rate, the accuracy increases , reaches threshold and drops down. The edge carries the weightage of each node from one level to another. This approach helps machine to learn in a different way from the machine learning algorithms.

Pros: The neural network’s output is better than all other machine learning algorithms performed individually.

Cons: in many situations hybrid ML algorithms can perform better than them. It’s complex to understand the working of nodes in neural networks.

2. A survey of intrusion detection systems based on ensemble and hybrid classifiers

Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz. "A survey of intrusion detection systems based on ensemble and hybrid classifiers." *Computers & Security* 65 (2017): 135-152.

Methodology: Due to no of times malicious network activities and the network policy violations, intrusion detection systems (IDSs) have come out as proper methods that combats the unauthorized use of a network's resources. Recent advances in the information technology have produced some wide variety of machine learning methods, which we can integrate into an IDS. This study gives an overview of intrusion classification algorithms based on the popular methods in the field of machine learning. Various ensemble and hybrid techniques are also examined, considering both homogeneous and heterogeneous types of ensemble methods. survey of the recent literature shows that hybrid methods in which the feature selection or a feature reduction component is combined with a single-stage classifier, have become commonplace. Therefore, the scope of this study has been expanded to make and incorporate hybrid classifiers.

Pros: Special attention was paid to those ensemble methods that are based on voting techniques, as those methods are the simplest to implement and generally produce favourable results.

Cons: These Hybrid dataset as combination of any 2 algorithms has higher chances of failure for different dataset.

3. Hybrid optimization scheme for intrusion detection using considerable feature selection

Velliangiri, S., and P. Karthikeyan. "Hybrid optimization scheme for intrusion detection using considerable feature selection." *Neural Computing and Applications* (2019): 1-15.

Methodology: The intrusion detection is an essential section in network security because of its immense volume of threats which bothers the computing systems. The real time intrusion detection dataset comprises the redundant and irrelevant features. The duplicate or similar features in context make it quite hard for algorithm to locate the patterns for intrusion detection systems. Hybrid optimization scheme is designed for combining adaptive artificial bee colony along with adaptive particle swarm optimization for detecting intrusive activities. The aim of the proposed system is to improve the rate of precision in intrusion activities in internetwork by choosing the most relevant features.

Pros: The schemes discussed in the methodology of the paper resulted in optimization-based outcomes and yields higher precision. Effectiveness and usefulness of hybrid categorization system is accessed using the NSL- KDD dataset. Both the random feature selection method and the Single feature selection method are used to assess the proposed HOS intrusion detection approaches. The effectiveness of this scheme is evaluated with existing machine learning algorithms such as Naive Bayes, AABC, APSO, and support vector machine, which outperform the HOS.

Cons: The processing of this algorithm may take longer time even for simple task. It will have to parse entire dataset completely many times even if the input data is the first row of training dataset.

4. Algorithm selection for classification problems

Pise, Nitin, and Parag Kulkarni. "Algorithm selection for classification problems." In *2016 SAI Computing Conference (SAI)*, pp. 203-211. IEEE, 2016.

Methodology: In this paper, algorithm selection is proposed for classification problems in data mining. The characteristics of datasets and the performance of classification algorithms are found out. Then based on the problem of classification, or the new problem at hand, mapping is done between the datasets and the benchmark performance of different classifiers. K-similar datasets are returned. Then ranking of classification algorithms is performed and based on the highest rank, the classification algorithm is recommended for the problem at hand. Hence the user doesn't need to waste time for working on different data mining algorithms, fine tuning the parameters for different algorithms. The algorithm is directly recommended for his problem.

Pros and Cons: The learning parameters used for training classifiers affects the performance of the classifiers. So, the further work is required to discover the suitable values for a given dataset.

Cons: Grid search and evolutionary algorithms can be used for this task. This will be helpful in model selection which includes not only algorithm selection but also parameter optimization.

5. Network Intrusion Detection System Using Voting Ensemble Machine Learning

Raihan-Al-Masud, Md, and Hossen Asiful Mustafa. "Network Intrusion Detection System Using Voting Ensemble Machine Learning." In *2019 IEEE International Conference on Telecommunications and Photonics (ICTP)*, pp. 1-4. IEEE, 2019.

Methodology: In this paper, a network intrusion detection system (NIDS) model is proposed with ensemble machine learning methods. The proposed system can detect known attacks as well as can prevent unknown attacks.The proposed system uses ensemble machine learning methods with Voting. The voting-based IDS model with selecting features has a strong capability for intrusion detection compared with base level machine learning methods such as support machines (SVM) as well as deep learning methods.

Pros: Ensemble machine learning methods have the potential to detect and prevent different types of attacks compared to traditional machine learning methods. Using voting technique, we increased the detection rate of U2R which more than double compared to the existing methods.

Cons: Drawback of not having effectiveness in measuring diversity and shows no or little relation with the accuracy of the ensemble because it only considers the difference of two models and hence, they are not valuable.

6. An efficient XGBoost–DNN-based classification model for network intrusion detection system

Devan, Preethi, and Neelu Khare. "An efficient XGBoost–DNN-based classification model for network intrusion detection system." *Neural Computing and Applications* (2020): 1-16.

Methodology: In this paper, they have discussed the existing IDS and also an attempt has been made to implement XGBoost–DNN-based classification model for the intrusion detection system. XGBoost model is primarily used for classification, and in our work, we have utilized the feature importance score generated by XGBoost model for feature selection. This feature selection method is first of its kind, which is applied for intrusion detection and DNN for classification of intrusions using NSL-KDD dataset. The results are then are compared with existing logistic regression, naive Bayes, and support vector machine. The efficient classifier is identified based on the following results of accuracy, precision, recall, and F1-score.

Pros: XGBoost deals with the regularization and helps in preventing overfitting issues. XGBoost provides a faster detection mechanism for network intrusions than the existing models applied for binary classification of intrusion detection. From the observed results, a deep learning model reveals a consistent level of 97% classification accuracy then existing models. It has the edge over others. The proposed work is applied for binary classification.

Cons: Even though the algorithm accuracy, there is always a possibility that the output will fall into its wrong prediction categories. In the future, the work can be polated and stretched for multi class classification.

7. New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN

Hosseini, Soodeh, and Behnam Mohammad Hasani Zade. "New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN." *Computer Networks* 173 (2020): 107168.

Methodology: This paper describes a brand new hybrid IDS method with two steps, a feature selection step and an attack detection phase. In feature selection phase, the wrapper technique, namely MGA-SVM, is used. This technique combines the features of the support vector machine and the genetic algorithm with multi-parent crossover and multi-parent mutation. Where as in the attack detection phase, the artificial neural network is used to detect attacks. For improving the performance, a mix of a hybrid gravitational search with a particle swarm optimization is used to combine and train the classifier.

Pros: For evaluating the capability of the proposed method, standard classification metrics along with training and testing time and the number of selected features has been used as evaluation criteria, based on the dataset. Obtained results have shown that the proposed method outperforms all other methods under each criterion used in the comparison. also, its training and testing time are also the lowest

Cons: Disadvantage of the particle swarm optimization (PSO) algorithm are that it is easy to fall into local optimum in high-dimensional space and has a low convergence rate in the iterative process..

8. Intrusion detection system using Machine Learning

Ugochukwu, Chibuzor John, E. O. Bennett, and P. Harcourt. *An intrusion detection system using machine learning algorithm*. LAP LAMBERT Academic Publishing, 2019.

Methodology: The proposed architecture eradicates malicious behaviours by detecting known attacks using log files, blocks suspicious behaviour in real time on behalf of recent architectures requests, secures sensitive data, and it also establishes better adaptations of security measures by dynamically updating security rules. Rule-based analysis identifies malicious behaviours based on static rules such as source code, serial numbers of developers, session fixation or attack signatures. Misuse detection system helps in predicting anomalies in the short and medium term but considers each recorded event that is not on a list of known attack signatures as normal. The algorithms used are KNN, Logistic Regression, SVM and Naïve Bayes.

Pros: Advantage is that a hybrid machine learning architecture analyses all the log events automatically to detect anomalies, and crafts real-time user profiles to decide whether behaviour is legitimate.

Cons: Disadvantage is that it does not calculate the distance between a suspicious behaviour and a certain attack, nor does it predict new anomalies and it is static and needs frequent updates.

9. Hierarchical Anomaly Detection Model for In-Vehicle Networks Using Machine Learning Algorithms

Park, Seunghyun, and Jin-Young Choi. "Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms." *Sensors* 20, no. 14 (2020): 3934.

Methodology: Proposed a multi-labelled hierarchical classification (MLHC) intrusion detection model that analyzes and detects external attacks caused by message injection. This algorithm quickly determines the occurrences of attacks and classifies the attacks using only existing classified attack data. This proposed model is also able to classify both the type and existence or absence of attacks with high accuracy and can be used in interior communication environments of high- speed vehicles with a high throughput. analysis model is trained by injecting the same training data, and the performance of the trained model is evaluated using the test data. The intrusion detection module, with the trained model in an real application environment, is used to find follow-up information, such as attack or benign, vehicle-type and attack type, after receiving the CAN (Controller Area Network) message frame as input.

Pros: Thus, we conclude that the DT and RF algorithms are applicable to high-speed internal communication environments, as well as in CAN for analyzing 43 million and 46 million CAN amessage frames per second, respectively. The simulation results show that the proposed MLHC model achieved high accuracy when based on the RF algorithm and rapid detection when based on the DT algorithm. Both algorithms derived. F1 scores higher than 0.998.

Cons: The disadvantage with this model is that, there can be loss of information and can lead to lots of classes with small number of data.

10. Genetic convolutional neural network for intrusion detection systems

Nguyen, Minh Tuan, and Kiseon Kim. "Genetic convolutional neural network for intrusion detection systems." *Future Generation Computer Systems* 113 (2020): 418-427.

Methodology: This paper creates an algorithm for a network intrusion detection system (NIDS) using an redesigned feature subset selected directly by a fuzzy C-means clustering (FCM) and genetic algorithm (GA)-based exhaustive search .This algorithm identifies the bagging classifier and convolutional neural network model as an effective extractor by implementing the GA in combination with 5-fold cross validation (CV) to select the CNN model structure. This deep feature subset extracted by selected CNN model was put into the BG classifier to validate the performance with the 5-fold CV. The high-quality feature set obtained by the three-layered feature construction using the GA, FCM, CNN extractor, and a hybrid CNN and BG learning method significantly improves the final detection performance. High quality Deep Feature Subset (DFS) was produced by the extractor, which learned the most representative characteristics of the attack types through multiple layers.

Pros: the advantage of the model is that the robust learning of the hybrid learning method containing the CNN model as an extractor and the BG classifier helped improve the final classification performance of the algorithm. Deployment of the proposed algorithm over the practical internet systems would improve the computer network security against the illegal activities.

Cons: Disadvantage is that most likelihood-based methods compute the likelihood of only a few features of the data (only one), and therefore additional information that could improve accuracy of the model is ignored.

11. An Adaptive Synchronous Parallel Strategy for Distributed Machine Learning

Zhang, Jilin, Hangdi Tu, Yongjian Ren, Jian Wan, Li Zhou, Mingwei Li, and Jue Wang. "An adaptive synchronous parallel strategy for distributed machine learning." *IEEE Access* 6 (2018): 19222-19230.

Methodology: In recent years, distributed systems have mainly been used to train machine learning (ML) models. Due to the result of the different performances among computational nodes in the

distributed cluster and delays in network transmission, the accuracies rate and convergence rate of ML models are relatively low. So, there is a necessity to design reasonable strategy that provides dynamic communication optimization to improve the utilization of the cluster, accelerate the training times, and strengthen the accuracy of the training model. In this paper, they have proposed the adaptive synchronous parallel strategy for distributed ML. The synchronization strategy of every compute node with the parameter server is tuned adaptively by taking into consideration the full performance of each node, thereby ensuring higher accuracy.

Pros: Strategies used in this paper prevents the ML model from being affected by irrelevant tasks in the same cluster. The experiments proves that their strategy fully improves clustering performance, and it ensures the accuracy and convergence speed of the model, increases the model training speed, and has good expansibility.

Cons: For higher memory dataset, more cores will be required for running all ML models parallelly.

12. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost

Shahraki, Amin, Mahmoud Abbasi, and Øystein Haugen. "Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost." *Engineering Applications of Artificial Intelligence* 94 (2020): 103770.

Methodology: In this paper, a summary of the latest advances in the context of intrusion detection methods, present a technical background on boosting, and show the ability of the three popular boosting algorithms (Real Adaboost, Gentle Adaboost, and Modest Adaboost) as IDSs by using five IDS public benchmark datasets is prepared. Necessary pre- processing steps are applied and Real AdaBoost, Gentle AdaBoost, and Modest AdaBoost algorithms are used to deal with the binary classification problem.

Pros: Boosting comes with an easy to read and interpret algorithm, making its prediction interpretations easy to handle. Prediction capability is very efficient through the use of its clone methods, such as bagging, random forest, and decision trees. Boosting is a method that curbs over-fitting easily and efficiently.

Cons: The disadvantage is that running the Modest AdaBoost in more depths increases the running time considerably which is a big challenge in IDSs. The simulation results on the datasets reveal that the proposed methods are efficient and make boosting approach a strong player for intrusion data classification.

4. Proposed work:

We have tried to detect the intruders by implementing the network intrusion detection system using neural network with the help of technologies: Python. This will be done using 9 different algorithms as below:

Algorithms used:

- Naive Bayes
- Support Vector Machine
- Decision Tree
- Random Forest
- K- Nearest Neighbors
- Along with machine learning, we will be using parallel computing to decrease the computational time and improve the performance of the model.

4.1 Existing models:

Today, in the field of data science and artificial intelligence, network intrusion detection is done using the individual classification models : Support Vector Machine, Logical Regression, Naïve Bayes, K-Nearest Neighbour's,[4] Decision Tree, Random forest based on their accuracy rate and outcome of evaluation metrics using confusion matrix; merging of individual ML to get hybrid models that are more effective than individual ones; forming network of nodes with each classifier in the nodes and their weightage carried in the edges for computation of result from one level to another(ensemble methods); running neural network where input attributes of the dataset lead to the binary outcome by parsing the network again and again.

4.2 Dataset:

The dataset for Network Intrusion Detection System (NIDS) is taken from Kaggle.

https://www.kaggle.com/sampadab17/network-intrusion-detection?select=Train_data.csv

The dataset consists of the ‘protocol service’ (tcp, udp etc) along with the ‘ftp service' and other attributes are categorized as 0 or 1 and the last column ‘class’ depicts whether the protocol service is normal or anomaly.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	duration	protocol_t	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_faile	logged_in	num_com	root_shell	su_attemp	num_root	num_file_	num_shell	num_acce	num_outb	is_host_lo
2	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	tcp	remote_jo	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	tcp	http	SF	287	2251	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15	0	tcp	ftp_data	SF	334	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
16	0	tcp	name	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	tcp	netbios_n	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	tcp	http	SF	300	13788	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19	0	icmp	eco_i	SF	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	tcp	http	SF	233	616	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
21	0	tcp	http	SF	343	1178	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22	0	tcp	mtp	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	tcp	http	SF	253	11905	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 1: First 21 columns of the Dataset

	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
1	is_host_lo	is_guest_l	count	srv_count	serror_rat	srv_serror	rerror_rat	srv_rerror	same_srv_	diff_srv_r	srv_diff_h	dst_host_c	dst_host_s	dst_host_s	dst_host_s	dst_host_r	dst_host_r	dst_host_s	dst_host_s	dst_host_r	dst_host_r	class
2	0	0	2	2	0	0	0	0	1	0	0	150	25	0.17	0.03	0.17	0	0	0	0.05	0	normal
3	0	0	13	1	0	0	0	0	0.08	0.15	0	255	1	0	0.6	0.88	0	0	0	0	0	normal
4	0	0	123	6	1	1	0	0	0.05	0.07	0	255	26	0.1	0.05	0	0	1	1	0	0	anomaly
5	0	0	5	5	0.2	0.2	0	0	1	0	0	30	255	1	0	0.03	0.04	0.03	0.01	0	0.01	normal
6	0	0	30	32	0	0	0	0	1	0	0.09	255	255	1	0	0	0	0	0	0	0	normal
7	0	0	121	19	0	0	1	1	0.16	0.06	0	255	19	0.07	0.07	0	0	0	0	1	1	anomaly
8	0	0	166	9	1	1	0	0	0.05	0.06	0	255	9	0.04	0.05	0	0	1	1	0	0	anomaly
9	0	0	117	16	1	1	0	0	0.14	0.06	0	255	15	0.06	0.07	0	0	1	1	0	0	anomaly
10	0	0	270	23	1	1	0	0	0.09	0.05	0	255	23	0.09	0.05	0	0	1	1	0	0	anomaly
11	0	0	133	8	1	1	0	0	0.06	0.06	0	255	13	0.05	0.06	0	0	1	1	0	0	anomaly
12	0	0	205	12	0	0	1	1	0.06	0.06	0	255	12	0.05	0.07	0	0	0	0	1	1	anomaly
13	0	0	199	3	1	1	0	0	0.02	0.06	0	255	13	0.05	0.07	0	0	1	1	0	0	anomaly
14	0	0	3	7	0	0	0	0	1	0	0.43	8	219	1	0	0.12	0.03	0	0	0	0	normal
15	0	0	2	2	0	0	0	0	1	0	0	2	20	1	0	1	0.2	0	0	0	0	anomaly
16	0	0	233	1	1	1	0	0	0	0.06	0	255	1	0	0.07	0	0	1	1	0	0	anomaly
17	0	0	96	16	1	1	0	0	0.17	0.05	0	255	2	0.01	0.06	0	0	1	1	0	0	anomaly
18	0	0	8	9	0	0.11	0	0	1	0	0.22	91	255	1	0	0.01	0.02	0	0	0	0	normal
19	0	0	1	1	0	0	0	0	1	0	0	1	16	1	0	1	1	0	0	0	0	anomaly
20	0	0	3	3	0	0	0	0	1	0	0	66	255	1	0	0.02	0.03	0	0	0.02	0	normal
21	0	0	9	10	0	0	0	0	1	0	0.2	157	255	1	0	0.01	0.04	0	0	0	0	normal
22	0	0	223	23	1	1	0	0	0.1	0.05	0	255	23	0.09	0.05	0	0	1	1	0	0	anomaly
23	0	0	280	17	1	1	0	0	0.06	0.05	0	238	17	0.07	0.06	0	0	0.99	1	0	0	anomaly
24	0	0	8	10	0	0	0	0	1	0	0.2	87	255	1	0	0.01	0.02	0	0	0	0	normal

Figure 2: Next 20 columns and the Class values of the Dataset

4.3. Modules:

- **Learning different Machine Learning models:** Includes learning Machine learning models in detail and analyzing them. Also deciding how these models are useful for our project.
- **Requirements Collecting requirements and analyzing them:** It will also include the specific requirements for each of the model and can be applies in our applied[3].
- **KNN model A machine learning model:** We will use the constraints acquired to predict and produce an output using the KNN algorithm [3].

- **Naïve Bayes A machine learning model:** We will use the constraints acquired to predict and produce an output using the Naïve Bayes algorithm.
- **Decision Tree A machine learning model:** We will use the constraints acquired to predict and produce an output using the Decision Tree algorithm.
- **Random forest A machine learning model:** We will use the constraints acquired to predict and produce an output using the Random forest algorithm [4].
- **Support Vector Machine (SVM):** A machine learning model. We will use the constraints acquired to predict and produce an output using the support vector [7].

4.4. Sub Modules:

- Collection of Dataset
- Data pre-processing
- Training the machine learning models
- Testing the trained machine learning models
- Validating the trained machine learning models
- Evaluating all the model
- Training the models by Principle Component Analysis
- Testing the machine learning models trained by PCA
- Validating the machine learning models trained by PCA
- Visualizing the testing result of PCA machine learning models

4.5. Network Intrusion Detection Flow Chart:

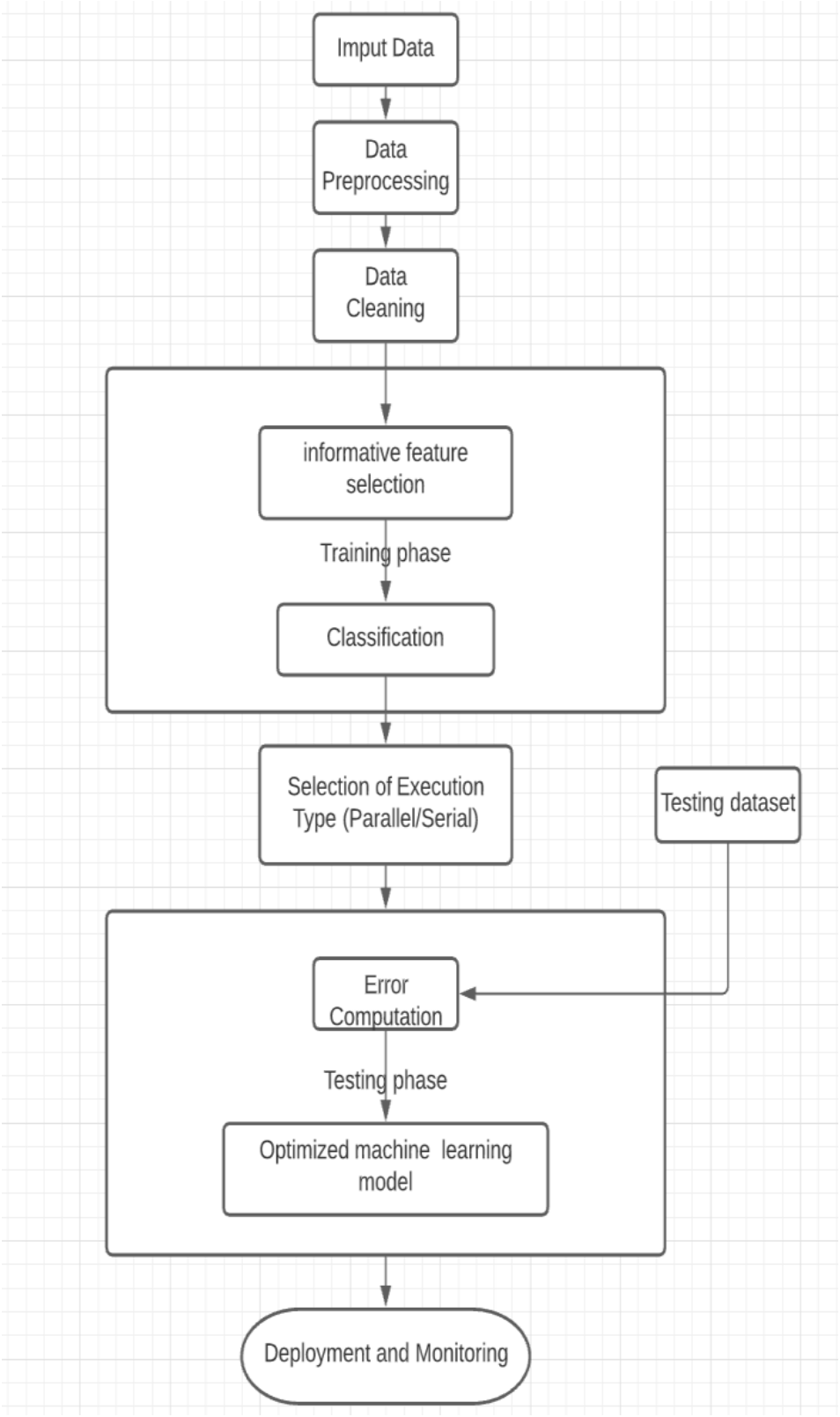


Figure 3: Flowchart for Incremental Process Model of IDS

Dataset: It's the database authorized by national authorities that will be used in the project.

Data Retrieval: It is the process of extracting data from the database for testing.

Data Processing, Wrangling: It is the process of cleaning and unifying untidy and complex datasets for analysis, this includes mapping data from one format to another to organize data.

Feature Extraction , Engineering: It is a process of reducing dimensionality by which an initial set of data is reduced to more easier groups for processing. Feature Scaling, Selection: This is a technique used to make the independent features present in the data in a fixed range standard. It is carried out during the data pre- processing phase to handle highly varying magnitudes or values or units.

Modeling: is the stage where the machine learning algorithm is applied and results are calculated.
Model Evaluation: This phase of the project tests the and prepares Machine Learning Models it for evaluation. Lastly, at this stage all the algorithms are compared and the optimum results corresponding to the best algorithm is chosen.

4.7 Algorithm Explanation:

1. Naive Bayes

Definition: Naive Bayes algorithm is based on Bayes' theorem by assuming to keep every pair of features independent. These classifiers work well in many situations such as classification of documents and spam filtering.

Pseudo Code:

Input: Training dataset T, F= (fi, f2, f3,.., fn) in testing dataset // value of the predictor variable Output:
A class of testing dataset.

Steps:

- Read the training dataset T
- Calculating the mean and standard deviation in each class of the predictor variables Repeat
- Calculate the probability of fi using the gauss density equation in each class until the probability of all predictor variables (fi, f2, f3,.., fn) has been calculated. Calculate the likelihood for each class.
- Get the greatest likelihood;

Advantages: This algorithm requires only needs a small amount of training data to estimate the necessary parameters. Naive Bayes classifiers are fast compared to more sophisticated methods[9].

Disadvantages: Naive Bayes is known for its bad estimations.

2. K- Nearest Neighbors:

Definition: K- Nearest Neighbours based classification is a class of lazy learning as it does not attempt to construct a general model, but simply stores instances of the training data. Classification is done from a majority vote of the k nearest neighboring points of each point[5].

Pseudo Code:

1. k-Nearest Neighbor Classify (X, Y,x) where X: training data, and Y: class labels of X. "x" is an unknown sample
2. for i = 1 to m do Compute distance d(x,x) end for
3. Compute set I containing indices for the k smallest distances d(x,x). return majority label for {Y, where i belongs to I}

Advantages: This algorithm is easy to implement, noisy to diverse training data, and effective if training data is large.

Disadvantages: Need to determine the value of K for many iterations and the computation cost is high as it needs to compute the distance of each instance to all of the training samples.

3. Random Forest:

Definition: Random forest classifier works as an estimator that fits a number of decision tree models on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub sample size is the same as the original input sample size but the samples are drawn with replacement.

Pseudo Code :

To generate c classifiers:
for i = 1 to c do
Sample the training data D randomly with replacement to generate Di Create a
root node, Ni containing Di

Call BuildTree(N;) end
for BuildTree(N):
if N contains occurrences of only one class then return
else
Randomly select x% of the possible splitting features in N Select the feature F with the highest information gain to split on Create f child nodes of 1, 2, ..., N, , where F has f possible values (F1, ... ,FN)
for i = 1 to f do
Set the contents of N; to D; //where D, is all instances in N that match Call
BuildTree(N)
end for
end if

Advantages: Reduction in the over fitting and the random forest classifier is more precise than decision trees in lots of cases.

Disadvantages: It is complex , with slow real time prediction and difficult to implement.

4. Decision Tree:

Definition: with the data of attributes together with its classes or features, a decision tree makes a sequence of rules that are used to classify the data.

Pseudo Code:

GenDecTree(Sample S, Features F) Steps:
1. If stopping_condition(S,F) = true then Leaf = createNode() leafLabel = classify(s)
return leaf root = createNode()
2. root.test_condition =findBestSpilt(S,F)
3. V = {vs va possible outcomecfroot.test_condition}
4. For each value v € V:
5. Sy = {s | root.test_condition(s) = v and s 6 S}; Child = TreeGrowth (S,F);
6. Add child as descent of root and label the edge {root → child} as v
7. returnroot

Advantages: its decision tree is simple to understand and visualize, it also requires very little data preparation, and can handle both numerical and categorical data types.

Disadvantages: these Decision tree may create complex trees that do not generalize well, and they can be unstable because of small variations in the data might result in a completely different tree being generated.

5. Support Vector Machine:

Definition: Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. The examples are mapped into that same space and predicted to be placed into a category based on the side of the gap they fall in.

Pseudo Code:

1. Plot the optimal separating hyperplane & margins
2. Plotting new test data points for classification
3. Plot the final Graph
4. Select kernel of support Vector Machine
5. Identify the right hyper-plane
6. Find the hyper-plane to segregate to classes
7. Plot the new final graph

Advantages: It is memory efficient and effective in high dimensional spaces and also uses a subset of training points in the decision function.

Disadvantages: The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

5. Implementation:

This section includes the implementation of the Network Intrusion Detection System[1] and its results, developed in this course of a project. The user can choose any of these classifications models individually or run all models serially or parallelly. On running any of these models, the system tests the model and gives it accuracy, precision and recall along with the Principal Component Analysis Graph corresponding to that model as shown below:

```
(base) Adarshs-MacBook-Pro:~ adarshreddy$ cd /Users/adarshreddy/Desktop/12
(base) Adarshs-MacBook-Pro:12 adarshreddy$ ls
IDS_dataset.csv main.py
(base) Adarshs-MacBook-Pro:12 adarshreddy$ python ./main.py

Welcome to Network Intrusion Detection System
                    Make your choice :
                    0. Logical Regression
                    1. Random Forest
                    2. Support Vector Machine - Linear
                    3. Support Vector Machine - Poly
                    4. Support Vector Machine - rbf
                    5. Support Vector Machine - Sigmoid
                    6. Decision Tree
                    7. Naive Bayes
                    8. K-nearest Neighbours
                    9. Run all Algorithms
                    10. Quit
```

Figure 4: Implementation of Network Intrusion Detection System

```
Welcome to Network Intrusion Detection System
                    Make your choice :
                    0. Logical Regression
                    1. Random Forest
                    2. Support Vector Machine - Linear
                    3. Support Vector Machine - Poly
                    4. Support Vector Machine - rbf
                    5. Support Vector Machine - Sigmoid
                    6. Decision Tree
                    7. Naive Bayes
                    8. K-nearest Neighbours
                    9. Run all Algorithms
                    10. Quit
9
Select how you want to run it :
                                0. Serially
                                1. Parallelly
                                2. Go Back
1
Accuracy of the Decision Tree Model is :  99.94046437785275
Precision of the Decision Tree Model is :  99.96025437201908
Recall of the Decision Tree Model is :  99.98012323593719
Accuracy of the Gaussian Naive Bayes Model is :  61.222464774756894
Precision of the Gaussian Naive Bayes Model is :  100.0
Recall of the Gaussian Naive Bayes Model is :  61.16080302126814
Accuracy of the Random Forest Model is :  99.8809287557055
Precision of the Random Forest Model is :  99.88088147706968
Recall of the Random Forest Model is :  100.0
Accuracy of the Support Vector Machine - poly Model is :  99.86108354832308
Accuracy of the Support Vector Machine - linear Model is :  99.86108354832308
Precision of the Support Vector Machine - poly Model is :  99.86105597459309
Recall of the Support Vector Machine - poly Model is :  100.0
Precision of the Support Vector Machine - linear Model is :  99.86105597459309
Recall of the Support Vector Machine - linear Model is :  100.0
Accuracy of the K-nearest Neighbours Model is :  99.8809287557055
Precision of the K-nearest Neighbours Model is :  99.88088147706968
Recall of the K-nearest Neighbours Model is :  100.0
Accuracy of the Support Vector Machine - sigmoid Model is :  99.78170271879341
Precision of the Support Vector Machine - sigmoid Model is :  99.86094557012316
Recall of the Support Vector Machine - sigmoid Model is :  99.92049294374876
Accuracy of the Support Vector Machine - rbf Model is :  99.84123834094066
Precision of the Support Vector Machine - rbf Model is :  99.84123834094066
Recall of the Support Vector Machine - rbf Model is :  100.0
```

Figure 5: Running all algorithms and recording their evaluation metrics

6. Results and Discussions:

6.1 Data Preprocessing and Feature Selection

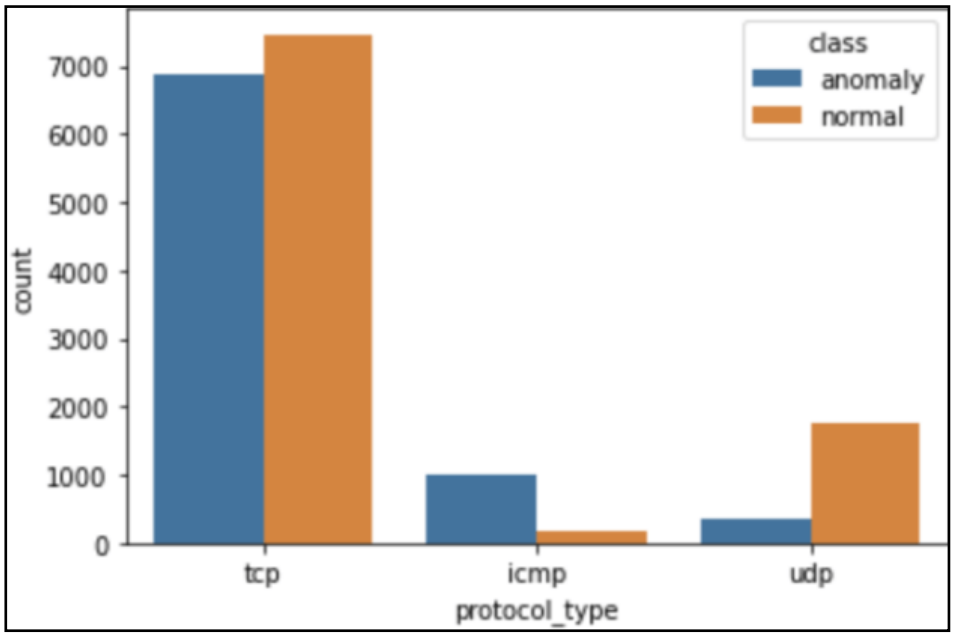


Figure 6: Class distribution according to the protocol type

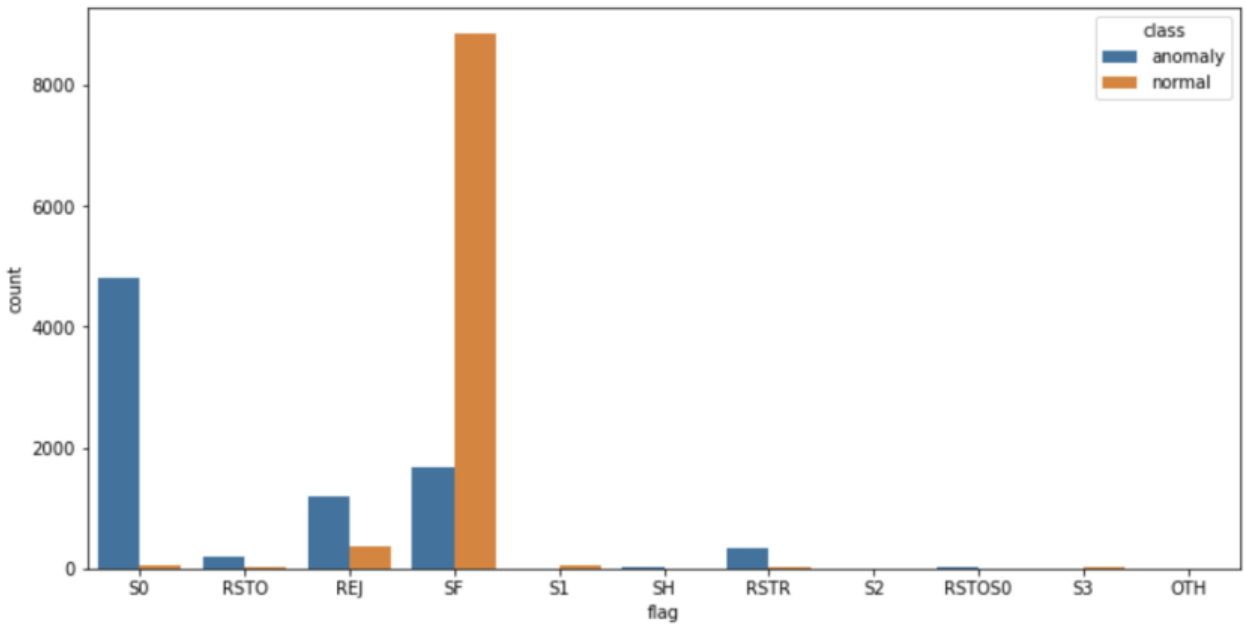


Figure 7: Class distribution with respect to mode of traffic

Observations from Data Analysis of (Figure 6 and Figure 7)

- We found an imbalance in the target column "class" from our dataset. But it is not significant, otherwise we could go for oversampling.
- 80% of all network traffic belongs to TCP while 12% to UDP and rest to ICMP.
- Most ICMP traffic had anomalies; most of the UDP traffic did not have anomalies; while the distribution was almost equal in case of TCP.
- The traffic distribution on the basis of flags was also uneven where most of it had SF(Sign Flag).
- SF flag tagged traffic was normal, while that set S0 flag had anomalies.
- A vast majority of the traffic was unique.
- Very low chance of the traffic to have the same destination host and service.

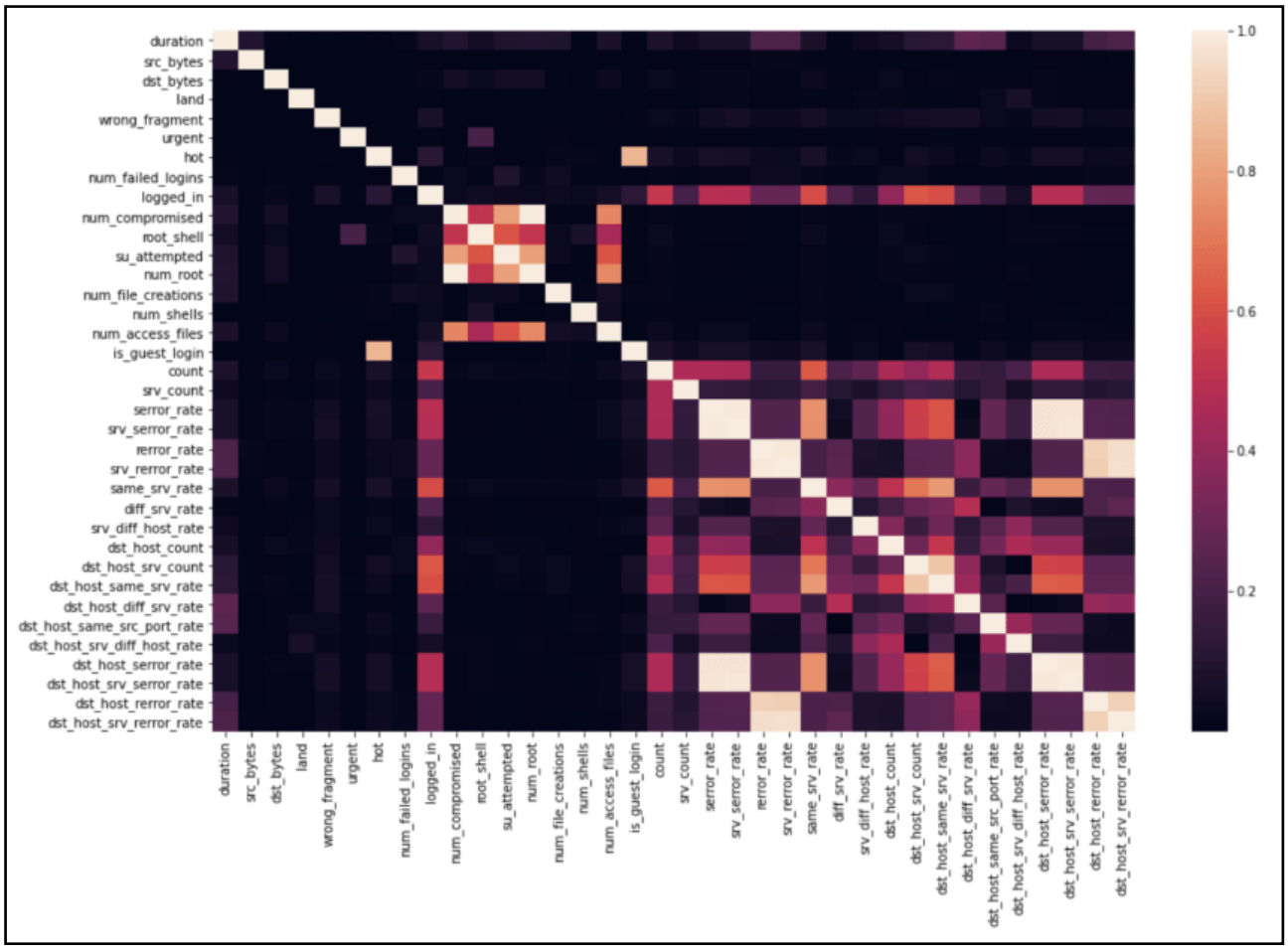


Figure 8: Correlation heatmap of the dataset

Figure 8 represents the correlation heatmap, we can see that most of the data has very low correlation. Very Few features were highly correlated with our target class namely, same_srv_rate, dst_host_srv_count, which will be helpful for our model. We will be using the only

the features with high correlation coefficient to obtain better accuracy and reduce the time consuming by removing unnecessary attributes from the equations.

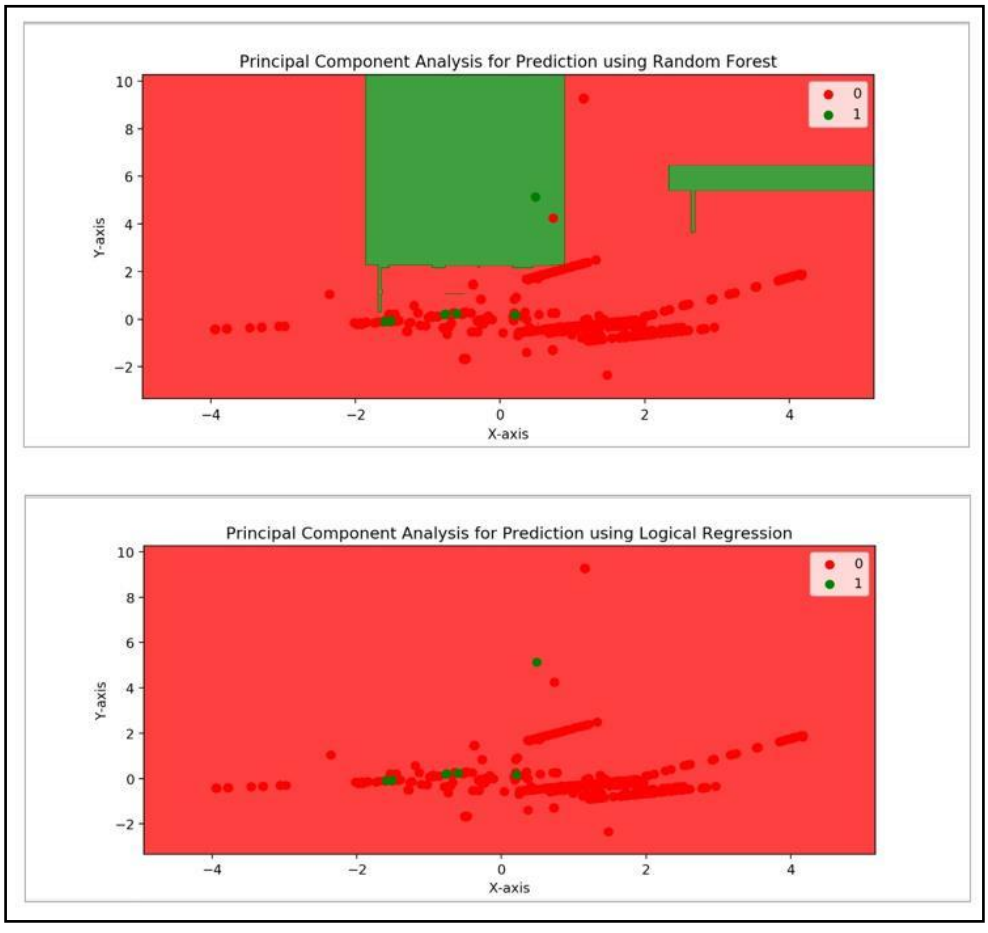
6.2 Principal Component Analysis plots

The arrangement of a PCA plot is like this:

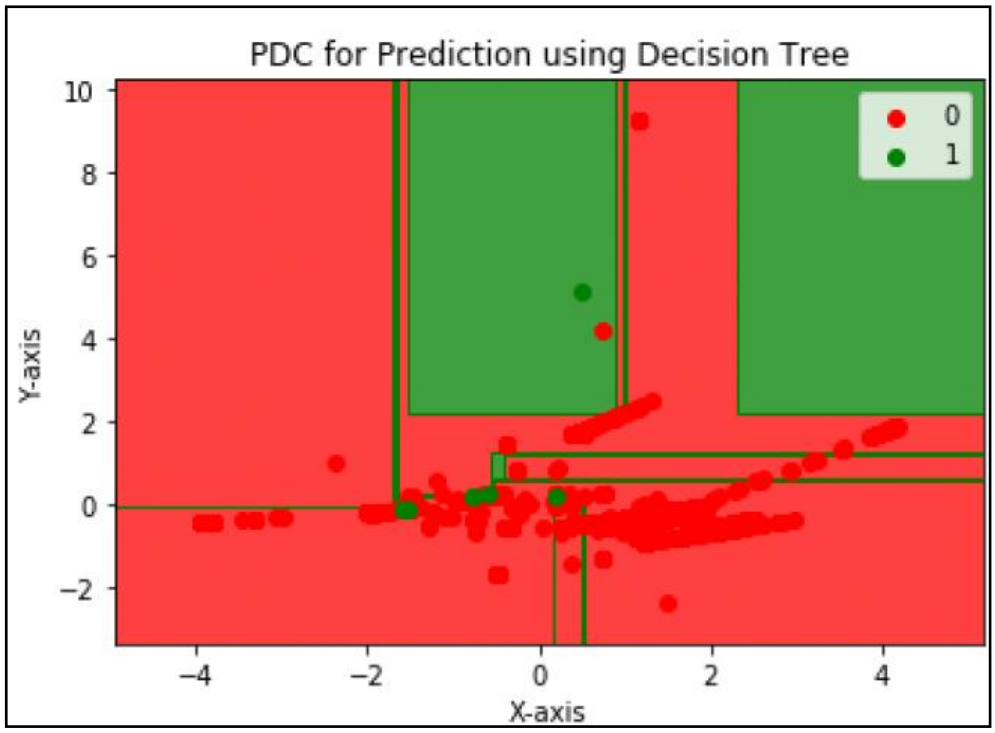
- Bottom axis: PC1 score.
- Left axis: PC2 score.

The bottom and left axes are of the PCA graph — used to read PCA scores of the samples. PCA doesn't discard any of the characteristics or samples . Instead, it will reduce the overwhelming number of dimensions by constructing the principal components (PCs). PCs describe the variation and account for the varied influences of the original characteristics. These can be traced back from the PCA graph to find out the differences among these clusters.

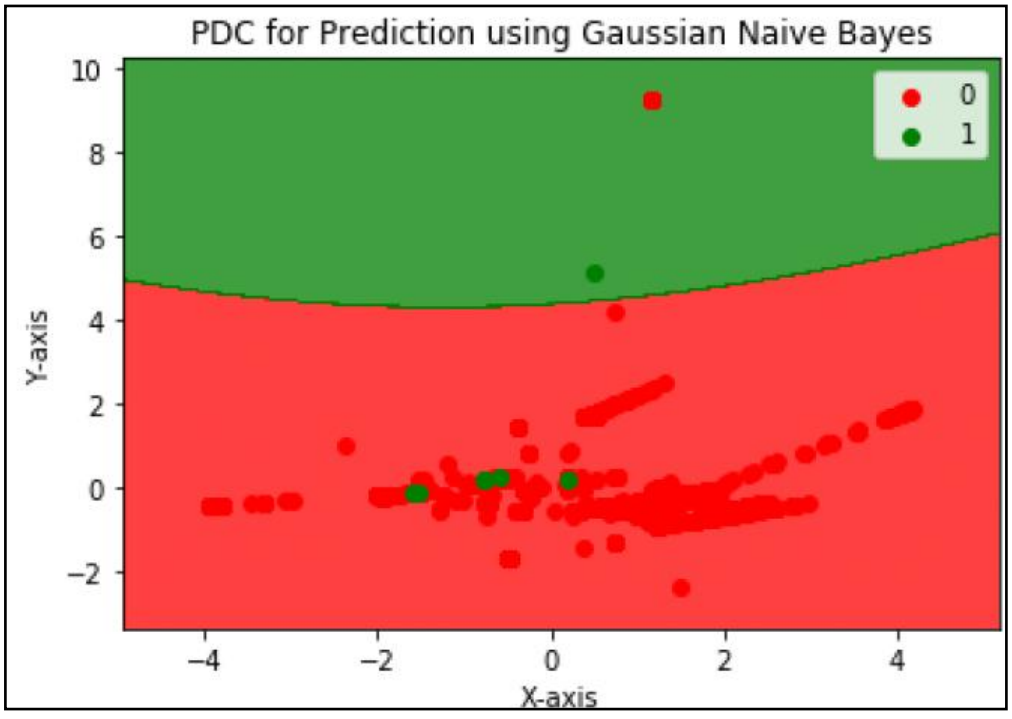
Principle Component Analysis Graph for Prediction using Random Forest and Logical Regression:



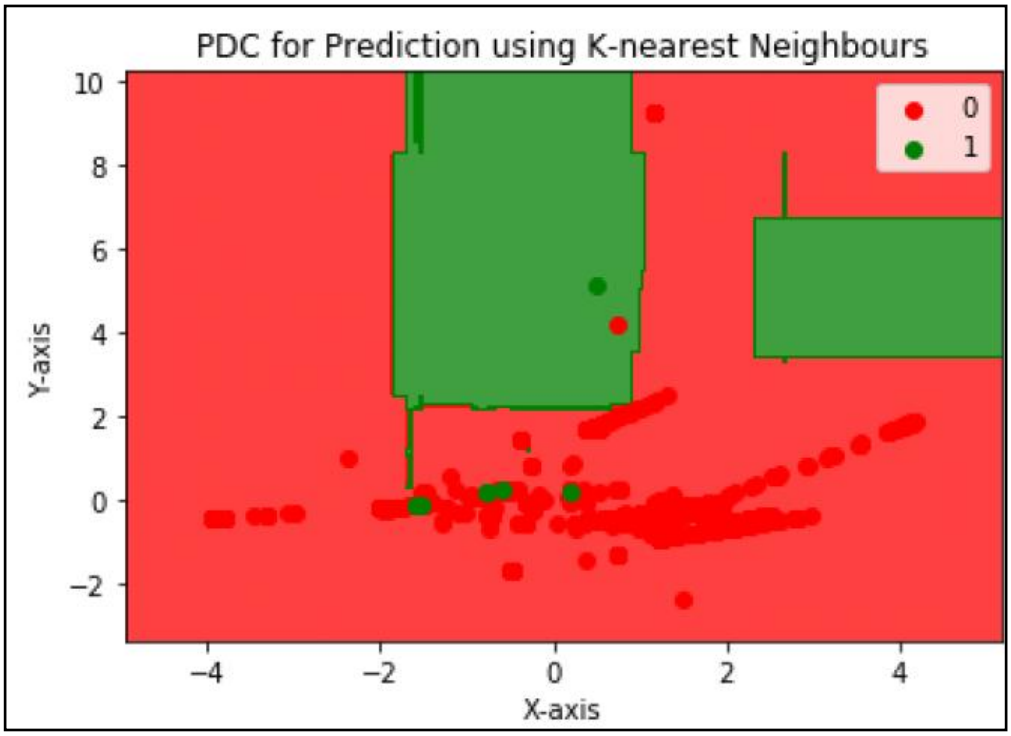
Principle Component Analysis Graph for Prediction using Decision Tree:



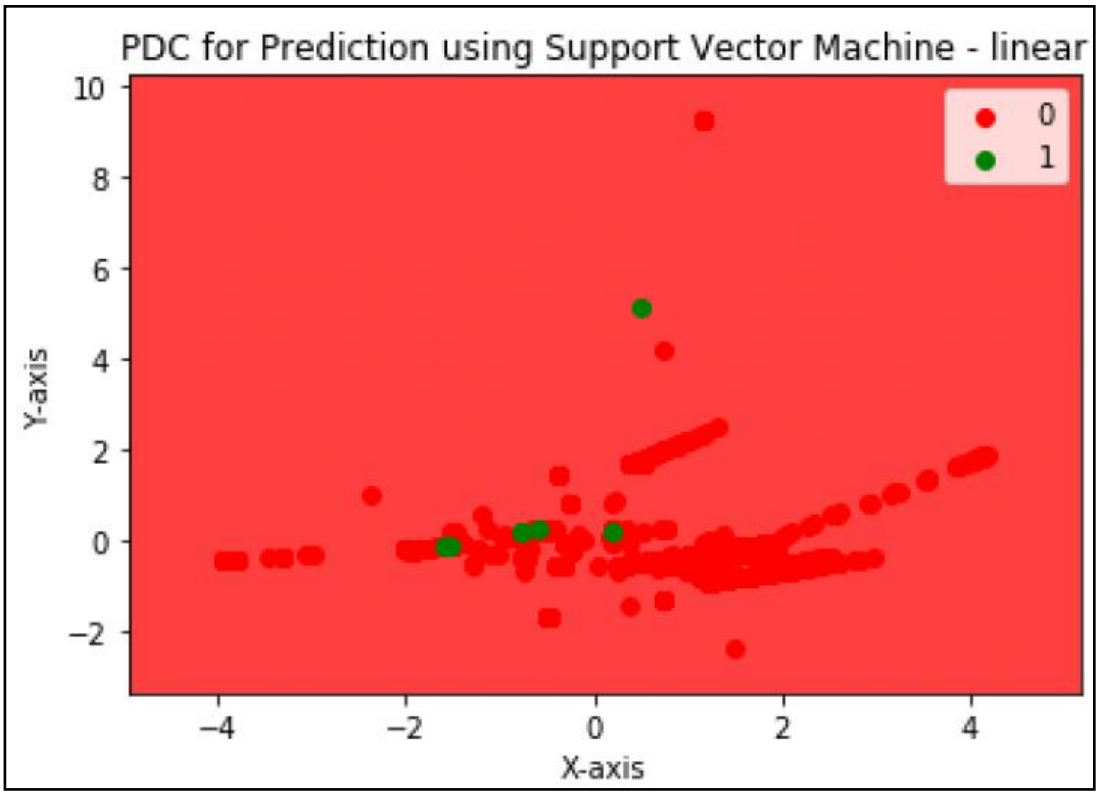
Principle Component Analysis Graph for Prediction using Naïve Bayes:



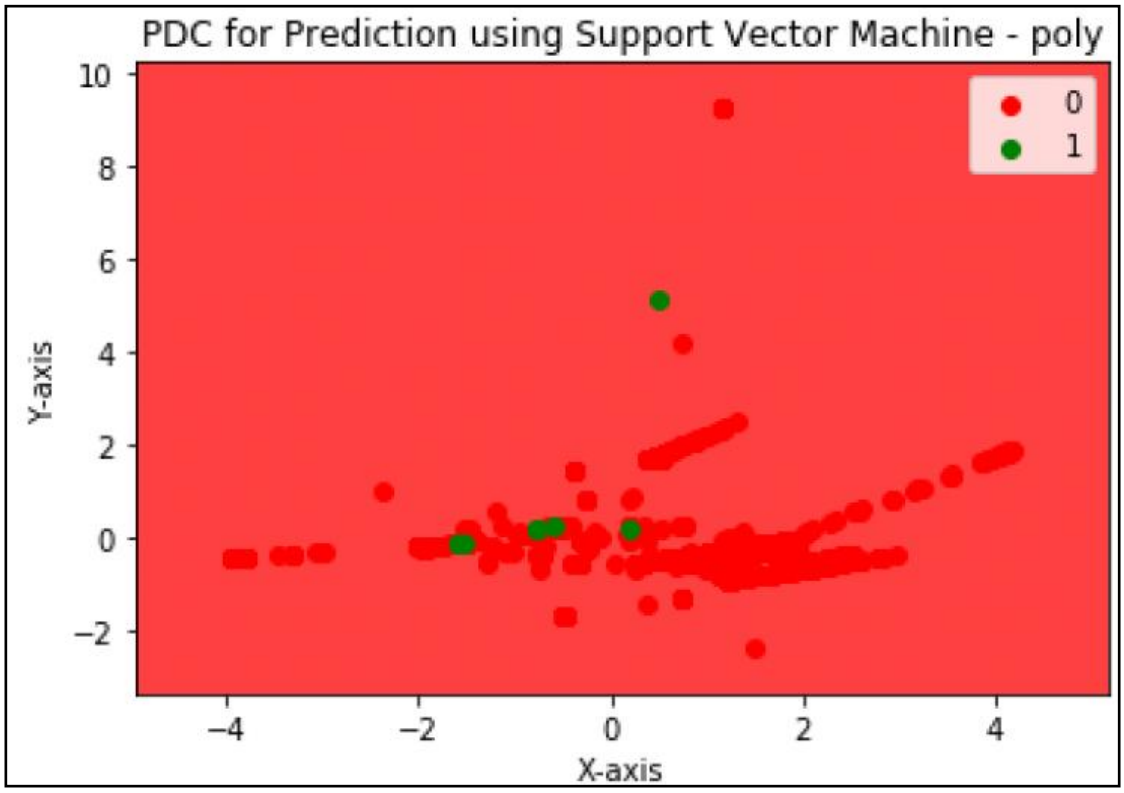
Principle Component Analysis Graph for Prediction using K-Nearest Neighbours:



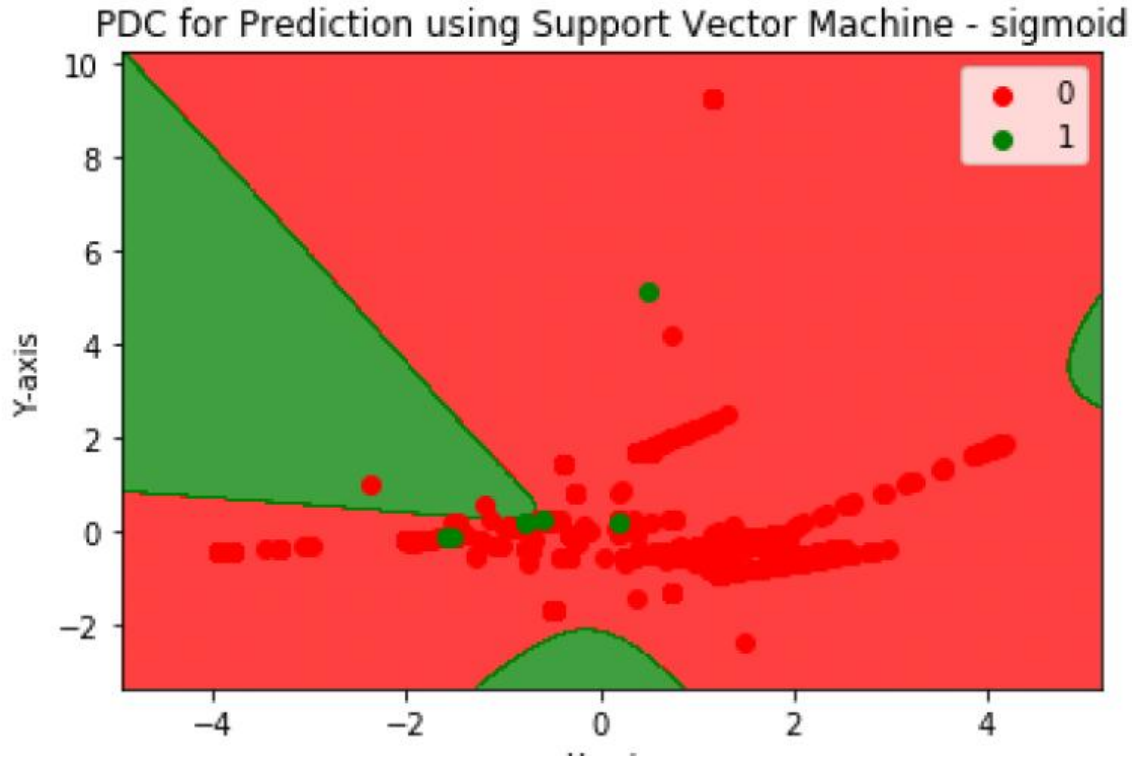
Principle Component Analysis Graph for Prediction using Support Vector Machine – linear:



Principle Component Analysis Graph for Prediction using Support Vector Machine – poly:



Principle Component Analysis Graph for Prediction using Support Vector Machine – sigmoid:



Output Table:

This table shows the accuracy, precision and recall of all 9 Classification models trained for the input dataset.

Classification model	Accuracy	Precision	Recall
Logical Regression	99.86	99.86	100
K-nearest neighbours	99.88	99.88	100
Naive bayes	61.22	100	61.16
Decision Tree	99.94	99.96	99.98
Random Forest	99.88	99.88	100
Support Vector Machine-Linear	99.86	99.86	100
Support Vector Machine-Poly	99.86	99.86	100
Support Vector Machine-rbf	99.78	99.86	99.92
Support Vector Machine-Sigmoid	99.84	99.84	100

Figure 7 shows the comparison chart among all the 9 Classification algorithm validated against the test set. It shows that the **KNN (K – Nearest Neighbours) and Random Forest** algorithms havethe highest scores and are more accurate than other models.

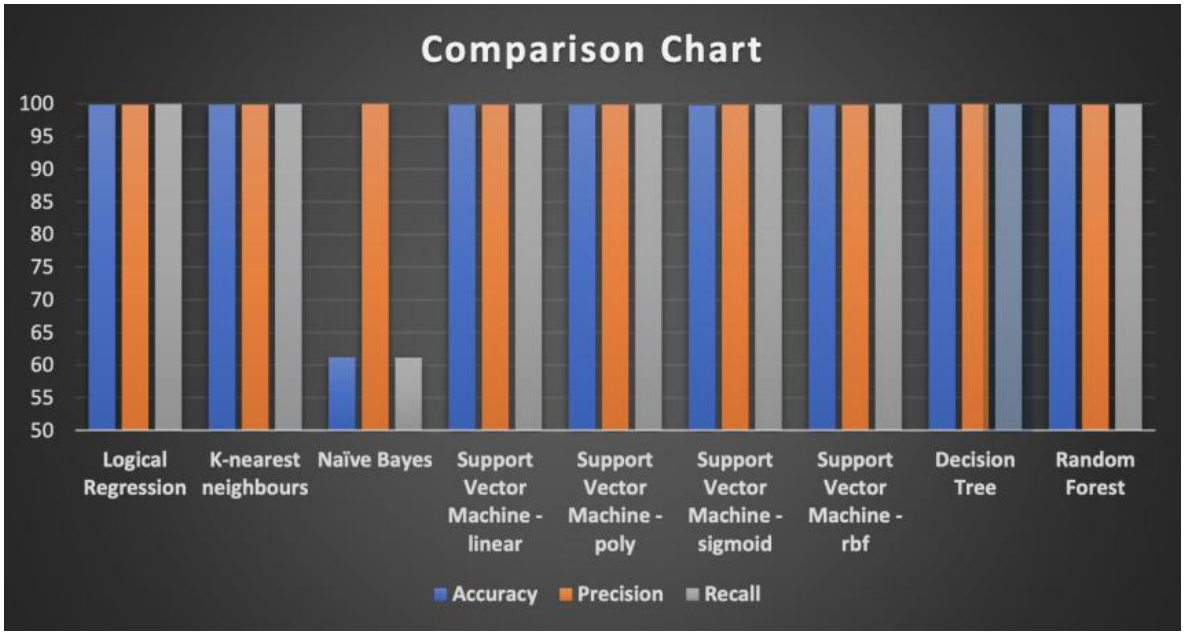


Figure 7: Comparison graph for Evaluation metrics of all algorithms

7. Conclusion

Intrusion Detection Systems (IDS) are the second layer of defence. It detects if any attacks are present within the traffic that goes in through the holes in the firewall. An Intrusion Detection System (IDS) constantly monitors actions in a certain environment and decides whether they are part of a possible hostile attack or a valid use of the environment. The intrusion discovery and intrusion avoidance fields are amazingly powerful, with new discoveries, capacities, and models being made constantly. A lot of research on information representation techniques for intrusion location information is additionally right now being led. The conclusion, of this study has shown that the data mining methods generate interesting rules that are crucial for intrusion detection and prevention in the networking industry. We also showed how intrusion detection can have a big benefit benefit from high performance computing and parallelization techniques . This project attempts to address the problem of intrusion attack detection with the use of data mining supervised model. In summary, the results from this study can contribute towards in improving the networking security.

7.1. Future Work:

In future, it is possible to provide extensions or modifications to the proposed clustering and classification algorithms using intelligent agents to achieve further increased performance. Apart from the experimented combination of data mining techniques, further combinations such as artificial intelligence, soft computing and other deep learning algorithms can be used to improve the detection accuracy and to reduce the rate of false negative alarm and false positive alarm. Finally, the intrusion detection system can be extended as an intrusion prevention system to enhance the performance of the system.

8. References:

[1] Midzic, A., Z. Avdagic, and S. Omanovic. "Intrusion detection system modeling based on neural networks and fuzzy logic." In *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*, pp. 189-194. IEEE, 2016.

[2] Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz. "A survey of intrusion detection systems based on ensemble and hybrid classifiers." *Computers & Security* 65 (2017): 135-152.

[3] Velliangiri, S., and P. Karthikeyan. "Hybrid optimization scheme for intrusion detection using considerable feature selection." *Neural Computing and Applications* (2019): 1-15.

[4] Pise, Nitin, and Parag Kulkarni. "Algorithm selection for classification problems." In *2016 SAI Computing Conference (SAI)*, pp. 203-211. IEEE, 2016.

[5] Raihan-Al-Masud, Md, and Hossen Asiful Mustafa. "Network Intrusion Detection System Using Voting Ensemble Machine Learning." In *2019 IEEE International Conference on Telecommunications and Photonics (ICTP)*, pp. 1-4. IEEE, 2019.

[6] Devan, Preethi, and Neelu Khare. "An efficient XGBoost–DNN-based classification model for network intrusion detection system." *Neural Computing and Applications* (2020): 1-16.

[7] Hosseini, Soodeh, and Behnam Mohammad Hasani Zade. "New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN." *Computer Networks* 173 (2020): 107168.

[8] Ugochukwu, Chibuzor John, E. O. Bennett, and P. Harcourt. An intrusion detection system using machine learning algorithm. LAP LAMBERT Academic Publishing, 2019.

[9] Park, Seunghyun, and Jin-Young Choi. "Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms." *Sensors* 20, no. 14 (2020): 3934.

[10] Nguyen, Minh Tuan, and Kiseon Kim. "Genetic convolutional neural network for intrusion detection systems." *Future Generation Computer Systems* 113 (2020): 418-427.

[11] Zhang, Jilin, Hangdi Tu, Yongjian Ren, Jian Wan, Li Zhou, Mingwei Li, and Jue Wang. "An adaptive synchronous parallel strategy for distributed machine learning." *IEEE Access* 6 (2018): 19222-19230.

[12] Shahraki, Amin, Mahmoud Abbasi, and Øystein Haugen. "Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost." *Engineering Applications of Artificial Intelligence* 94 (2020): 103770.

8. Appendix and Code:

```
#
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
#
dataset =pd.read_csv('IDS_dataset.csv')
X=dataset.iloc[:,0:13].values
y=dataset.iloc[:,13].values
#
def Multilabelencoder(X,k):
    from sklearn.preprocessing import LabelEncoder
    X[:,k]= LabelEncoder().fit_transform(X[:,k])
    return X

for i in range(1,4):
    X=Multilabelencoder(X,i)
#
from sklearn.preprocessing import LabelEncoder
y= LabelEncoder().fit_transform(y)
#
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.2,random_state=0)
#
from sklearn.preprocessing import StandardScaler
sc_X= StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
#
import time
s=time.time()
from sklearn.linear_model import LogisticRegression
classifier1 = LogisticRegression(random_state=0)
classifier1.fit(X_train,y_train)

from sklearn.ensemble import RandomForestClassifier
```

```
classifier2 = RandomForestClassifier(n_estimators=10, criterion='entropy',random_state=0)
classifier2.fit(X_train,y_train)

from sklearn.svm import SVC
classifier3 = SVC(kernel ='linear', random_state=0)
classifier3.fit(X_train,y_train)

from sklearn.svm import SVC
classifier4 = SVC(kernel ='poly', random_state=0)
classifier4.fit(X_train,y_train)

from sklearn.svm import SVC
classifier5 = SVC(kernel ='rbf', random_state=0)
classifier5.fit(X_train,y_train)

from sklearn.svm import SVC
classifier6 = SVC(kernel ='sigmoid', random_state=0)
classifier6.fit(X_train,y_train)

from sklearn.tree import DecisionTreeClassifier
classifier7 = DecisionTreeClassifier(criterion ='entropy',random_state=0)
classifier7.fit(X_train,y_train)

from sklearn.naive_bayes import GaussianNB
classifier8 = GaussianNB()
classifier8.fit(X_train,y_train)

from sklearn.neighbors import KNeighborsClassifier
classifier9 = KNeighborsClassifier(n_neighbors=3,metric='minkowski',p=2)
classifier9.fit(X_train,y_train)

e=[classifier1,classifier2,classifier3,classifier4,classifier5,classifier6,classifier7,classifier8,classifier9]

X_first=X_test
from sklearn.decomposition import PCA
pca= PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

```
explained_variance= pca.explained_variance_ratio_

from sklearn.linear_model import LogisticRegression
classifier1 = LogisticRegression(random_state=0)
classifier1.fit(X_train,y_train)

from sklearn.ensemble import RandomForestClassifier
classifier2 = RandomForestClassifier(n_estimators=10, criterion='entropy',random_state=0)
classifier2.fit(X_train,y_train)

from sklearn.svm import SVC
classifier3 = SVC(kernel ='linear', random_state=0)
classifier3.fit(X_train,y_train)

from sklearn.svm import SVC
classifier4 = SVC(kernel ='poly', random_state=0)
classifier4.fit(X_train,y_train)

from sklearn.svm import SVC
classifier5 = SVC(kernel ='rbf', random_state=0)
classifier5.fit(X_train,y_train)

from sklearn.svm import SVC
classifier6 = SVC(kernel ='sigmoid', random_state=0)
classifier6.fit(X_train,y_train)

from sklearn.tree import DecisionTreeClassifier
classifier7 = DecisionTreeClassifier(criterion ='entropy',random_state=0)
classifier7.fit(X_train,y_train)

from sklearn.naive_bayes import GaussianNB
classifier8 = GaussianNB()
classifier8.fit(X_train,y_train)

from sklearn.neighbors import KNeighborsClassifier
classifier9 = KNeighborsClassifier(n_neighbors=3,metric='minkowski',p=2)
classifier9.fit(X_train,y_train)
```

```


f=[classifier1,classifier2,classifier3,classifier4,classifier5,classifier6,classifier7,classifier8,classifier9]

g=["Logical Regression","Random Forest","Support Vector Machine - linear","Support Vector Machine - poly"
   ,"Support Vector Machine - rbf","Support Vector Machine - sigmoid","Decision Tree","Gaussian Naive Bayes"
   ,"K-nearest Neighbours"]

def abc(h):
    warnings.filterwarnings("ignore")
    classifier=e[h]
    Y_pred = classifier.predict(X_first)
    #
    from  sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test,Y_pred)
    #
    print("Accuracy of the "+g[h]+" Model is : ",(cm[0][0]+cm[1][1])*100/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0]))
    print("Precision of the "+g[h]+" Model is : ",(cm[0][0])*100/(cm[0][0]+cm[1][0]))
    print("Recall of the "+g[h]+" Model is : ",(cm[0][0])*100/(cm[0][0]+cm[0][1]))
    #
    classifier=f[h]
    #
    Y_pred = classifier.predict(X_test)
    #
    from matplotlib.colors import ListedColormap
    X_set,y_set=X_test,y_test
    X1,X2= np.meshgrid(np.arange(start=X_set[:,0].min()-1,stop=X_set[:,0].max()+1,step=0.01)
           ,np.arange(start=X_set[:,1].min()-1,stop=X_set[:,1].max()+1,step=0.01))
    plt.contourf(X1,X2,classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),alpha=0.75
        ,cmap =ListedColormap(('red','green')))
    plt.xlim(X1.min(),X1.max())
    plt.ylim(X2.min(),X2.max())
    for i,j in enumerate(np.unique(y_set)):
        plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],c=ListedColormap(('red','green'))(i),label=j)
    plt.title('Principal Component Analysis for Prediction using '+g[h])
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.legend()
```

```
    plt.show()

import multiprocessing as mp
def master():
    if __name__=='__main__':
        while(1):
            warnings.filterwarnings("ignore")
            s=int(input("""\nWelcome to Network Intrusion Detection System
                    Make your choice :
                    0. Logical Regression
                    1. Random Forest
                    2. Support Vector Machine - Linear
                    3. Support Vector Machine - Poly
                    4. Support Vector Machine - rbf
                    5. Support Vector Machine - Sigmoid
                    6. Decision Tree
                    7. Naive Bayes
                    8. K-nearest Neighbours
                    9. Run all Algorithms
                    10. Quit\n"""))
            if(s<9):
                abc(s)
            elif(s==9):
                s=int(input("""Select how you want to run it :
                                0. Serially
                                1. Parallelly
                                2. Go Back\n"""))
                if(s==0):
                    for i in range(9):
                        abc(i)
                elif(s==1):
                    parallel()
                elif(s==2):
                    pass;
                else:
                    print("INVALID CHOICE :( TRY AGAIN")
            elif(s==10):
                break;
            else:
                print("INVALID CHOICE :( TRY AGAIN")
    else:
        pass

def parallel():
    p = mp.Pool(9)
    for h in range(9):
        p.apply_async(abc, args=(h,))
    p.close()
    p.join()

master()
```