

Implementing Honeypot on IOT Environment to detect the Man-in-the-Middle Attack

A CAPSTONE PROJECT REPORT

*Submitted in partial fulfilment of the
requirement for the award of the
Degree of*

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

by

**DASARI SHIVA (20BCE7075)
DASARI SRIKANTH (20BCE7076)
MULLAPUDI SRI SURYA TARUN (20BCD7030)
GOLLENAR RUSHWANTH (20BCN7163)**

Under the Guidance of

Dr. Y Mohamed Sirajudeen



**SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237**

November 2023

CERTIFICATE

This is to certify that the Capstone Project work titled "**Implementing HoneyPot on IOT Environment to detect the Man-in-the-Middle Attack**" that is being submitted by **DASARI SHIVA (20BCE7075)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.


Dr. Prof. Y Mohamed Sirajudeen

Guide

✓
The thesis is satisfactory / ~~unsatisfactory~~

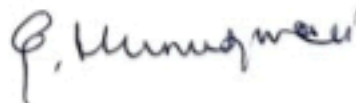

PROF. D KHASIM VALI

Internal Examiner 1


PROF. ESWARAIAH RAYACHOTI

Internal Examiner 2

Approved by



Dr. G Muneeswari (HoD), Data Science and Engineering
School of Computer Science and Engineering

CERTIFICATE

This is to certify that the Capstone Project work titled "**Implementing HoneyPot on IOT Environment to detect the Man-in-the-Middle Attack**" that is being submitted by **DASARI SRIKANTH (20BCE7076)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.



Dr. Prof. Y Mohamed Sirajudeen

Guide

✓
The thesis is satisfactory / ~~unsatisfactory~~



PROF. D KHASIM VALI

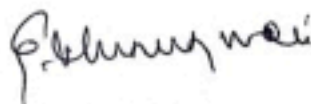
Internal Examiner 1



PROF. ESWARAIAH RAYACHOTI

Internal Examiner 2

Approved by



Dr. G Muneeswari (HoD), Data Science and Engineering

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the Capstone Project work titled "**Implementing Honeypot on IOT Environment to detect the Man-in-the-Middle Attack**" that is being submitted by **MULLAPUDI SRI SURYA TARUN (20BCD7030)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.


Dr. Prof. Y Mohamed Srarudeen

Guide


The thesis is satisfactory / unsatisfactory



PROF. D KHASIM VALI

Internal Examiner 1


PROF. ESWARAJAH RAYACHOTI

Internal Examiner 2

Approved by




Dr. G Muneeswari (HoD), Data Science and Engineering

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the Capstone Project work titled "**Implementing Honeypot on IOT Environment to detect the Man-in-the-Middle Attack**" that is being submitted by **GOLLENA RUSHWANTH (20BCN7163)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.


Dr. Prof. Y Mohamed Sirajudeen
Guide

✓
The thesis is satisfactory / ~~unsatisfactory~~



PROF. D KHASIM VALI

Internal Examiner 1


PROF. ESWARAIAH RAYACHOTI

Internal Examiner 2

Approved by


Dr. Anil V Turukmane (Hod), Networking and Security

School of Computer Science and Engineering

ACKNOWLEDGEMENTS

We would like to take this opportunity to express our deep sense of gratitude to all who helped us directly or indirectly during their thesis work.

Firstly, we would like to thank our supervisor, Dr. **Y Mohamed Sirajudeen** for being a great mentor and for constantly supporting and guiding us in the successful completion of this dissertation. The confidence shown on us by him was the biggest source of inspiration for us. It has been a privilege working with him.

We also wish to express our sincere thanks to all the faculty members of computer science and engineering department for their support and encouragement.

We would like to express our sincere appreciation and gratitude towards my team members for their encouragement, consistent support and invaluable suggestions at this time I needed the most.

- **DASARI SHIVA (20BCE7075)**
- **DASARI SRIKANTH (20BCE7076)**
- **MULLAPUDI SRI SURYA TARUN (20BCD7030)**
- **GOLLENAR RUSHWANTH (20BCN7163)**

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	4
1.	INTRODUCTION	
	1.1. General Introduction	5
	1.2. Project Objectives	6
	1.3. Problem Statement	7
2.	SYSTEM PROPOSAL	
	2.1. Existing System	8
	2.1.1 Disadvantages	8
	2.2. Proposed System	8
	2.2.1 Advantages	9
	2.3. Literature Survey	10
3.	SYSTEM DIAGRAMS	
	3.1. Architecture Diagram	16
	3.2. Flow Diagram	16
	3.3. UML Diagrams	17
4.	IMPLEMENTATION	
	4.1. Modules	20
	4.2. Modules Description	21
5.	SYSTEM REQUIREMENTS	
	5.1. Hardware Requirements	25
	5.2. Software Requirements	25
	5.3. Software Description	26
	5.4. Testing of Products	31
6.	CONCLUSION	35
	FUTURE ENHANCEMENT	35
7.	SAMPLE SCREENSHOT	36
	REFERENCES	43
	SAMPLE CODING	46

Abstract

Man in the middle (MITM) attacks can dramatically compromise the security of Wi-Fi network where an attacker eavesdrops and intercepts the communication medium over the wireless communication networks. This kind of attack aims to steal sensitive data such as credit card details, login accounts, and other important financial transactions. Even though that many detection techniques have been proposed to mitigate MITM attacks, however, this attack still occurs and causes tremendous damages. In this study, we propose a set of machine learning techniques to detect and identify MITM attacks on a wireless communication network. In addition, we evaluate and validate our approach based on the performance metrics, and compare the performance results with other machine learning techniques.

The use of Internet of Things appliances has grown significantly over the last few years, and with it, so have the security risks associated with them. The frequency of cyberattacks targeting Internet of Things devices is rising daily. Botnets are used to carry out the majority of malicious attacks on Internet of Things devices. These days, different gadgets call for various security precautions. In the event that multiple security options are available, we must determine which options are most appropriate for a given IoT device or application. It is critical to comprehend what is needed for a specific device's threat prevention. Devices on their own or an entire company require a system that can identify and react to various threats, malware, and system hacking.

CHAPTER 1

INTRODUCTION

1.1 General Introduction:

IoT environments are major targets of several types of intrusions and malicious activities which put the security and privacy of IoT consumers at risk. Thus, network security and user privacy are the major concerns in IoT environment which motivate researchers to develop an effective IDS. The proposed work is motivated by several challenges. The increasing size, autonomous nature, and attractive features of IoT network draws the attention of cybercriminals. The dramatic increase of crime rate in IoT ecosystem motivates researchers to develop more effective and intelligent solutions to prevent and detect such crimes. Cyber-attacks can be reduced by recognizing any suspicious activity occurring in the network to catch the malicious actors before they breach a particular network or a system. Hence, to monitor network and system assets for unexpected activities is also the motivation behind proposing an intrusion detection system.

Machine learning is an extensively used technique to harmonize IDS with intelligent information systems, to detect various types of malicious activities in a smart network . Network traffic passing through the nodes connected in IoT infrastructure needs to be discriminated between benign and malicious traffic. In most cases, the majority of network traffic showcases normal (benign) behavior. If malicious traffic also shows normal behavior, it could be more dangerous and may lead to the problem of high attack detection rate with low false alarm rate (FAR).

The performance of ML algorithms can be strengthened through various problem optimization techniques. The ensemble technique of ML is an advanced convincing tool that can upgrade the performance of existing models designed for application forecasting in different application areas. In ML, the ensemble-based learning methods perform classification by creating and integrating multiple models to solve a problem. Ensemble-based ML models combine multiple base models and provide better prediction performance than the conventional classification models.

Besides accuracy, latency and true positive rate must also be considered to be essential metrics for evaluating any proposed predictive model. The authors in the present paper aim to develop a model for intrusion detection for IoT systems using Random Forest Classification. This model aims to identify and classify Man-in-the-middle attack existing in an IoT environment.

This can be achieved by the Honeypot on IOT environment to detect the MIMT attack on those network.

1.2 Objectives:

The main objective of our project is,

- To design a time-efficient realistic MIMT detection using an ML-based algorithms such as SVM and RF classifier by predicting network traffic behavior in an IoT environment using Dataset collected by the Wireshark.
- To remove irrelevant and repetitive features using dimensionality reduction and feature reducing approaches.
- To evaluate and compare the performance of the proposed MIMT detection in terms of train and test accuracy, TPR, FPR, error-rate, and above all time efficiency.

1.3 Problem Statement:

One of the most common problems with detecting MIMT is the detection of false positives or false negatives, this occurs when the system blocks a activity on the network because it is out of the normal and so it assumes it is malicious, causing denial of service to a valid user, trying to do a valid procedure.

CHAPTER 2

SYSTEM PROPOSAL

2.1 EXISTING SYSTEM:

- The existing system used classification such as Support Vector Machine which gave less efficiency and accuracy to detect the MIMT Attack in the IOT Environment.
- The existing algorithms have not given the correctness in prediction accurately and also the time taken to detect is high compare to proposed classification such as Random Forest. So the existing system could sometimes lead to varying levels of accuracy.

2.1.1 DISADVANTAGES:

- The results is low when compared with proposed.
- Time consumption is high.
- Theoretical limits.

2.2 PROPOSED SYSTEM:

- Support vector machine gives low accuracy results in the existing system, so we move on to the RF Algorithm.
- Accurate prediction is done using Random Forest.
- Perfectly classify the MIMT attack and Normal.
- Result is generated and viewed by the Legitimate user.
- Fast convergence, skipping the local minima and computationally faster.

2.2.1 ADVANTAGES

- It is efficient for large number of datasets.
- The experimental result is high when compared with existing system.
- Time consumption is low.
- Provide accurate prediction results

2.3 LITERATURE SURVEY:

2.3.1 Digital forensic investigation of cloud storage services

Author: H. Chung, J. Park, S. Lee, and C. Kang

Year: 2022

Methodology

Current large distributed systems allow users to share and trade resources. In cloud computing, users purchase different types of resources from one or more resource providers using a fixed pricing scheme. Federated clouds, a topic of recent interest, allow different cloud providers to share resources for increased scalability and reliability. However, users and providers of cloud resources are rational and maximize their own interest when consuming and contributing shared resources. In this paper, we present a dynamic pricing scheme suitable for rational users requests containing multiple resource types. Using simulations, we compare the efficiency of our proposed strategy-proof dynamic scheme with fixed pricing, and show that user welfare and the percentage of successful requests is increased by using dynamic pricing.

Searching on the Internet today can be compared to dragging a net across the surface of the ocean. While a great deal may be caught in the net, there is still a wealth of information that is deep, and therefore, missed.

Advantages:

- New procedure for investigating and analyzing the artifacts of all accessible devices, such as Windows system, Mac system, iPhone, and Android smartphone.

Disadvantages:

- Here they failed revoke the traitors

2.3.2 Google drive: Forensic analysis of data remnants

Author: D. Quick and K. R. Choo

Year: 2020

Cloud computing has the potential to provide low-cost, scalable computing, but cloud security is a major area of concern. Many organizations are therefore considering using a combination of a secure internal cloud, along with (what they perceive to be) less secure public clouds. However, this raises the issue of how to partition applications across a set of clouds, while meeting security requirements. Currently, this is usually done on an ad-hoc basis, which is potentially error-prone, or for simplicity the whole application is deployed on a single cloud, so removing the possible performance and availability benefits of exploiting multiple clouds within a single application. This paper describes an alternative to ad-hoc approaches – a method that determines all ways in which applications structured as workflows can be partitioned over the set of available clouds such that security requirements are met. The approach is based on a Multi-Level Security model that extends Bell-LaPadula to encompass cloud computing. This includes introducing workflow transformations that are needed where data is communicated between clouds. In specific cases these transformations can result in security breaches, but the paper describes how these can be detected. Once a set of valid options has been generated, a cost model is used to rank them.

Advantages:

- Digital evidence can be stored in cloud storage services, such as Google Drive.
- Identification of potential data storage is a challenge to forensic examiners.

Disadvantages:

- Investigation points not include directory listings, prefetch, link and registry files.

2.3.3 Fully collusion resistant traitor tracing with short cipher texts and private keys

Author: D.Boneh, A.Sahai, and B.Waters

Year: 2022

This paper reports the first results of an investigation into solutions to problems of security in computer systems; it establishes the basis for rigorous investigation by providing a general descriptive model of a computer system. Borrowing basic concepts and constructs from general systems theory, we present a basic result concerning security in computer systems, using precise notions of "security" and "compromise". We also demonstrate how a change in requirements can be reflected in the resulting mathematical model. A lengthy introductory section is included in order to bridge the gap between general systems theory and practical problem solving.

The Web has been quickly "deepened" by horde searchable databases online, where information is hidden behind query interfaces. The Deep Web, i.e., content hidden behind HTML forms, has long been recognized as a noteworthy gap in search engine coverage. Since it speaks to an extensive segment of the structured data on the Web, accessing to Deep-Web content has been a long-standing challenge for the database community. The rapid development of the World-Wide Web poses phenomenal scaling difficulties for universally useful crawlers and web search engines.

Advantages:

- This would allow both for the tracer to be untrusted and could be used to give a solution that is secure against stateful receivers

Disadvantages:

- Some complexities are in the encryption strategy.

2.3.4 Identity-based encryption from the weil pairing

Author: D.Boneh and M.K.Franklin

Year: 2021

Recently, there has been increased interest in the retrieval and integration of hidden-Web data with a view to leverage high-quality information available in online databases. Although previous works have addressed many aspects of the actual integration, including matching form schemata and automatically filling out forms, the problem of locating relevant data sources has been largely overlooked. However, considering the number of documents on the Web (Google already indexes over 8 billion documents), automatically finding tens, hundreds or even thousands of forms that are relevant to the integration task is really like looking for a few needles in a haystack. Besides, since the vocabulary and structure of forms for a given domain are unknown until the forms are actually found, it is hard to define exactly what to look for. We propose a new crawling strategy to automatically locate hidden- Web databases which aims to achieve a balance between the two conflicting requirements of this problem: the need to perform a broad search while at the same time avoiding the need to crawl a large number of irrelevant pages

Advantages:

- Cocks' system uses bit-by-bit encryption and consequently outputs long cipher texts

Disadvantages:

- Problem to build Chosen ciphertext secure identity based Systems that are secure in the standard Computation model (rather than the random oracle model).

2.3.5 Attribute-based encryption for fine-grained access control of encrypted data

Author: V. Goyal, O. Pandey, A. Sahai, and B. Waters

Year: 2022

New Web technologies, like AJAX, result in more responsive and interactive Web applications, sometimes called Rich Internet Applications (RIAs). Crawling techniques developed for traditional Web applications are not sufficient for crawling RIAs. The inability to crawl RIAs is a problem that needs to be addressed for at least making RIAs searchable and testable. We present a new methodology, called “model-based crawling”, that can be used as a basis to design efficient crawling strategies for RIAs. We illustrate model-based crawling with a sample strategy, called the “hypercube strategy”. The performances of our model-based crawling strategies are compared against existing standard crawling strategies, including breadth-first, depth-first, and a greedy strategy. Crawling is the activity of capturing all the reachable pages of a Web application together with the information on how to reach them. A crawler is a tool that performs crawling. A crawler navigates through the pages of the application starting from an initial page, just like a human navigates using a browser, but in an automated, systematic, and efficient way.

Advantages:

- A search query could potentially be any monotone boolean formula of any number of keywords. We leave the problem of hiding the set of attributes as open.

Disadvantages:

- One drawback of encrypting data, is that it can be selectively shared only at a coarse-grained level (i.e., giving another party your private key).

2.3.6 Early Intrusion Detection System using honeypot for industrial control networks

Author: Abbasgholi Pashaei , Mohammad Esmaeil Akbari , Mina Zolfy

Lighvan , Asghar Charmin

Year: 2022

Man-in-the-Middle (MITM) and Distributed Denial of Service (DDoS) attacks are significant threats, especially to Industrial Control Systems (ICS). The honeypot is one of the most common approaches to protecting the network against such attacks. This study proposes a Markov Decision Process (MDP) called the state-action-reward-state-action (SARSA) for honeypot design. The proposed system using environmental experiments can achieve greater accuracy and convergence speed than traditional IDSs. Here, we use two types of agents, one for classification and the other for the environment. The environmental agent tries to minimize the rewards given to the classifying agent. Therefore, the classification agent is forced to learn the most complicated policies, increasing its learning capability in the long term. Thus, the proposed method improves the level of interaction for the early detection of honeypots by recording aggressive behavior.

Advantages:

- To improve the classification of different DDoS attacks. To solve the problem in the honeypot environment, we use one of the model-free RL techniques called State-Action-Reward-State-Action (SARSA). Previous experiences of an agent are captured by the Q-value and controlled by the reward function.

Disadvantages:

- One drawback of encrypting data, is that it can be selectively shared only at a coarse-grained level (i.e., giving another party your private key).

CHAPTER 3

SYSTEM DIAGRAMS

3.1 SYSTEM ARCHITECTURE:

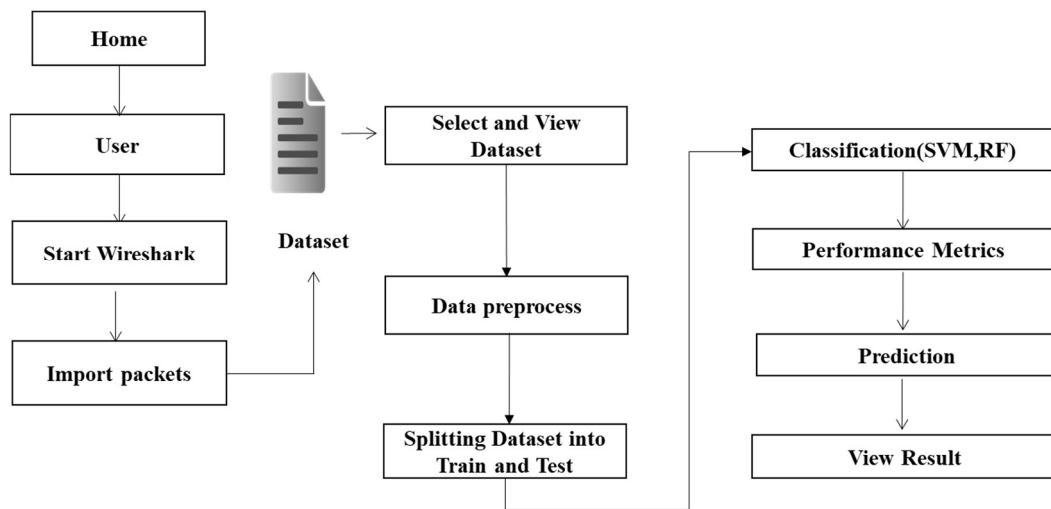


Fig 3.1 System architecture

3.2 Flow Diagram

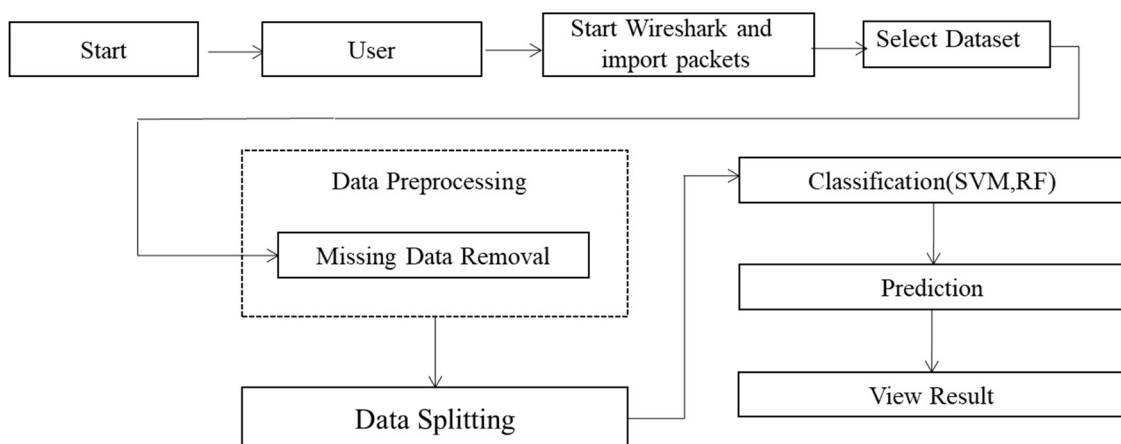


Fig 3.2 Flow Diagram

3.3 Use Case Diagram

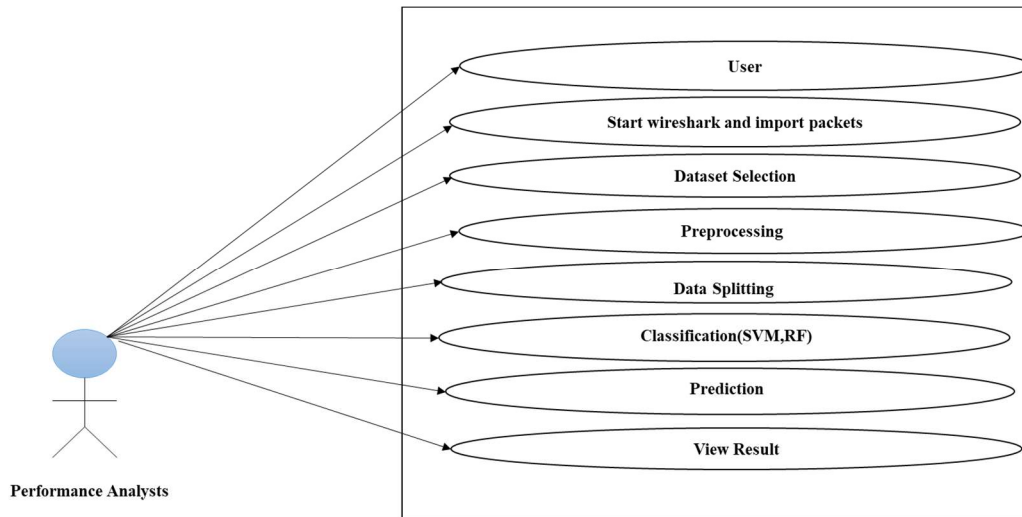


Fig 3.3 Use Case Diagram

3.4 Activity Diagram

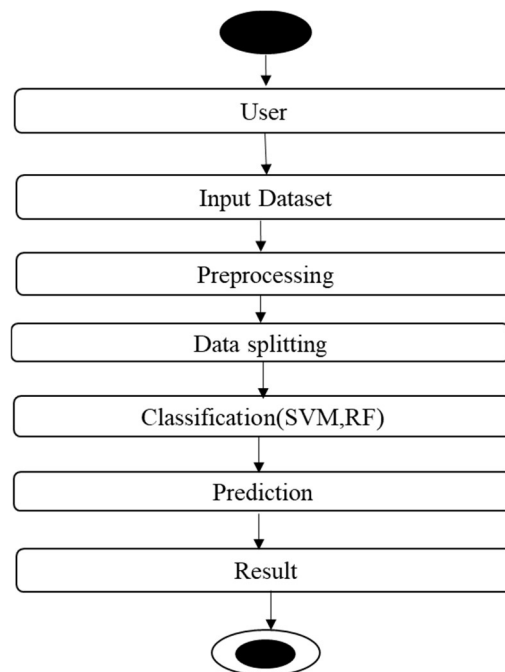


Fig 3.4 Activity Diagram

3.5 Sequence Diagram

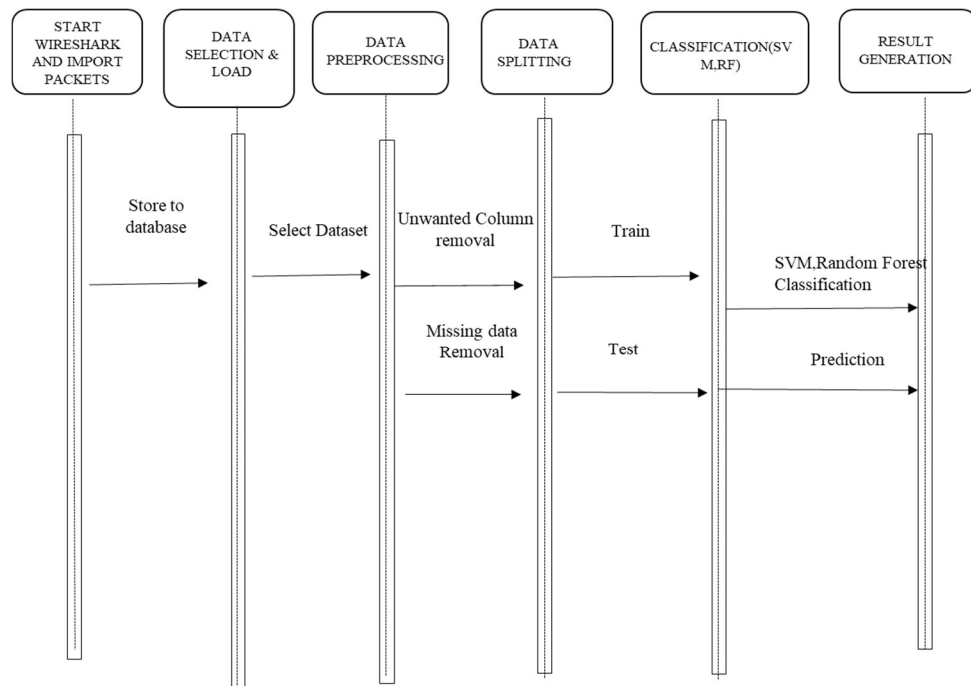


Fig 3.5 Sequence Diagram

3.6. Class Diagram

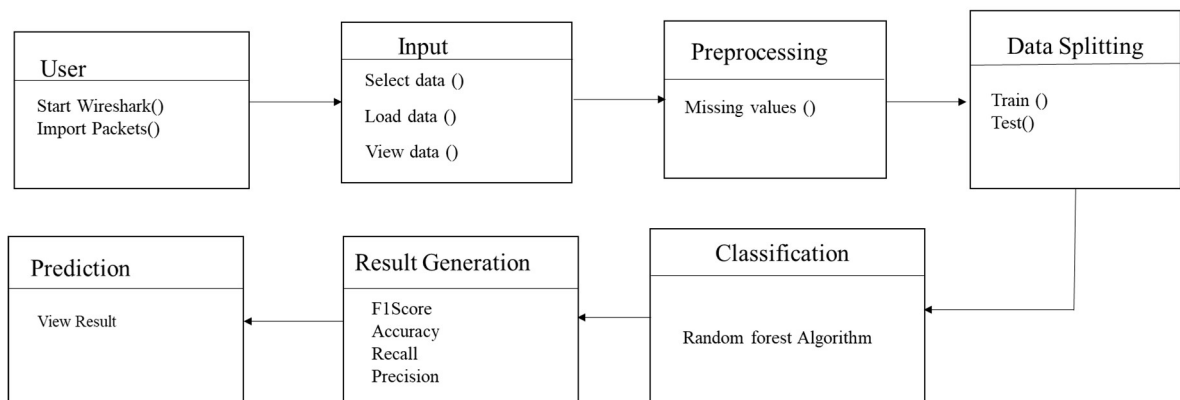


Fig 3.6 Class Diagram

3.7 ER DIAGRAM:

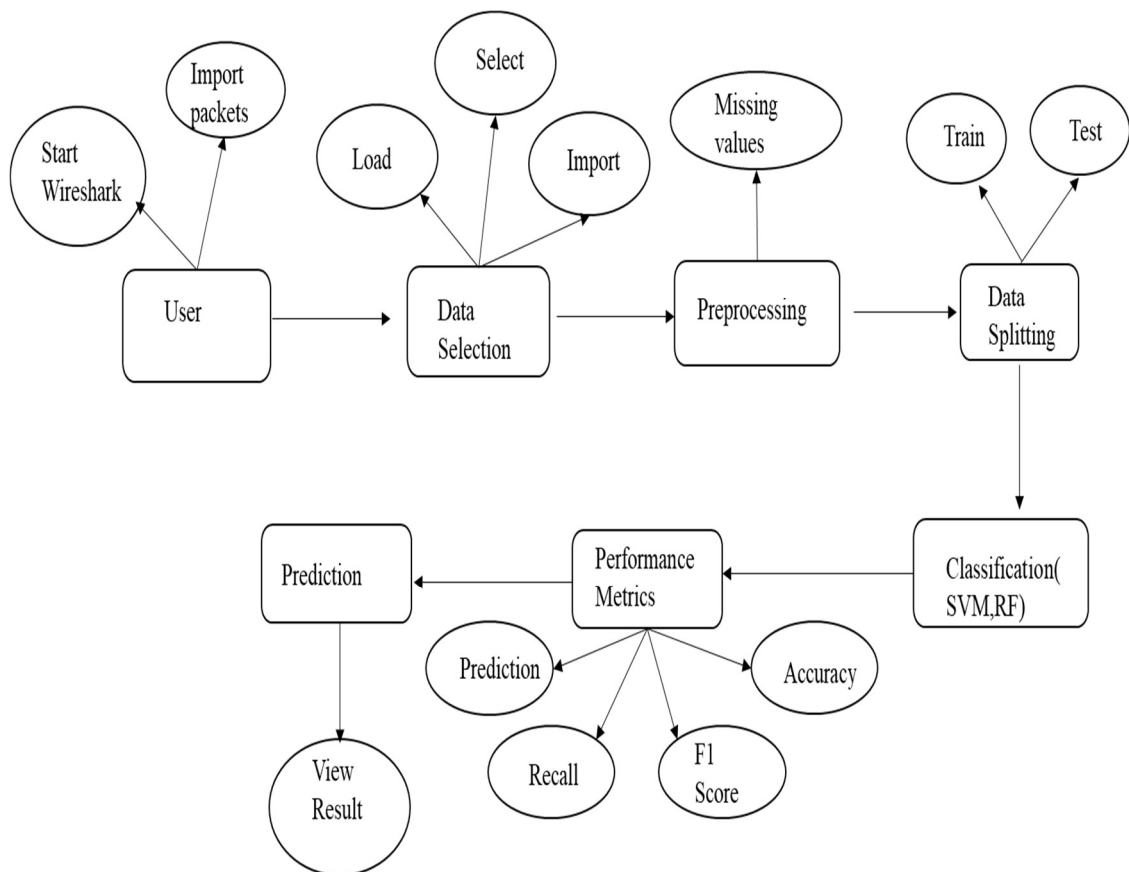


Fig 3.7 ER Diagram

CHAPTER-4

IMPLEMENTATION

4.1 MODULES:

- Honeypot Node Setup
- Data selection and Load dataset
- Data preprocessing
- Data splitting
- Classification
- Performance metrics
- Prediction

4.2 MODULES DESCRIPTION:

4.2.1 Honeypot Node Setup: Capturing packet in WIRESHARK and import packets for detecting MIMT attack.

- **Start WIRESHARK:** Start WIRESHARK and capture the packets and save it.
- **Import Packets:** Browse and import the packets where we can view the file path, file size, date created, time created importing dump data and to proceed.

4.2.2 DATA SELECTION:

- The input dataset was collected from the internet , website called kaggle.com.
- Dataset consist of 698 records with their outcome (normal,anomalous(spying,scan,malitious)).
- Now load the dataset to the cloud for applying ML algorithm over the dataset.

4.2.3 DATA PREPROCESSING:

- Data pre-processing is the process of removing the unwanted data from the dataset.
- Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.
- This step also includes cleaning the dataset by removing irrelevant or corrupted data that can affect the accuracy of the dataset, which makes it more efficient.
- Missing data removal
- Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.
- Missing and duplicate values were removed and data was cleaned of any abnormalities.

4.2.4 Data Splitting

- During the machine learning process, data are needed so that learning can take place.

- In addition to the data required for training, test data are needed to evaluate the performance of the algorithm but here we have training and testing dataset separately.
- In our process, we have to divide as training and testing.
- 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.

4.2.5 Classifications

Random Forest Algorithm

- **Random forest** is a machine learning algorithm for anomaly detection. It's an unsupervised learning algorithm that identifies anomaly by isolating outliers in the data.
- Random Forest is based on the Decision Tree algorithm. It isolates the outliers by randomly selecting a feature from the given set of features and then randomly selecting a split value between the max and min values of that feature.

Support Vector Machine:

- Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset.
- SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The

algorithm creates a line or a hyperplane which separates the data into classes.

4.2.6 Performance Metrics

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,

- **Accuracy:** Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

$$AC = (TP+TN) / (TP+TN+FP+FN)$$

- **Precision:** Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

$$Precision = TP / (TP+FP)$$

- **Recall:** Recall is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

$$Recall = TP / (TP+FN)$$

- **Specificity:** Specificity itself can be described as the algorithm/model's ability to predict a true negative of each category available. In literature, it

is also known simply as the true negative rate. Formally it can be calculated by the equation below.

$$\text{Specificity} = \text{TN} / \text{TN} + \text{FP}$$

4.2.7 Prediction

- Predict the dataset values are Anomalous / Normal by using LGBM classification algorithm

CHAPTER 5

5.3 SOFTWARE DESCRIPTION:

5.3.1 Java

Java is a programming language originally developed by James Gosling at Sun Microsystems (now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere." Java is currently one of the most popular programming languages in use, particularly for client-server web applications.

- **Java Platform**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific machine code. Java byte code instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware.

End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executable would. Just-in-Time compilers were introduced from an early stage that compiles byte codes to machine code during runtime.

Just as application servers such as Glass Fish provide lifecycle services to web applications, the Net Beans runtime container provides them to Swing applications. All new shortcuts should be registered in "Key maps/Net Beans" folder. Shortcuts installed INS Shortcuts folder will be added to all key maps, if there is no conflict. It means that if the same shortcut is mapped to different actions in Shortcut folder and current key map folder (like Key map/Net Beans), the Shortcuts folder mapping will be ignored.

- ✓ Database Explorer Layer API in Database Explorer
- ✓ Loaders-images-dB schema-Actions in Database Explorer
- ✓ Loaders-images-sq.-Actions in Database Explorer
- ✓ Plug-in Registration in Java EE Server Registry

The keyword `public` denotes that a method can be called from code in other classes, or that a class may be used by classes outside the class hierarchy. The class hierarchy is related to the name of the directory in which the `.java` file is located.

The keyword `static` in front of a method indicates a static method, which is associated only with the class and not with any specific instance of that class. Only static methods can be invoked without a reference to an object. Static methods cannot access any class members that are not also static. The keyword `void` indicates that the main method does not return any value to the caller. If a Java program is to exit with an error code, it must call `System. Exit ()` explicitly.

The method name "main" is not a keyword in the Java language. It is simply the name of the method the Java launcher calls to pass control to the program. Java classes that run in managed environments such as applets and Enterprise JavaBeans do not use or need a main () method. A Java program may contain multiple classes that have main methods, which means that the VM needs to be explicitly told which class to launch from.

The Java launcher launches Java by loading a given class (specified on the command line or as an attribute in a JAR) and starting its public static void main(String[]) method. Stand-alone programs must declare this method explicitly. The String [] rags parameter is an array of String objects containing any arguments passed to the class. The parameters to main are often passed by means of a command line.

- **Java a High-level Language**

A high-level programming language developed by Sun Microsystems. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

Java source code files (files with a .java extension) are compiled into a format called byte code (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs). Byte code can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Object-Oriented Software Development using Java: Principles, Patterns, and Frameworks contain a much applied focus that develops skills in designing software-particularly in writing well-designed, medium-sized object-oriented programs. It provides a broad and coherent coverage of object-oriented technology, including object-oriented modeling using the Unified Modeling Language (UML) object-oriented design using Design Patterns, and object-oriented programming using Java.

- **Net Beans**

The **Net Beans Platform** is a reusable framework for simplifying the development of Java Swing desktop applications. The Net Beans IDE bundle for Java SE contains what is needed to start developing Net Beans plug-in and Net Beans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally-signed upgrades and new features directly into the running application.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- Net Beans Visual Library
- Integrated Development Tools

J2EE

A **Java EE application** or a **Java Platform, Enterprise Edition application** is any deployable unit of Java EE functionality. This can be a single Java EE module or a group of modules packaged into an EAR file along with a Java EE application deployment descriptor.

Enterprise applications can consist of the following:

- EJB modules (packaged in JAR files)
- Web modules (packaged in WAR files)
- connector modules or resource adapters (packaged in RAR files)
- Session Initiation Protocol (SIP) modules (packaged in SAR files)
- application client modules
- Additional JAR files containing dependent classes or other components required by the application.

Xamp Server

- **XAMPs** are packages has the ability to serve web pages on World Wide Web.
- Apache is a web server. MySQL is an open-source database. PHP is a scripting language that can manipulate information held in a database and generate web pages dynamically each time content is requested by a

browser. Other programs may also be included in a package, such as phpMyAdmin which provides a graphical user interface for the MySQL database manager, or the alternative scripting languages Python or Perl.

MySQL

- The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.
- Free-software-open source projects that require a full-featured database management system often use MySQL. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, hob, Drupal and other software built on the LAMP software stack.

Platforms and interfaces

- Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called Modoc allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

5.4 TESTING PRODUCTS:

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. . 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.

5.4.1 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.

5.4.2 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:

- i) Top-down integration testing.
- ii) Bottom-up integration testing.

5.4.3 TESTING TECHNIQUES/STRATEGIES:

➤ BLACK BOX TESTING:

- 1. Black box testing is done to find incorrect or missing function
- 2. Interface error
- 3. Errors in external database access
- 4. Performance errors.
- 5. 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.

➤ 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.

5.4.4 SOFTWARE TESTING STRATEGIES

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.

CHAPTER 6

CONCLUSION

In this project, we propose an approach to utilise the Support Vector Machine and Random Forest algorithm for detecting man-in-the-middle attack. We call the approach the RF on datasets with significantly reduced dimensionality. The Classifications classifier gives high accuracy results that are comparable to SVM algorithm techniques in spite of working with reduced data.

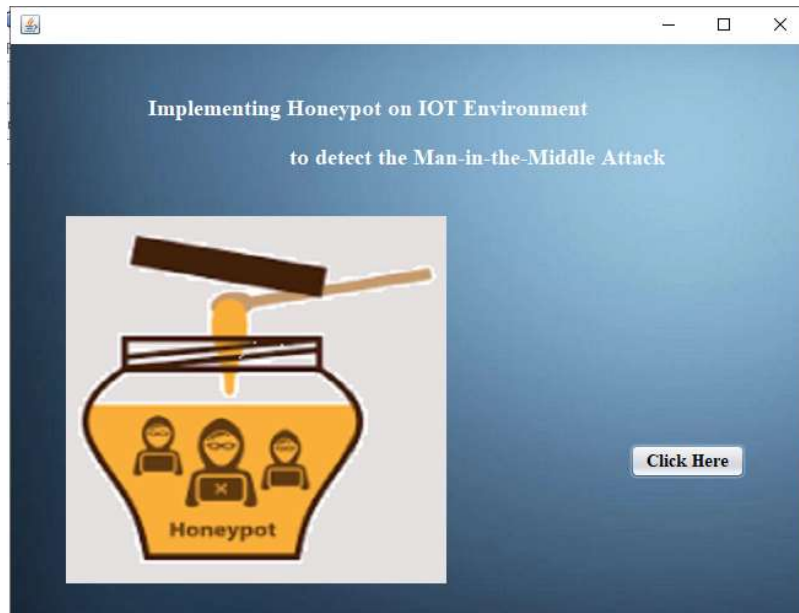
FUTURE ENHANCEMENT

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

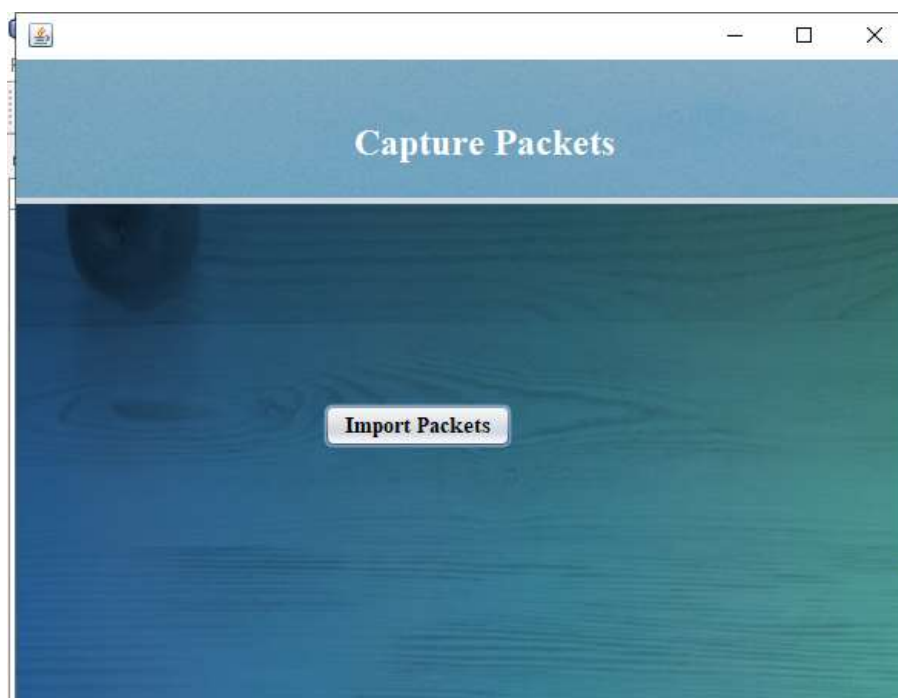
CHAPTER 7

SAMPLE SCREENSHOTS

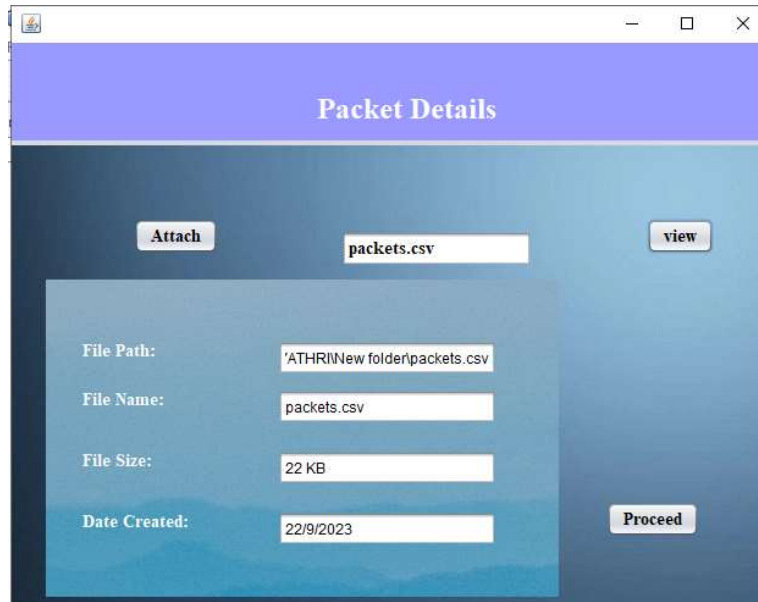
HOME PAGE:



CAPTURE PACKETS:



PACKET DETAILS:



The 'Packet Details' window has a purple header bar with the title 'Packet Details'. Below the header, there is a light blue background. On the left, there is a dark blue sidebar with a 'Load' button. The main area contains a form with the following fields:

- Attach** button
- packets.csv** (file name)
- view** button
- File Path:** 'ATHR\New folder\packets.csv'
- File Name:** packets.csv
- File Size:** 22 KB
- Date Created:** 22/9/2023
- Proceed** button

LOAD and STORE DATASET:



The 'Import CSV' window has a dark green header bar with the title 'Import CSV'. Below the header, there is a dark green background. On the left, there is a light blue sidebar with a 'Load' button and a 'Proceed' button. The main area contains a form with the following fields:

- Browse** button
- packets.csv** (file name)
- view** button
- 261,8.116995,192.168.0.24,239.255.255.250,SSDP,330,NOTIFY**
- HTTP/1.1**
- ,1**
- 262,8.117729,192.168.4.184,239.255.255.250,SSDP,217,M-SEARCH**
- HTTP/1.1**
- ,1**
- 263,8.138689,Routerbo_3f8a:9d,Broadcast,ARP,60,Who**
- has**
- 14.102.45.28?**
- Tell**
- 14.102.45.17,1**



VIEW DATA:

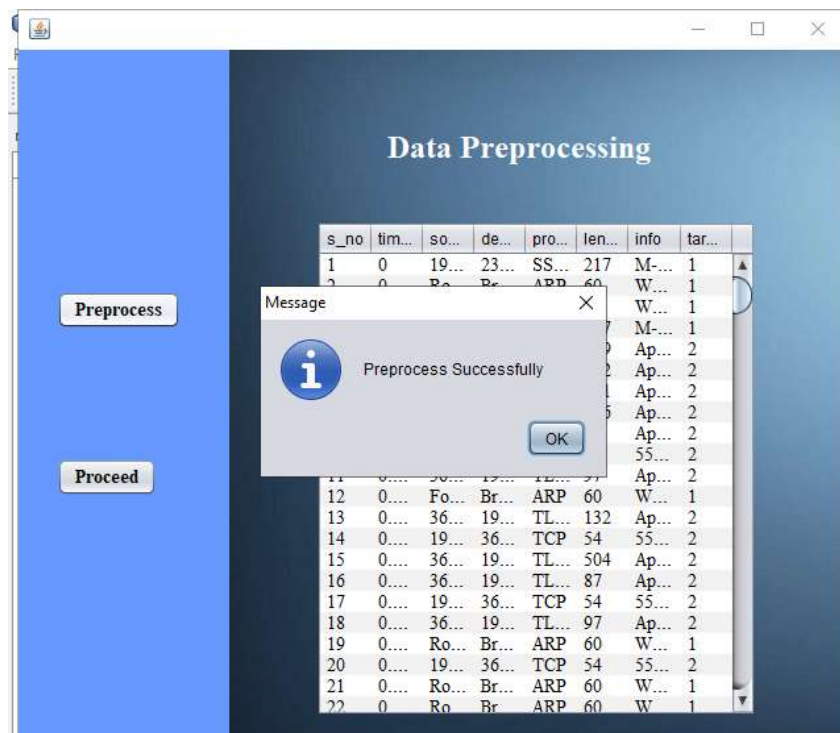
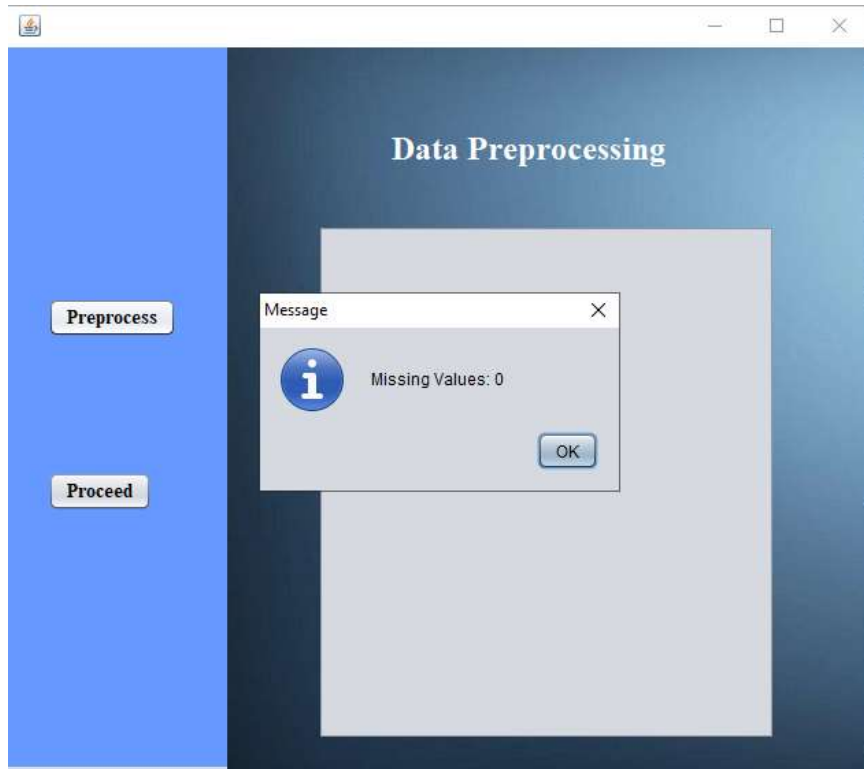
View Data

View

Proceed

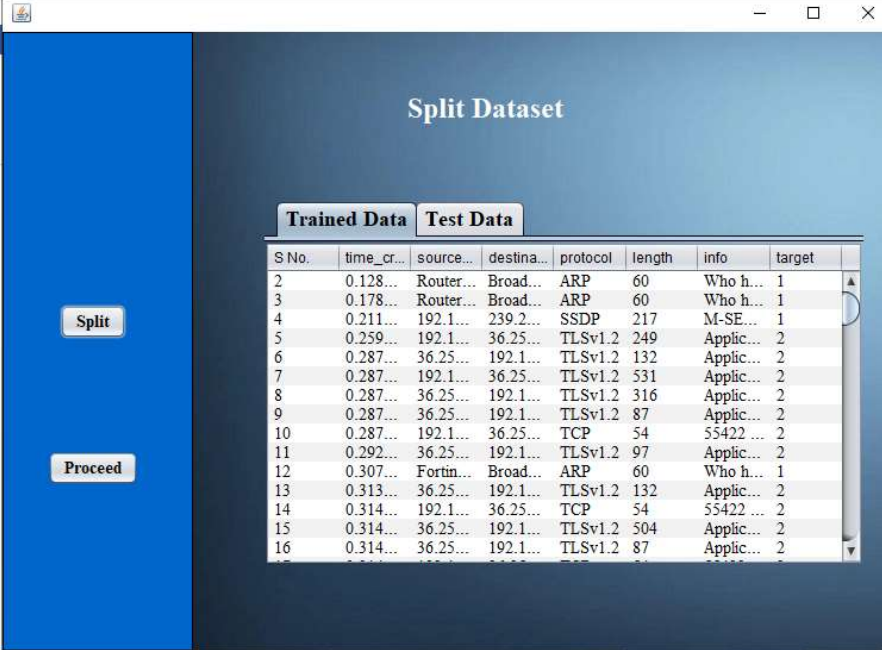
s_no	time_...	sourc...	destin...	protocol	length	info	target
1	0	192.1...	239.2...	SSDP	217	M-SEA...	1
2	0.128...	Route...	Broad...	ARP	60	Who h...	1
3	0.178...	Route...	Broad...	ARP	60	Who h...	1
4	0.211...	192.1...	239.2...	SSDP	217	M-SEA...	1
5	0.259...	192.1...	36.25...	TLSv1.2	249	Applic...	2
6	0.287...	36.25...	192.1...	TLSv1.2	132	Applic...	2
7	0.287...	192.1...	36.25...	TLSv1.2	531	Applic...	2
8	0.287...	36.25...	192.1...	TLSv1.2	316	Applic...	2
9	0.287...	36.25...	192.1...	TLSv1.2	87	Applic...	2
10	0.287...	192.1...	36.25...	TCP	54	55422...	2
11	0.292...	36.25...	192.1...	TLSv1.2	97	Applic...	2
12	0.307...	Fortin...	Broad...	ARP	60	Who h...	1
13	0.313...	36.25...	192.1...	TLSv1.2	132	Applic...	2
14	0.314...	192.1...	36.25...	TCP	54	55422...	2
15	0.314...	36.25...	192.1...	TLSv1.2	504	Applic...	2
16	0.314...	36.25...	192.1...	TLSv1.2	87	Applic...	2
17	0.314...	192.1...	36.25...	TCP	54	55422...	2
18	0.317...	36.25...	192.1...	TLSv1.2	97	Applic...	2
19	0.328...	Route...	Broad...	ARP	60	Who h...	1
20	0.358...	192.1...	36.25...	TCP	54	55422...	2

PREPROCESSING DATA:



DATA SPLITTING:

TRAIN DATA:

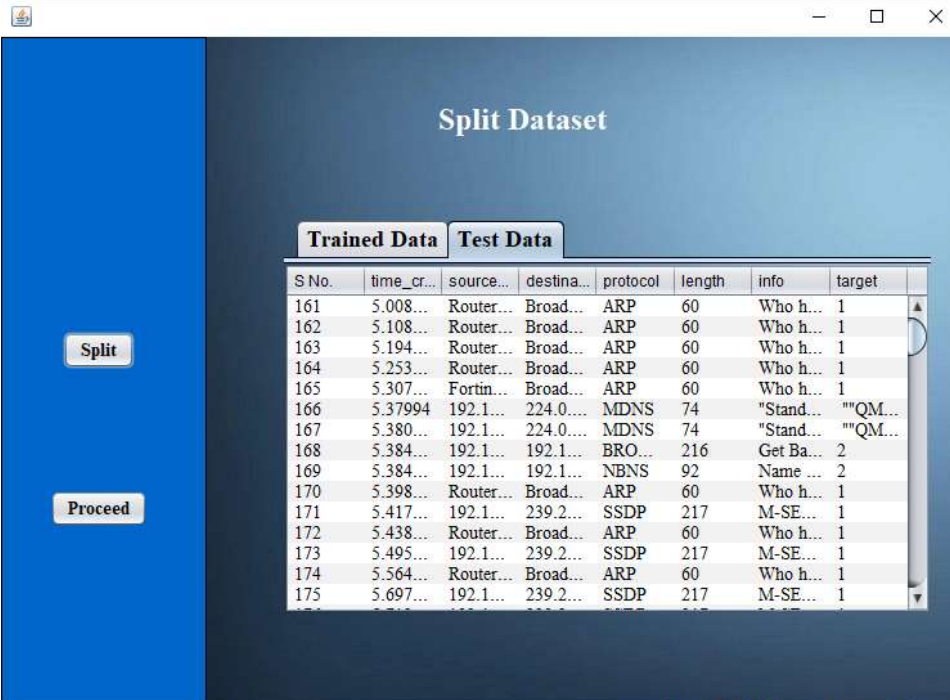


Split Dataset

Trained Data Test Data

S No.	time_cr...	source...	destina...	protocol	length	info	target
2	0.128...	Router...	Broad...	ARP	60	Who h...	1
3	0.178...	Router...	Broad...	ARP	60	Who h...	1
4	0.211...	192.1...	239.2...	SSDP	217	M-SE...	1
5	0.259...	192.1...	36.25...	TLSv1.2	249	Applic...	2
6	0.287...	36.25...	192.1...	TLSv1.2	132	Applic...	2
7	0.287...	192.1...	36.25...	TLSv1.2	531	Applic...	2
8	0.287...	36.25...	192.1...	TLSv1.2	316	Applic...	2
9	0.287...	36.25...	192.1...	TLSv1.2	87	Applic...	2
10	0.287...	192.1...	36.25...	TCP	54	55422 ...	2
11	0.292...	36.25...	192.1...	TLSv1.2	97	Applic...	2
12	0.307...	Fortin...	Broad...	ARP	60	Who h...	1
13	0.313...	36.25...	192.1...	TLSv1.2	132	Applic...	2
14	0.314...	192.1...	36.25...	TCP	54	55422 ...	2
15	0.314...	36.25...	192.1...	TLSv1.2	504	Applic...	2
16	0.314...	36.25...	192.1...	TLSv1.2	87	Applic...	2

TEST DATA:

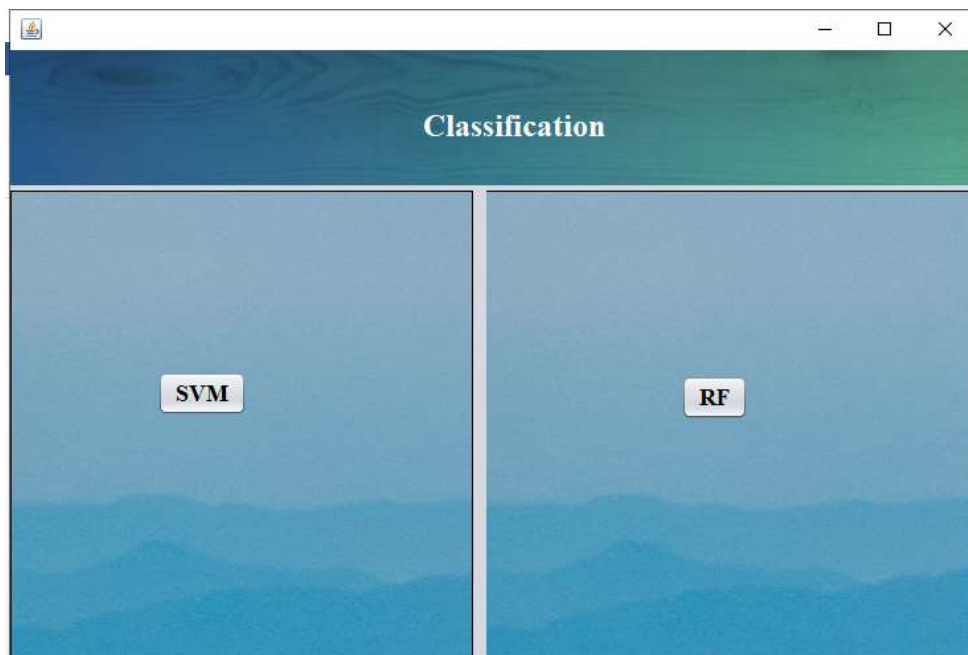


Split Dataset

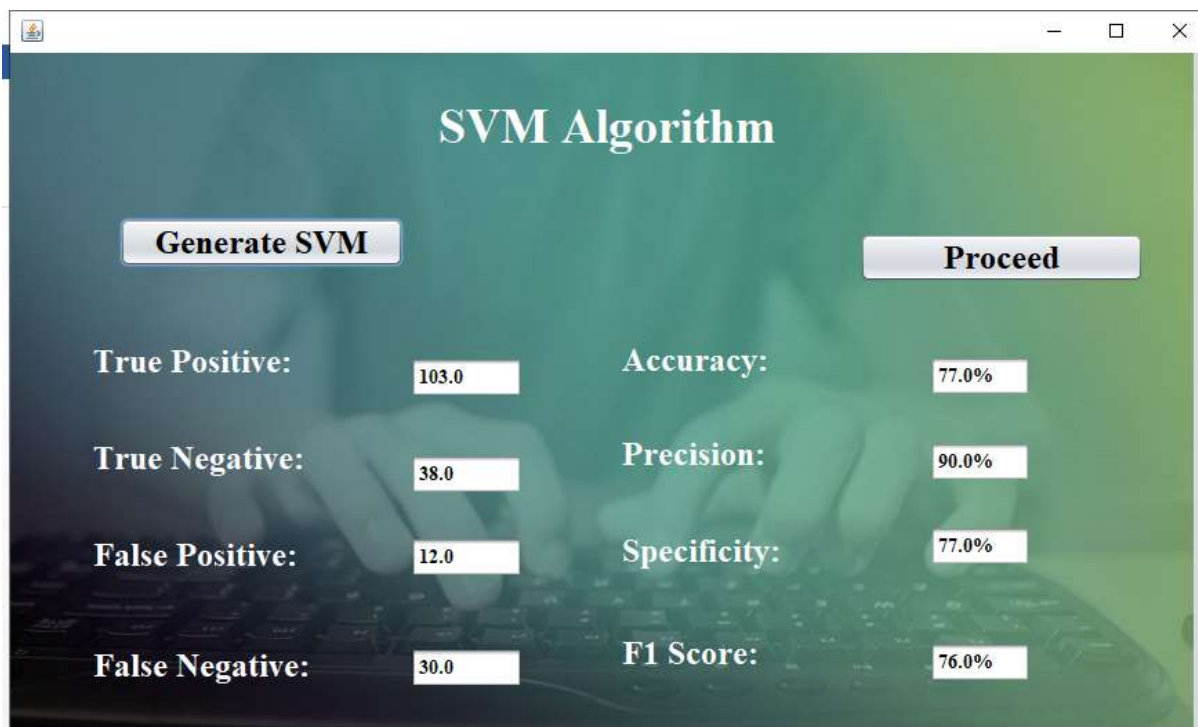
Trained Data Test Data

S No.	time_cr...	source...	destina...	protocol	length	info	target
161	5.008...	Router...	Broad...	ARP	60	Who h...	1
162	5.108...	Router...	Broad...	ARP	60	Who h...	1
163	5.194...	Router...	Broad...	ARP	60	Who h...	1
164	5.253...	Router...	Broad...	ARP	60	Who h...	1
165	5.307...	Fortin...	Broad...	ARP	60	Who h...	1
166	5.37994	192.1...	224.0...	MDNS	74	"Stand...	""QM...
167	5.380...	192.1...	224.0...	MDNS	74	"Stand...	""QM...
168	5.384...	192.1...	192.1...	BRO...	216	Get Ba...	2
169	5.384...	192.1...	192.1...	NBNS	92	Name ...	2
170	5.398...	Router...	Broad...	ARP	60	Who h...	1
171	5.417...	192.1...	239.2...	SSDP	217	M-SE...	1
172	5.438...	Router...	Broad...	ARP	60	Who h...	1
173	5.495...	192.1...	239.2...	SSDP	217	M-SE...	1
174	5.564...	Router...	Broad...	ARP	60	Who h...	1
175	5.697...	192.1...	239.2...	SSDP	217	M-SE...	1

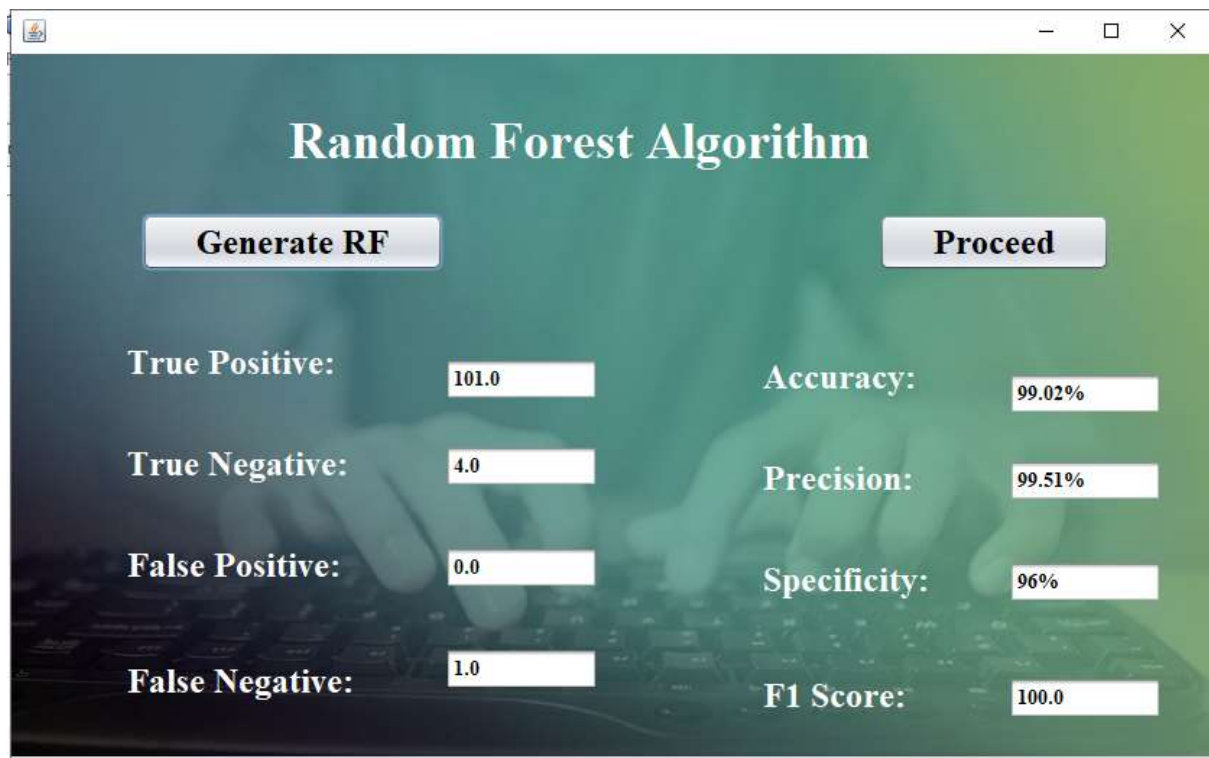
CLASSIFICATION ALGORITHMS:



CLASSIFICATION(SVM):



CLASSIFICATION(RF):



A screenshot of a web application titled "Random Forest Algorithm". It features two buttons at the top: "Generate RF" and "Proceed". Below these, there are eight input fields arranged in two columns. The left column contains "True Positive:", "True Negative:", "False Positive:", and "False Negative:". The right column contains "Accuracy:", "Precision:", "Specificity:", and "F1 Score:". Each label is followed by a text input field containing a numerical value.

Metric	Value
True Positive:	101.0
True Negative:	4.0
False Positive:	0.0
False Negative:	1.0
Accuracy:	99.02%
Precision:	99.51%
Specificity:	96%
F1 Score:	100.0

PREDICT:



A screenshot of a web application titled "Prediction". It contains several input fields for user data and a "Predict" button. The fields are arranged in two columns. The left column contains "Time Created:", "Source_ip:", and "Destination_ip:". The right column contains "Protocol:", "Length:", and "Info:". Each label is followed by a text input field. The "Protocol:" field is a dropdown menu. At the bottom, there is a "Predict" button and a "Result:" label followed by a text input field.

Field	Value
Time Created:	0.259913
Source_ip:	2.168.0.53
Destination_ip:	55.253.181
Protocol:	TLS
Length:	249
Info:	ration Data
Predict	
Result:	Not Attack

REFERENCES

1. Moghaddass, Ramin and Wang, Jianhui. "A Hierarchical Framework for smart grid Anomaly Detection Using Large-Scale Smart Meter Data." IEEE Transactions on smart grid (2018).
2. P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," International Journal of Critical Infrastructure Protection, 2008.
3. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purpy, and D. Kundur, "Implementing attacks for Modbus/tcp protocol in a real-time cyber physical system test bed," in 2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR). IEEE, 2015.
4. S.-C. Li, Y. Huang, B.-C. Tai, and C.-T. Lin, "Using data mining methods to detect simulated intrusions on a Modbus network," in 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2). IEEE, 2017.
5. A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A Modbus/tcp fuzzer for testing internetworked industrial systems," in 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA). IEEE, 2015.
6. Bhatia, N. Kush, C. Djamaludin, A. Akande, and E. Foo, "Practical Modbus flooding attack and detection," in Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)
7. T. H. Morris, B. A. Jones, R. B. Vaughn, and Y. S. Dandass, "Deterministic intrusion detection rules for Modbus protocols," in 2013 46th Hawaii International Conference on System Sciences. IEEE, 2013.
8. N. Goldenberg and A. Wool, "Accurate modeling of Modbus/tcp for intrusion detection in scada systems," international journal of critical infrastructure protection, 2013.
9. S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial Modbus/tcp data set," in Proceedings of the 13th International Conference on Availability, Reliability and Security, 2018.
10. P.-H. Wang, I.-E. Liao, K.-F. Kao, and J.-Y. Huang, "An intrusion detection method based on log sequence clustering of honeypot for Modbus tcp protocol," in 2018 IEEE International Conference on Applied System Invention (ICASI). IEEE, 2018.

11. Bashendy, May, et al. "Design and implementation of cyber-physical attacks on Modbus/tcp protocol." World Congress on Industrial Control Systems Security (WCICSS-2020). 2020.
12. P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems," 2019.
13. P. Radoglou-Grammatikis, I. Siniosoglou, T. Liatifis, A. Kourouniadis, K. Rompolos and P. Sarigiannidis, "Implementation and Detection of Modbus Cyberattacks," 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, Germany, 2020.
14. J. Luswata, P. Zavarisky, B. Swar, and D. Zvabva, "Analysis of scada security using penetration testing: A case study on Modbus tcp protocol," in 2018 29th Biennial Symposium on Communications (BSC), June 2018.
15. Ametov, F. R., E. A. Bekirov, and M. M. Asanov. "Organizing the information security in Modbus TCP interfaces for use in the energy complex." IOP Conference Series: Materials Science and Engineering, 2021.
16. Zakaria El Mrabet, Naima Kaabouch, Hassan El Ghazi, Hamid El Ghazi, Cyber-security in smart grid: Survey and challenges, Computers & Electrical Engineering, 2018.
17. D. Von Doller, Report to NIST on the Smart Grid Interoperability Standards Roadmap, Electric Power Research Institute (EPRI) and National Institute of Standards and Technology.
18. Ayesha Rahman, Ghulam Mustafa, Abdul Qayyum Khan, Muhammad Abid, Muhammad Hanif Durad, Launch of denial of service attacks on the Modbus/TCP protocol and development of its protection mechanisms, 2022.
19. Morris, Thomas H., and Wei Gao. "Industrial control system cyber attacks." 1. 2013.
20. Ortega-Fernandez, I.; Liberati, F. A Review of Denial of Service Attack and Mitigation in the Smart Grid Using Reinforcement Learning. Energies 2023.
21. Zhang, H., Min, Y., Liu, S. et al. Improve the Security of Industrial Control System: A Fine-Grained Classification Method for DoS Attacks on Modbus/TCP. Mobile Netw Appl (2023).
22. Pasiadis, A., Kotsiopoulos, T., Lazaridis, G., Drosou, A., Tzovaras, D., Sarigiannidis, P. (2023). Cyber-Resilience Enhancement Framework in Smart Grids. In: Haes Alhelou, H., Hatziargyriou, N., Dong, Z.Y. (eds) Power Systems Cybersecurity. Power Systems. Springer, Cham.

23. J. J. Fritz, J. Sagisi, J. James, A. S. Leger, K. King and K. J. Duncan, "Simulation of Man in the Middle Attack On Smart Grid Testbed," SoutheastCon, Huntsville, AL, USA, 2019.
24. Khan, A.A., Beg, O.A. (2023). Cyber network vulnerabilities of Modern Power Systems. In: Haes Alhelou, H., Hatziargyriou, N., Dong, Z.Y. (eds) Power Systems Cybersecurity. Power Systems. Springer, Cham.
25. Ulysse Boudier. Design of a Prototype for Inverter Monitoring with SunSpec Modbus Protocol. Energy Technology EGI-2017

ANNEXURE

SAMPLE CODING

View data:

```
package mimt_honeypot;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author EGC
```

```

*/

public class View_data extends javax.swing.JFrame {

    /**
     * Creates new form View_data
     */
    public View_data() {
        initComponents();
    }
    Vector columnNames =new Vector();
    Vector data1=new Vector();
    Vector columnNames1 = new Vector();
    Vector data2 = new Vector();
    DefaultTableModel model,model1;

    View_data(String filepath) {
        String path=filepath; //To change body of generated methods, choose Tools
| Templates.
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

```

```

jScrollPane4 = new javax.swing.JScrollPane();
jTable4 = new javax.swing.JTable();
jPanel1 = new javax.swing.JPanel();
jButton10 = new javax.swing.JButton();
jButton12 = new javax.swing.JButton();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

jTable4.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
    new String [] {

        ""
    }
));
jScrollPane4.setViewportView(jTable4);

getContentPane().add(jScrollPane4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(210, 110, 463, 355));

jPanel1.setBackground(new java.awt.Color(0, 102, 204));
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

```

```

        jButton10.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        jButton10.setText("View");
        jButton10.setContentAreaFilled(false);
        jButton10.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jButton10.setOpaque(true);
        jButton10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton10ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(45, 197, -1, -1));

```

```

        jButton12.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        jButton12.setText("Proceed");
        jButton12.setContentAreaFilled(false);
        jButton12.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jButton12.setOpaque(true);
        jButton12.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton12ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(45, 347, -1, -1));

```

```
        getContentPane().add(jPanel1, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 160, 500));
```

```
        jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
```

```
        jLabel1.setText("View Data");
```

```
        getContentPane().add(jLabel1, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(362, 21, -1, -1));
```

```
        jLabel2.setIcon(new  
javax.swing.ImageIcon(getClass().getResource("/images/background-  
learner1.jpg"))); // NOI18N
```

```
        getContentPane().add(jLabel2, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(150, -20, 570, 530));
```

```
        pack();
```

```
    } // </editor-fold>
```

```
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
    try
```

```
    {
```

```
        System.out.println("conn established");
```

```
        Class.forName("com.mysql.jdbc.Driver");
```

```
        Connection con =
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/honeypot_mimt",  
"root", "");
```

```
        Statement stmt = con.createStatement();
```

```
        Statement stmt2 = con.createStatement();
```

```
        ResultSet rs2 = stmt2.executeQuery("SELECT * FROM data");
```



```

        System.out.println("conn established1");
        ResultSetMetaData rsmd = rs2.getMetaData();
        int columns = rsmd.getColumnCount();
        System.out.println("conn established2");
        for(int i=1;i<=columns;i++)
        {
            columnNames1.addElement(rsmd.getColumnName(i));
        }
        while(rs2.next())
        {
            Vector row=new Vector(columns);
            for(int i=1;i<=columns;i++)
            {
                row.addElement(rs2.getObject(i));
            }
            data2.addElement(row);
        }
        rs2.close();
        model1=new DefaultTableModel(data2, columnNames1);
        jTable4.setModel(model1);
    }
    catch(Exception e)
    {
        JOptionPane.showMessageDialog(this, e);
    }
}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

```

```

        this.setVisible(false);
        new Preprocess().setVisible(true);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(View_data.class.getName()).log(java.util.lo
                gging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {

```

```
java.util.logging.Logger.getLogger(View_data.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
    } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(View_data.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(View_data.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
    }
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new View_data().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JButton jButton10;
```

```
private javax.swing.JButton jButton12;
```

```
private javax.swing.JLabel jLabel1;
```

```
private javax.swing.JLabel jLabel2;
```

```
private javax.swing.JPanel jPanel1;
```

```
private javax.swing.JScrollPane jScrollPane4;
```

```
private javax.swing.JTable jTable4;
```

```
// End of variables declaration
```

```
}
```

Classification using RF:

```
package mimt_honeypot;
import static mimt_honeypot.RandomForest.accuracy;
import java.io.FileNotFoundException;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
/**
```

```
*
```

```
* @author EGC
```

```
*/
```

```
public class RF extends javax.swing.JFrame {
```

```
/**
```

```
* Creates new form RF
```

```
*/
```

```
public RF() {
    initComponents();
}
```

```
/**
```

```

* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is
always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

```

```

    jButton1 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel14 = new javax.swing.JLabel();
    jLabel16 = new javax.swing.JLabel();
    jLabel12 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jButton2 = new javax.swing.JButton();
    jTextField1 = new javax.swing.JTextField();
    jTextField2 = new javax.swing.JTextField();
    jTextField3 = new javax.swing.JTextField();
    jTextField4 = new javax.swing.JTextField();
    jTextField5 = new javax.swing.JTextField();
    jTextField6 = new javax.swing.JTextField();
    jTextField7 = new javax.swing.JTextField();
    jTextField8 = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();

```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
    getContentPane().setLayout(new  
org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
        jButton1.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jButton1.setText("Generate RF");
```

```
        jButton1.addActionListener(new java.awt.event.ActionListener() {  
            public void actionPerformed(java.awt.event.ActionEvent evt) {  
                jButton1ActionPerformed(evt);  
            }  
        });
```

```
        getContentPane().add(jButton1, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(90, 110, 209, 40));
```

```
        jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 36)); //  
NOI18N
```

```
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
```

```
        jLabel1.setText("Random Forest Algorithm");
```

```
        getContentPane().add(jLabel1, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(192, 37, 421, 46));
```

```
        jLabel10.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jLabel10.setForeground(new java.awt.Color(255, 255, 255));
```

```
        jLabel10.setText("True Positive: ");
```

```
        getContentPane().add(jLabel10, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 200, 159, 26));
```

```
jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N
```

```
jLabel6.setForeground(new java.awt.Color(255, 255, 255));
jLabel6.setText("True Negative: ");
getContentPane().add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 270, -1, 26));
```

```
jLabel8.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N
```

```
jLabel8.setForeground(new java.awt.Color(255, 255, 255));
jLabel8.setText("False Positive: ");
getContentPane().add(jLabel8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 340, 159, 26));
```

```
jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N
```

```
jLabel3.setForeground(new java.awt.Color(255, 255, 255));
jLabel3.setText("False Negative:");
getContentPane().add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 420, 159, 26));
```

```
jLabel14.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N
```

```
jLabel14.setForeground(new java.awt.Color(255, 255, 255));
jLabel14.setText("F1 Score:");
getContentPane().add(jLabel14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(520, 430, 159, 26));
```

```
jLabel16.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N
```

```
jLabel16.setForeground(new java.awt.Color(255, 255, 255));
jLabel16.setText("Specificity:");
```

```
        getContentPane().add(jLabel16, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(520, 350, 159, 26));
```

```
        jLabel12.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jLabel12.setForeground(new java.awt.Color(255, 255, 255));  
        jLabel12.setText("Precision:");  
        getContentPane().add(jLabel12, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(520, 280, 159, 26));
```

```
        jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jLabel4.setForeground(new java.awt.Color(255, 255, 255));  
        jLabel4.setText("Accuracy:");  
        getContentPane().add(jLabel4, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(520, 210, 159, 26));
```

```
        jButton2.setFont(new java.awt.Font("Times New Roman", 1, 24)); //  
NOI18N
```

```
        jButton2.setText("Proceed");  
        jButton2.addActionListener(new java.awt.event.ActionListener() {  
            public void actionPerformed(java.awt.event.ActionEvent evt) {  
                jButton2ActionPerformed(evt);  
            }  
        });  
        getContentPane().add(jButton2, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(600, 110, 160, 40));
```

```
        jTextField1.setFont(new java.awt.Font("Times New Roman", 1, 14)); //  
NOI18N
```

```
        getContentPane().add(jTextField1, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(300, 210, 106, -1));
```



```
        jTextField2.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(300, 270, 106, -1));
```

```
        jTextField3.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(300, 340, 106, -1));
```

```
        jTextField4.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(300, 410, 106, -1));
```

```
        jTextField5.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(690, 430, 106, -1));
```

```
        jTextField6.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(690, 350, 106, -1));
```

```
        jTextField7.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(690, 280, 106, -1));
```

```
        jTextField8.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
```

```
        getContentPane().add(jTextField8, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(690, 220, 106, -1));
```

```
        jLabel5.setForeground(new java.awt.Color(255, 255, 255));  
        jLabel5.setIcon(new  
javax.swing.ImageIcon(getClass().getResource("/images/background-  
learner1.jpg"))); // NOI18N
```

```
        getContentPane().add(jLabel5, new  
org.netbeans.lib.awtextra.AbsoluteConstraints(0, -20, 830, 520));
```

```
        pack();  
    } // </editor-fold>
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    try {  
        File trainfile = new File("train.csv");  
        File testfile = new File("test.csv");  
        //System.out.println("file name:"+trainfile.getName());  
        List<String[]> traincontent = new ArrayList<>();  
        List<String[]> testcontent = new ArrayList<>();
```

```
        BufferedReader trainreadFile,testreadFile = null;
```

```
        trainreadFile = new BufferedReader(new FileReader(trainfile));
```

```
        testreadFile = new BufferedReader(new FileReader(testfile));
```

```
        String trainline,testline;
```

```
        while ((trainline = trainreadFile.readLine()) != null) {
```

```

        traincontent.add(trainline.split(", "));
    }
    while ((testline = testreadFile.readLine()) != null) {
        testcontent.add(testline.split(", "));
    }
    RandomForest rf = new RandomForest();
    rf.Start();

    jTextField1.setText(Double. toString(RandomForest.TP));
    jTextField2.setText(Double. toString(RandomForest.TN));
    jTextField3.setText(Double. toString(RandomForest.FP));
    jTextField4.setText(Double. toString(RandomForest.FN));
    //String acc1=(RandomForest.accuracy);
    String recall1=String.format("%.2f",RandomForest.Recall);
    jTextField8.setText(recall1+"%");
    String f1=String.format("%.2f",RandomForest.f1_score);
    jTextField7.setText(f1+"%");
    jTextField6.setText(RandomForest.accuracy);
    jTextField5.setText(Double. toString(RandomForest.Precision));

}
catch (FileNotFoundException ex) {
    Logger.getLogger(RF.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
    Logger.getLogger(RF.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // new Userinputs().setVisible(true);
    new Predict().setVisible(true);
    this.setVisible(false);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(RF.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);

    } catch (InstantiationException ex) {

```

```
java.util.logging.Logger.getLogger(RF.class.getName()).log(java.util.logging.L  
evel.SEVERE, null, ex);
```

```
    } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(RF.class.getName()).log(java.util.logging.L  
evel.SEVERE, null, ex);
```

```
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(RF.class.getName()).log(java.util.logging.L  
evel.SEVERE, null, ex);
```

```
    }
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new RF().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JButton jButton1;
```

```
private javax.swing.JButton jButton2;
```

```
private javax.swing.JLabel jLabel1;
```

```
private javax.swing.JLabel jLabel10;
```

```
private javax.swing.JLabel jLabel12;
```

```
private javax.swing.JLabel jLabel14;
```

```
private javax.swing.JLabel jLabel16;
```

```
private javax.swing.JLabel jLabel3;
```

```
private javax.swing.JLabel jLabel4;  
private javax.swing.JLabel jLabel5;  
private javax.swing.JLabel jLabel6;  
private javax.swing.JLabel jLabel8;  
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;  
private javax.swing.JTextField jTextField7;  
private javax.swing.JTextField jTextField8;  
// End of variables declaration  
}
```