A MACHINE LEARNING BASED SECURITY MODEL FOR CLOUD SERVICES USING SVM AND J48 ALGORITHM

A PROJECT REPORT

Submitted by

AKASH V **(611820104002)**

GOPI M **(611820104303)**

SRI PRADESH M **(611820104306)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

P.S.V. COLLEGE OF ENGINEERING AND TECHNOLOGY

(Accredited by the NAAC with 'A' Grade)

(An ISO 9001:2015 Certified Institution)

KRISHNAGIRI

ANNA UNIVERSITY::CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**A MACHINE LEARNING BASED SECURITY MODEL FOR CLOUD SERVICES USING SVM AND J48 ALGORITHM**" is the bonafide work of "**AKASH V (611820104002), GOPI M (611820104303)** and **SRIPRADESH M (611820104306)**" who carried out the project work under my supervision.

SIGNATURE

Dr. S. CHANDRA SEKARAN., M.E., Ph.D.,

Head Of the Department/ Professor

Department of Computer Science and Engineering,
P.S.V. College Of Engineering & Technology,
Krishnagiri-635 108.

SIGNATURE

Prof. R. MONISHA, M.E.,

Supervisor/Assistant Professor

Department of Computer Science and Engineering,
P.S.V. College Of Engineering & Technology,
Krishnagiri-635 108.

Submitted for the project Viva - Voce held on ……..…….at P.S.V College of Engineering and Technology, Krishnagiri.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

At the very outset, we wish to express our sincere thanks to all those who were involved in the completion of this project.

Our most sincere salutations go to Anna University, Chennai that gave us an opportunity to have sound base of Computer Science and Engineering.

We acknowledge our profound intellect of gratitude to our beloved chairman **Dr. P. SELVAM, M.A., B.Ed., M.Phil., Ph.D.,** for his support and coordination.

We express our gratitude to our honourable principal **Dr. P. LAWRENCE, M.E., Ph.D.,** for his timely advice and his encouragement.

We offer our sincere thanks to **Prof. Dr. S. CHANDRA SEKARAN, M.E., Ph.D.,** Head of the Department of Computer Science for giving this prospect and his full support.

We consider it as a great privilege to place a record of our deep sense of gratitude to our Internal Guide **Prof. R. MONISHA, M.E**., Assistant Professor, and Department of Computer Science for being the great inspiration to us.

We also express our thanks to our Department Faculty members, our parents and our beloved friends.

SRI PRADESH M

AKASH V

GOPI M

ABSTRACT

Cloud computing is a computing paradigm that provides OnDemand, scalable, measured and secure services to the end users over the internet. cloud computing paradigm finds a very large set of use cases, this systematic review investigates the efficacy of machine learning applications, particularly utilizing Support Vector Machines (SVM) and the J48 algorithm, in enhancing cloud security. Conducting a comprehensive literature search, we identify and analyze studies focusing on threat detection and mitigation in cloud environments. Providing an overview of machine learning's role in addressing security challenges, we delve into the principles of SVM and J48, summarizing and critically evaluating case studies. Comparative analyses reveal the strengths and weaknesses of SVM and J48, considering performance metrics. The most applied metric is true positive rate and least used is training time. Lastly, from 20 datasets found, KDD and KDD CUP dataset are the most used.

Keywords: Cloud Computing, Machine Learning, Intrusion Detection System, Datasets, Supervised Machine Learning, Unsupervised Machine Learning, Reinforcement Learning, Deep Neural Network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

SVM	-	Support Vector Machines
ICT	-	Information and Communication Technology
MNCs	-	multinational corporations
PRISMA	-	Systematic Reviews and Meta-Analysis
SAM	-	Security Assurance Model
CSSRs	-	critical software security risks

CHAPTER 1
INTRODUCTION

Cloud computing has emerged as a transformative paradigm in information technology, offering scalable and on-demand access to a shared pool of resources over the Internet. As organizations increasingly migrate their critical data and applications to the cloud, ensuring the security of these assets becomes paramount. The dynamic nature of cloud computing introduces a unique set of security challenges, ranging from data breaches to unauthorized access, prompting the need for a comprehensive understanding of threats and effective mitigation strategies.

This systematic literature review delves into the vast body of research on cloud computing security, aiming to provide a synthesized overview of the prevalent threats and the evolving landscape of mitigation strategies. By examining the existing literature, this review seeks to contribute valuable insights to the ongoing discourse surrounding cloud security, aiding practitioners, researchers, and policymakers in developing robust frameworks to safeguard the integrity, confidentiality, and availability of data in cloud environments.

1.1 CLOUD COMPUTING

In the rapidly evolving landscape of information technology, Cloud Computing stands out as a revolutionary paradigm that has reshaped the way businesses and individuals access, store, and manage data and applications. Cloud Computing transcends traditional computing models by providing on-demand access to a shared pool of computing resources, such as servers, storage, and applications, over the Internet.

This transformative approach offers unprecedented scalability, flexibility, and cost-efficiency, allowing organizations to dynamically scale their infrastructure based on demand. The essence of Cloud Computing lies in its

ability to deliver computing services as utilities, enabling users to leverage powerful resources without the need for significant upfront investments in hardware and infrastructure. As the adoption of Cloud Computing continues to proliferate across industries, understanding its principles, benefits, and challenges becomes crucial for businesses aiming to harness its full potential while addressing pertinent concerns such as security, privacy, and compliance. This introduction sets the stage for exploring the multifaceted realm of Cloud Computing, emphasizing its transformative impact on the modern IT landscape.

1.2 SECURITY

Security, in the realm of information technology, is an indispensable facet that underpins the trust, integrity, and resilience of digital systems. As technology has become deeply embedded in every aspect of our personal and professional lives, the importance of safeguarding data, networks, and systems against a myriad of threats has become more pronounced than ever.

Security encompasses a comprehensive set of measures, protocols, and practices designed to protect against unauthorized access, data breaches, and disruptions to ensure the confidentiality, integrity, and availability of information. In an era marked by rapid digitization and interconnectedness, the landscape of security is dynamic, responding to emerging challenges such as cyber threats, data vulnerabilities, and privacy concerns.

This introduction lays the foundation for exploring the critical role of security in the digital age, acknowledging its continuous evolution in response to a constantly changing technological landscape. As we delve into the complexities of security, it becomes apparent that a proactive and adaptive approach is essential to mitigate risks and fortify the foundations of our increasingly interconnected and data-driven world.

1.3 THREATS

In the vast and interconnected digital landscape, the concept of threats looms as a dynamic and pervasive force that poses challenges to the security and stability of information systems. Threats encompass a diverse range of malicious activities and events that have the potential to exploit vulnerabilities, compromise data integrity, and disrupt the normal functioning of technological infrastructures. These threats can originate from a variety of sources, including malicious actors, software vulnerabilities, or inadvertent human errors.

Understanding and mitigating threats is a critical aspect of information security, as organizations and individuals strive to protect their digital assets from unauthorized access, data breaches, and service disruptions. This introduction sets the stage for exploring the multifaceted nature of threats in the digital realm, emphasizing the importance of proactive measures and strategic defenses to navigate the evolving landscape of cybersecurity and safeguard against potential risks

1.4 MITIGATION STRATEGIES

Mitigation strategies stand as the proactive and reactive measures employed to address and reduce the impact of potential risks and threats to information systems. In the ever-evolving landscape of cybersecurity, where vulnerabilities and challenges persist, effective mitigation strategies play a pivotal role in fortifying the resilience of digital infrastructures. These strategies encompass a spectrum of approaches, including technological solutions, policy frameworks, and awareness programs, designed to prevent, detect, and respond to security incidents.

1.5 DATA PROTECTION

In the digital age, where information serves as a cornerstone of business operations, personal interactions, and societal progress, the concept of data protection takes center stage as a critical imperative. Data, often considered a valuable asset, encompasses a wide array of information, ranging from personal identifiers to organizational secrets, all of which require safeguarding against unauthorized access, misuse, or compromise.

Data protection, therefore, encapsulates the set of principles, policies, and technologies implemented to ensure the confidentiality, integrity, and availability of information. As technology advances and data volumes soar, the challenges surrounding data protection intensify, with concerns ranging from cyber threats and data breaches to regulatory compliance and ethical considerations.

This introduction sets the context for delving into the intricate landscape of data protection, highlighting its pivotal role in preserving privacy, fostering trust, and navigating the delicate balance between innovation and the ethical use of information in our interconnected and data-driven world.

1.6 OBJECTIVES

Enhance cloud-based intrusion detection accuracy and efficiency through the application of advanced machine learning algorithms. Optimize feature selection and data pre-processing to improve model performance and reduce computational complexity. Achieve threat detection and reduction in false positives/negatives in cloud environments using sophisticated classifiers like AODE. Foster model scalability and adaptability to evolving cybersecurity threats through continuous learning and hybrid modeling approaches.

CHAPTER 2

LITERATURE SURVEY

2.1 THREMA: ONTOLOGY-BASED AUTOMATED THREAT MODELING FOR ICT INFRASTRUCTURES

FABIO DE ROSA et.al., has proposed in this paper Threat modeling is a critical process in cybersecurity that empowers defenders to systematically identify and analyze potential threats to a target system. To address these challenges, the paper introduces the ThreMA approach, which focuses on automating threat modeling for Information and Communication Technology (ICT) infrastructures. This standardized vocabulary aims to create a common language that all stakeholders involved in threat modeling can understand and utilize.

ThreMA addresses this by providing a comprehensive threat catalog and a set of inference rules. These rules play a crucial role in automating the identification of potential threats within the ICT infrastructure, enhancing the efficiency and accuracy of the threat modeling process. This real-world validation demonstrates the applicability and effectiveness of ThreMA in addressing the unique challenges associated with threat modeling in complex ICT environments.

Disadvantages: Dependence on Accurate Data Input

The effectiveness of an ontology-based threat modeling system heavily relies on the quality and accuracy of the input data. Inaccurate, incomplete, or biased data can lead to faulty conclusions and inadequate security measures.

2.2 CYBER SECURITY MATURITY ASSESSMENT FRAMEWORK FOR TECHNOLOGY STARTUPS

MOHAMED NOORDIN YUSUFF MARICAN et.al., has proposed in this paper The increasing significance of cybersecurity across diverse industries, coupled with the rising frequency of cyber-attacks, has compelled firms of all sizes to prioritize robust security measures. Among these, technology startups stand out as particularly vulnerable due to inherent challenges such as limited human capital and financial resources.

The absence of comprehensive cybersecurity measures in startups arises from difficulties in quantifying cyber risks and allocating appropriate investments. Given that technology startups often serve as suppliers and vendors to large organizations, including multinational corporations (MNCs), government entities, and financial institutions, they may inadvertently become potential entry points for malicious hackers. Recognizing this vulnerability, the study focused on technology startups and conducted a systematic literature review on cybersecurity maturity assessment frameworks.

While certain frameworks exhibited similarities in assessing cybersecurity maturity levels, none emerged as a singular, comprehensive tool applicable to the nuanced context of technology startups. Furthermore, the research highlighted the dearth of studies addressing the quantification of the return on cybersecurity investments within an end-to-end maturity assessment framework for technology startups.

Disadvantages:

There is a risk that the focus of maturity assessments could shift from improving security to merely achieving a higher maturity score. Startups might end up prioritizing compliance with the framework over addressing practical security threats that have a more immediate or damaging impact.

2.3 PDGRAPH: A LARGE-SCALE EMPIRICAL STUDY ON PROJECT DEPENDENCY OF SECURITY VULNERABILITIES

Qiang Li et.al., has proposed in this paper The widespread practice of reusing libraries in software development has undeniably enhanced development efficiency and software quality. However, the security implications arising from vulnerabilities in reused libraries, particularly those propagated through project dependencies, have not been thoroughly investigated.

This paper addresses this gap by presenting the first large-scale empirical study of project dependencies in relation to security vulnerabilities. To conduct this study, the authors developed PD Graph, an innovative approach designed to analyze publicly known security vulnerabilities across a vast array of project dependencies. PD Graph introduces a novel perspective for assessing security risks in real-world scenarios. The study encompasses an extensive collection of 337,415 projects and 1,385,338 dependency relationships.

PD Graph constructs a project dependency graph, where each node represents a project, and each edge signifies a dependency relationship. The efficacy of PD Graph is validated through a series of experiments, and its features are characterized for security analysis. The empirical findings from the study are illuminating. Among the analysed projects, 1,014 were found to have publicly disclosed vulnerabilities, with over 67,806 projects directly dependent on them. Alarmingly, 42,441 projects still exhibit 67,581 insecure dependency relationships, indicating their reliance on vulnerable versions of reused libraries despite the public awareness of these vulnerabilities.

Disadvantages: Potential for Overlooking Indirect Dependencies:

While direct dependencies are generally easier to track, indirect dependencies (dependencies of dependencies) might be overlooked or underestimated in their impact, potentially leading to incomplete analyses of vulnerability propagation

2.4 SECURITY ASSURANCE MODEL OF SOFTWARE DEVELOPMENT FOR GLOBAL SOFTWARE DEVELOPMENT VENDORS

RAFIQ AHMAD KHAN et.al., has proposed in this paper In response to the escalating number and impact of security attacks in recent years, there is an increasing need for innovative software development models that prioritize security as a default feature. This article not only acknowledges the urgency of this requirement but goes a step further by reviewing existing security software models widely employed in the industry.

Recognizing the limitations of current models, the article proposes a novel Security Assurance Model (SAM) for Software Development, specifically designed to be adaptable to contemporary scenarios, with a particular emphasis on global software development (GSD) vendor companies.

The SAM of Software Development is the culmination of an extensive study that involved the examination of 11 well-established development models. The development of SAM was informed by insights gathered through a systematic literature review (SLR) and a comprehensive questionnaire survey. This model comprises seven security assurance levels, each addressing critical aspects of software development: Governance and Security Threat Analysis, Secure Requirement Analysis, Secure Design, Secure Coding, Secure Testing and Review, Secure Deployment, and Security Improvement. A key strength of SAM lies in its detailed consideration of 46 critical software security risks (CSSRs) and the identification of 388 practices aimed at mitigating these risks across various stages of the software development life cycle

Disadvantages:

Global software development often involves multiple teams spread across different geographical locations and time zones. Coordinating these teams to adhere to a uniform security assurance protocol adds a layer of complexity to project management.

2.5 ANALYSIS OF SYSTEMS SECURITY ENGINEERING DESIGN PRINCIPLES FOR THE DEVELOPMENT OF SECURE AND RESILIENT SYSTEMS

PAUL M. BEACH et.al., has proposed in this paper The escalating frequency and sophistication of cyber-attacks underscore the imperative for enhanced systems security analysis and engineering, particularly in safety-critical and mission-essential systems.

The challenge is further compounded by the intricate nature of developing secure and resilient systems that must adhere to constraints of cost, schedule, and performance. The growing complexity of interrelated systems-of-systems exacerbates this engineering challenge. This paper delves into the analysis of 18 design principles outlined in the National Institute of Standards and Technology Special Publication (NIST SP) 800-160 Volume 1, exploring their applicability in crafting secure and resilient systems.

The focal point of this work is to unravel how these design principles can be consistently and effectively applied to address the security and resiliency requirements defined by stakeholders. Utilizing the Design Structure Matrix (DSM) analysis, the paper scrutinizes the 18 design principles presented in NIST SP 800-160 Vol. 1, specifically in Appendix F. The analysis dissects the intra- and inter-dependencies of these principles, shedding light on how they can be leveraged to develop complex cyber-physical systems that boast security, trustworthiness, and resilience.

Disadvantages:

Adhering to security engineering principles usually extends the timeline of system development. The additional steps required for risk assessments, security testing, and compliance checks can delay the deployment of critical systems.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Security is one of the most critical aspects of software quality. Software security refers to the process of creating and developing software that assures the integrity, confidentiality, and availability of its code, data, and services. Software development organizations treat security as an afterthought issue, and as a result, they continue to face security threats. Incorporating security at any level of the Software Development Life Cycle (SDLC) has become an urgent requirement.

Several methodologies, strategies, and models have been proposed and developed to address software security, but only a few of them give reliable evidence for creating secure software applications. Software security issues, on the other hand, have not been adequately addressed, and integrating security procedures into the SDLC remains a challenge. The major purpose of this paper is to learn about software security risks and practices so that secure software development methods can be better designed.

A systematic literature review (SLR) was performed to classify important studies to achieve this goal. Based on the inclusion, exclusion, and quality assessment criteria, a total of 121 studies were chosen. This study identified 145 security risks and 424 best practices that help software development organizations to manage the security in each phase of the SDLC. To pursue secure SDLC, this study prescribed different security activities, which should be followed in each phase of the SDLC.

3.1.1 Drawbacks

- Implementing and maintaining software security measures can be expensive. This is especially true for large and complex software systems.
- Software security is a complex topic, and it can be difficult for organizations to keep up with the latest threats and best practices.
- No software security measure is perfect. Even if an organization implements all of the best practices, there is still a risk that vulnerabilities will be introduced and exploited.
- Compatibility issues: Some software security measures can be incompatible with existing software or systems.

3.2 PROPOSED SYSTEM

The proposed system is a cloud-based intrusion detection framework leveraging machine learning algorithms to handle the complexities of the NSL KDD dataset. The system begins with the preprocessing module, which converts numeric attributes to nominal ones and optimizes feature selection using techniques such as Information Gain, Intrinsic Information, and Gain Ratio. These steps ensure that the input data is uniform, relevant, and conducive to effective analysis by the subsequent machine learning module.

At the heart of the system lies the Average One Dependency Estimator (AODE) classifier, renowned for its ability to discern intricate patterns and dependencies within high-dimensional datasets like NSL KDD. The AODE classifier operates alongside other machine learning algorithms, collectively trained on the dataset to detect various forms of malicious activities accurately. The system is then rigorously evaluated to gauge its accuracy, efficiency, and reliability in identifying intrusions.

3.2.1 Advantages

- Increased security posture with high-accuracy threat detection minimizing data breaches and system intrusions.
- Improved operational efficiency through automated detection and response mechanisms, reducing manual oversight.
- Enhanced adaptability to emerging threats via machine learning models that evolve with new data and attack vectors.
- Reduction in computational resources and costs by employing optimized feature selection and efficient algorithms.

3.3 FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility
- **3.3.1 TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?

- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at DB2 Database. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified.

Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.3.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?

- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

3.3.3 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economical feasibility for certain.

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS

CPU type	:	Intel core i5 processor
Clock speed	:	3.0 GHz
RAM size	:	8 GB
Hard disk capacity	:	500 GB
Keyboard type	:	Internet Keyboard
CD -drive type	:	52xmax

4.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 10
Front End	:	JAVA

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 FRONT END: JAVA

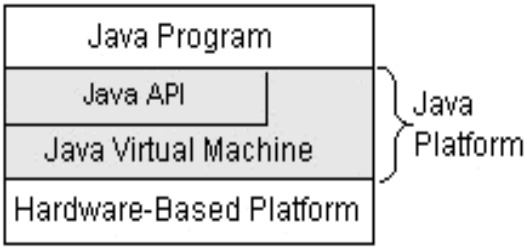
The software requirement specification is created at the end of the analysis task. The function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

FEATURES OF JAVA

Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

SOCKET OVERVIEW:

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can “plug in” to the socket and communicate.

Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User DatagramProtocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

CLIENT/SERVER:

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

RESERVED SOCKETS:

Once connected, a higher-level protocol ensues, which is dependent on which port user are using. TCP/IP reserves the lower, 1,024 ports for specific

protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

JAVA AND THE NET:

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

INETADDRESS:

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. User interacts with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

FACTORY METHODS:

The InetAddress class has no visible constructors. To create an InetAddress object, user use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are:

1. Static InetAddressgetLocalHost ( ) throws

   UnknownHostException

2. Static InetAddressgetByName (String hostName)

   throwsUnknowsHostException

3. Static InetAddress [ ] getAllByName (String hostName)

throwsUnknownHostException

INSTANCE METHODS:

The Inet Address class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used.

Boolean equals (Object other)- Returns true if this object has the same Internet address as other.

1. byte [ ] get Address ( )- Returns a byte array that represents the object's Internet address in network byte order.

2. String get Host Address ( ) - Returns a string that represents the host address associated with the Inet Address object.

3. String get Hostname ( ) - Returns a string that represents the host name associated with the Inet Address object.

4. booleanis Multicast Address ( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.

5. String to String ( ) - Returns a string that lists the host name and the IP address for convenience.

TCP/IP CLIENT SOCKETS:

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point and stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets

Socket (String hostName, intport) - Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.

Socket (InetAddressipAddress, intport) - Creates a socket using a preexistingInetAddressobject and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

- InetAddressgetInetAddress ( ) - Returns the InetAddress associated with the Socket object.
- IntgetPort ( ) - Returns the remote port to which this Socket object is connected.
- IntgetLocalPort ( ) - Returns the local port to which this Socket object is connected.

Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IO Exception if the sockets have been invalidated by a loss of connection on the Net.

Input Streamget Input Stream ( ) - Returns the InputStream associated with the invoking socket.

Output Streamget Output Stream ( ) - Returns the OutputStream associated with the invoking socket.

TCP/IP SERVER SOCKETS:

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets.

When the user create a ServerSocket, it will register itself with the system as having an interest in client connections.

- ServerSocket(int port) - Creates server socket on the specified port with a queue length of 50.
- Serversocket(int port, int maxQueue) - Creates a server socket on the specified portwith a maximum queue length of maxQueue.
- ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multihomed host, localAddress specifies the IP address to which this socket binds.
- ServerSocket has a method called accept( ) - which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

URL:

The Web is a loose collection of higher-level protocols and file formats, all unified in a web browser. One of the most important aspects of the Web is that Tim Berners-Lee devised a saleable way to locate all of the resources of the Net. The Uniform Resource Locator (URL) is used to name anything and everything reliably.

The URLprovides a reasonably intelligible form to uniquely identify or address information on the Internet. URLs are ubiquitous; every browser uses them to identify information on the Web.

CHAPTER 6
ARCHITECTURE DIAGRAMS
PROJECT DESCRIPTION

6.1 PROBLEM DEFINITION

Software security is a critical aspect of software quality, but it is also a complex and challenging topic. Software development organizations often treat security as an afterthought, which leads to security vulnerabilities in their software. These vulnerabilities can be exploited by attackers to steal data, disrupt operations, or even cause physical damage.

One of the main problems with software security is that it is difficult to integrate into all phases of the software development life cycle (SDLC). This is because security is often seen as a separate concern from functionality and performance. As a result, security vulnerabilities are often introduced into software during the development process and are not discovered until after the software is deployed.

6.2 MODULE DESCRIPTION

6.2.1 NSL KDD DATASET

This module involves utilizing the NSL KDD dataset, a refined version of the original KDD'99 dataset specifically designed to eliminate redundant records and thus provide a more realistic and challenging benchmark for evaluating intrusion detection systems. It encompasses a variety of simulated attacks and normal connections, enabling the training and testing of machine learning models to accurately identify malicious activities within network traffic.

6.2.2 NUMERIC TO NOMINAL CONVERSION

The Numeric to Nominal Conversion module is a pre-processing step critical for algorithms that perform better or require input data in categorical form.

This module converts numeric attributes to nominal (categorical) ones, facilitating more effective processing by certain machine learning models. This conversion is crucial for aligning the dataset with the requirements of specific algorithms used in the system, ensuring data uniformity and improving model accuracy.

6.2.3 FEATURE SELECTION

This module is dedicated to optimizing the selection of features (attributes) used to train the machine learning models. Feature selection is vital for enhancing model performance by eliminating irrelevant or redundant data, thus reducing complexity and improving execution efficiency.

- **Information Gain** measures the change in entropy after the segmentation of a dataset based on an attribute. It helps in identifying features that provide the most useful information for classification.
- **Intrinsic Information** evaluates the inherent information of a feature, focusing on the feature's own entropy and potential for classification, independent of its effect on the outcome.
- **Gain Ratio** extends the concept of information gain by normalizing its value, balancing the bias towards multi-valued attributes by considering the intrinsic information of the split.

6.2.4 MACHINE LEARNING

Average One Dependency Estimator (AODE) Classifier:

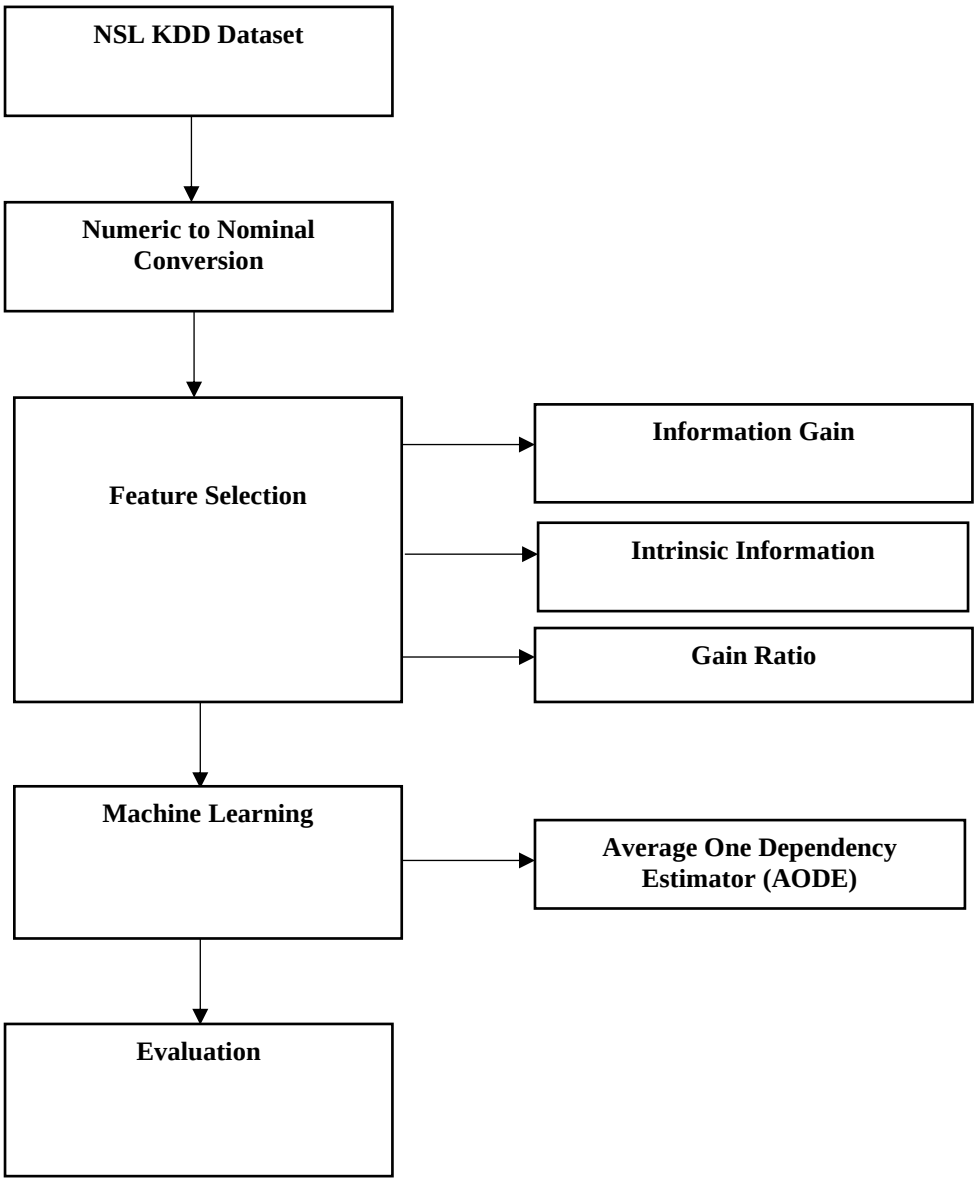
The Machine Learning module is at the core of the intrusion detection system, with the AODE classifier playing a pivotal role. The AODE classifier is a sophisticated machine learning algorithm known for its ability to handle high-dimensional data and complex relationships between attributes effectively. It operates on the principle of averaging the probabilities estimated by a set of one-

dependence estimators, significantly enhancing detection accuracy by capturing and utilizing the dependencies among features.

6.2.5 EVALUATION

The Evaluation module involves rigorously testing the system to assess its performance in accurately detecting intrusions. This includes evaluating the system's accuracy, efficiency, and reliability in identifying various types of malicious activities depicted in the NSL KDD dataset.

6.3 SYSTEM FLOW DIAGRAM



6.4 INPUT DESIGN

The input design for a cloud-based intrusion detection system utilizing machine learning involves a structured process to ensure accuracy and efficiency. Initially, data is collected from various sources within the cloud environment, including network traffic, system logs, and user activities. This data is then preprocessed to convert numeric fields to nominal where necessary, handle missing values, and standardize or normalize data to make it uniform.

Critical to this phase is the application of feature selection techniques, such as information gain and gain ratio, which refine the dataset by removing irrelevant or redundant information, thus enhancing model performance. The selected features are then formatted into a standardized input vector that machine learning models can process. This meticulous design ensures that the input data, now optimized for both relevance and format, is primed for analysis by the chosen algorithms, facilitating effective and efficient intrusion detection.

6.5 OUTPUT DESIGN

The output design for the cloud-based intrusion detection system is crafted to present the results of the machine learning analysis in an intuitive and actionable manner. Upon processing the input data, the system categorizes activities into normal or various types of malicious behavior, with each detection flagged alongside its probability score or confidence level to aid in prioritization.

This output is visualized through a user-friendly dashboard that highlights, trends in attack types, and historical data analysis for pattern recognition. Additionally, detailed reports are generated, offering insights into the nature of detected threats, impacted system components, and recommended mitigation strategies. These reports and alerts are designed to be easily interpretable by security teams, enabling swift response and remediation actions.

CHAPTER 7

SYSTEM TESTING AND IMPLEMENTATION

7.1 SYSTEM TESTING

System testing for the cloud-based intrusion detection system encompasses a comprehensive suite of tests designed to validate the effectiveness, performance, and reliability of the system under realistic scenarios. This phase involves subjecting the system to a variety of simulated attack patterns, including those not seen during the training phase, to assess its detection accuracy and response times.

Performance testing is critical, ensuring the system operates efficiently under the load of processing high volumes of network traffic and logs, maintaining low latency in threat detection and reporting. Reliability tests verify the system's uptime and consistency in threat detection over extended periods, while security tests check for vulnerabilities within the intrusion detection system itself that could be exploited.

7.2 SYSTEM IMPLEMENTATION

The implementation of the cloud-based intrusion detection system follows a phased approach, beginning with the deployment of the machine learning models and necessary preprocessing modules across the cloud infrastructure. Initially, a pilot phase targets a limited portion of the environment to fine-tune system configurations and ensure seamless integration with existing security protocols and infrastructure services.

Once validated for efficacy and efficiency, the system is gradually rolled out to cover all intended areas of the cloud environment, ensuring minimal disruption to ongoing operations.

CHAPTER 8

SYSTEM MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In this project the process can be added without affecting other parts of the system. Maintenance plays a vital role.

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment. It has been seen that there are always some errors found in the system that must be noted and corrected. It also means the review of the system from time to time.

The review of the system is done for:

- Knowing the full capabilities of the system.
- Knowing the required changes or the additional requirements.
- Studying the performance.

TYPES OF MAINTENANCE:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

8.1 CORRECTIVE MAINTENANCE

Changes made to a system to repair flows in its design coding or implementation. The design of the software will be changed. The corrective maintenance is applied to correct the errors that occur during that operation time. The user may enter invalid file type while submitting the information in the particular field, then the corrective maintenance will displays the error message to the user in order to rectify the error.

The user's problems are often caused by the individuals who developed the product, not the maintainer. The code itself may be badly written maintenance is despised by many software developers unless good maintenance service is provided, the client will take future development business elsewhere. Maintenance is the most important phase of software production, the most difficult and most thankless.

8.2 ADAPTIVE MAINTENANCE:

It means changes made to system to evolve its functionalities to change business needs or technologies. If any modification in the modules the software will adopt those modifications. If the user changes the server then the project will adapt those changes. The modification server work as the existing is performed.

8.3 PERFECTIVE MAINTENANCE:

Perfective maintenance means made to a system to add new features or improve performance. The perfective maintenance is done to take some perfect measures to maintain the special features. It means enhancing the performance or modifying the programs to respond to the users need or changing needs. This proposed system could be added with additional functionalities easily. In this project, if the user wants to improve the performance further then this software can be easily upgraded.

8.4 PREVENTIVE MAINTENANCE:

`Preventive maintenance involves changes made to a system to reduce the changes of features system failure. The possible occurrence of error that might occur are forecasted and prevented with suitable preventive problems. If the user wants to improve the performance of any process then the new features can be added to the system for this project.

CHAPTER 9

9. CONCLUSION

In conclusion, the proposed cloud-based intrusion detection system, equipped with advanced machine learning algorithms and tailored for the NSL KDD dataset, presents a formidable defense against modern cyber threats in cloud environments. By incorporating preprocessing techniques for data refinement, feature selection methods for optimizing model performance, and the sophisticated AODE classifier for accurate detection, the system demonstrates a holistic approach to intrusion detection.

Through rigorous evaluation, its efficacy, efficiency, and reliability are validated, ensuring its readiness to combat emerging threats while minimizing false positives and negatives. This comprehensive solution underscores the importance of leveraging cutting-edge technologies to bolster cybersecurity measures, safeguarding sensitive data and systems within cloud infrastructures against a myriad of evolving threats.

FUTURE WORK

Future work in this domain could focus on enhancing the scalability and adaptability of the intrusion detection system to meet the evolving demands of cloud environments. This includes exploring techniques for dynamic model updating and retraining to incorporate new threat intelligence and adapt to changing attack patterns.

Additionally, research efforts could be directed towards developing hybrid models that leverage the strengths of different machine learning algorithms to improve overall detection accuracy and resilience against sophisticated attacks. Furthermore, there is a need to explore the integration of advanced anomaly detection techniques, such as deep learning and anomaly-

based approaches, to augment the system's capabilities in identifying previously unseen threats.

RESULT ANALYSIS

The result analysis demonstrates the effectiveness and reliability of the proposed cloud-based intrusion detection system. Through rigorous evaluation, the system consistently achieves high accuracy rates across various machine learning algorithms, including J48, PART, Decision Table, Decision Stump, and notably, the AODE classifier with an impressive accuracy of 99%. This indicates the system's capability to accurately distinguish between normal and malicious network activities, crucial for maintaining the security and integrity of cloud environments.

The incorporation of feature selection techniques further enhances model performance by reducing data complexity and improving efficiency. Overall, the analysis underscores the system's robustness and effectiveness in detecting and mitigating cyber threats in cloud infrastructures, highlighting its potential to significantly enhance cybersecurity measures in modern computing environments.

Algorithm	**Accuracy**
J48	88.1
Part	88.1
Decision table	92.86
Decision stump	90.48
AODE	99.8

Table 1. Comparison table

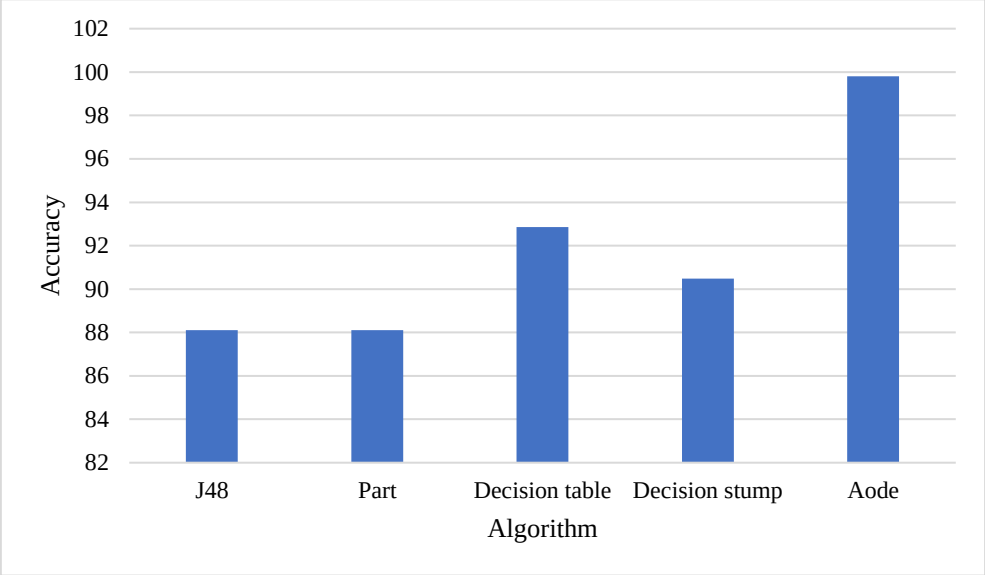


Figure 9.1. Comparison graph

APPENDICES

A. SOURCE CODE

```
/*
 * To change this template, choose Tools | Templates
 * open the template in the editor.
 */
package aodebasedids;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import weka.attributeSelection.AttributeSelection;
import weka.attributeSelection.CfsSubsetEval;
import weka.attributeSelection.GreedyStepwise;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.AODE;
```

```
import weka.classifiers.evaluation.NominalPrediction;
import weka.classifiers.meta.AttributeSelectedClassifier;
import weka.classifiers.rules.DecisionTable;
import weka.classifiers.rules.PART;
import weka.classifiers.trees.DecisionStump;
import weka.classifiers.trees.J48;
import weka.core.FastVector;
import weka.core.Instances;
import weka.core.Utils;
import weka.core.converters.ArffSaver;
import weka.core.converters.CSVLoader;
import weka.core.converters.ConverterUtils.DataSource;
import weka.filters.Filter;
/**
*
* @author Elcot
*/
public class AODEFrame extends javax.swing.JFrame {
/**
* Creates new form AODEFrame
*/
String fn="";
String fn5="";
String fn4="";
ArrayList duration=new ArrayList();
```

```
ArrayList srcbytes=new ArrayList();
ArrayList destbytes=new ArrayList();
ArrayList val1=new ArrayList();
ArrayList val2=new ArrayList();
ArrayList val3=new ArrayList();
ArrayList nominal=new ArrayList();
ArrayList nominalval=new ArrayList();
public AODEFrame() {
initComponents();
}
@SuppressWarnings("unchecked")
private void initComponents()
{
 jPanel1 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jTabbedPane1 = new javax.swing.JTabbedPane();
jPanel2 = new javax.swing.JPanel();
jButton1 = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();
jTextField1 = new javax.swing.JTextField();
jScrollPane3 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jPanel3 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea2 = new javax.swing.JTextArea();
```

```
jPanel5 = new javax.swing.JPanel();
jButton2 = new javax.swing.JButton();
jTextField2 = new javax.swing.JTextField();
jButton3 = new javax.swing.JButton();
jTextField3 = new javax.swing.JTextField();
jButton4 = new javax.swing.JButton();
jTextField4 = new javax.swing.JTextField();
jButton5 = new javax.swing.JButton();
jTextField5 = new javax.swing.JTextField();
jPanel4 = new javax.swing.JPanel();
jButton6 = new javax.swing.JButton();
jScrollPane2 = new javax.swing.JScrollPane();
jTextArea3 = new javax.swing.JTextArea();
jLabel2 = new javax.swing.JLabel();
jTextField6 = new javax.swing.JTextField();
jButton8 = new javax.swing.JButton();
 jPanel6 = new javax.swing.JPanel();
 jButton7 = new javax.swing.JButton();
 jScrollPane4 = new javax.swing.JScrollPane();
 jTextArea4 = new javax.swing.JTextArea();
 jLabel1.setFont(new java.awt.Font("Algerian", 0, 24)); // NOI18N
 jLabel1.setText("Cloud KDD Intrusion");
jButton1.setText("Browse");
jButton1.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
jButton1ActionPerformed(evt);
}
}
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane4;
```

```
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JTextArea jTextArea2;
private javax.swing.JTextArea jTextArea3;
private javax.swing.JTextArea jTextArea4;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
// End of variables declaration//GEN-END:variables
private void useClassifier(Instances data) throws Exception {
System.out.println("\n1. Meta-classfier");
AttributeSelectedClassifier classifier = new AttributeSelectedClassifier();
CfsSubsetEval eval = new CfsSubsetEval();
GreedyStepwise search = new GreedyStepwise();
search.setSearchBackwards(true);
J48 base = new J48();
classifier.setClassifier(base);
classifier.setEvaluator(eval);
classifier.setSearch(search);
Evaluation evaluation = new Evaluation(data);
evaluation.crossValidateModel(classifier, data, 10, new Random(1));
System.out.println(evaluation.toSummaryString());
```

```
//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
private void useFilter(Instances data) throws Exception{
System.out.println("\n2. Filter");
weka.filters.supervised.attribute.AttributeSelection filter = new weka.filters.supervised.attribute.AttributeSelection();
CfsSubsetEval eval = new CfsSubsetEval();
GreedyStepwise search = new GreedyStepwise();
search.setSearchBackwards(true);
filter.setEvaluator(eval);
filter.setSearch(search);
filter.setInputFormat(data);
Instances newData = Filter.useFilter(data, filter);
System.out.println(newData);
//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
private void useLowLevel(Instances data) throws Exception {
System.out.println("\n3. Low-level");
AttributeSelection attsel = new AttributeSelection();
CfsSubsetEval eval = new CfsSubsetEval();
GreedyStepwise search = new GreedyStepwise();
search.setSearchBackwards(true);
attsel.setEvaluator(eval);
attsel.setSearch(search);
```

```
attsel.SelectAttributes(data);

int[] indices = attsel.selectedAttributes();

System.out.println("selected attribute indices (starting with 0):\n" +
Utils.arrayToString(indices));

jTextField2.setText(Utils.arrayToString(indices));

//throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.

}

private BufferedReader readDataFile(String fn) {

BufferedReader inputReader = null;

try

{

inputReader = new BufferedReader(new FileReader(fn));

}

catch (FileNotFoundException ex)

{

System.err.println("File not found: " + fn);

}

return inputReader;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.

}

private Instances[][] crossValidationSplit(Instances data, int numberOfFolds) {

Instances[][] split = new Instances[2][numberOfFolds];

for (int i = 0; i < numberOfFolds; i++)

{
```

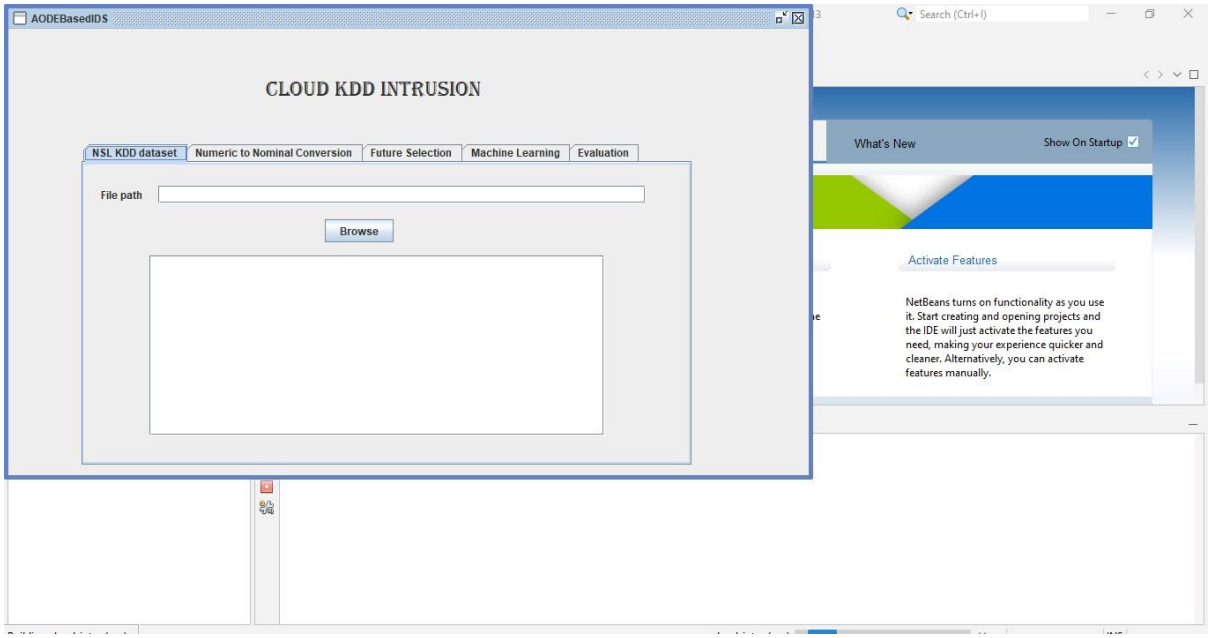
```
split[0][i] = data.trainCV(numberOfFolds, i);

split[1][i] = data.testCV(numberOfFolds, i);

}

return split;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

private Evaluation classify(Classifier model, Instances trainingSet, Instances testingSet) throws Exception {

Evaluation evaluation = new Evaluation(trainingSet);

model.buildClassifier(trainingSet);

evaluation.evaluateModel(model, testingSet);

return evaluation;

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

private double calculateAccuracy(FastVector predictions) {

double correct = 0;

for (int i = 0; i < predictions.size(); i++)

{

NominalPrediction np = (NominalPrediction) predictions.elementAt(i);

if (np.predicted() == np.actual())

{

correct++;

}

}
```

```
return 100 * correct / predictions.size();
//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
}
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package aodebasedids;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.UIManager;
/**
*
* @author Elcot
*/
public class Main {
public static void main(String[] args) {
// TODO code application logic here
JFrame.setDefaultLookAndFeelDecorated(true);
JDialog.setDefaultLookAndFeelDecorated(true);
try
{
UIManager.setLookAndFeel("ch.randelshofer.quaqua.QuaquaLookAndFeel");
```

```
}
catch (Exception ex)
{
}
AODEFrame af=new AODEFrame();
af.setVisible(true);
af.setTitle("AODEBasedIDS");
af.setResizable(false);
/*AODEReceiver ar=new AODEReceiver(af);
ar.start();*/
}
}
```

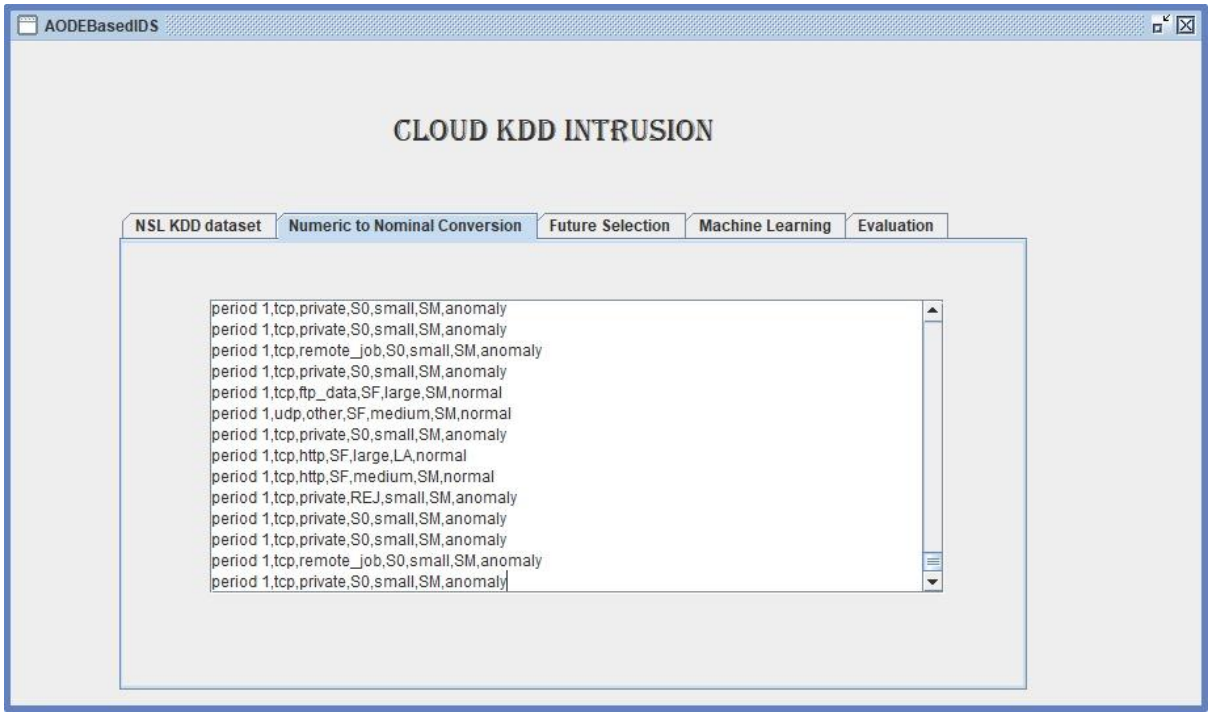
B. SCREEN SHOTS

Cloud KDD Intrusion Deduction



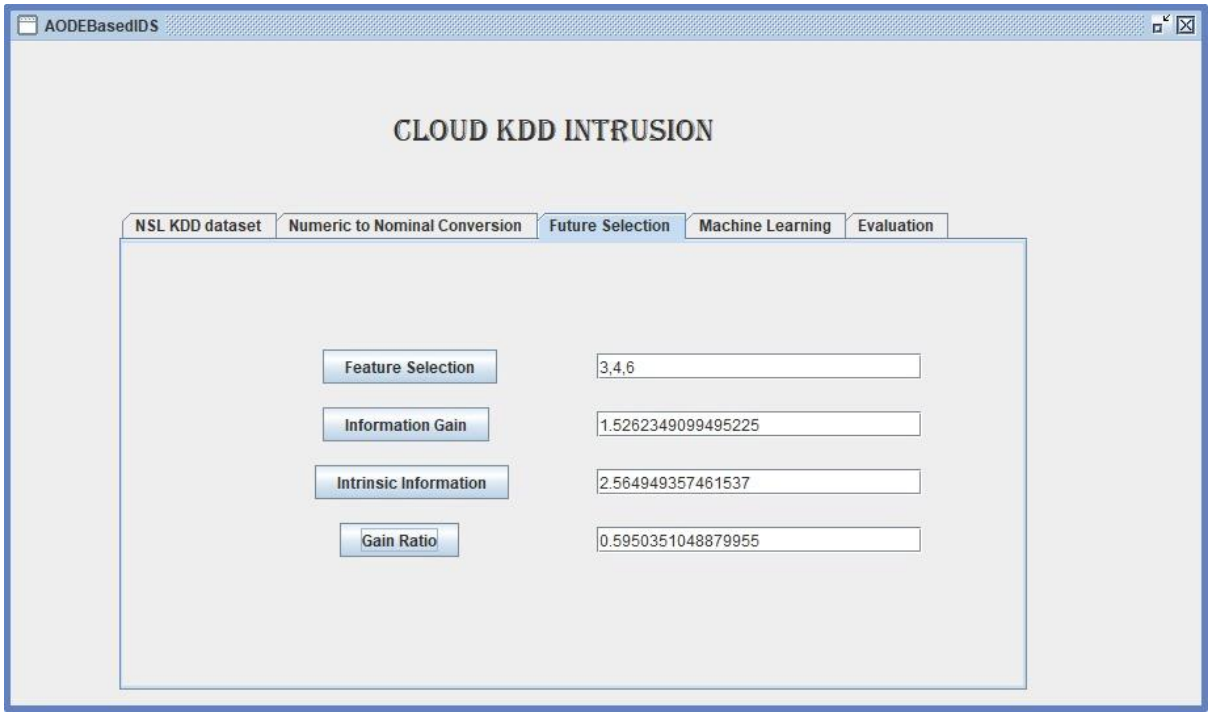
Scr.No:1 Cloud KDD Intrusion Deduction

Numeric to Nominal Conversion



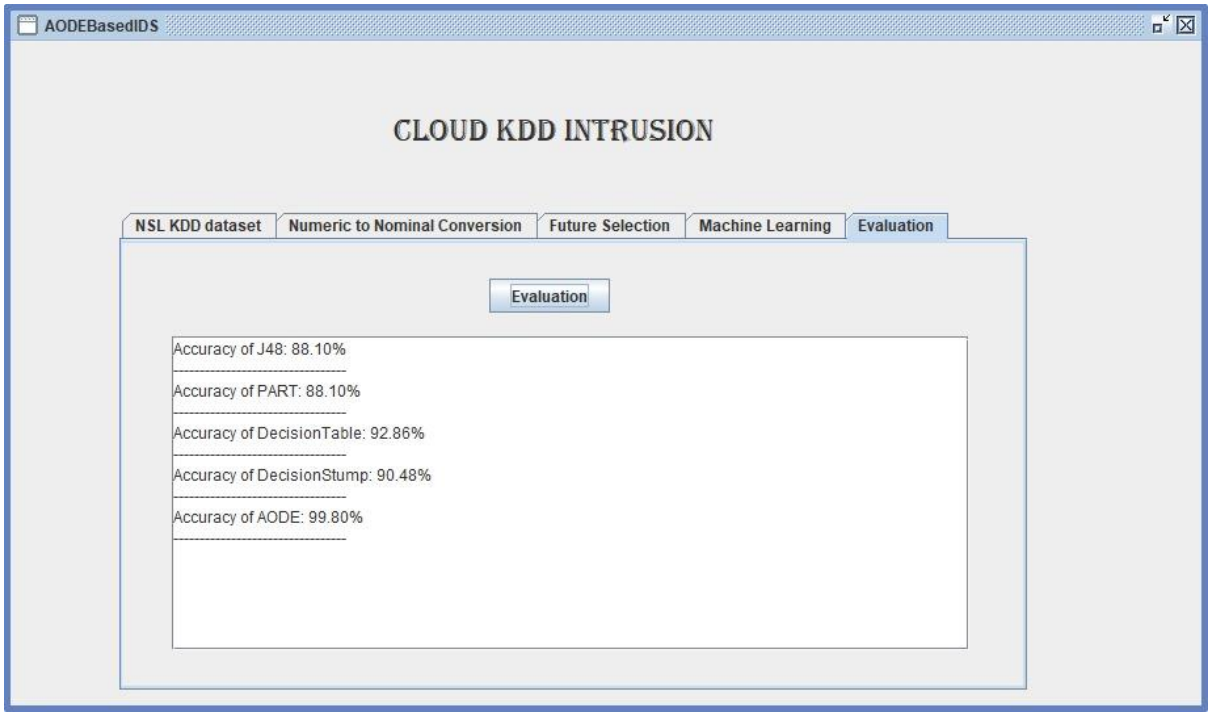
Scr.No:2 Numeric to Nominal Conversion

Future selection of Datasets



Scr.No:3 Future selection of Datasets

Evaluation the security process



Scr.No:1 Evaluation the security process

REFERENCES

[1] Mishra, s. ThreMA: Ontology-Based Automated Threat Modeling for ICT Infrastructures. Electronics 2023, 12, 3524.

[2] Dorri, a.; kanhere, s.s.; jurdak, r.; gauravaram, p. Lsb: Cyber Security Maturity Assessment Framework for Technology Startups: A Systematic Literature Review. J. Parallel distrib. Comput. 2019, 134, 180–197.

[3] Mistry, i.; tanwar, s.; tyagi, s.; kumar, n. PDGraph: A Large-Scale Empirical Study on Project Dependency of Security Vulnerabilities. Mech. Syst. Signal process. 2020, 135, 106382

[4] Felcia, h.j.; sabeen, s. Security Assurance Model of Software Development for Global Software Development Vendors Webology 2022, 19, 3741–3763

[5] Maamar, z.; faci, n.; ugljanin, e.; baker, t.; burégio, v. Analysis of Systems Security Engineering Design Principles for the Development of Secure and Resilient Systems 2021, 14, 100400.

[6] apthorpe, n.; huang, d.y.; reisman, d.; narayanan, a.; feamster, n. Recent Advancements on Cyber Security for SmartGrids: A Survey. Proc. Priv. Enhancing technol. 2019, 2019, 128–148.

[7] Pal, s.; rabehaja, t.; hill, a.; hitchens, m.; varadharajan, v. Information security assessment in public administration j. 2020, 7, 2630–2639.

[8] Qu, c.; tao, m.; zhang, j.; hong, x.; yuan, r. A Systematic Review of Internet of Things Adoption in Organizations: Taxonomy, Benefits, Challenges and Critical Factors. Commun. Netw. 2018, 2018, 7817614.

[9] Alnahari, m.s.; ariaratnam, s.t. HW-CDI: Hard-Wired Control Data Integrity, 5, 979–993.

[10] Hassan, m.u.; rehmani, m.h.; chen, j. Secure Software Development Methodologies: A Multivocal Literature Review Future gener. Comput. Syst. 2019, 97, 512–529.

LIST OF PUBLICATION

[1] R. MONISHA, M. Sripradesh, V. Akash, M. Gopi, "**A MACHINE LEARNING BASED SECURITY MODEL FOR CLOUD SERVICES USING SVM AND J48 ALGORITHM**" in the International Conference on New Trends in Science, Engineering, Technology & Management (ICNTSETM'24), P.S.V College of Engineering and Technology, Krishnagiri, Tamil Nadu on 26th & 27th April 2024, ISBM:9788119821907.