STUDENTS PROJECT REPORT COVERAGE[V1.1]

The following sequence should be followed and maintained.

1. **Cover page, along with the title of the project and name of the candidate.**
2. **Certificate obtained from industry (in case of the external project)**
3. **Certificate obtained from guides (in case of an internal project)**
4. **Acknowledgment**
5. **List of symbols, nomenclature, and abbreviations used**
6. **List of Figures and Graphs**
7. **List of Tables**
8. **Abstract [PURPOSE-METHODOLOGY-FINDINGS]**
9. **Tables of Content**
10. **Chapters organization**

    Chapter-1: Project description and outline

    Chapter-2: Related work investigation

    Chapter-3: Requirement Artifacts

    Chapter-4: Design methodology and its novelty

    Chapter-5: Technical Implementations and Analysis

    Chapter-6: Project Outcome and Applicability

    Chapter-7: Conclusions and Recommendation

    References

    Appendices (Additional Information if necessary).

[Report should be in A4 size paper with flexible cover]

Anomaly Detection System based on Computing Intelligence

A PROJECT REPORT

Submitted by

Aakash Vats (18BCY10002)
Adil Mustafa Khokhawala (18BCY10007)
Devyani Senwar (18BCY10031)
Het Bhavin Patel (18BCY10040)

in partial fulfillment for the award of the degree of

BACHLEORS OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION IN CYBER SECURITY AND DIGITAL FORENSICS

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY
KOTHRIKALAN, SEHORE
MADHYA PRADESH - 466114

APRIL & 2022

BONAFIDE CERTIFICATE

Certified that this project report titled **"Anomaly Detection System based on Computing Intelligence"** is the bonafide work of **"Aakash Vats (18BCY10002), Adil Mustafa Khokhawala (18BCY10007), Devyani Senwar (18BCY10031), Het Bhavin Patel (18BCY10040)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Rakesh R, Asst. Professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Thangavel M., Asst. Professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on **30th April 2022**

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. Pushpindar Singh Patheja, Head of the Department, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Mr. Thangavel M, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

KNN	K Nearest Neighbor
SVM	Support Vector Machine
NB	Naïve Bayes
DT	Decision Tree
ML	Machine Learning
LSTM	Long Short-Term Memory
AM	Attention Mechanism
DL	Deep Learning
MSD	Mahalanobis-Squared Distance
BM	Block Maxima
IDS	Intrusion Detection System
GA	Genetic Algorithm
ASNM	Advance Security Network Metrics
ROC	Receiver Operating Characteristic
ANN	Artificial Neural Network
ROC	Receiver Operating Characteristic Curve

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

ABSTRACT

Our research is built on machine learning and artificial intelligence technologies, namely stacking models. Stacking is an algorithm that takes the outputs of sub-models as input and tries to figure out how to combine the input predictions in the best way possible to get a superior output prediction.

Detecting and diagnosing the root cause of network traffic log problems is a time-consuming and labour-intensive process, especially for previously unknown failure modes. However, in the context of troubleshooting anomalies in network traffic logs, our project is based on a stacking method to identify malicious logs from the Advanced Security Network Metrics (ASNM) datasets.

According to the input training data, there have been roughly three orthogonal techniques to creating intrusion detectors: (1) Knowledge-based detection, which models and matches the characteristics of harmful intrusions. (2) Detection based on anomalies, which models typical behaviour and finds departures from it. (3) classification-based detection, which simultaneously models harmful and legitimate behaviour. The problems with these approaches are it showed a high false negative rate in case of evasions by unknown or zero-day attacks, long time required for training and profiling and susceptibility.

To overcome all of the flaws, our project is entirely built on a stacking model, in which we took four machine learning algorithms and employed one of them at a time at level 1 and the other at level 0 for a higher testing accuracy rate. K-Nearest Neighbor, Nave Bayes, Support Vector Machine, and Decision Tree are the four methods employed.

The performance of these methods is quite comparable, with nave bayes being the most effective when retained at level 1, followed by support vector machine, Decision Tree, and K-Nearest Neighbor when kept at level 0.

TABLE OF CONTENTS (SPECIMEN)

RELATED WORK INVESTIGATION

Peng lin et al (2019) The author has used the LSTM approach in the above-mentioned article. It is a particular recurrent neural network structure which helps in handling the problem of long-term reliance. Attention Mechanism has been used. The Attention System is a simulation of the human brain's attention mechanism, AM's role in anomaly detection is to calculate the impacts of each network traffic on the previous network traffic. The above-mentioned algorithm has a 93 percent accuracy.

Junhong Kim et al (2019) Insider threats are generally the security issues that emerge from people who have access to a company's network, systems, and data. Insider threats are less often than external incursions, but they are more dangerous. The correctness of the paper is 53.77 percent. After considering all of the aforementioned information (all log data), the author devised an algorithm for detecting insider risks in a company.

HassanSarmadi et al(2020) Unsupervised anomaly detection using mahalanobis-squared distance (MSD) which is a well-known method. The primary purpose of this newsletter is to promote AMSD-KNN. The key idea behind the suggested method is to find adequate local friends of education and inspect datasets in a -degree method for avoiding environmental variability situations and estimating nearby covariance matrices. The block maxima (BM) approach is used to determine a reasonable threshold limit using specialized excessive cost distribution modeling.

Bingming Wang et al (2020) In this paper the author approaches Log-based Anomaly Detection Using K-Nearest Neighbor Logs play a critical role in the safety of large-scale systems in this research article. Detecting anomalies from logs manually using the K-Nearest Neighbor (KNN) set of rules, an outlier detection technique with high accuracy, is an effective way to find abnormalities. As a result, we suggest a more advanced KNN set of rules-primarily based strategy that takes use of the current mean-shift clustering set of rules to successfully choose the educational set from large logs.

Bo Long et al (2020) The goal of this research study's author is to give a full grasp of entirely anomaly detection techniques based on deep learning in a variety of software applications. The book then examines today's deep mastery trends in depth, as well as the strategies used to overcome the restrictions imposed by traditional algorithms. The research concludes with a consideration of future trends. Then we realize how important it is to add modern deep anomaly detection algorithms. We discuss the following tasks in deep version anomaly detection strategies: Using RNN, LSTM, Auto-Encoder, and other approaches, learn regular patterns from complicated data. Detecting anomalies, in which we look at how to correctly detect anomalous behavior using just reconstruction errors, reconstruction probabilities, and the use of a single class, KNN.

Harsh H. Patel et al (2018) Machine learning entails researching devices based on a variety of educational backgrounds, experimenting with statistics, and deciding the outcomes in each circumstance without the use of pre-programmed algorithms. Decision Tree algorithms were employed in a variety of industries and for a variety of applications. It may be quite important for us to know how to apply a large range of rules in any case when we must make a decision. ID3, C4.five, and CART are three different Decision Tree methods presented in this study. The dataset was subjected to the Decision Tree algorithms ID3 C4.five and CART. In terms of effectiveness, time, and accuracy, the decision tree exceeds the competition. It's built on a set of guidelines for locating interesting resources. Finally, the study of choice tree algorithms is concluded, and this research says that CART is a set of rules for this dataset that may be extremely particular and correct in comparison to others.

Shi Ying et al (2021) The author of this study concludes that outliers are the logs that contain uncommon log states (anomaly logs), and that the k-Nearest Neighbor (KNN). As a result, author has employed the KNN set of rules to find abnormalities within the log data. A green log anomaly detection solution based entirely on a stepped forward KNN set of rules with a mechanically classified pattern set to solve those three issues. This method presents a log parsing method based only on N-grams and a common sample mining (FPM) method that decreases the measurement of the log vector modified with Frequency distribution. Inverse Document Frequency (TF-IDF) is a technique that uses inverse document frequency. Then, using clustering and self-schooling, we automatically extract classified log information patterns from old logs.

Azam Davahli et al (2020) Anomaly or intrusion detection system (IDS) is an effective defense mechanism for stressed networks. However, modern technology with high computational complexity is mistaken for aid-restricted WSNs in IoT, and they also fail to identify new WSN assaults. This study offers a novel version of the GAB GWO to improve a support vector machine (SVM)-based completely light-weight IDS (LIDS) by combining concepts from genetic algorithms (GA) with mathematical equations from the gray wolf optimizer (GWO). The hybrid set of rules has been compared to natural GA and GWO, as well as other recent approaches, and it has been established that its overall performance is superior.

Harsh Patel et al (2018) The methods may be used to extract text, clinically licensed fields, and search engines, as well as identify statistics in a number of statistical activities. On the basis of their accuracy and efficiency rate, many decision tree algorithms were constructed. It may be critical for us to understand how to employ a good set of guidelines in any decision-making situation. In terms of accuracy, time, and precision, the decision tree exceeds the competition. It is based on a set of rules that are used to provide recommendations for finding interesting sites. Finally, a thorough examination of choice tree algorithms is completed, and this study finds that CART is a set of rules for this dataset that is more specific and accurate than many others.

Hossein Saeedi Emadi et al (2018) Here the author discusses the fact that such networks cannot be supervised, and hence this research addresses the problem of anomaly detection. Here the community traffic is used to extract the three functions of temperature, humidity, and the voltage. The density-primarily depends upon fully spatial clustering of packages with noise (DBSCAN) collection of guidelines is then used to cluster community information. It also uses density-based total detection techniques to assess the correctness of the DBSCAN set of criteria for entering information. This collection of criteria identifies variables in low-density areas as anomalous. It trains to assist vector machines by using daily information. Finally, it eliminates outliers from community data. Using coefficient correlation, we should be able to solve DBSCAN's problem of deciding on entry parameters in this study.

Kun Xie et al (2018) The author of this paper highlights how traffic anomaly detection is important for better Internet administration. Traditional detection algorithms frequently transform excessively dimensional recordings to a prolonged vector, which reduces detection accuracy due to a lack of spatial analysis in the records. Furthermore, they're frequently created based wholly on the segregation of regular and abnormal records in a time period, which not only adds extra trash and computing price, but also inhibits timely identification of anomalies. It is critical to discover online and fix website navigation anomalies, but it is difficult to do so. To deal with the problem, this research creates a 2-D matrix from the monitoring records in each time slot and uses bilateral big thing analysis to discover anomalies inside the new time frame (B-PCA).

Chapter 1: Project Description and Outline:

1.1 Introduction:

Cyberattacks and breaches are nearly daily occurrences these days. The rising frequency of cyberattacks is also due to technological advancements and information shared with hackers on which tools to employ and which techniques are successful on specific networks. In these situations, businesses should prepare themselves so that they can endure a breach or cyberattack without jeopardising data or vital services. Anomaly detection may help companies enhance their networks. Anomaly detection is the process of identifying or recognising any abnormal behaviour in the network and then deleting it to make your network resistant to assaults caused by that anomaly. What we've come up with is a hybrid model in which one method is employed at Tier 1 and another is used at Tier 0, also known as Stacking. Stacking is an extended form of the machine learning technique that may be expressed as an aggregate. Stacking Machine learning gives you the benefit of mixing the meta-learning algorithm with the training of your dataset, allowing you to anticipate numerous machine learning algorithms and models.

Stacking allows you to take advantage of the capabilities of a number of well-known regression and classification models.

When it comes to stacking, it is classified into 4 different parts:

- *Generalization:* A generalization is a composition of numerous machine learning models performed on a similar dataset.
- *Scikit-Learn API:* This is among the most popular libraries and contains tools for machine learning and statistical modelling.
- Classification of stacking.
- Regression of stacking.

The basic technique of stacking in machine learning:

- Divide the training data into 2 disjoint sets.
- The level to which you train data depends on the base learner.
- Test base learner and make a prediction.
- Collect correct responses from the output.

Moreover, in our project along with the stacking machine learning model we have used four machine learning algorithms – K-Nearest Neighbor, Support Vector Machine, Naïve Bayes and Decision tree. There are a total 12 different results when we apply permutation and combination on these algorithms. All in all, among these any three machine learning algorithms are being used at level 0 of the stacking model and with their output as predictions the remaining one is used to test the dataset based on the predictions obtained from the level 0 algorithms.

1.2 Motivation for the work:

1. Face Recognition: Face Recognition has become one of the most common areas under pattern recognition for identification and verification of an image based on its saved history. It plays a great role currently in the expanses regarding mobile devices security.

2. Remote Sensing: Remote sensing obtains data from objects and occurrences without actually having to make any physical contact.

- **Land Cover Mapping:** It happens to be an application of earth observatory satellite sensors using the data that is acquired from remote sensing as well as geospatial data. This is done to identify the objects and materials in the required target zones.
- **Change Detection:** It deals with the problem related to analysis of images, these images mostly are of the ones that have had changes in the cover or surface over time. One of the main uses for change detection is to be able to assess and monitor disasters for the safety of the civilians.

3. Medical Field: the process of stacking has been successfully implemented in multiple sectors of medicine such as diagnosis for neuro cognitive disorders such as Alzheimer's or myotonic dystrophy which can be obtained from MRI datasets.

4. Fraud Detection: It mainly deals with discovery of money laundering, bank and credit card fraud, which also comprises telecommunication fraud. They have enormous domains in research and application of machine learning. Hence stacking helps in creating a model which improves the robustness of the behaviour of the model and detects fraud in the banking systems.

5. Malware Detection: Computer viruses, worms, trojans, etc. are classifications of malware codes with respect to machine learning, document categorization problem arises from this and hence blended ensemble method can provide proper efficiency in the area.

1.3 Techniques/Model used in the project:

In our project we have total of five supervised machine learning model which have can be used to solve both classification and regression problems and their working is described below:

1. Stacking Model: Model stacking is a way to improve model prediction by combining the outputs of multiple models and running them in another machine learning model called Meta-learner.

2. K-Nearest Neighbor (KNN): The KNN algorithm assumes that similar things exist in close proximity. The steps are:

a) Load the data and initialize K to your chosen number of neighbors.
b) For each example in the data, calculate the distance between the query example and the current example from the data and add the distance and the index of the example to an ordered collection.

3. Decision tree (DT): Sort the ordered collection of distances and indices from the smallest to largest by the distances.

4. Support Vector Machine (SVM): Pick the first K entries from the sorted collection and get the labels of the selected K entries.

5. Naive Bayes (NB): If the dataset is regression or classification return and mean or mode of the K Labels respectively.

1.4 Problem Statement:

Anomalies create traffic in a network or cause breach in security and there is a loss of sensitive data. Any change or anomaly in a network is a potential threat to its functionality, hence may result in security errors and hinder the smooth functioning of an organization or network. The detection for anomaly is done in data sets, logs, company records and other data which are important or crucial results or outcomes. If in a network, during monitoring something unusual and unexpected is detected which is not normal it can be called an anomaly, thus it indicates there is something unusual which needs our attention.

Hence, the common methods for anomaly detection are not that efficient as they do not contain a very high accuracy percentage for detecting the outliers in their network/datasets.

1.5 Objective of Work:

We made several objectives in this work, which are summarized as follows:

- The main goal of anomaly detection analysis is to identify the observations that do not adhere to general patterns considered as normal behaviour.
- Anomaly detection methods could be used in branch applications, e.g., data cleaning from the noise data points and observations mistakes. On the other hand, anomaly detection methods could be helpful in business applications such as intrusion detection or credit card fraud detection systems.
- It is critical for network admins to be able to identify and react to changing operational conditions. Any nuances in the operational conditions of data centres or cloud applications can signal unacceptable levels of business risk.

- An evidence-based, well-constructed behavioural model can not only represent data behaviour, but also helps users identify outliers and engage in meaningful predictive analysis. Static alerts and thresholds are not enough, because of the overwhelming scale of the operational parameters and because it's too easy to miss anomalies in false positives and negatives.

1.6 Organization of the project:

The rest of the paper is organized as follows. In chapter 2, we investigate the recent papers on our related work, in chapter 3 we tell you about the dataset, hardware and software components, followed by in chapter 4 we tell you about the methodology about our research work, subsequently, in chapter 5 we define about the analysis and technical part and in chapter 6 we tell you about the applications about out related work. At last, in chapter 7 we conclude our work.

1.7 Summary:

We summarize our chapter one, starting with giving you a small glance on the technology or our concept of this project, moving forward our interest grew in machine learning which motivates us to complete this project. Also, it discusses the algorithms that have been used and the problem faced by us doing this wonderful project.

Chapter 2: Related work

2.1 Introduction:

In this chapter we will discuss the recent progress that has been made so far in the field of anomaly detection based on AI (Artificial intelligence), Stacking and Machine Learning (ML). Under this chapter we have taken into consideration papers which are written from 2018 to 2022, For every paper there will be a summary about what approach the author have followed to tackle the problem with that we will also discuss the accuracy of the algorithm and the results which the author obtained, all the paper which are to be discussed in the upcoming paragraphs are to be judged or summarized on the basis of how the author have perceived the problem, how is the algorithms behaving in practical , Efficiency and false alarm rate (i.e. How many times the algorithms falsely notifies the organization about the anomaly). In the Below section you would see that different algorithms have been used by different authors and with the change of algorithm accuracy too is affected in one way or another, we have taken a logical approach towards all of the papers below and we have taken the papers from journals only and the literature survey papers have not been included.

2.2 Background of this project:

In the paper as discussed above different authors have taken different approaches towards the problem i.e., anomaly detection and have come up with a solution i.e., the final algorithms all the approaches have different pros and cons and all the points have been described in the short summary above, the different technologies that have been used by different authors are: SVM, KNN, DBSCAN, Stacking.

2.3 Existing Approaches:

Paper 1: Dynamic Network Anomaly Detection System by Using Deep Learning Techniques.

The author employed the LSTM (Long Short-Term Memory) strategy in the above-mentioned article. About LSTM = LSTM is a particular recurrent neural network structure suggested to handle the problem of long-term reliance. The forget gate instructs the neural network to forget the worthless information, the input gate instructs the neural network to add new content, and the output gate determines the current node's ultimate output.

In addition, he has made use of the Attention Mechanism. The Attention System (AM deep learning) is a simulation of the human brain's attention mechanism. When we read a piece of text, we usually focus on a few keywords so that we can quickly summarise the main content of the text; similarly, AM's role in anomaly detection is to calculate the impacts of each network traffic on the previous network traffic. The above-mentioned algorithm has a 93 percent accuracy. The author has employed a loss function to fine tune the algorithm's efficiency, which is one of the benefits of utilising the aforesaid technique (in the given function we calculate the loss).

Paper 2: Insider Threat Detection Based on User Behaviour modelling and Anomaly Detection Algorithms

Insider threats are security issues that emerge from people who have access to a company's network, systems, and data, such as workers and trusted partners. Insider threats are less often than external incursions, but the scale of harm is larger. The correctness of the suggested paper is 53.77 percent.

Assumptions & data used:

Individual user activity records that have been logged in the corporate system are gathered. Then, by describing individual actions, potential traits are extracted. If the system logs collect details on when a user connects his or her personal USB drive to the system, for example, the overall number of USB connections per day can be retrieved as an applicant variable, and subscriber contents, such as the body of an e-mail, can also be used to create candidate features.

-Proposed method:

After considering all of the aforementioned information, the author devised an algorithm for detecting insider risks in a company.

Paper 3: A novel anomaly detection method based on adaptive MahalaNobis-squared distance and one-class KNN rule for structural health monitoring under environmental effects

Unsupervised anomaly detection using Mahalanobis-squared distance (MSD) is a well-known method. Despite the MSD-based totally anomaly detection technique's widespread acceptance and wide applicability, a few key challenges and obstacles, such as climatic variables, commitment of a beyond the point threshold limit, prediction of an incorrect covariance matrix, and non-Gaussian of schooling numbers, can result in false alarms and inaccurate harm detection impacts. The primary purpose of this newsletter is to promote AMSD-KNN, an anomaly detection approach based entirely on adaptive Mahalanobis-squared distance and one-elegance KNN rule for SHM under diverse environmental conditions. The key idea behind the suggested method is to find adequate local friends of education and inspecting datasets in a -degree method for avoiding environmental variability situations and estimating nearby covariance matrices. To find enough nearest mates that ensure the estimate of well-conditioned neighbouring covariance matrices, a strong approach based entirely on a multivariate normality assumption check is provided. The suggested AMSD-KNN strategy is unique in that it uses a new multivariate distance degree and one-elegance KNN rule to build a unique unsupervised learning method for SHM. The block maxima (BM) approach is used to determine a reasonable threshold limit using specialized excessive cost distribution modelling. Due to the importance of picking excellent blocks inside the BM approach, a goodness-of-in-shape grade is calculated using the Kolmogorov-Smirnov hypothesis test to determine an optimal block number.

The suggested methodologies' overall performance and efficacy are determined using well-known benchmark setups. A number of comparison studies are also carried out to demonstrate the superiority of the suggested procedures over a number of ultra-modern techniques.

The suggested AMSD-KNN and BM approaches excel in detecting harm in low-variability environments, according to the results.

PAPER 4: Log-Based Anomaly Detection with the Improved K-Nearest Neighbor

Log-based Anomaly Detection Using K-Nearest Neighbor Logs play a critical role in the safety of large-scale systems in this research article. The large range of logs that imply daily (everyday logs) differs greatly from the wide range of logs that suggest anomalies (bizarre logs), and the two types of logs have distinct characteristics. Detecting anomalies from logs manually using the K-Nearest Neighbor (KNN) set of rules, an outlier detection technique with high accuracy, is an effective way to find abnormalities. However, logs have the characteristics of a large scale and extremely choppy samples, which can influence the results of the KNN set of rules on log-based completely anomaly detection. As a result, we suggest a more advanced KNN set of rules-primarily based strategy that takes use of the current mean-shift clustering set of rules to successfully choose the educational set from large logs. Then, for samples with unique distances, we assign unique weights, which lessens the disastrous impact of unbalanced log statistical distribution at the efficiency of the KNN set of regulations. The results of experiments on log units from five supercomputers show that the approach we proposed can be efficiently applied to log-primarily based completely anomaly detection, and that the exactness, account fee, and F degree with our methodology are superior to those of the conventional key-word seek approach.

Paper 5:Deep Learning for Anomaly Detection

The goal of this research study's author is to give a full grasp of entirely anomaly detection techniques based on deep learning in a variety of software applications. To begin, it outlines the paradox detection problem, the tactics employed prior to the production of deep versions, and the difficult cases encountered. The book then examines today's deep mastery trends in depth, as well as the strategies used to overcome the restrictions imposed by traditional algorithms. It comes in second to last, looking into deep version anomaly detection techniques in real-world samples from LinkedIn production systems. The research concludes with a consideration of future trends. Then we realise how important it is to add modern deep anomaly detection algorithms. We discuss the following tasks in deep version anomaly detection strategies:

1) Using RNN, LSTM, Auto-Encoder, and other approaches, learn regular patterns from complicated data.

2) Detecting anomalies, in which we look at how to correctly detect anomalous behaviour using just reconstruction errors, reconstruction probabilities, and the use of a single class, KNN.

Paper 6: Study and Analysis of Decision Tree Based Classification Algorithms

Machine learning entails researching devices based on a variety of educational backgrounds, experimenting with statistics, and deciding the outcomes in each circumstance without the use of pre-programmed algorithms. Decision Tree is one of the device research methodologies. Decision Tree algorithms were employed in a variety of industries and for a variety of applications. These algorithms can be used to find statistics in other statistical procedures, to extract text, in scientific licenced sectors, and in search engines, among other things. Different decision tree algorithms have been developed based on their correctness and performance rate. It may be quite important for us to know how to apply a large range of rules in any case when we must make a decision. ID3, C4.five, and CART are three different Decision Tree methods presented in this study.

The dataset was subjected to the Decision Tree algorithms ID3 C4.five and CART. In terms of effectiveness, time, and accuracy, the decision tree exceeds the competition. It's built on a set of guidelines for locating interesting resources. Finally, the study of choice tree algorithms is concluded, and this research says that CART is a set of rules for this dataset that may be extremely particular and correct in comparison to others.

Paper 7: An Improved KNN-Based Efficient Log Anomaly Detection Method with Automatically Labelled Samples

The author of this study work concludes that outliers are logs that contain uncommon log states (anomaly logs), and that the k-Nearest Neighbor (KNN) set of rules has exceptionally high accuracy in outlier identification approaches. As a result, we employ the KNN set of rules to find abnormalities within the log data. However, there are a few issues with using the KNN set of rules to find anomalies, three of which are: excessive vector measurement results in inefficient KNN set of rules, unlabelled log information is useless to the KNN set of rules, and the imbalance of the range of log information distorts the type selection of the KNN set of rules. We offer a green log anomaly detection solution based entirely on a stepped forward KNN set of rules with a mechanically classified pattern set to solve those three issues. This method presents a log parsing method based only on N-grams and a common sample mining (FPM) method that decreases the measurement of the log vector modified with Frequency distribution. Inverse Document Frequency (TF-IDF) is a technique that uses inverse document frequency. Then, using clustering and self-schooling, we automatically extract classified log information patterns from old logs.
For odd logs with tiny amounts and long distances from conventional logs, we employ a clustering and self-training technique to provide classified log statistics pattern set on a regular basis. Finally, we apply common weighting distance to improve accuracy of the KNN algorithm, reducing the detrimental effects of log pattern imbalance. The results show that our approach can improve the effectiveness of log-based totally aberration detection with the KNN algorithm while ensuring accuracy at the same time, based on experiments on log units generated through six datasets of various types and comparisons with three different log-based totally anomaly detection methods.

Paper 8: A Lightweight Anomaly Detection Model using SVM for WSNs in IoT, through a Hybrid Feature Selection Algorithm based on GA and GWO

Anomaly or intrusion detection system (IDS) is an effective defence mechanism for stressed networks. However, modern technology with high computational complexity is mistaken for aid-restricted WSNs in IoT, and they also fail to identify new WSN assaults. Managing the large number of incursion wi-fi site visitors obtained by sensors, imposing a gradual detecting procedure, improved assistance use, and inaccurate detection As a result, considering WSN hurdles for developing an IDS in the IoT is a significant challenge for security researchers. This study offers a novel version of the GAB GWO to improve a support vector machine (SVM)-based completely light-weight IDS (LIDS) by combining concepts from genetic algorithms (GA) with mathematical equations from the grey wolf optimizer (GWO). The GABGWO uses novel crossover and mutation operators to try to find the most useful site visitor functions and eliminate the useless ones, in order to improve the LIDS' overall performance. The overall performance of LIDS is assessed using the AWID real-world wi-fi dataset in scenarios with and without the use of GAB GWO. The results supported the suggested GAB GWO set of rules' promising performance in choosing on the most desirable traffics, lowering computing expenses, and delivering excessive accuracies for LIDS. The hybrid set of rules has been compared to natural GA and GWO, as well as other recent approaches, and it has been established that its overall performance is superior.

Paper 9: Study and Analysis of Decision Tree Based Classification Algorithms

Deep learning is the technique of analysing devices based on a variety of academic and testing information and deciding the outcomes in different situations without being explicitly programmed. One of the device learning approaches is the Decision Tree. Decision Tree algorithms have been used in a diverse range of companies and applications. These methods may be used to extract text, clinically licenced fields, and search engines, as well as identify statistics in a number of statistical activities. On the basis of their accuracy and efficiency rate, many decision tree algorithms were constructed. It may be critical for us to understand how to employ a good set of guidelines in any decision-making situation. In terms of accuracy, time, and precision, the decision tree exceeds the competition. It is based on a set of rules that are used to provide recommendations for finding interesting sites. Finally, a thorough examination of choice tree algorithms is completed, and this study finds that CART is a set of rules for this dataset that is more specific and accurate than many others.

Paper 10: A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks

The author discusses the fact that such networks cannot be supervised, and hence this research addresses the problem of anomaly detection. First, the community traffic is used to extract the three functions of temperature, humidity, and voltage. The density-primarily based fully spatial clustering of packages with noise (DBSCAN) collection of guidelines is then used to cluster community information. It also uses density-based totally detection techniques to assess the correctness of the DBSCAN set of criteria for entering information. This collection of criteria identifies variables in low-density areas as anomalous. It trains to assist vector machines by using daily information. Finally, it eliminates outliers from community data. The suggested set of rules is examined using Intel Berkeley Research lab's normal and standard facts set (IRLB). Using coefficient correlation, we should be able to solve DBSCAN's problem of deciding on entry parameters in this study. The suggested set of rules has an advantage over previous ones in that it uses gentle computing methods, is simple to implement, and improves detection accuracy by evaluating these three functions simultaneously.

Paper 11: On-Line Anomaly Detection with High Accuracy

The author of this paper highlights how traffic anomaly detection is important for better Internet administration. Traditional detection algorithms frequently transform excessively dimensional recordings to a prolonged vector, which reduces detection accuracy due to a lack of spatial analysis in the records. Furthermore, they're frequently created based wholly on the segregation of regular and abnormal records in a time period, which not only adds extra trash and computing price, but also inhibits timely identification of anomalies. It is critical to discover online and fix website navigation anomalies, but it is difficult to do so. To deal with the problem, this research creates a 2-D matrix from the monitoring records in each time slot and uses bilateral big thing analysis to discover anomalies inside the new time frame (B-PCA). We recommend a number of novel strategies in Online B-PCA to aid quick and accurate anomaly detection in real time, including a unique B-PCA-based totally anomaly detection precept that considers the variant of each row and column major instructions for more accurate anomaly detection, an approximate set of rules to avoid using the generation process to calculate the major instructions in a close-form, and a sequential anomaly set of rules. That is, to the best of our knowledge, the first artwork to use 2-D PCA for anomaly detection. We ran massive simulations to test our Online B-PCA using state-of-the-art anomaly detection techniques and real-world site visitor strains Abilene and GANT. Our simulation results demonstrate that, when compared to other algorithms, our Online B-PCA can achieve much better overall detection performance with a low fake effective rate, a high legitimate effective rate, and an excessive caffeine calculation value.

2.4 Summary:

We examined all of the related research provided by different writers about outlier detection in this chapter, as well as their perspectives on various methods utilised to increase in performance percentages. We may also observe the operation of many machine learning modules.

Chapter-3: Requirement Artifacts

3.1 Introduction:

Here we will discuss all the hardware and software requirements of our project which helped in the building idea behind it. We gathered around 11 research papers with the same research backgrounds, data sets for our algorithm, we used a laptop with windows 10 with a RAM of 8 GB and others as mentioned in the next section.

3.2 Hardware and Software requirements:

The hardware and software requirements are:

Table for hardware and software requirement

OS	Windows 10
RAM	8GB
GPU	4GB
IDE	Visual Studio Code (Python)

The entire algorithm is implemented with the above configuration. As we can observe from the above table, the physical requirements are negligible and the proposed model can be compatible with almost any physical machine.

3.3 Specific Project requirements

3.3.1 Data requirement

- ***Importing the Dataset:*** Here we have three sub-dataset of ASNM dataset namely, ASNM-CDX-2009, ASNM-NBPOv2 and ASNM-TUN subsequently we will be importing them one by one for 12 programs each.

- ***Segmentation and Masking:*** Since our dataset is not organized, we need to make it neat and clean so for that we need to separate and label features and store them into two different arrays.
- ***Resizing:*** Here all the network traffic data which was in the form of IP address, MAC address, port number, VLAN Id is converted into binary form for better calculation and effective output.
- ***Standardization:*** All the entries of the CSV dataset were applied for standardization so that the resulting average is calculated as 0, and a unit standard deviation can be observed.
- ***Data Source:*** The publicly available data source was collected from centre of Excellence IT4Innovations, Faculty of Information Technology, Brno University of Technology, 612 00 Brno, Czech Republic, which consists of three datasets that have been built from network traffic traces using ASNM (Advanced Security Network Metrics) features. Each of this dataset consists of 5000-6000 rows and 50-60 columns.

  Our dataset comprises of three dataset and they are:

  1. ASNM-CDX 2009 dataset
  2. ASNM-NBPOv2 dataset
  3. ASNM-TUN dataset

(i.) ASNM-CDX 2009: This dataset consists of ASNM features extracted from tcp-dump capture of malicious and legitimate TCP communications on network services which are vulnerable to buffer overflow attacks and are included in CDX-2009 dataset of network traffic dumps.

The final composition of the dataset is depicted in table ASNM-CDX-2009 dataset contains two types of labels that are enumerated by increasing order of their granularity in the following:

Label_2: Is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or legitimate traffic.

· **Lebel_poly:** is composed of two parts that are eliminated be a separator:

(a) A two-class label where legitimate and malicious communications are represented by symbols 0 and 1 respectively.

(b) An acronym of network service. This label represents the type of communication on a particular network service.

(ii.) ASNM-NPBO v2: This dataset contains non-payload-based obfuscation techniques applied onto malicious traffic and onto several samples of legitimate TCP communications on selected vulnerable network services. The selection of vulnerable services was aimed on high severity of their successful exploitation leading to remote shell code execution through established backdoor communication.

legitimate representatives of the dataset were collected from two sources:

a) Legitimate traffic simulation in our virtual network architecture and also employed non-payload-based obfuscations for the purpose of real network simulation.

b) Common usage of all selected services was captured in the campus network, and all traffic was anonymized and further filtered on high severity alerts by signature-based NIDS Suricata and Snort through virus total API.

(iii.) ASNM-TUN: It consists of ASNM features extracted from tcp-dump capture tunnelling obfuscation techniques applied onto malicious traffic created with the intention to evade and improve machine learning classifiers and besides legitimate network traffic samples.

ASNM-TUN dataset contains four types of labels that are listed by increasing order of their level in the following:

· **Label_2:** It is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or a legitimate communication

· **Lable_3:** It is a three-class label, which distinguishes among legitimate traffic, direct attacks and obfuscated network attacks.

· **Label_poly:** It is a label that is composed of two parts:

(a) A three-class label

(b) An acronym of a network service

· **Label_poly_s:** It is composed of three parts:

(a) A three-class label

(b) An acronym of network service

(c) A network modification technique involved.

3.3.2 Functions requirement:

The functions that we have used in our program are:

1. Stacking Model
2. K Nearest Neighbor (KNN)
3. Support Vector Machine (SVM)
4. Naive Bayes (NB)
5. Decision Tree (DT)

1. Stacking Model:

Stacking is an ensemble learning technique that meta-learning uses to generate predictions. Here, the original data is further divided into n-folds (training and test data), which are embedded in various models (this study used ANN, SVM, Naive Bayes, and decision tree models). You can get some predictions. Returned to level 2 for final results.

2. K Nearest Neighbor (KNN):

K-nearest neighbor algorithms fall into the supervised learning category and are used for classification. Predict the class or contiguous value of a new data point, taking into account the K-nearest neighbors (data points).

3 Support Vector Machine (SVM):

Support vector machines are supervised machine learning algorithms that can be used for both classification and regression tasks. This algorithm plots each data item as a point in N-Dimensional space (where N is the number of features used). Where the value of each feature is the value of a particular coordinate. Then perform the classification by finding the hyperplane that distinguishes the two classes.

4. Naïve Bayes:

Naive Bayes is an easy-to-create classification technique, especially useful for large datasets where the presence of a particular feature in a class is assumed to be independent of the presence of other features.

5. Decision Tree:

Decision trees can be used for classification and regression problems. This name indicates that the flowchart is used like a tree structure to indicate the predictions that result from a series of feature-based divisions. It starts at the root node and ends with a leaf decision.

3.3.3 Performance and security requirement:

Performance requirements typically include a set of criteria that define how things work or the criteria that must be met under certain circumstances. The performance of your dataset and the algorithms we have used in this research project is somewhat same for all these as the accuracy rate for each and all models were above 90%, highest being 100% when we use naive bayes at level 1, other at level 0 for all the 3 sub-datasets and lowest being 97.96% when we use decision tree algorithm at level 1 and rest others at level 0.

A security requirement is a statement of security features required to ensure that one of the various security properties of the software is met. Since our project is based on finding anomalies the only thing to take care was to not to alter with the dataset, otherwise the result could differ a lot

3.4 Summary:

To summarize this chapter, we discussed about the hardware and requirement used in this project, moreover we tell you about the dataset used and the processing and standardizing part of it not only this but also discuss about the algorithms or models used by us to get the 100% accuracy rate in all the three sub-datasets of ASNM dataset. Since our project is based on finding anomalies the only thing to take care was to not to alter with the dataset, otherwise the result could differ a lot.

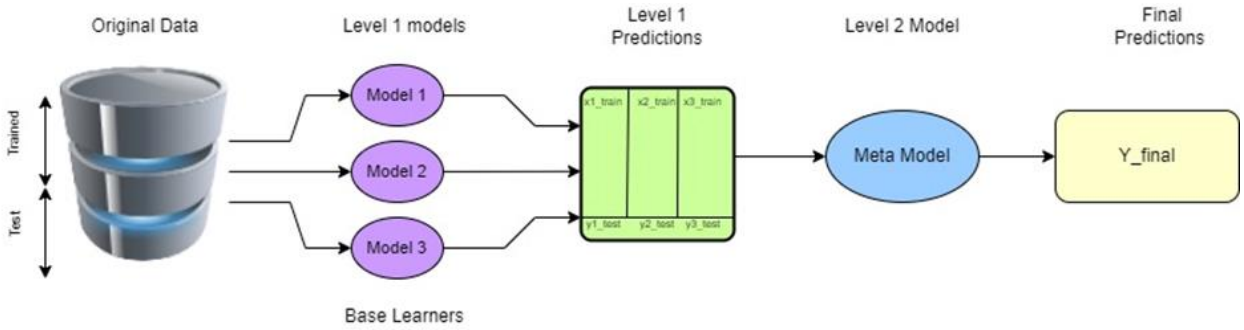
Chapter-4: Design methodology and its novelty

4.1 Methodology and goal:

The goal of our project was to find anomalies inside any networking device with 100% accuracy rate by combining the best algorithms present in the machine learning domain and to help firms preventing network attacks which can cause them millions of dollars not only this but also to spread awareness among the people and to help students to dig more in this domain.

For that, we took the publicly available dataset from the internet, which consists of three datasets that have been built from network traffic traces using ASNM (Advanced Security Network Metrics) features.

The proposed architecture diagram of stacking model



1. The following steps are involved in implementation:
2. The Original Train data is split into n-folds using the RepeatedStratifiedKFold.
3. Then the base learner (Model 1) is fitted on the first n1 folds and predictions are made for the nth part.
4. This prediction is added to the x1_train list.
5. Steps 2 and 3 are repeated for the remaining n1 parts to get an x1_train array of size n
6. Then train the model on all ‘n’ parts and make predictions with test data. Save this prediction in y1_test.
7. Similarly, we obtain x2_train, y2_test, x3_train and y3_test by using Model 2 and 3 for training respectively to obtain Level 2 predictions.

8. Now we train a Meta Learner on Level 1 Predictions (using these predictions as features for the model).
9. Metallerers are now used to predict test data.

Proposed Solution for our project:

We have used a stacking method to find the best algorithm which finds better accuracies in our dataset so we have 4 programs for each dataset and we have 3 dataset and in total we have 12 programs. Below is the split of the four programs for each dataset:

1. Level 0: DT, SVM, NB **Level 1:** KNN

2. Level 0: KNN, SVM, NB **Level 1:** DT

3. Level 0: KNN, DT, NB **Level 1:** SVM

4. Level 0: KNN, DT, SVM **Level 1:** NB

4.2 Functional modules design and analysis:

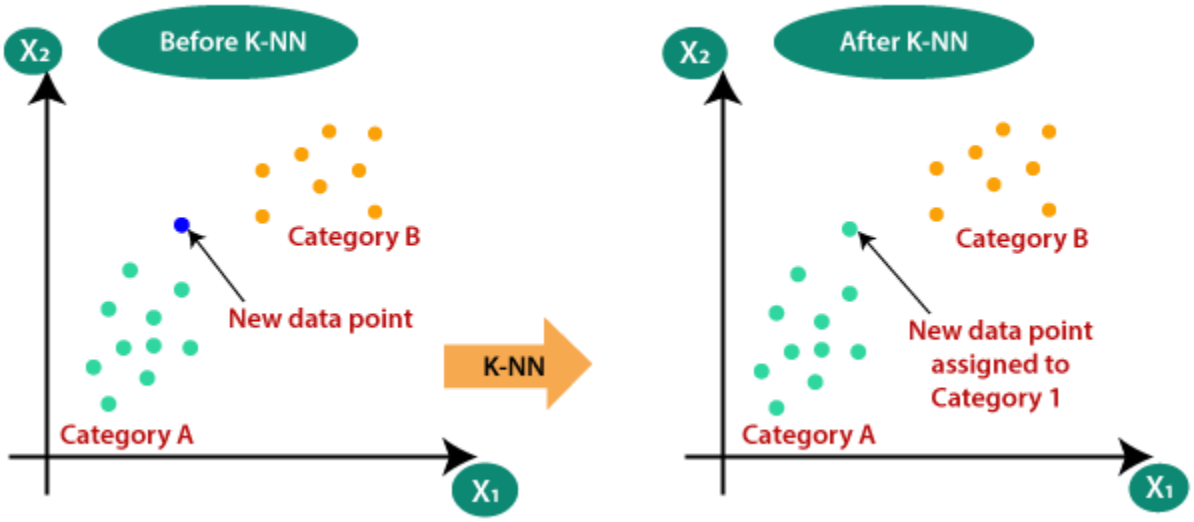
The list of modules we have used are

1. Stacking Model
2. K Nearest Neighbor (KNN)
3. Support Vector Machine (SVM)
4. Naive Bayes (NB)
5. Decision Tree (DT)

1. **Stacking Model:**

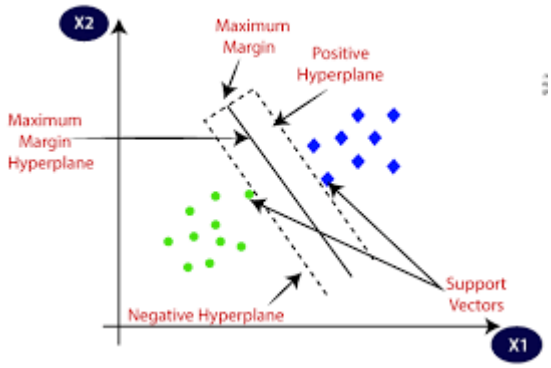
   Discussed in Section 4.1 with architecture diagram

2. K Nearest Neighbor (KNN):



1. Select the number K of neighbors
2. Calculate the Euclidean distance near K
3. Take the nearest K-nearest neighbor according to the calculated Euclidean distance.
4. Count the number of data points in each category within these k-nearest neighbors.
5. Assign new data points to the category with the largest number of neighbors.
6. The model is ready.

3. Support Vector Machine (SVM):



The behaviour of the SVM algorithm can be understood using an example:

- Suppose you have a dataset with two tags (green and blue), and the dataset has two characteristics x1 and x2. I need a classifier that can classify coordinate pairs (x1, x2) into green or blue.
- Since it is a 2D space, you can easily separate the two classes just by using a straight line. However, there can be multiple rows that can separate these classes.
- Therefore, the SVM algorithm helps to find the best line or decision boundary. This best boundary or region is called a hyperplane. The SVM algorithm finds the closest point on the lines of both classes. These points are called support vectors. The distance between the vector and the hyperplane is called the margin. And the goal of SVM is to maximize that margin. The hyperplane with the maximum margin is called the optimal hyperplane.

4. Naive Bayes (NB):

Naive Bayes algorithm works on bayes theorem and it is defined as,

Bayes Theorem: **P(A/B) = [P(B/A) * P(A)] / P(B)**

a) **P (A | B) is the posterior probability:** the probability of hypothesis A at the observed event B**.**

b) **P (B | A) is a Probability:** The probability of proving that the probability of the hypothesis is true.

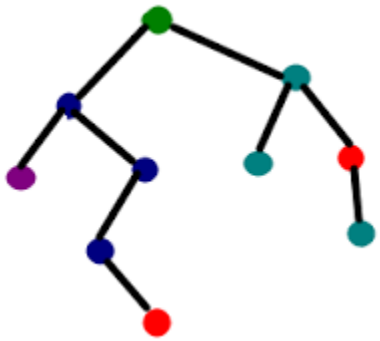
c) **P (A) is the prior probability:** the probability of the hypothesis before observing the evidence.

d) **P (B) is Peripheral Probability:** Probability of Probability.

You need to perform the following steps:

1. Converts the specified dataset to a frequency table.
2. Find the probabilities for a particular characteristic and create a probability table.
3. Then use Bayes' theorem to calculate posterior probabilities.

5. Decision Tree (DT):



Assumptions we make are:

1. At the beginning, we consider the whole training set as the root.
2. Attributes are assumed to be categorical for information gain and for Gini index, attributes are assumed to be continuous.
3. On the basis of attribute values records are distributed recursively.
4. We use statistical methods for ordering attributes as root or internal nodes.

Pseudocode:

1. Find the best attribute and place it on the root node of the tree.
2. Then divide the training set of the dataset into subsets. When creating subsets, make sure that the attribute values for each subset of the training dataset are the same.
3. Repeat 1 and 2 for each subset to find the leaf node in every branch.

4.3 Subsystem services:

The services which are described below plays equal importance role in making our project projects and they are:

1. **Draw.io:** This software aids us in creating architecture diagrams for algorithms.
2. **Google Docs:** This is the software where our team used to import our research paper.
3. **Microsoft Excel:** On this platform we have made charts for comparing which algorithms gives us a better accuracy rate.
4. **Google Drive:** Helps us in uploading all the necessary files.
5. **Google Meet/ Microsoft Teams:** This is where we were discussing the strategies for our project via video calling.
6. **WhatsApp:** This helps us in sharing files, photos and other essential materials for our research work.
7. **Sci-hub:** so, this is a platform where we used to get paid research papers for free of cost.
8. **Google scholar / ResearchGate:** This both were the most essential part in our project as we used to gather research papers from these websites.

4.4 Summary:

To summarize, we started with describing our goal and methodology then we talked about the four machine learning algorithms that we used in our project namely - KNN, NB, SVM and DT, explained about their working with steps and its diagrams and at the end we concluded what other software we have used in this journey of ours from beginning of the day 1 till the last last. We haven’t used any hardware except our laptop.

Chapter 5 Technical Implementations and Analysis:

5.1 Outline:

The following are the prerequisites for the machine that will be used to carry out the research:

Tested Environment

OS	Windows 10
RAM	8GB
GPU	4GB
IDE	Visual Studio Code (Python)

The entire algorithm is implemented with the above configuration. As we can observe from the above table, the physical requirements are negligible and the proposed model can be compatible with almost any physical machine.

5.2 Technical Code and Code Solutions:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import StackingClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score, roc_curve, auc
import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

Here we are importing all the necessary modules required to run the project and the explanation of these modules are previously explained.

```
data = pd.read_csv(r'D:\Python\Project\Dataset\ASNM-CDX-2009.csv', sep = ';')
y = data.iloc[:, 1]
X = data.iloc[:, 2:]

y = [int(bool(i)) for i in y]
y = np.array(y)

encoder = LabelEncoder()
X['label_poly'] = encoder.fit_transform(X['label_poly'].astype(str))
X['SrcIP'] = encoder.fit_transform(X['SrcIP'].astype(str))
X['DstIP'] = encoder.fit_transform(X['DstIP'].astype(str))
X['SrcMAC'] = encoder.fit_transform(X['SrcMAC'].as
X['DstMAC'] = encoder.fit_transform(X['DstMAC'].astype(str))
X['SrcIPInVlan'] = X['SrcIPInVlan'].astype(np.int32)
X['DstIPInVlan'] = X['DstIPInVlan'].astype(np.int32)
X.iloc[:, 535] = X.iloc[:, 535].astype(np.int32)
X.iloc[:, 569] = X.iloc[:, 569].astype(np.int32)
```

Loading...

Moving forward, here we are importing and formatting the dataset for our better understanding and making the process easier for us so that we can process this data easily. Here we are converting all the labels into string variables and in binary form further, we are arranging them in an array with different names to them.

```
33
34  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 101)
35
36  scaler = StandardScaler().fit(X_train)
37  X_train = scaler.transform(X_train)
38  X_test = scaler.transform(X_test)
39
40  level0 = list()
41  ##level0.append(('knn', KNeighborsClassifier()))
42  level0.append(('cart', DecisionTreeClassifier()))
43  level0.append(('svm', SVC()))
44  level0.append(('nb', GaussianNB()))
45  level1 = KNeighborsClassifier()
46
```

Next, we are training the 75% of the dataset and based on its predictions we are testing our 25% of the dataset to predict the same outcome as we got while training the dataset.

Since, we are using stacking machine learning model we have took four algorithms in level 0 and level 1 of the model, in which level 0 comprises of any 3 machine learning algorithms and the remaining algorithm is used at level 1 of the stacking model what this means is that based on the predictions of the first 3 algorithms at level 0 we are expecting the same outcome when we input the outcome of first three algorithms into the level 1 algorithm.

```
model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test).round()

a = confusion_matrix(y_test, y_pred)
print(a)
print('Classification Report:\n', classification_report(y_test, y_pred))
print('Training Accuracy : {:.2f}%'.format(accuracy_score(model.predict(X_train).round(), y_train) * 100))
print('Testing Accuracy : {:.2f}%'.format(accuracy_score(y_pred, y_test) * 100))
print('Sensitivity: {:.2f}'.format(a[1][1]/(a[1][0]+a[1][1])))
print('Specificity: {:.2f}'.format(a[0][0]/(a[0][0]+a[0][1])))
print('Precision: {:.2f}'.format(precision_score(y_test, y_pred)))
print('F1 Score: {:.2f}'.format(f1_score(y_test, y_pred)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_pred)))
```

Subsequently, we have used a stacking classifier to train and test data, based on the working of the stacking model (which has just been explained) and we are calculating the metrics like training accuracy, testing accuracy, F1-score, precision, sensitivity, recall value and confusion matrix.

```
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Naive Bayes (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
plt.show()
```

At last, we are finding the ROC (Receiver Operating Characteristic) curve which tells us the true positive to false positive ratio with area under the curve.

5.3 Test and Validation:

We employed the ASNM dataset, which has three datasets: ASNM-CDX 2009, ASNM-NBPOv2, and ASNM-TUN datasets, and we have four methods in total, making a total of 12 programs.

Our dataset is divided into three parts:

(i.) 2009 ASNM-CDX: This dataset contains ASNM characteristics derived from tcp-dump captures malicious and normal TCP communications on network services that are vulnerable to buffer overflow attacks, as well as network traffic dumps from the CDX-2009 dataset.

This dataset's link is:
https://drive.google.com/drive/folders/1227eiS1BXC7oZJXZaxdRMyRmVfpIQsaC?usp=sharing

(ii.) ASNM-NPBO v2: On chosen susceptible network services, this dataset comprises non-payload-based obfuscation techniques deployed to malicious traffic and various instances of legal TCP interactions. The vulnerable services chosen were chosen based on the likelihood of successful exploitation leading to remote shell code execution via establishing backend connectivity.

This dataset's link is:
https://drive.google.com/drive/folders/1227eiS1BXC7oZJXZaxdRMyRmVfpIQsaC?usp=sharing

(iii.) ASNM-TUN: It includes ASNM characteristics collected from tcp-dump capture tunnelling obfuscation methods applied to malicious traffic built to elude and enhance machine learning classifiers, as well as normal network traffic samples.

The dataset’s link is:
https://drive.google.com/drive/folders/1227eiS1BXC7oZJXZaxdRMyRmVfpIQsaC?usp=sharing

Now, the output of all the 12 programs is:

1. **ASNM-CDX 2009 Dataset:**

   a. **KNN is at level 1 and DT, SVC and NB at level 0:**

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe
[[1429    0]
 [   0   14]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1429
           1       1.00      1.00      1.00        14

    accuracy                           1.00      1443
   macro avg       1.00      1.00      1.00      1443
weighted avg       1.00      1.00      1.00      1443

Training Accuracy : 99.98%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here, the confusion matrix is:

1429	**0**
0	**14**

which indicates that 1429 numbers of negative examples are classified correctly and only 14 numbers of positive samples were classified correctly.

Here the Metrics was 100% correct with testing accuracy 100%.

b. DT is at level 1 and KNN, SVC and NB at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe
[[1425    4]
 [   8    6]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1429
           1       0.60      0.43      0.50        14

    accuracy                           0.99      1443
   macro avg       0.80      0.71      0.75      1443
weighted avg       0.99      0.99      0.99      1443

Training Accuracy : 99.91%
Testing Accuracy : 99.17%
Sensitivity: 0.43
Specificity: 1.00
Precision: 0.60
F1 Score: 0.50
Recall: 0.43
PS D:\Python\Project\Final 30 Programs>
```

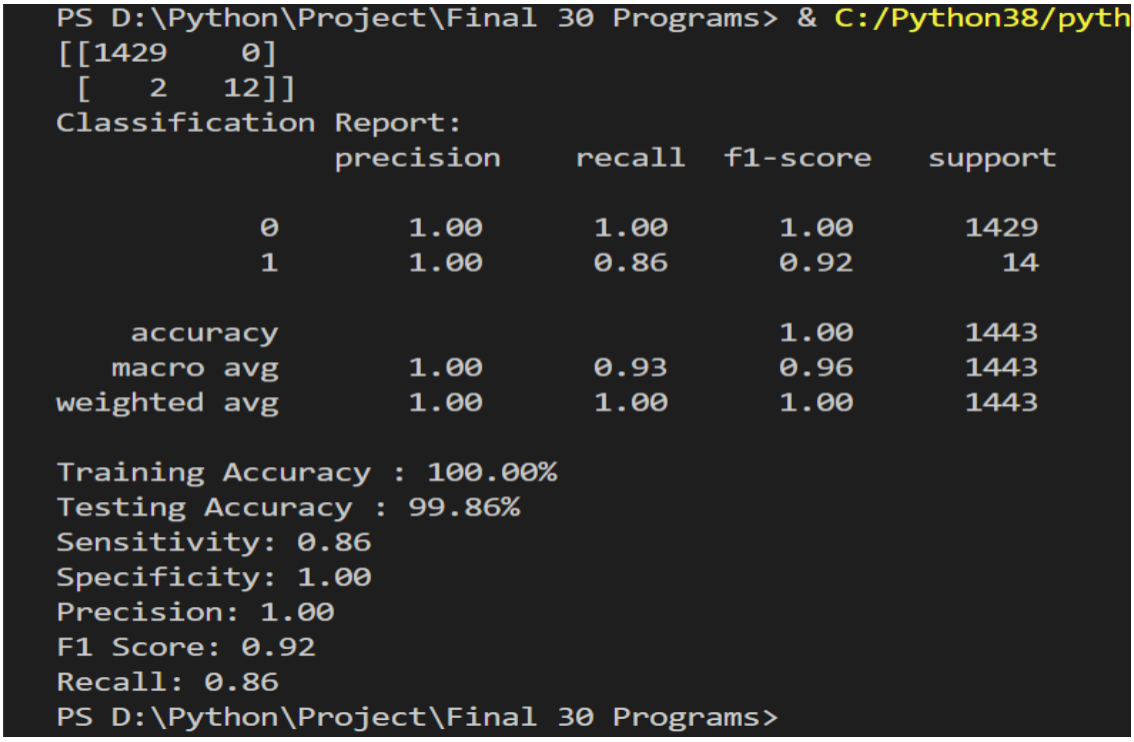
Here the **Confusion matrix** is:

1425	**4**
8	**6**

Which indicates that 1425 number of negative samples are classified correctly, 6 number of positive samples were classified correctly, 4 number of actual negative examples classified as positive and 8 number of actual positive examples are classified as negative.

Here the Metrics is varying with F1- score being half the value of specificity, sensitivity and recall value are same with precision value being 0.43 correct with testing accuracy being 99.17%.

c. SVC is at level 1 and KNN, DT and NB at level 0:



```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/pyth
[[1429    0]
 [   2   12]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1429
           1       1.00      0.86      0.92        14

    accuracy                           1.00      1443
   macro avg       1.00      0.93      0.96      1443
weighted avg       1.00      1.00      1.00      1443

Training Accuracy : 100.00%
Testing Accuracy : 99.86%
Sensitivity: 0.86
Specificity: 1.00
Precision: 1.00
F1 Score: 0.92
Recall: 0.86
PS D:\Python\Project\Final 30 Programs>
```

Here the **confusion matrix** is:

1429	**0**
2	**12**

Which indicates that 1429 number of negative samples are classified correctly, 12 number of positive samples were classified correctly, nil number of actual negative examples classified as positive and 2 number of actual positive examples are classified as negative.

Here the metrics of precision and specificity is 100%, sensitivity and recall score are the same, 0.86, while F1-Score is 92% and at last testing accuracy being 99.17%.

d. NB is at level 1 and KNN, DT and SVC at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python
[[1429    0]
 [   0   14]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1429
           1       1.00      1.00      1.00        14

    accuracy                           1.00      1443
   macro avg       1.00      1.00      1.00      1443
weighted avg       1.00      1.00      1.00      1443

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here the **confusion matrix** is:

1429	**0**
0	**14**

Which indicates that 1429 number of negative samples are classified correctly, 14 number of positive samples were classified correctly.

Here the Metrics is 100% with testing accuracy being 99.17%.

2. ASNM-NBPOv2 Dataset:

a. KNN is at level 1 and DT, SVC and NB at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe "d:/Python/Project/Final 30 Programs/2_A.py"
[[2693    0]
 [   0  169]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2693
           1       1.00      1.00      1.00       169

    accuracy                           1.00      2862
   macro avg       1.00      1.00      1.00      2862
weighted avg       1.00      1.00      1.00      2862

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

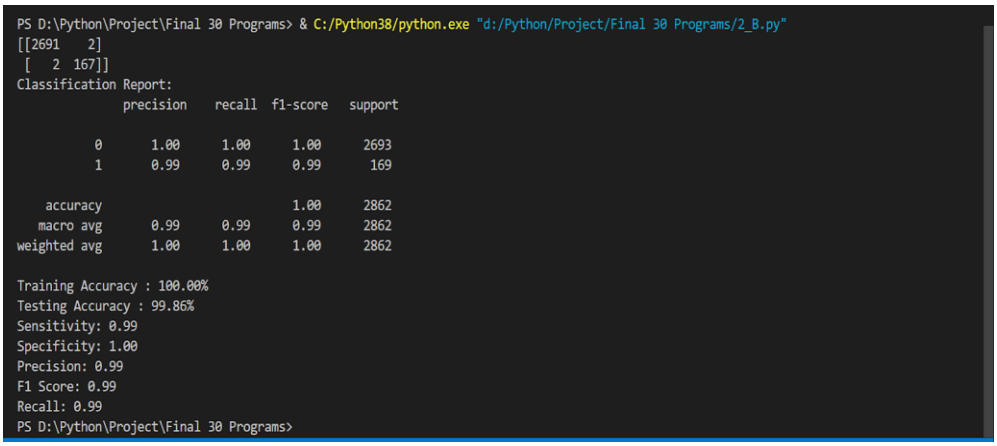
Here the **confusion matrix** is:

2693	**0**
0	**169**

Which indicates that 2693 numbers of negative samples are classified correctly, 169 numbers of positive samples were classified correctly.

Here the Metrics is 100% with testing accuracy being 99.17%.

b. DT is at level 1 and KNN, SVC and NB at level 0:



```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe "d:/Python/Project/Final 30 Programs/2_B.py"
[[2691    2]
 [   2  167]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2693
           1       0.99      0.99      0.99       169

    accuracy                           1.00      2862
   macro avg       0.99      0.99      0.99      2862
weighted avg       1.00      1.00      1.00      2862

Training Accuracy : 100.00%
Testing Accuracy : 99.86%
Sensitivity: 0.99
Specificity: 1.00
Precision: 0.99
F1 Score: 0.99
Recall: 0.99
PS D:\Python\Project\Final 30 Programs>
```

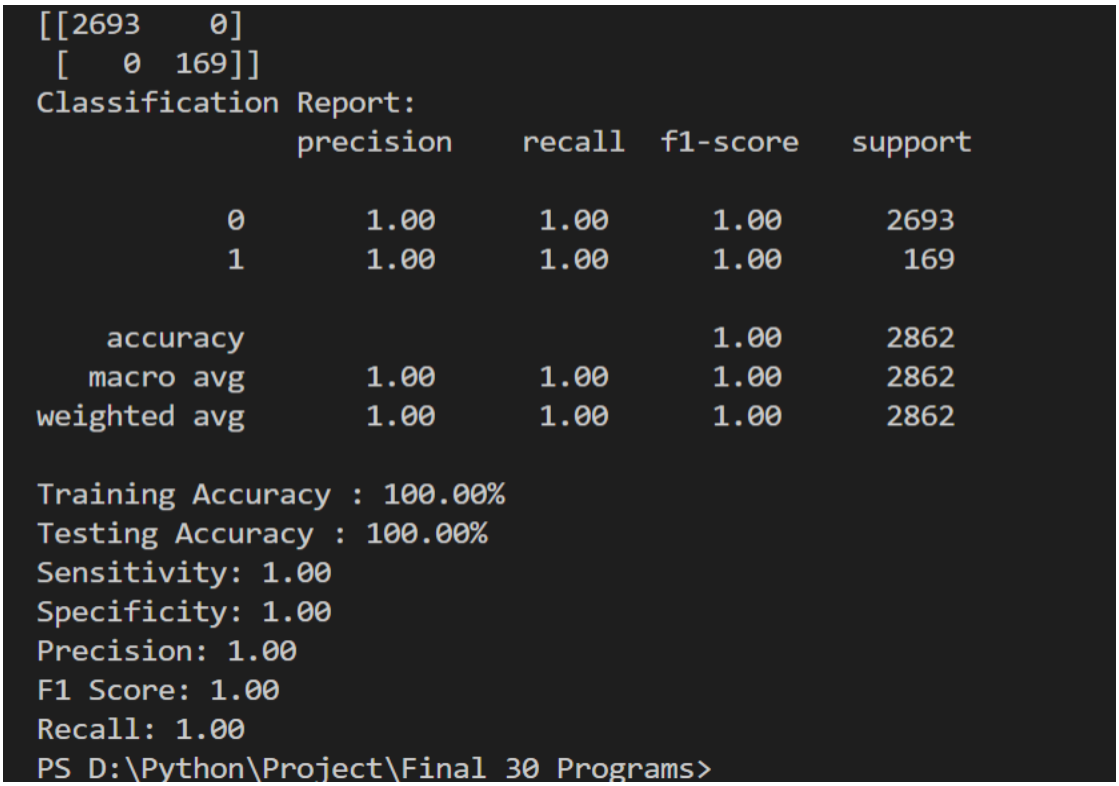
Here the **confusion matrix** is:

2691	**2**
2	**167**

Which indicates that 2691 number of negative samples are classified correctly, 167 number of positive samples were classified correctly, 2 number of actual negative examples classified as positive and 2 number of actual positive examples are classified as negative.

Here the Metrics score is 99% except for specificity, it’s 100% for it and testing accuracy being 99.86%.

c. SVC is at level 1 and KNN, DT and NB at level 0:



```
[[2693    0]
 [   0  169]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2693
           1       1.00      1.00      1.00       169

    accuracy                           1.00      2862
   macro avg       1.00      1.00      1.00      2862
weighted avg       1.00      1.00      1.00      2862

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here the **confusion matrix** is:

2693	**0**
0	**169**

Which indicates that 2693 numbers of negative samples are classified correctly, 169 numbers of positive samples were classified correctly.

Here the Metrics score is 100% with testing accuracy being 100%.

d. NB is at level 1 and KNN, SVC and NB at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe "
[[2693    0]
 [   0  169]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2693
           1       1.00      1.00      1.00       169

    accuracy                           1.00      2862
   macro avg       1.00      1.00      1.00      2862
weighted avg       1.00      1.00      1.00      2862

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here, the **confusion matrix** is:

2693	**0**
0	**169**

Which indicates that 2693 numbers of negative samples are classified correctly, 169 numbers of positive samples were classified correctly.

Here, the Metrics score is 100% with testing accuracy being 100%.

3. ASNM-TUN Dataset:

a. KNN is at level 1 and DT, SVC and NB at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe
[[42  1]
 [ 0 55]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        43
           1       0.98      1.00      0.99        55

    accuracy                           0.99        98
   macro avg       0.99      0.99      0.99        98
weighted avg       0.99      0.99      0.99        98

Training Accuracy : 100.00%
Testing Accuracy : 98.98%
Sensitivity: 1.00
Specificity: 0.98
Precision: 0.98
F1 Score: 0.99
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here, the **Confusion matrix** is:

42	**1**
0	**55**

Which indicates that 42 numbers of negative samples are classified correctly, 55 numbers of positive samples were classified correctly, 1 number of actual negative examples classified as positive.

Here the Metrics Score is 100% for Sensitivity and Recall value, while 98% for Specificity and precision and 99% for F1 Score. Testing accuracy being 98.98%.

b. DT is at level 1 and KNN, SVC and NB at level 0:

```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe "d
[[42  1]
 [ 3 52]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.98      0.95      0.96        55

    accuracy                           0.96        98
   macro avg       0.96      0.96      0.96        98
weighted avg       0.96      0.96      0.96        98

Training Accuracy : 100.00%
Testing Accuracy : 95.92%
Sensitivity: 0.95
Specificity: 0.98
Precision: 0.98
F1 Score: 0.96
Recall: 0.95
PS D:\Python\Project\Final 30 Programs>
```

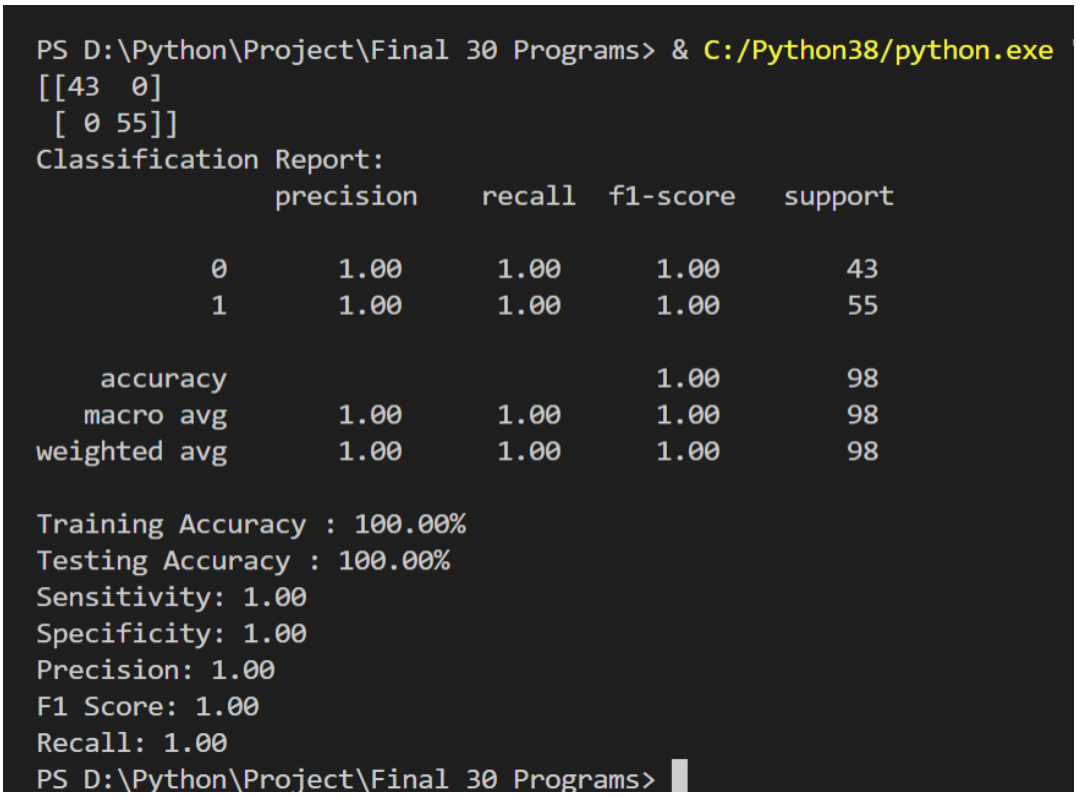
Here, the **confusion matrix** is:

42	**1**
3	**52**

Which indicates that 42 numbers of negative samples are classified correctly, 52 numbers of positive samples were classified correctly, 1 number of actual negative examples classified as positive and 3 numbers of actual positive examples are classified as negative.

The metrics scores are for sensitivity and recall value it's 95%, for specificity and precision it's 98% while F1 score is 96% and testing accuracy is 95.92%.

c. SVC is at level 1 and KNN, DT and NB at level 0:



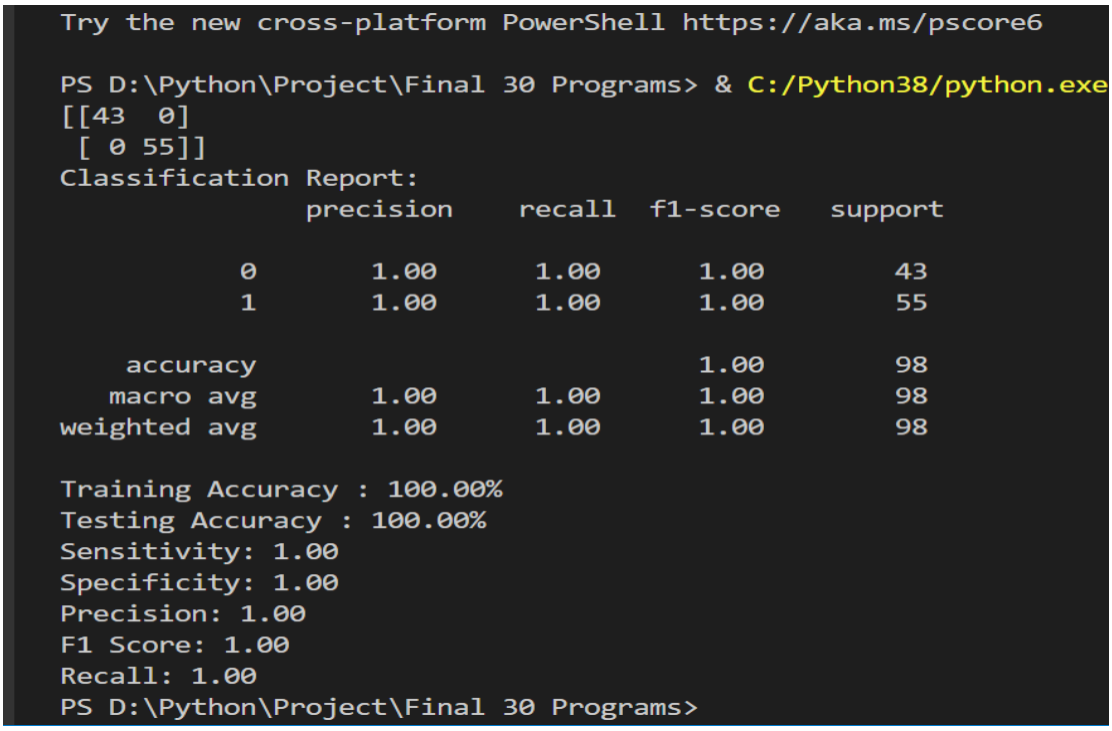
```
PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe
[[43  0]
 [ 0 55]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        55

    accuracy                           1.00        98
   macro avg       1.00      1.00      1.00        98
weighted avg       1.00      1.00      1.00        98

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

Here, the **confusion matrix** is:

43	**0**
0	**55**

The metrics score and testing accuracy are 100%

d. NB is at level 1 and KNN, DT and SVC at level 0:



```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Python\Project\Final 30 Programs> & C:/Python38/python.exe
[[43  0]
 [ 0 55]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        55

    accuracy                           1.00        98
   macro avg       1.00      1.00      1.00        98
weighted avg       1.00      1.00      1.00        98

Training Accuracy : 100.00%
Testing Accuracy : 100.00%
Sensitivity: 1.00
Specificity: 1.00
Precision: 1.00
F1 Score: 1.00
Recall: 1.00
PS D:\Python\Project\Final 30 Programs>
```

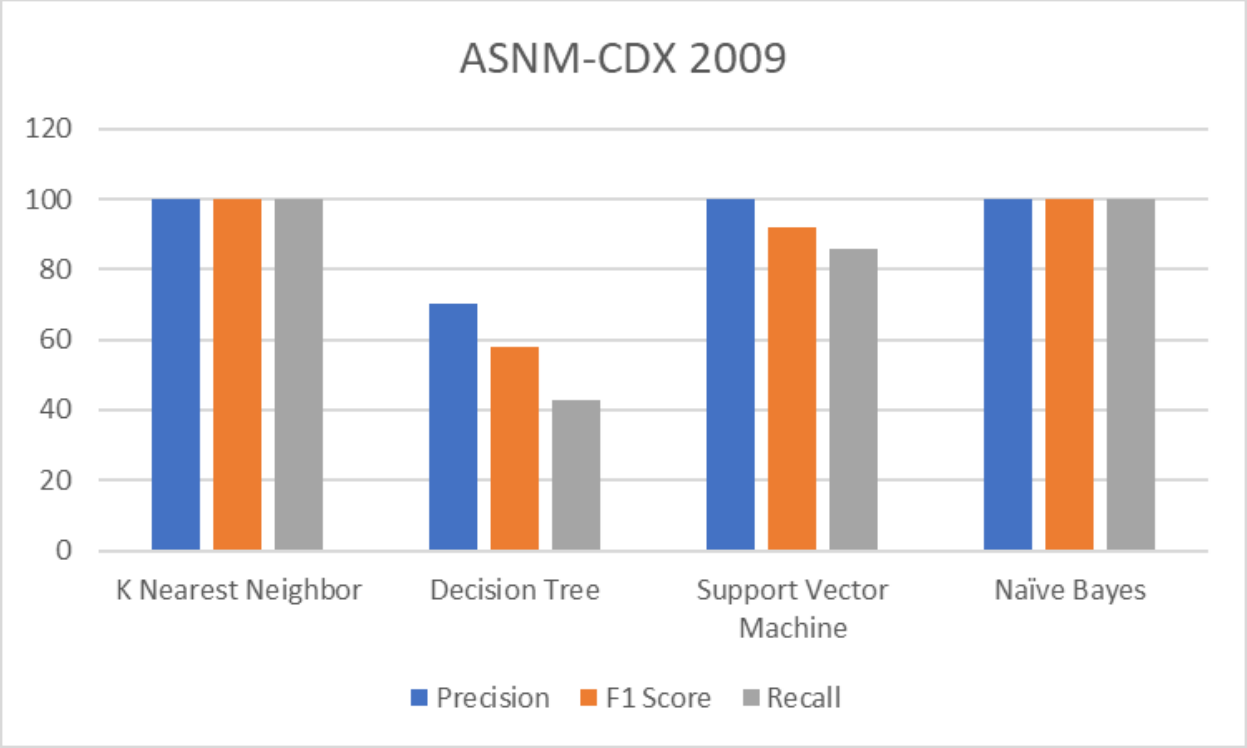
Here, the **confusion matrix** is:

43	**0**
0	**55**

The metrics score and testing accuracy are 100%.

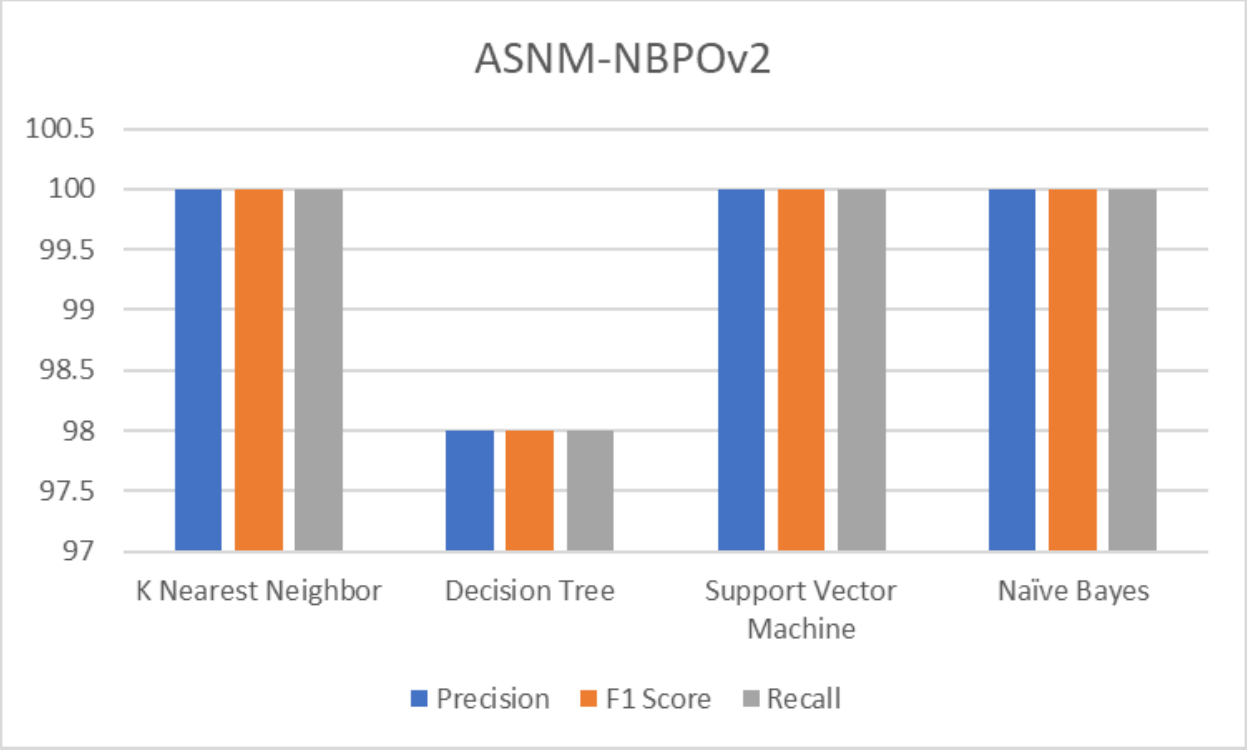
5.4 Performance Analysis:

Comparison of metrics – (Precision, Recall, F1-score) for **ASNM-CDX 2009** Dataset



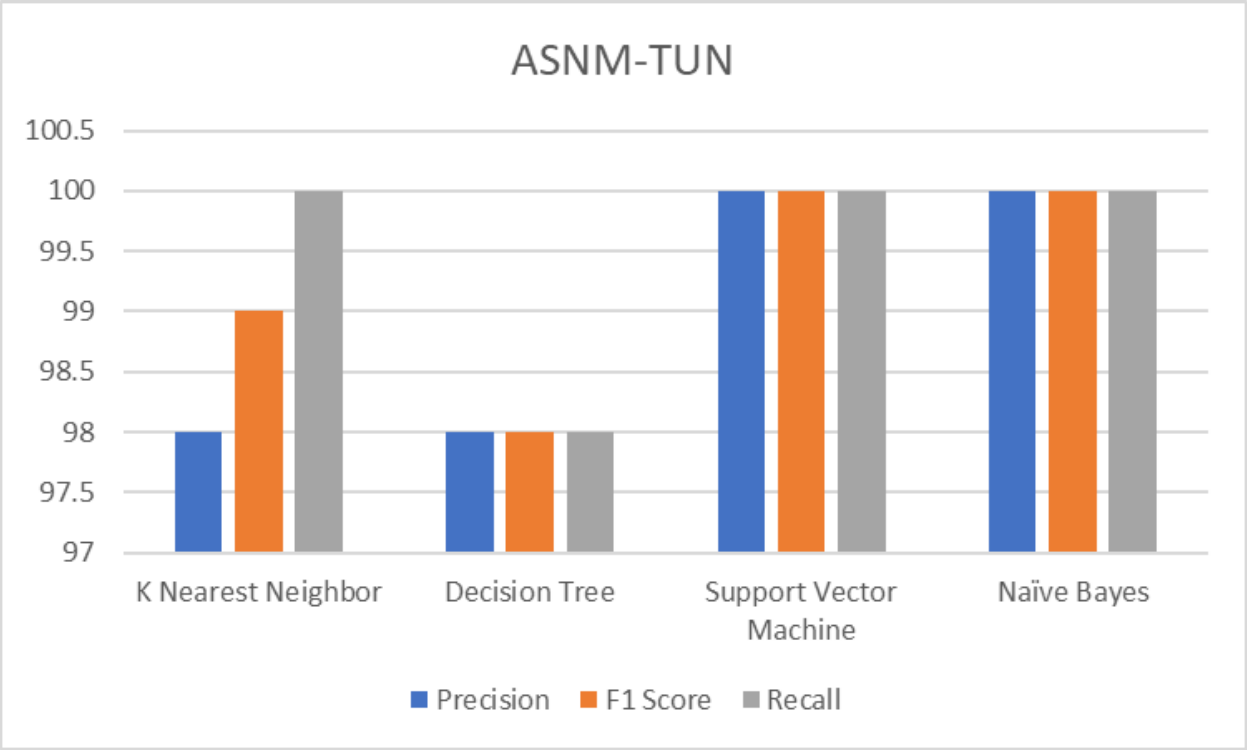
The above chart depicts the performance shown by four algorithms on ASNM-CDX 2009 dataset. K Nearest Neighbor algorithm and Support Naïve Bayes algorithm have a perfect score of metrics. The Support Vector Machine model has a high precision score only while least metrics are shown by the decision tree model.

Comparison of metrics – (Precision, Recall, F1-Score) for **ASNM-NBPOv2** Dataset



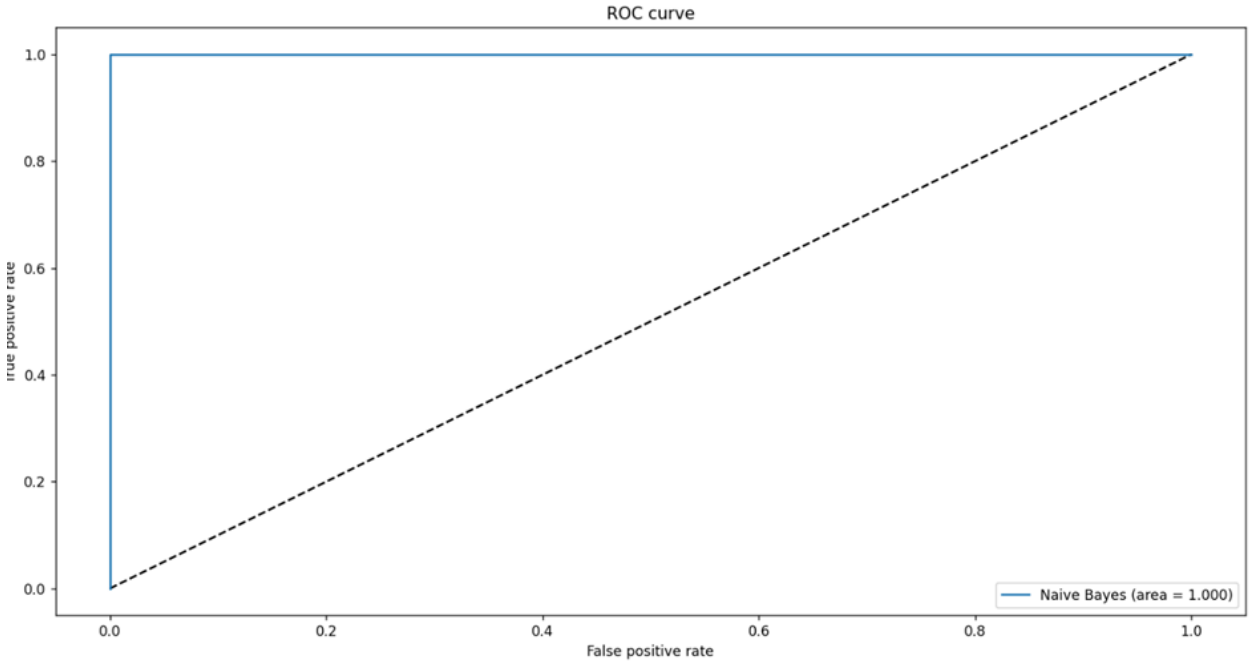
It shows the performance of all AI models on ASNM-NBPOv2 dataset. Support Vector Machine, Naïve Bayes and K Nearest Neighbor show high values in all the three metrics while Decision Tree accounted for 98%.

Comparison of Metrics – (Precision, Recall, F1-Score) for **ASNM-TUN** dataset



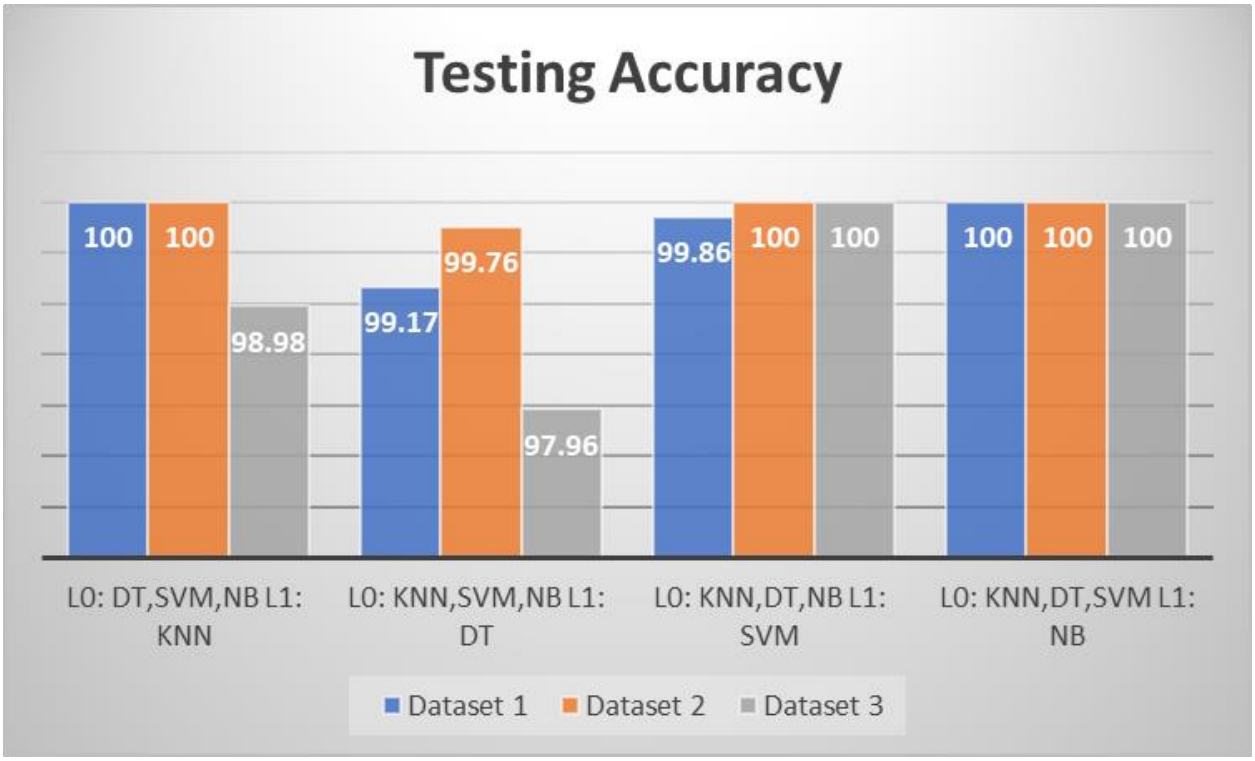
The bar graph of figure _ shows the performance of the 3 Artificial Intelligence models on ASNM-TUN dataset. Support Vector Machine and Naïve Bayes display all the metrics with utmost percentage. The Precision of K Nearest Neighbor is 100% while the decision tree has equal amount of proportion in all the three domains.

ROC curve for KNN, Decision tree, SVM at level 0 and naïve bayes at level 1 model.



The above model of KNN, SVM and Decision Tree at level 0 and Naïve Bayes at level 1 model shows utmost true positive to false positive ratio with area under curve of 1.000.

After a through comparison and contrast, it is empirical that the proposed model of Naïve Bayes at level 1 and other three algorithms at level 0 has surpassed all the other techniques, with an accuracy of 100%, and the side metrics of Precision, F1 Score and Recall score were found to be 100% in all the sub dataset of ASNM dataset.



From the above graph it is clearly visible that the model in which Naïve Bayes is at level 1 has 100% accuracy in all the three datasets.

5.5 Summary:

This chapter discusses all about the technical content of this research work. Starting off with explaining the code and the sub-dataset of ASNM dataset that we have used in this project. Subsequently, we have provided in-depth outcomes of your code and at the end we compared all the algorithms on the basis of their performance and metrics.

Chapter-6: Project Outcome and Applicability

6.1 Outline:

Anomaly detection helps in numerous ways as it tells an organization where they are vulnerable for attacks and also gives them the time to improve upon them. The accuracy of the hybrid algorithms ensures better results and keeps the organization safe from possible breach.

6.2 key implementations outline of the System

Here we propose a novel technique to determine the malicious intent of a network request. Utmost accuracy in all the three datasets. In our project

6.3 Significant project outcomes:

Attained utmost accuracy with anomaly detection in a data set by using a hybrid model of algorithms to get perfect results compared to that of singular algorithms.

6.4 Real-world applications:

This project can be used by various organizations to keep track of their network logs and monitor any irregular changes in the log patterns. This is achieved with higher accuracy using our proposed hybrid model.

6.4 Inference:

In this chapter we discuss the various outcomes achieved from this project and also how it can be used in real life by giving a major hand in resolving anomalies with high accuracy percentages.

Chapter-7: Conclusions and Recommendation

7.1 Outline:

Our project is based on Stacking the reason behind choosing stacking is that it is more efficient than other algorithms which are present, it has less false rate and is less susceptible to attacks, In, our method we have taken 4 machine learning algorithms and at any instance one of the algorithms is used at Tier 1 and the other one at Tier 0 the reason is to achieve higher accuracy.

7.2 Constraints of the System

The constraints of the system are:

1. To fit an artificial Neural Network (ANN) model in a stacking model we require a high-performance PC/laptop.
2. It probably won't help in small datasets.
3. The improvement of the ensemble model over the best individual model tends to be relatively small. Oftentimes, there is no improvement whatsoever.
4. Time consuming to train the model as we have to assemble many base models.
5. More memory consumption.

7.3 Future Enhancements

With the help of our idea in the research paper we can implement it in projects for better accuracy and precision from anomaly detection in data sets. We can improve the accuracy by improving time complexity of machine learning algorithms.

7.4 Inference

In the chapter we have described the basic outline of our project about what limitations or constraints are there in our current model and have also talked about what addition can be done in the algorithm in future to make it more efficient and accurate.

References:

1. https://ieeexplore.ieee.org/document/9115004
2. http://www.fit.vutbr.cz/~ihomoliak/asnm/
3. https://www.edureka.co/blog/what-is-a-neural-network/
4. https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207#:~:text=Artificial%20Neural%20Network(ANN)%20uses,complex%20patterns%20and%20prediction%20problems.
5. https://www.ijitee.org/wp-content/uploads/papers/v8i9/I7914078919.pdf
6. https://www.mdpi.com/1099-4300/23/5/529
7. https://www.analyticssteps.com/blogs/8-applications-neural-networks
8. https://www.xenonstack.com/blog/artificial-neural-network-applications
9. https://towardsdatascience.com/building-our-first-neural-network-in-keras-bdc8abbc17f5
10. https://link.springer.com/article/10.1007/s11277-017-4961-1
11. https://www.worldscientific.com/doi/abs/10.1142/S0218194020500114
12. https://dl.acm.org/doi/abs/10.1145/3336191.3371876
13. https://dl.acm.org/doi/abs/10.1145/3441448
14. https://www.researchgate.net/profile/Purvi-Prajapati/publication/330138092_Study_and_Analysis_of_Decision_Tree_Based_Classification_Algorithms/links/5d2c4a91458515c11c3166b3/Study-and-Analysis-of-Decision-Tree-Based-Classification-Algorithms.pdf
15. https://jcomsec.ui.ac.ir/article_24558_4491.html
16. (PDF) Insider Threat Detection Based on User Behavior Modeling and Anomaly Detection Algorithms (researchgate.net)
17. https://www.researchgate.net/publication/342118050_ASNM_Datasets_A_Collection_of_Network_Attacks_for_Testing_of_Adversarial_Classifiers_and_Intrusion_Detectors
18. https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2109

19. https://link.springer.com/chapter/10.1007/978-3-642-31537-4_46
20. https://www.researchgate.net/publication/347635021_Cyberattacks_Detection_in_IoT-Based_Smart_City_Applications_Using_Machine_Learning_Techniques
21. https://www.irjet.net/
22. https://www.researchgate.net/publication/336767849_ASNM_Datasets_A_Collection_of_Network_Traffic_Features_for_Testing_of_Adversarial_Classifiers_and_Network_Intrusion_Detectors