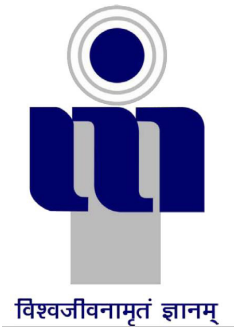
Deploying and analyzing classification algorithms for Intrusion Detection

A project report submitted in partial fulfillment of the requirements for B.Tech. Project

B.Tech.

by

Himanshu Pandey (2018IMT-038)



ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT
GWALIOR-474 015

2021

CANDIDATES DECLARATION

We hereby certify that the work, which is being presented in the report, entitled **Deploying and analyzing classification algorithms for Intrusion Detection**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the institution is an authentic record of our own work carried out during the period *June 2021* to *october 2021* under the supervision of **Dr. Saumya Bhadauria**. We also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Himanshu Pandey

Date: Signatures of the Candidates

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: Signatures of the Research Supervisors

ABSTRACT

Hacking has grown in popularity in recent years, increasing cyber-attack quantity and variety. Malware, denial of service attacks, phishing, and social engineering are all examples of computer network threats. Antiviruses and firewalls are no longer sufficient for effective cybersecurity. To prevent these dangers, you can't only rely on antivirus and firewalls: you need multiple levels of defense. With the capacity to monitor packets from OSI layer 2 (Datalink) through layer 7 (Application), network-based Intrusion Detection Systems (IDSs) offer a supplementary technique of increasing security. IDSs that use anomaly detection can detect unknown assaults, but they are less accurate, resulting in a high amount of false alarms. Machine learning methods are investigated in this thesis to develop IDSs that can be deployed in real-world computer networks. To begin, a three-step optimization strategy is given to increase detection quality: 1) data augmentation to rebalance the dataset, 2) model performance optimization, and 3) ensemble learning to integrate the findings of the best models. This method has the disadvantage of requiring labeled datasets, which are rarely available in real-life situations. As a result, transfer learning is explored to train machine learning models on huge labeled datasets and subsequently finetune them on innocuous network traffic. This method also has problems because the models are trained on previously known assaults and so do not do anomaly detection. As a result, an unsupervised learning-based method is provided. It takes advantage of network prototyping. When anomalies are discovered, they are grouped into attacks or disregarded if they are isolated. Finally, network congestion detection is investigated. The bandwidth usage of various links is forecasted to prevent problems from arising.

Keywords: (TO WRITE) Intrusion detection, machine learning, neural networks, cyber security, unsupervised learning, ensemble learning.

ACKNOWLEDGEMENTS

I am highly indebted to Dr. Saumya Bhadauria and obliged for giving me the autonomy of functioning and experimenting with ideas. I would like to take this opportunity to express my profound gratitude to her not only for her academic guidance but also for her interest in my report and constant support coupled with confidence-boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within me. The nurturing and blossoming of the present work is mainly due to her valuable guidance, suggestions, astute judgment, constructive criticism, and an eye for perfection. My mentor always answered a myriad of my doubts with smiling graciousness and prodigious patience, never letting me feel like a novice by always lending an ear to my views, appreciating and improving them, and by giving me a free hand in my report. It's only because of her overwhelming interest and helpful attitude, the present work has attained the stage it has.

Finally, I am grateful to our Institution and colleagues whose constant encouragement served to renew my spirit, refocus my attention and energy, and helped me in carrying out this work.

Himanshu Pandey 2018IMT-038

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IDSs	IntrusionDetection Systems
SVM	Support Vector Machine
MLP	Multi-Layer Perceptron
LSTM	Long Short-Term Mem-ory network
GAN	enera-tive Adversarial Networks
PCA	Principle Component Analysis
EDA	Exploratory Data Analysis
TN	True Negative
FN	False Negative
TP	True Positive
FP	False Positive
TPR	True Positive Rate
FPR	False Positive Rate
TNR	True Negative Rate
FNR	False Negative Rate
DT	Decision Tree
RF	Random Forest
ROC	Receiver Operating Characteristic

CHAPTER 1

INTRODUCTION AND LITERATURE SURVEY

Computers have become an inextricable aspect of our lives. They are widely employed in all government agencies, businesses, private organizations, hospitals, and private residences. Because of their importance in our lives, protecting them from invasions is a huge task for us. Intrusions are still the most persistent hazards in the cyber world, despite significant advancements in protection technologies. Many ways for analyzing intrusions have been developed to date.This process and the design of the machine learning models are usually managed by a framework such as scikit-learn F. Pedregosa and Duchesnay (2011), Tensorflow M. Abadi and Zheng (2007), PyTorch A. Paszke and Lin (2010), Matlab or Weka M. Hall and ten (2009).These techniques rely on algorithms with the ability to learn directly from the data, without being explicitly programmed. This is particularly convenient considering the great diversity of the traffic. However, despite these advantages, anomaly detection algorithms are rarely deployed in the real world and misuse detection still prevails. The problem of the high false positive rate is often cited as the main reason for the lack of adoption of anomaly-based IDS. Indeed, even a false positive rate of 1 percent can create so many false alarms on a high traffic network that they become impossible for an administrator to process Axelsson (2000). The objective of this thesis is to propose solutions to improve the quality of detection of anomaly-based IDS using machine learning techniques for deployment on real networks. Improving the accuracy of detection on known datasets is not enough to achieve this goal, because the results obtained are not transferable to real networks. Indeed, machine learning models learn the traffic of a dataset and not the traffic to be monitored. They need to be re-trained on the monitored network, which is hardly possible as it requires labeled datasets containing attacks on a real network. The second objective of the thesis is therefore to develop IDSs that can be deployed on unknown networks without labeled datasets.

1.1 Motivation

With time, many various computer attacks are stretching their arms intending to harm the targeted system. For a company, antiviruses or firewalls are no longer sufficient to ensure the security of the systems of the company. Now we need multiple layers of security to ensure truly secured systems among which one of the most important layers is provided by the Intrusion Detection System (IDS) which is designed to protect its target against any potential attack through continuous monitoring of the system. In our thesis, we are going to deploy machine learning models which can detect known attacks through supervised intrusion detection and unknown attacks through unsupervised intrusion detection. Further, the thesis model will be deployed to predict bandwidth utilization and at last, all the machine learning techniques will be compared in detecting intrusions so that we can come up with better and efficient algorithms and others can have a knowledge to which algorithm is to be used in case of intrusion detection.

1.2 Literature Survey

1.2.1 Background

During past few years, the growth of computer networks created new problems related to monitoring user activities and access. Intrusion Detection Systems have signature-based detection (or "misuse detection") and anomaly detection. In signature-based detection, the data monitored by the IDS is compared to known patterns of attacks and it can only detect known attacks while anomaly detection builds a model of the normal behavior of the system and then looks for deviations in the monitored data. Anomaly detection has a wider aspect but it can also detect unknown attacks which can generate irrelevant alarms. Many machine learning techniques have come into the picture to detect anomalies. Some of them can rely on algorithms with the ability to learn directly from the data, without being explicitly programmed. In our thesis, we will analyse the already deployed supervised and unsupervised machine learning algorithms for intrusion detection and work on new technologies to improve there time complexity and accuracy.

1.2.1.1 Key related research and Research Gaps

Another crossover intrusion detection technique described by Kim and Lee (2014) in March 2014, gradually joins misuse location and peculiarity identification in a deteriorated structure. To begin, a decision tree was utilized to develop a misuse detection model, which was then used to break down the normal training data into smaller groups. Then, in each deconstructed region, a one-class support vector machine (1-class

SVM) was employed to create an anomaly detection model. During the integration, the anomaly detection model might leverage the known attack information to improve its capabilities while creating normal behavior profiles. This is the first time a misuse detection model has been used to improve the ability of an anomaly detection model. The C4.5 decision tree does not create a cluster, which can reduce the system's profiling ability and accuracy.

Horng and Su (2011) devised a system for intrusion detection that incorporates a clustering technique, a basic feature selection algorithm, and the Support Vector Machine (SVM). This paper suggested an SVM-based network intrusion detection system with BIRCH hierarchical clustering for data pre-processing, in addition to a basic feature selection technique. In place of the original huge dataset, BIRCH hierarchical clustering gives a well-qualified and reduced dataset for SVM training. In addition to saving time during training, the generated classifiers outperformed SVM classifiers trained on the previously duplicated dataset. In terms of accuracy, however, the proposed method could achieve the highest score of 95.72 percent. In comparison to the other NIDS, this technique performs better in terms of accuracy (Network-based IDS). Only Dos and Probe attacks are detected, not U2L or R2L attacks.

Panda and Patra (2008) introduced hybrid intelligent decision technologies that use data filtering in combination with directed learning methods and a classifier to produce better-classified judgments to detect network assaults. The Naive Bayes model is highly appealing because of its purity, elegance, robustness, and effectiveness, as shown by the findings. Decision trees, on the other hand, have demonstrated their effectiveness in both generalizing and detecting new assaults. The findings reveal that there is no single optimum algorithm that can consistently outperform others in all cases. There may be some reliance on the data's properties in some circumstances. To make better decisions, a domain expert or expert system may use the categorization findings to choose a suitable algorithm.

Juan Wang (2009) presented a decision tree-based intrusion detection system. The information gain ratio is utilized instead of information gain when creating incursion rules. The findings of the experiment reveal that the C4.5 decision tree is viable, effective, and accurate. His research demonstrates that the C4.5 decision tree is a viable technique for implementing decision trees, with about 90% classifier accuracy. However, the mistake rate remains the same with this method.

et.al (2013) provide work on feature reduction using the ADTree algorithm. ADTree also performs well in classification. Furthermore, its easy-to-understand decision rules enable the user to find factors that lead to a higher classification. This knowledge base makes it easier to create support vectors with a reduced dimension for a better classifier. The experiment backs up the idea of using this algorithm as a categorization and knowledge-finding tool. Due to the fewer procedures required to accomplish the cate-

gorization, the process has been simplified and the speed has increased dramatically.
Tavallaee M (2009) submitted a study on the KDD CUP 99 Data Set, and after analyzing the full KDD dataset, it was discovered that there were two major flaws in the data set that impacted the performance of the assessed systems, resulting in a poor evaluation of anomaly detection algorithms. NSL-KDD, which comprises selected records from the KDD data set, was presented as a solution to the problems. Even though the proposed data set has some flaws and may not be a perfect representation of existing networks, they believe that it can still be used as a useful benchmark to help researchers compare different intrusion detection systems due to the lack of public data sets for network-based IDSs.
F. Amiri and Yazdani (2011) proposed the Feature Selection approach to improving the performance of existing classifiers by removing irrelevant information. In addition, PLSSVM, an enhanced Partial Least Squares Support Vector Machine, has been introduced. In this study, a linear and non-linear measure for feature selection during the pre-processing phase was investigated. PLSSVM classified normal and probing assaults data with an accuracy of 95.69 percent and 86.46 percent, respectively. By using linear correlation-based feature selection (LCFS), forward feature selection (FFSA), and modified mutual information, the effect of adjusting feature goodness measure and evaluation function has been examined in this work. Experiments on the KDDcup99 dataset show that feature selection techniques can enhance classification accuracy significantly. PLSSVM, on the other hand, missed a large number of dynamic attacks, such as DoS and U2R attacks, which behaved similarly to normal behavior (78.76 percent and 30.7 percent, respectively).
An alternating decision tree with boosting is proposed by Freund (2004). The new learning algorithm combines decision trees and boosting. They compared the alternating decision tree to the C5.0 method in their article. On smaller datasets, ADtree quickly fits the data, and after 50 iterations, ADtree has a relatively modest error, whereas the stump boost's error remains substantial even after 200 iterations. This is an instance where ADtree's big capacity is advantageous. When comparing the size of classifiers, the ADtree classifiers are substantially smaller than those generated by C5.0 by boosting in all but three cases.

1.3 PYTHON

We have used the PYTHON libraries to implement our project. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.We have used many python libraries like numpy, pandas, matplotlib, seaborn, sklearn, and keras in the deployment of our project.
Key Features

- sklearn library used in the implementation of DecisionTreeClassifier, LogisticRegression, accuracy_ score, confusion_matrix.
- matplotlib library is used for visualisation of figurs.
- numpy and pandas libraries are used for general creating and reading data.

Version - Pyhton3

1.4 OBJECTIVE

The main objective is to design improve the accuracy and time complexity of the already existing supervised and unsupervised machine learning algorithms for Intrusion Detection System(IDS). We have Naive Bayes Classifier, Decision Tree, Logistic Regression, Support Vector Machine(SVM), Random Forest, and XG Boost in supervised machine learning algorithms. In unsupervised machine learning algorithms, we have Ensemble Learning, Autoencoder, Multi-Layer Perceptron(MLP), LSTM and GAN.

CHAPTER 2

DESIGN DETAILS AND IMPLEMENTATION

The proposed strategy is to split the project into two sections. In the first phase, three attribute assessment methods are used to choose features from the retrieved feature set. The second phase trains machine learning classifiers with the selected feature and compares them by estimating their efficiency. We downloaded the KDD Cup dataset for the supervised Intrusion Detection System. Intrusion detection should be possible with the algorithms. It is accomplished by identifying the characteristics that are responsible for their actions. In the presence of better features, classifiers will be better trained, which will improve their performance in predicting incursions.

2.1 MODEL

We use the concept of classifiers to classify our dataset into various categories. There are three stages of our machine learning model:

(i) **Phase I:**Feature normalization is done in Phase I by rescaling one or more attributes to have a mean of 0 and a standard deviation of 1.

(ii) **Phase II:**Machine Learning classifiers such as Naive Bayes, Logistic Regression, SVM, Decision Tree, Random Forest, and GBDT / XGBoost are utilized in Phase II for training purposes.

(iii) **Phase III:**We used Clustering features, PCA transformed features, and Feature engineering employing existing features in Phase III to build some extra features to our dataset.

2.1.1 Feature Normalization

Feature Normalization is a scaling technique in which values of features are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling. Here's the formula for normalization:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.1}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

1. When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0.
2. On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1.
3. If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1.

2.1.2 Classification Algorithms

2.1.2.1 Naive Bayes

It's a classification approach that uses probabilistic data. The classification is based on Bayes' theorem and the premise of conditional independence of attributes, which means that the existence of one feature does not affect the presence of others. The Bayes' theorem is expressed in the following formula:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)} \tag{2.2}$$

where:

(i) P(A/B)=Probability of happening of A given that B has occurred.

(ii) P(B/A)=Probability of happening of B given that A has occurred.

(iii) P(A)= Probability of happening of A.

(iv) P(B)= Probability of happening of B.

2.1.2.2 Logistic Regression

Logistic regression models the probability of the default class. Logistic regression is named for the function used at the core of the method, the logistic function. Logistic

function is also known as sigmoid function which is given as:

$$Y = \frac{1}{1 + e^{-z}} \tag{2.3}$$

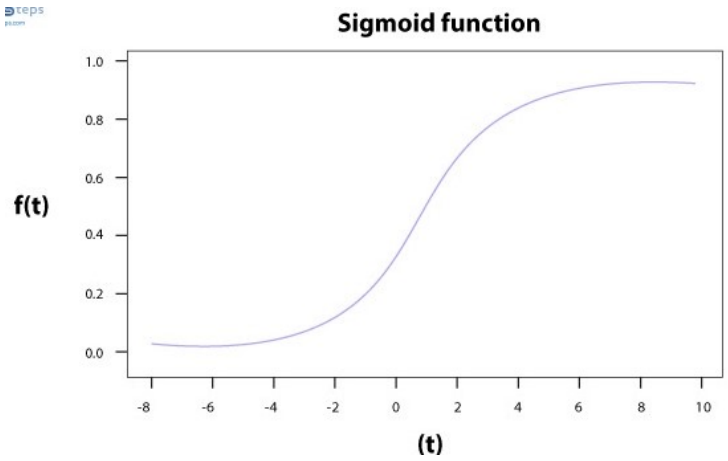


Figure 2.1: Sigmoid Function.

2.1.2.3 Support Vector Machine(SVM)

For two-group classification issues, a support vector machine (SVM) is a supervised machine learning model that uses classification techniques. The hyperplane (which in two dimensions is simply a line) that best separates the tags is produced by a support vector machine using these data points.

2.1.2.4 Decision Tree

A Decision Tree Classifier is a non-linear ML classifier that makes judgments and classifies points into distinct categories using numerous lines/planes/hyperplanes, similar to an if-else statement.

2.1.2.5 Random Forest

It pertains to the ensemble classification approach. A decision tree is the basic unit of a random forest. The classification rules are used to create a decision tree, which is represented as a flowchart. It is constructed in such a way that the internal nodes represent a feature test, the branches represent the test result, and the leaf nodes represent class labels. Using the training data, Random Forest creates many decision trees and then calculates the mode of the output produced or the class predicted by the trees.

2.1.2.6 GBDT / XGBoost

The main advantage of XGB over gradient boosting machines is that it has several hyper-parameters that can be tweaked. Missing values are handled automatically by XGBoost. Parallelization, distributed computing, cache optimization, and other intuitive features are included.

2.2 IMPLEMENTATION/EXECUTION OF PROJECT

After studying the classification algorithms, we have implemented them in Python3 and applying Clustering features, PCA transformed features, and Feature engineering to improve the time complexity and accuracy of the algorithms.

2.2.1 Dataset

We have used KDD Cup 99 collected from Kaggle in our study.

2.2.2 Identifying the features

First Task is to identify the features that are used in the various classification models. For knowing the dataset, we have plotted the graph between duraion and features.

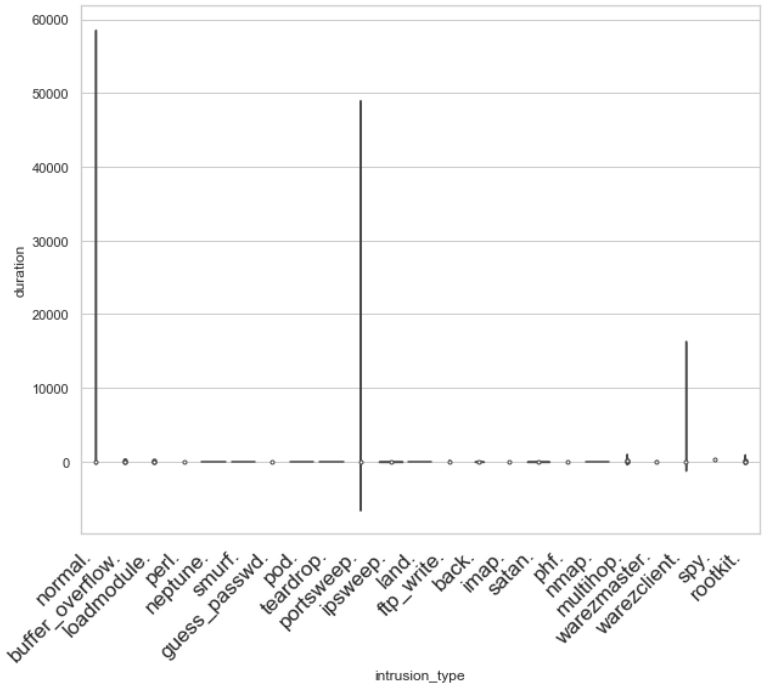


Figure 2.2: Features vs duration.

2.2.3 Dependency of intrusion type on different features

To know the dependency of different features, we have plotted the graphs between different features and intrusion types using the pairplot function.

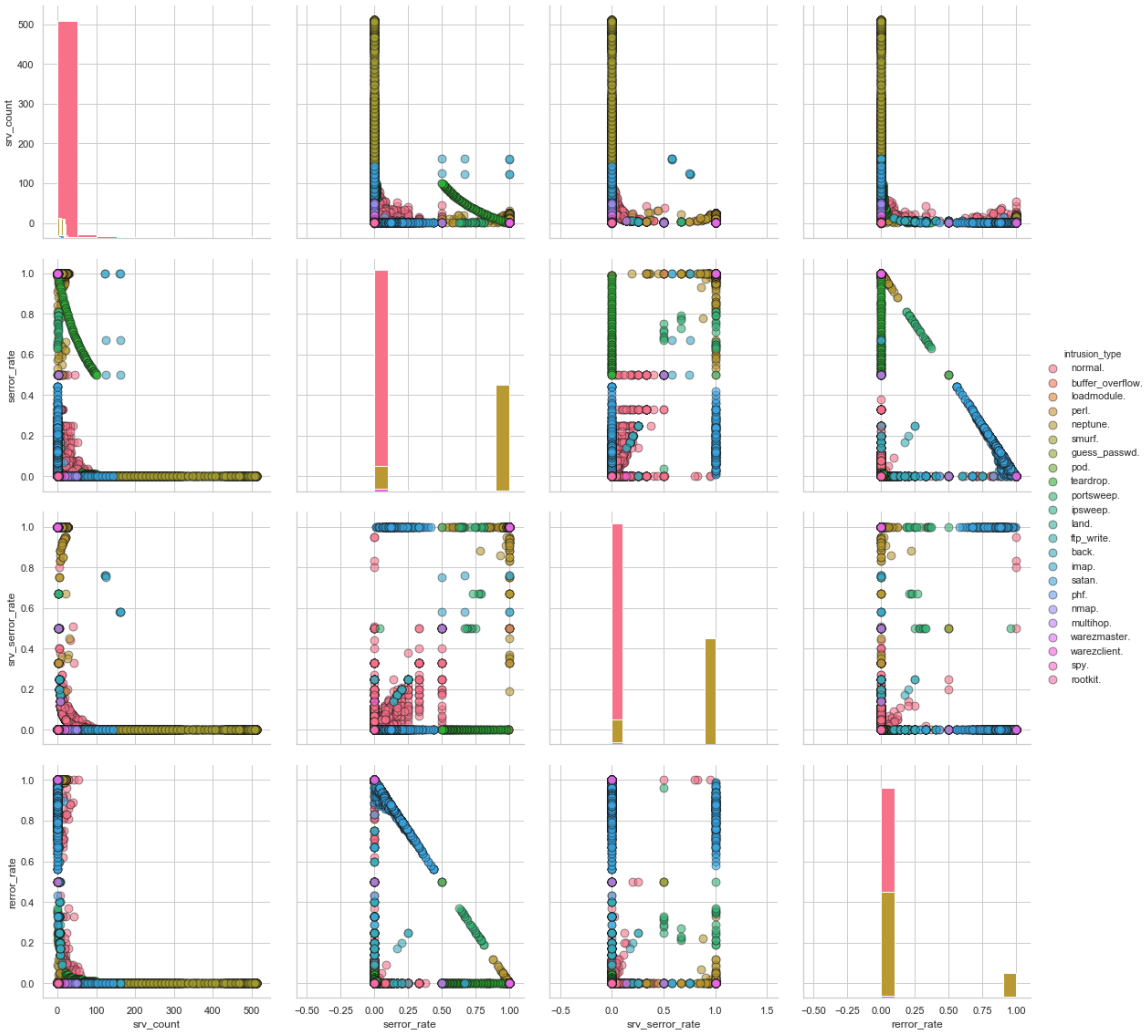


Figure 2.3: Features vs intrusion type

2.2.4 Exploratory Data Analysis(EDA)

There are 4,94,021 data points and 42 characteristics in the dataset. After the dataset has been imported, it is cleaned to remove/impute NULL values and duplicates. After that, we used Exploratory Data Analysis (EDA) to discover what the data could tell us in addition to the formal modelling. Python packages such as matplotlib, pandas, and seaborn are used in EDA.
Observations from EDA:

(i) Data points as “normal” (good connections):60.33%

(ii) Bad connections, class “Neptune” : 35.594 % and “back”: 0.665 %

(iii) Classes “rootkit.”, “load_module.”, “ftp_write.”, “multi-hop.”, “phf.”, “Perl.” and “spy.” have the least no. of data points.

The performances of the algorithms were measured using Confusion Matrix, precision, recall, and weighted f1-score.

2.2.4.1 Confusion Matrix

It's a table that shows the classifier's performance on real-world data. The number of False Positives, True Positives, False Negatives, and True Negatives acquired after classification is shown in the confusion matrix.

	Predicted Class (Negative)	Predicted Class (Positive)
Actual Class (Negative)	True Negative(TN)	False Postive(FP)
Actual Class (Positive)	False Negative(FN)	True Positive(TN)

Table 2.1: Confusion Matrix

Using them, we calculate three quantities to determine the performance of the classifier:

(i) **Sensitivity (True Positive Rate):** It is the ratio of the number of correctly classified intrusions to the total number of intrusions in the dataset.

(ii) **Specificity (True Negative Rate):** It is the ratio of the number of true negatives to the total number of actual negatives.

(iii) **Accuracy:** It is the ratio of the number of correctly classified files to the total number of files in the dataset.

In our models, we employed TPR for sensitivity and TNR for specificity. Vectorizing categorical data using One-hot encoding into service, flag, and protocol was the next stage. The purpose of data standardisation was to rescale one or more attributes to have a mean of 0 and a standard deviation of 1.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 RESULTS

Intrusion detection was done using supervised machine learning algorithms like naive bayes, logistic regression, support vector machine, decision tree, random forest, XG Boost which resulted in the accuracy given in the table.

Model	Train f1 score	Train TDR	Train FPR	Test f1-score	Test TPR	Test FPR
Naive Bayes	0.9671	99.40%	5.13%	0.9679	99.34%	4.91%
Logistic Regression	0.9813	99.81%	2.95%	0.9819	99.81%	2.76%
SVM	0.9967	99.87%	0.48%	0.9966	99.87%	0.43%
Decision Tree - 1	0.9997	99.96%	0.0%	0.9986	99.90%	0.13%
Random Forest - 1	0.9999	99.98%	0.0%	0.9992	99.98%	0.13%
XG Boost - 1	0.9999	100.0%	0.0%	0.9994	99.98%	0.083%

Table 3.1: Conclusion after running the given models

We will employ DT, RF, and XGBoost classifiers ahead of the existing and feature engineered data because they had the best performance.

Feature engineering : Clustering features (using MiniBatchKmeans), PCA features, and constructing new features from the data (such as adding two current features and deleting two existing features) are used.

Conclusion after applying feature engineering is given in the table.

Model	Train f1 score	Train TDR	Train FPR	Test f1-score	Test TPR	Test FPR
Naive Bayes	0.9671	99.40%	5.13%	0.9679	99.34%	4.91%
Logistic Regression	0.9813	99.81%	2.95%	0.9819	99.81%	2.76%
SVM	0.9967	99.87%	0.48%	0.9966	99.87%	0.43%
Decision Tree - 1	0.9997	99.96%	0.0%	0.9986	99.90%	0.13%
Random Forest - 1	0.9999	99.98%	0.0%	0.9992	99.98%	0.13%
XG Boost - 1	0.9999	100.0%	0.0%	0.9994	99.98%	0.083%
Decision Tree - 2	0.9998	99.97%	0.0%	0.9992	99.98%	0.83%
Random Forest - 2	0.9999	99.99%	0.0%	0.9990	99.99%	0.15%
XG Boost - 2	0.9999	99.99%	0.0%	0.9994	99.98%	0.083%

Table 3.2: Conclusion after running the given models post feature engineering

3.1.1 Conclusion

We have come up with the techniques where we used the KDD Cup 99 dataset and applied different ML techniques to build a Network Intrusion Detection System that is able to classify between Good and Bad connections with good precision while reducing the number of False Positives. There is also variation in the time complexity of the algorithms which is faced after using our techniques. Time complexities before and after the feature engineering is shown in the given figures:

```
Fitting the model and prediction on train data:
Fitting 3 folds for each of 25 candidates, totalling 75 fits

C:\Users\91969\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:657: Warning: The least populated class in y has o
nly 2 members, which is too few. The minimum number of members in any class cannot be less than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed: 142.5min
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed: 204.7min finished
C:\Users\91969\Anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=F
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[16:07:36] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
Completed
Time taken: 3:27:02.333159
================================================
{'tp': 65873, 'tpr': 0.9999848195039014, 'fp': 0, 'fpr': 0.0}
Prediction on test data:
{'tp': 21954, 'tpr': 0.9998178340468167, 'fp': 15, 'fpr': 0.0010388531061707874}
Completed
Time taken: 0:00:00.160556
```

Figure 3.1: Time Taken by xgb classifier before applying new techniques

```
Fitting the model and prediction on train data:
Fitting 3 folds for each of 10 candidates, totalling 30 fits

C:\Users\91969\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:657: Warning: The least populated class in y has o
nly 2 members, which is too few. The minimum number of members in any class cannot be less than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 47.4min finished
C:\Users\91969\Anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=F
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[13:16:00] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
Completed
Time taken: 0:52:39.313878
==================================================
{'tp': 65874, 'tpr': 1.0, 'fp': 0, 'fpr': 0.0}
Prediction on test data:
{'tp': 21950, 'tpr': 0.9996356680936332, 'fp': 16, 'fpr': 0.0011081099799155065}
Completed
Time taken: 0:00:00.225398
==================================================
```

Figure 3.2: Time Taken by xgb classifier after applying new techniques

From the figures, it can be clearly noted that before applying the new techniques, the xgb classifier takes more than 3 hours which takes only 52mins after applying the techniques.

CHAPTER 4

Tasks to be completed

4.1 Future Work

(a) Analyzing Unsupervised machine learning techniques like ensemble learning, Autoencoder Ng (2011),the Multilayer Perceptron (MLP), the Long Short-Term Memory network (LSTM) algorithms) M. Labonne and Olivereau (2020), and Generative Adversarial Networks(GAN).

(b) Applying new techniques to improve the timw complexity and accuracy of the above mentioned models.

(c) To check the performance of supervised and unsupervised Machine Learning techniques in detecting intrusions by comparing their sensitivity, specificity, and accuracy. (Applying detection rate, Accuracy, Precision, F1-score and ROC Curve) G. Gu and Skorić (2006)

4.2 Gantt Chart

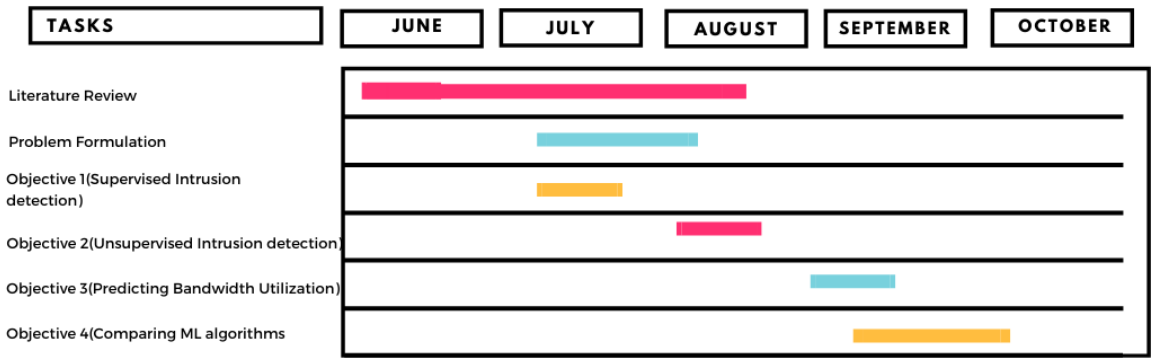


Figure 4.1: Gantt Chart.

REFERENCES

[1] A. Paszke, S. G. and Lin, Z.: 2010, Automatic differentiation in pytorch, *Computer Technology and Development (ICCTD), 2010 2nd International Conference on* pp. 374 – 378.

[2] Axelsson, S.: 2000, The base-rate fallacy and the difficulty of intrusion detection, **3**, 186–205.

[3] et.al, H. K. S.: 2013, Using the adtree for feature reduction through knowledge discovery, PHI Learning Pvt. Ltd., pp. 1010 – 1044.

[4] F. Amiri, M. Yousefi, C. L. A. S. and Yazdani, N.: 2011, Mutual information-based feature selection for intrusion detection systems, *Journal of Network and Computer Applications* **34**, 1184–1199.

[5] F. Pedregosa, G. V. and Duchesnay, E.: 2011, Scikit-learn: Ma chine learning in python, *Computer and Automation Engineering (ICCAE)*, Vol. 12, pp. 2825 – 2830.

[6] Freund, Y.: 2004, The alternating decision tree algorithm, *Wireless Communication Systems*, pp. 124– 133.

[7] G. Gu, P. Fogla, D. D. W. L. and Skorić, B.: 2006, Measuring intrusion detection capability: An information-theoretic approach, Vol. 3, p. 90–101.

[8] Horng, S.-J. and Su, M.-Y.: 2011, Novel intrusion detection system based on hierarchical clustering and support vector machines, *ELSEVIER, Expert Systems with Applications.* **38**, 306 – 313.

[9] Juan Wang, Qiren Yang, D. R.: 2009, An intrusion detection algorithm based on decision tree technology, Vol. 29, pp. 1040 – 1044.

[10] Kim, G. and Lee, S.: 2014, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, elsevier, expert systems with applications, **41**, 1690 – 1700.

[11] M. Abadi, A. A. and Zheng, X.: 2007, Ten sorflow: Large-scale machine learning on heterogeneous distributed systems, *The International Arab Journal of Information Technology* **4**, 50 – 59.

[12] M. Hall, E. F. and ten, I. H. W.: 2009, The weka data mining software: an update, *Computer Technology and Development (ICCTD), 2010 2nd International Conference on* pp. 374 – 378.

[13] M. Labonne, C. C. and Olivereau, A.: 2020, Multicasting in ad hoc networks using NTP protocol, *Computer communication International Council for Computer Communication*, pp. 144 – 160.

[14] Ng, A.: 2011, Sparse autoencoder, *CS294A Lecture notes* **8**, 1–19.

[15] Panda, M. and Patra, M. R.: 2008, A comparative study of data mining algorithms for network intrusion detection, Vol. 11, pp. 504 – 507.

[16] Tavallaee M, Bagheri E, L. W. G. A.: 2009, A detailed analysis of the kdd cup 99 data set, *INTERNATIONAL JOURNAL OF COMPUTATIONAL COGNITION* **8**, 1– 6.