**A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | |
| **S. NO.** | **TITLE** |
| 1 | Abstract |
| 2 | Introduction |
| 3 | Existing System |
| 4 | Disadvantages |
| 5 | Proposed System |
| 6 | Advantages |
| 7 | Flow Diagram |
| 8 | Use Case Diagram |
| 9 | Sequential Diagram |
| 10 | ER Diagram |
| 11 | Architecture Diagram |
| 12 | Modules |
| 13 | Modules Description |
| 14 | Literature Survey |
| 15 | Hardware & Software Requirements |
| 16 | Hardware & Software Description |
| 17 | Testing of Product |
| 18 | Conclusion |
| 19 | Future Enhancement |
| 20 | Coding & Screenshot |
| 21 | References |

**ABSTRACT**

Intrusion detection is one of the important security problems in today’s cyber world. A significant number of techniques have been developed which are based on machine learning approaches. So for identifying the intrusion we have designed the machine learning algorithms. By using the algorithm we find out intrusion and we can identify the attacker’s details also. IDS are mainly two types: Host based and Network based. A Host based Intrusion Detection System (HIDS) monitors individual host or device and sends alerts to the user if suspicious activities such as modifying or deleting a system file, unwanted sequence of system calls, unwanted configuration changes are detected. A Network based Intrusion Detection System (NIDS) [16] is usually placed at network points such as a gateway and routers to check for intrusions in the network traffic. Here we use SVM(Support Vector machine is used for classification.The basics of Support Vector Machines and how it works are best understood with a simple example. Let’s imagine we have two tags: red and blue, and our data has two features: x and y. We want a classifier that, given a pair of (x,y) coordinates, outputs if it’s either red or blue. We plot our already labeled training data on a plane:

**INTRODUCTION**

A detailed investigation and analysis of various machine learning techniques have been carried out for finding the cause of problems associated with various machine learning techniques in detecting intrusive activities. Attack classification and mapping of the attack features is provided corresponding to each attack. Issues which are related to detecting low-frequency attacks using network attack dataset are also discussed and viable methods are suggested for improvement. Machine learning techniques have been analysed and compared in terms of their detection capability for detecting the various category of attacks.

**EXISTING SYSTEM**

HACKING incidents are increasing day by day as technology rolls out. A large number of hacking incidents are reported by companies each year. The existing system doesn’t effectively classify and predict the attack which is presented in the network.

**DISADVANTAGES**

* Doesn’t Efficient for handling large volume of data.
* Theoretical Limits
* Incorrect Classification Results.
* Less Prediction Accuracy.

**PROPOSED SYSTEM**

The proposed model is introduced to overcome all the disadvantages that arises in the existing system. This system will increase the accuracy of the classification results by classifying the data based on the social network mental disorders and others using random forest classification algorithm. It enhances the performance of the overall classification results.

**ADVANTAGES**

* High performance.
* Provide accurate prediction results.
* It avoid sparsity problems.
* Reduces the information Loss and the bias of the inference due to the multiple estimates.

**SYSTEM ARCHITECTURE**

DATASET

FORMATTED DATSET

TRAIN

TEST

CLASSIFICATION

PREDICTION

**2/22/2019**

**A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection**

15

**FLOW DIAGRAM**

SELECT DATSET

CLEANING DATSET

SPLIT TRAIN AND TEST

SVM CLASSIFICATION

PREDICTION

**USE CASE DIAGRAM**

USER

**SEQUENCE DIAGRAM**

Clean

Split

Classification

Prediction

Select

Select dataset

 Load dataset

Start

Result Generation

**ER DIAGRAM**

**DATA SELECTION & LOAD**

**Split Train and Test**

**CLASSIFICATION**

**Result Generation**

**MODULES**

* Data Selection and Loading
* Data Preprocessing
* Splitting Dataset into Train and Test Data
* Feature Extraction
* Classification
* Prediction
* Result Generation

**DATA SELECTION AND LOADING**

* The data selection is the process of selecting the data for detecting the attacks.
* In this project, the KDDCUP dataset is used for detecting attacks.
* The dataset which contains the information about the duration, flag, service, src bytes, dest bytes and class labels.

**DATA PREPROCESSING**

* Data pre-processing is the process of removing the unwanted data from the dataset.
* Missing data removal
* Encoding Categorical data
* Missing data removal: In this process, the null values such as missing values are removed using imputer library.
* Encoding Categorical data: That categorical data is defined as variables with a finite set of label values. That most machine learning algorithms require numerical input and output variables. That an integer and one hot encoding is used to convert categorical data to integer data.

**SPLITTING DATASET INTO TRAIN AND TEST DATA**

* Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating data mining models.
* Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

**FEATURE EXTRACTION**

* Feature scaling. Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.
* Feature Scaling or Standardization: It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.

**CLASSIFICATION**

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships.

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyper planes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1.

**PREDICTION**

* It’s a process of predicting the attacks in the network from the dataset.
* This project will effectively predict the data from dataset by enhancing the performance of the overall prediction results.

**RESULT GENERATION**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* True Positive
* True Negative
* False Positive
* False Negative
* Accuracy
* Precision
* Recall
* F1-Score

**LITERATURE SURVEY**

**Title**: The Internet of Things - How the Next Evolution of the Internet is Changing Everything.

**Year**: 2011

**Author**: D. Evans

**Methodology**

This research describes the methodology and the development process of creating an IoT platform. This paper also presents the architecture and implementation for the IoT platform. The goal of this research is to develop an analytics engine which can gather sensor data from different devices and provide the ability to gain meaningful information from IoT data and act on it using machine learning algorithms.

**Advantage**

The proposed system is introducing the use of a messaging system to improve the overall system performance as well as provide easy scalability.

**Disadvantage**

Low cost devices are easily able to connect wirelessly to the Internet, from handhelds to coffee machines, also known as Internet of Things (IoT).

**Title:** The Internet of Things: A survey

**Year:** 2010.

**Author:** L. Atzori, A. Iera, and G. Morabito

**Methodology**

The object unique addressing and the representation and storing of the exchanged information become the mostchallenging issues, bringing directly to a third, ‘‘Semantic oriented”, perspective of IoT.

**Advantage**

People are informed of the scope and the way in which their movements are tracked by the system (taking peo-ple informed about possible leaks of their privacy is essential and required by most legislations).

**Disadvantage**

The user can set the preferences of the proxy. When sensor networks and RFID systems are in-cluded in the network, then the proxy operates between them and the services.

**Title**: Addressing the Class Imbalance Problem in Medical Datasets

**Year**: 2012

**Author**: M. Mostafizur Rahman and D. N. Davis

**Methodology**

A balanced dataset is very important for creating a good training set. They aim to optimize the overall accuracy without considering the relative distribution of each class. Typically real world data are usually imbalanced and it is one of the main causes for the decrease of generalization in machine learning algorithms.

**Advantage**

The aim was to reduce the ratio gap between the majority classes with the minority class. The proposed method is found to be useful for such datasets where the class labels are not certain and can also help to overcome the class imbalance problem of clinical datasets and also for other data domains.

**Disadvantage**

The outcome labels of most of the clinical datasets are not consistent with the underlying data. The conventional over-sampling and under-sampling technique may not always be appropriate for such datasets.

**Title:** A survey on cloud computing security

**Year:** 2010.

**Author:** R. Kanday

**Methodology**

This survey paper provides a general overview on Cloud Computing. The topics that are discussed include characteristics, deployment and service models as well drawbacks.

**Advantage**

The major part of countermeasures focuses on Intrusion Detection Systems. Moving towards Mobile Cloud Computing and Internet of Things, this survey paper gives a general explanation on the applications and potential that comes with the integration of Cloud Computing with any device that has Internet connectivity as well as the challenges that are before it.

**Disadvantage**

Several security issues and countermeasures are also discussed to show the major issues and obstacles that Cloud Computing faces as it is being implemented further.

**Title**: Data Mining: Practical Machine Learning Tool and Technique with Java Implementation

**Year**: 2000.

**Author**: Ian H. Witten and Eibe Frank

**Methodology**

The convergence of computing and communication has produced a society that feeds on information. Yet most of the information is in its raw form: data. If data is characterized as recorded facts, then information is the set of patterns, or expectations, that underlie the data. There is a huge amount of information locked up in databases—information that is potentially important but has not yet been discovered or articulated. Our mission is to bring it forth. The weather data (Tables 1.2 and 1.3) presents a set of days together with a decision for each as to whether to play the game or not.

**Advantage**

In these cases the output took the form of decision trees and classification rules, which are basic knowledge representation styles that many machine learning methods used.

**Disadvantage**

The weather problem is a tiny dataset that we will use repeatedly to illustrate machine learning methods.

**SYSTEM REQUIREMENTS**

**Software Requirements**

* O/S : Windows XP.
* Language : Java.
* IDE : NetBeans 8.2
* Data Base : MySQL

**Hardware Requirements**

* System : Pentium IV 2.4 GHz
* Hard Disk : 160 GB
* Monitor : 15 VGA color
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 2GB

**SOFTWARE DESCRIPTION**

**Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the [Python Package Index](http://pypi.python.org/pypi).

**FEASIBILITY STUDY**

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely.

**•** Economic Feasibility

• Technical Feasibility

• Behavioural Feasibility

**Economic Feasibility**

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

**Technical Feasibility**

This study centre around the system’s department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

**Behavioural Feasibility**

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user’s request, instead of getting a report, which doesn’t contain much detail.

**TESTING OF PRODUCT**

**Testing of Product**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system.  System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved.  The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test.  A series of tests are performed before the system is ready for the user acceptance testing.  Any engineered product can be tested in one of the following ways.  Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational.  Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**UNIT TESTING**

Unit testing is the testing of each module and the integration of the overall system is done.  Unit testing becomes verification efforts on the smallest unit of software design in the module.  This is also known as ‘module testing’.  The modules of the system are tested separately.  This testing is carried out during the programming itself.  In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module.  There are some validation checks for the fields.  For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included.  It is very easy to find error and debug the system.

**INTEGRATION TESTING**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function.  Integrated testing is systematic testing that can be done with sample data.  The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

1. Top-down integration testing.
2. Bottom-up integration testing.

**WHITE BOX TESTING**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases.  Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

**BLACK BOX TESTING**

* Black box testing is done to find incorrect or missing function
* Interface error
* Errors in external database access
* Performance errors
* Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’.  It tests the external behaviour of the system.  Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**VALIDATION TESTING**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

# **USER ACCEPTANCE TESTING**

User acceptance of the system is the key factor for the success of the system.  The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

# **OUTPUT TESTING**

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format.  The output displayed or generated by the system under consideration.  Here the output format is considered in two ways.  One is screen and the other is printed format.  The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.  For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**System Implementation**

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

* The active user must be aware of the benefits of using the system
* Their confidence in the software built up
* Proper guidance is impaired to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

**User Training**

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important. Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

**Training on the Application Software**

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. This training may be different across different user groups and across different levels of hierarchy.

**Operational Documentation**

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

**System Maintenance**

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affect a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only thru proper system maintenance procedures, the system can be adapted to cope up with these changes. Software maintenance is of course, far more than “finding mistakes”.

**Corrective Maintenance**

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called Corrective Maintenance.

**Adaptive Maintenance**

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore Adaptive maintenance termed as an activity that modifies software properly with a changing environment is both necessary & common place.

**Perceptive Maintenance**

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, Perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

**Preventive Maintenance**

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

**Types of Software Testing**

**Ad-hoc testing**

This type of software testing is very informal and unstructured and can be performed by any stakeholder with no reference to any test case or test design documents. The person performing Ad-hoc testing has a good understanding of the domain and workflows of the application to try to find defects and break the software. Ad-hoc testing is intended to find defects that were not found by existing test cases.

**Acceptance Testing**

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge.

**Accessibility Testing**

In accessibility testing, the aim of the testing is to determine if the contents of the website can be easily accessed by disable people. Various checks such as colour and contrast (for colour blind people), font size for visually impaired, clear and concise text that is easy to read and understand.

**Agile Testing**

Agile Testing is a type of software testing that accommodates agile software development approach and practices. In an Agile development environment, testing is an integral part of software development and is done along with coding. Agile testing allows incremental and iterative coding and testing.

**API Testing**

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

**Automated testing**

This is a testing approach that makes use of testing tools and/or programming to run the test cases using software or custom developed test utilities. Most of the automated tools provided capture and playback facility, however there are tools that require writing extensive scripting or programming to automate test cases.

**All Pairs testing**

Also known as Pair wise testing, is a black box testing approach and a testing method where in for each input is tested in pairs of inputs, which helps to test software works as expected with all possible input combinations.

**Beta Testing**

This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.

**Black Box testing**

Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing method relies on testing software with various inputs and validating results against expected output.

**Backward Compatibility Testing**

Type of software testing performed to check newer version of the software can work successfully installed over previous version of the software and newer version of the software works as fine with table structure, data structures, files that were created by previous version of the software.

**Boundary Value Testing (BVT)**

Boundary Value Testing is a testing technique that is based on concept “error aggregates at boundaries”. In this testing technique, testing is done extensively to check for defects at boundary conditions. If a field accepts value 1 to 100 then testing is done for values 0, 1, 2, 99, 100 and 101.

**Big Bang Integration testing**

This is one of the integration testing approaches, in Big Bang integration testing all or all most all of the modules are developed and then coupled together.

**Bottom up Integration testing**

Bottom up integration testing is an integration testing approach where in testing starts with smaller pieces or sub systems of the software till all the way up covering entire software system. Bottom up integration testing begins with smaller portion of the software and eventually scale up in terms of size, complexity and completeness.

**Branch Testing**

Is a white box testing method for designing test cases to test code for every branching condition? Branch testing method is applied during unit testing.

**Browser compatibility Testing**

It is one of the sub types of testing of compatibility testing performed by testing team. Browser compatibility testing is performed for web applications with combination of different browsers and operating systems.

**Compatibility testing**

Compatibility testing is one of the test types performed by testing team. Compatibility testing checks if the software can be run on different hardware, operating system, bandwidth, databases, web servers, application servers, hardware peripherals, emulators, different configuration, processor, different browsers and different versions of the browsers etc.

**Component Testing**

This type of software testing is performed by developers. Component testing is carried out after completing unit testing. Component testing involves testing a group of units as code together as a whole rather than testing individual functions, methods.

**Condition Coverage Testing**

Condition coverage testing is a testing technique used during unit testing, where in developer tests for all the condition statements like if, if else, case etc., in the code being unit tested.

**Dynamic Testing**

Testing can be performed as Static Testing and Dynamic testing, Dynamic testing is a testing approach where-in testing can be done only by executing code or software are classified as Dynamic Testing. Unit testing, Functional testing, regression testing, performance testing etc.

**Decision Coverage Testing**

Is a testing technique that is used in Unit testing, objective of decision coverage testing is to expertise and validate each and every decisions made in the code e.g. if, if else, case statements.

**End-to-end Testing**

End to end testing is performed by testing team, focus of end to end testing is to test end to end flows e.g. right from order creation till reporting or order creation till item return etc. and checking. End to end testing is usually focused mimicking real life scenarios and usage. End to end testing involves testing information flow across applications.

**Exploratory Testing**

Exploratory testing is an informal type of testing conducted to learn the software at the same time looking for errors or application behaviour that seems non-obvious. Exploratory testing is usually done by testers but can be done by other stake holders as well like Business Analysts, developers, end users etc. who are interested in learning functions of the software and at the same time looking for errors or behaviour is seems non-obvious.

**Equivalence Partitioning**

Equivalence partitioning is also known as Equivalence Class Partitioning is a software testing technique and not a type of testing by itself. Equivalence partitioning technique is used in black box and grey box testing types. Equivalence partitioning classifies test data into Equivalence classes as positive Equivalence classes and negative Equivalence classes, such classification ensures both positive and negative conditions are tested.

**Functional Testing**

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing software against design document, Use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing.

**Fuzz Testing**

Fuzz testing or fuzzing is a software testing technique that involves testing with unexpected or random inputs. Software is monitored for failures or error messages that are presented due to the input errors.

**GUI (Graphical User Interface) testing**

This type of software testing is aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mock-ups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mock-ups, detailed design documents and functional requirements. Most of the functional test automation tools work on GUI capture and playback capabilities. This makes script recording faster at the same time increases the effort on script maintenance.

**Glass box Testing**

Glass box testing is another name for White box testing. Glass box testing is a testing method that involves testing individual statements, functions etc., Unit testing is one of the Glass box testing methods.

**Gorilla Testing**

This type of software testing is done by software testing team, has a scary name though? Objective of Gorilla Testing is to exercise one or few functionality thoroughly or exhaustively by having multiple people test the same functionality.

**Happy Path Testing**

Also known as Golden path testing, this type of testing focuses on selective execution of tests that do not exercise the software for negative or error conditions.

**Integration Testing**

Integration testing also known as met in short, in one of the important types of software testing. Once the individual units or components are tested by developers as working then testing team will run tests that will test the connectivity among these units/component or multiple units/components. There are different approaches for Integration testing namely, Top-down integration testing, Bottom-up integration testing and a combination of these two known as Sand witch testing.

**Interface Testing**

Software provides support for one or more interfaces like “Graphical user interface”, “Command Line Interface” or “Application programming interface” to interact with its users or other software. Interfaces serves as medium for software to accept input from user and provide result. Approach for interface testing depends on the type of the interface being testing like GUI or API or CLI.

**Internationalization Testing**

Internationalization testing is a type of testing that is performed by software testing team to check the extent to which software can support Internationalization i.e., usage of different languages, different character sets, double byte characters etc., For e.g.: Gmail, is a web application that is used by people all over work with different languages, single by or multi byte character sets.

**Keyword-driven Testing**

Keyword driver testing is more of an automated software testing approach than a type of testing itself. Keyword driven testing is known as action driven testing or table driven testing.

**Load Testing**

Load testing is a type of non-functional testing; load testing is done to check the behaviour of the software under normal and over peak load conditions. Load testing is usually performed using automated testing tools. Load testing intends to find bottlenecks or issues that prevent software from performing as intended at its peak workloads.

**Localization Testing**

Localization testing a type of software testing performed by software testers, in this type of testing, software is expected to adapt to a particular locale, it should support a particular locale/language in terms of display, accepting input in that particular locale, display, font, date time, currency etc., related to a particular locale. For e.g. many web applications allow choice of locale like English, French, German or Japanese. So once locale is defined or set in the configuration of software, software is expected to work as expected with a set language/locale.

**Negative Testing**

This type of software testing approach, which calls out the “attitude to break”, these are functional and non-functional tests that are intended to break the software by entering incorrect data like incorrect date, time or string or upload binary file when text files supposed to be upload or enter huge text string for input fields etc. It is also a positive test for an error condition.

**Non-functional testing**

Software are built to fulfil functional and non-functional requirements, non-functional requirements like performance, usability, localization etc., There are many types of testing like compatibility testing, compliance testing, localization testing, usability testing, volume testing etc., that are carried out for checking non-functional requirements.

**Pair Testing**

**It** is a software testing technique that can be done by software testers, developers or Business analysts (BA). As the name suggests, two people are paired together, one to test and other to monitor and record test results. Pair testing can also be performed in combination of tester-developer, tester-business analyst or developer-business analyst combination. Combining testers and developers in pair testing helps to detect defects faster, identify root cause, fix and test the fix.

**Performance Testing**

**It** is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability. Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements. Performance testing checks how well software works in anticipated and peak workloads. There are different variations or sub types of performance like load testing, stress testing, volume testing, soak testing and configuration testing.

**Penetration Testing**

**It** is a type of security testing, also known as pen test in short. Penetration testing is done to tests how secure software and its environments (Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a human/hacker or malicious programs. Pen test uses methods to forcibly intrude (by brute force attack) or by using a weakness (vulnerability) to gain access to a software or data or hardware with an intent to expose ways to steal, manipulate or corrupt data, software files or configuration. Penetration Testing is a way of ethical hacking, an experienced Penetration tester will use the same methods and tools that a hacker would use but the intention of Penetration tester is to identify vulnerability and get them fixed before a real hacker or malicious program exploits it.

**Regression Testing**

**It** is a type of software testing that is carried out by software testers as functional regression tests and developers as Unit regression tests. Objective of regression tests are to find defects that got introduced to defect fix (is) or introduction of new feature(s). Regression tests are ideal candidate for automation.

**Retesting**

**It** is a type of retesting that is carried out by software testers as a part of defect fix verification. For e.g. a tester is verifying a defect fix and let us say that there are 3 test cases failed due to this defect. Once tester verifies defect fix as resolved, test will retest or test the same functionality again by executing the test cases that were failed earlier.

**Risk based Testing**

It is a type of software testing and a different approach towards testing a software. In Risk based testing, requirements and functionality of software to be tested are prioritized as Critical, High, Medium and low. In this approach, all critical and high priority tests are tested and them followed by Medium. Low priority or low risk functionality are tested at the end or may not base on the time available for testing.

**Smoke testing**

**It** is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e., major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find “show stopper” defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as build verification test.

**Security Testing**

**It** is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.

**Sanity Testing**

**It** is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented.

**Scalability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Scalability”. Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc., It is not necessary that software’s performance increases with increase in hardware configuration, scalability tests helps to find out how much more workload the software can support with expanding user base, transactions, data storage etc.,

**Stability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Stability”. Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc…

**Static Testing** is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc. Static testing usually has check lists against which deliverables are evaluated. Static testing can be applied for requirements, designs, and test cases by using approaches like reviews or walkthroughs.

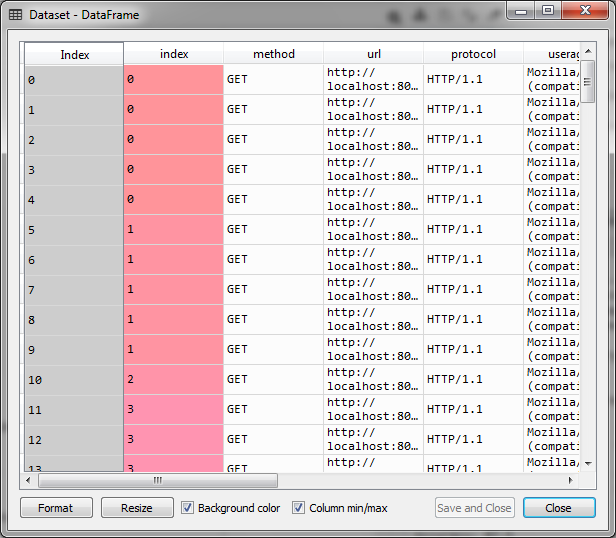
**Stress Testing** is a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behaviour of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.

**CONCLUSION**

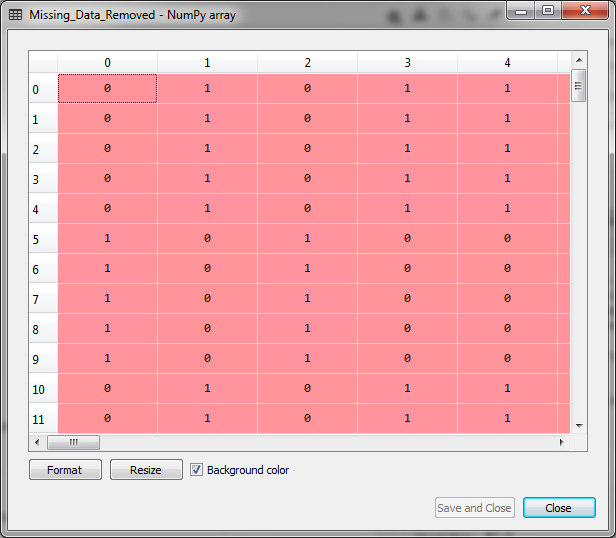
We reviewed several inﬂuential algorithms for intrusion detection based on various machine learning techniques. Characteristics of ML techniques makes it possible to design IDS that have high detection rates and low false positive rates while the system quickly adapts itself to changing malicious behaviours. We divided these algorithms into two types of ML-based schemes: Artiﬁcial Intelligence (AI) and Computational Intelligence (CI). Although these two categories of algorithms share many similarities, several features of CI-based techniques, such as adaptation, fault tolerance, high computational speed and error resilience in the face of noisy information, conform the requirement of building eﬃcient intrusion detection systems.

**SCREENSHOTS:**

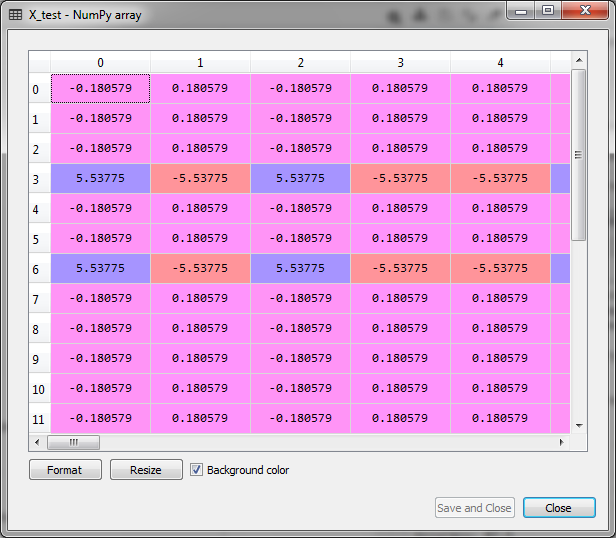
**Dataset:**



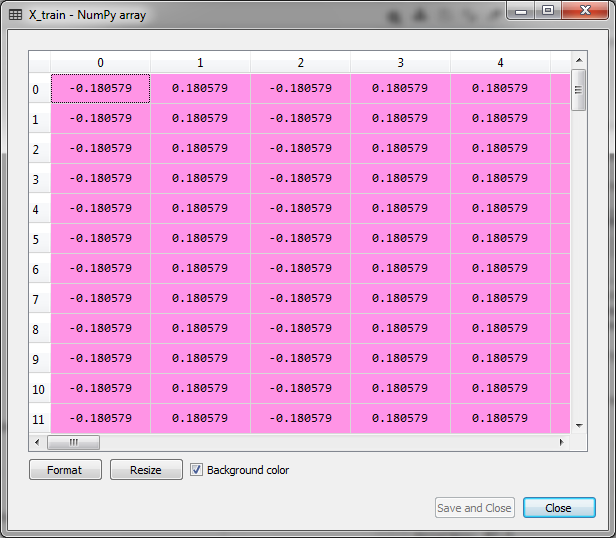
**Missing Data Removed:**



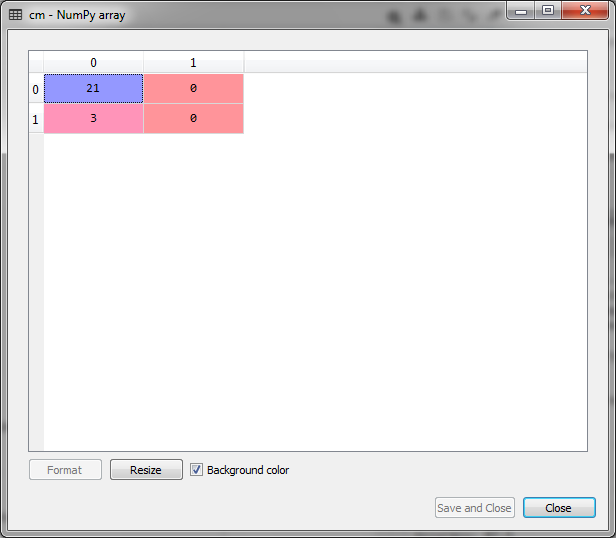
**X\_test:**



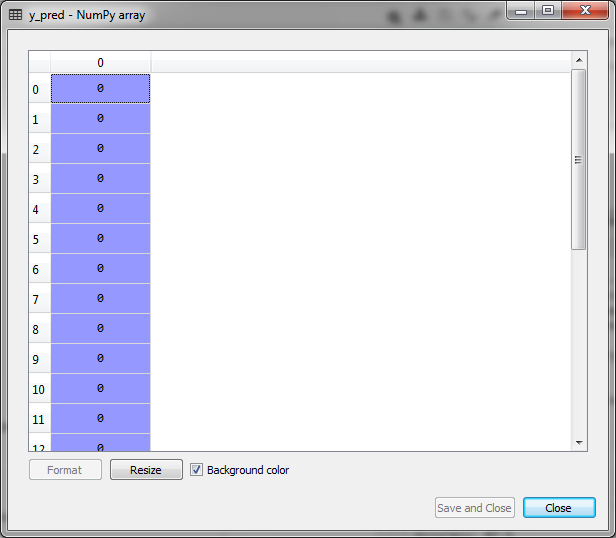
**X\_train:**



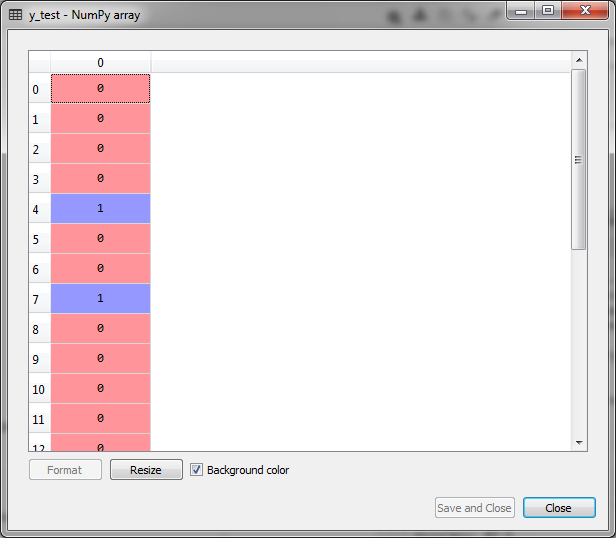
**Confusion Matrix:**



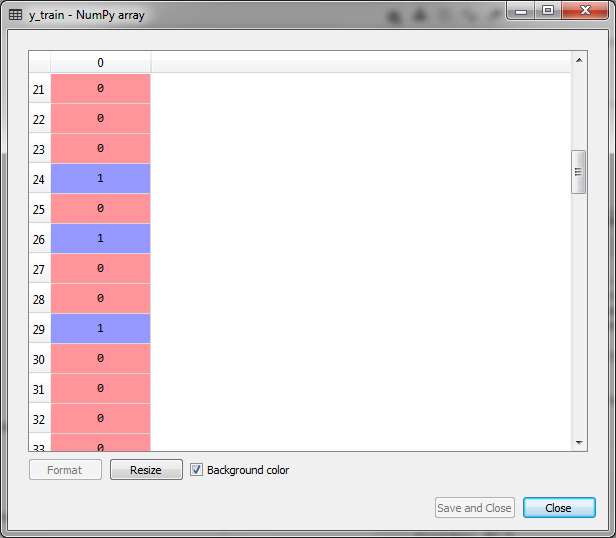
**Y\_pred:**



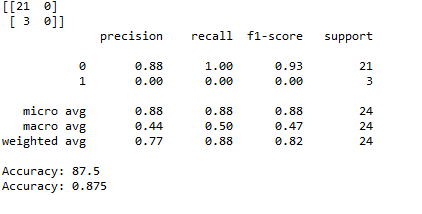
**Y\_test:**



**Y\_train:**



**Result:**



**REFERENCES**

1. S. Yinbiao and K. Lee, “Internet of Things: Wireless Sensor Networks Executive summary,” 2014. [5] F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci
2. “A survey on sensor networks,” IEEE Commun. Mag., vol. 40, no. 8, pp. 102–105, 2002. [6]
3. X. Chen, K. Makki, K. Yen, and N. Pissinou, “Sensor network security: a survey,” IEEE Commun. Surv. Tutorials, vol. 11, no. 2, pp. 52–73, 2009. [7]
4. A.-S. K. Pathan, H.-W. Lee, and C. S. Hong, “Security in wireless sensor networks: issues and challenges,” 2006 8th Int. Conf. Adv. Commun. Technol., vol. 2, p. 6 pp.-pp.1048, 2006. [8]
5. P. Yi, Y. Jiang, Y. Zhong, and S. Zhang, “Distributed Intrusion Detection for Mobile Ad Hoc Networks,” 2005 Symp. Appl. Internet Work. (SAINT 2005 Work., pp. 94–97, 2005. [9]
6. H. Sedjelmaci and M. Feham, “Novel Hybrid Intrusion Detection System for Clustered Wireless Sensor Network,” Int. J. Netw. Secur. Its Appl. (IJNSA), Vol.3, No.4, July 2011, vol. 3, no. 4, pp. 1–14, 2011. [10]
7. L. Khan, M. Awad, and B. Thuraisingham, “A new intrusion detection system using support vector machines and hierarchical clustering,” VLDB J., vol. 16, no. 4, pp. 507–521, 2007. [11]
8. S. K. Sahu, S. Sarangi, and S. K. Jena, “A detail analysis on intrusion detection datasets,” Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC 2014, pp. 1348–1353, 2014. [12]
9. O. Can, C. Turguner, and O. K. Sahingoz, “A Neural Network Based Intrusion Detection System For Wireless Sensor Networks,” Signal Process. Commun. Appl. Conf. (SIU), 2015 23th, pp. 2302–2305, 2015. [13]
10. F. Lu and L. Wang, “Intrusion Detection System Based on Integration of Neural Network for Wireless Sensor Network,” J. Softw. Eng. 2014. [14]
11. Y. Y. Li and L. E. Parker, “Intruder detection using a wireless sensor network with an intelligent mobile robot response,” Southeastcon, 2008. IEEE, pp. 37–42, 2008. [15] A. Kulakov and D. Davcev, “Tracking of unusual events in wireless sensor networks based on artificial neural-networks algorithms,” Inf. Technol. Coding Comput. 2005. ITCC 2005. Int. Conf., pp. 534–539, 2005. [16]
12. M. Panda, “Security Threats at Each Layer of Wireless Sensor Networks,” Int. J. Adv. Res. Comput.Sci. Softw. Eng., vol. 3, no. 11, pp. 61–67, 2013. [17]
13. Karlof and D. Wagner, “Secure routing in wireless sensor networks: attacks and countermeasures,” Proc. First IEEE Int. Work. Sens. Netw. Protoc. Appl. 2003., pp. 113–127, 2003.