

SENTRY SHIELD FOR WIRELESS DATA PACKETS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

DHARUN KRITHIK.K (411720104013)

GOPINATH.V (411720104018)

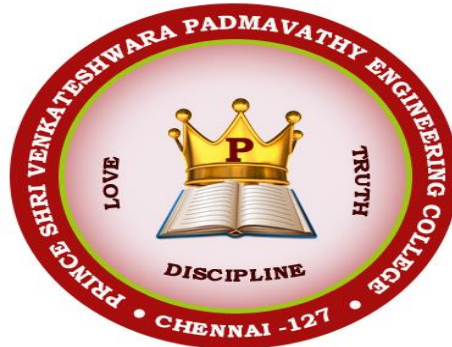
in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



**PRINCE SHRI VENKATESHWARA PADMAVATHY ENGINEERING
COLLEGE [AN AUTONOMOUS INSTITUTION], CHENNAI-127**

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2024

BONAFIDE CERTIFICATE

Certified that this project report **“SENTRY SHIELD FOR WIRELESS DATA PACKETS USING MACHINE LEARNING”** is the bonafide work of **“DHARUN KRITHIK.K (411720104013), GOPINATH.V (411720104018)”**, who carried out the project work under my supervision.

SIGNATURE

Dr. M. PREETHA, M.TECH., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Computer Science and
Engineering
Prince Shri Venkateshwara
Padmavathy Engineering College,
Chennai-127

SIGNATURE

Mrs. S. Bhuvaneswari, M.E

SUPERVISOR

Assistant Professor

Department of Computer Science
and Engineering
Prince Shri Venkateshwara
Padmavathy Engineering College,
Chennai-127

Submitted for viva-voce held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, we bow our head to the Almighty for being our light and for his gracious showers of blessing throughout the course of this project.

We would like to express our sincere thanks to our founder and Chairman, **Dr. K. Vasudevan, M.A., B.Ed., Ph.D.**, for his endeavor in educating us in his premier institution. We are grateful to our Vice Chairman, **Dr. V. Vishnu Karthik, M.D.**, for his keen interest in our studies and the facilities offered in the premier institution. We would like to express our sincere gratitude to our Administrative Officer **Mr. K. Parthasarathy, B.E.**, for his assistance in all our endeavors.

We thank our Principal, **Dr. G. Indira, M.E., Ph.D.**, for her valuable support and encouragement in all our efforts throughout this course. We would like to express our sincere thanks to our Dean Academics, **Dr. V. Mahalakshmi, M.E., Ph.D.**, for her great support and encouragement and moral support given to us during this course.

We would like to express our sincere thanks to our beloved Head of the Department, **Dr. M. Preetha, M.TECH., Ph.D.**, for his support and providing us with ample time to complete our project. We wish to express our great deal of gratitude to our project Co-Ordinator, **Dr. M. Preetha, M.TECH., Ph.D.**, and our Guide, **Mrs. S. Bhuvaneswari, M.E.**, for their cooperation towards us at all times of need, for their guidance and valuable suggestions in every aspect for completion of this project.

We are also thankful to all faculty members and non-teaching staffs of all Departments for their support. Finally, we are grateful to our family and friends for their help, encouragement and moral support given to us during our project work.

ABSTRACT

Network attacks pose a significant threat to the security and integrity of computer networks. The ability to predict and prevent these attacks is crucial for maintaining a secure network environment. Supervised machine learning techniques have emerged as effective tools for network attack prediction due to their ability to analyse large amounts of network data and identify patterns indicative of malicious activity. We present a comprehensive analysis of supervised machine learning techniques for the prediction of network attacks. We collect and pre-process the data, extracting relevant features and transforming them into a suitable format for machine learning algorithms. We evaluate the performance of these algorithms. We investigate the interpretability of the trained models to gain insights into the underlying patterns and characteristics of network attacks. This allows network administrators to understand the nature of attacks and develop appropriate defences strategies. Additionally, we discuss the challenges and limitations associated with the application of supervised machine learning techniques in the domain of network attack prediction, such as the need for real-time analysis and the emergence of sophisticated evasion techniques.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE.NO
	ABSTRACT	i
	LIST OF FIGURES	v
	LIST OF SYMBOLS	vi
01.	INTRODUCTION	1
	1.1 DOMAIN OVERVIEW	
	1.1.1 DATA SCIENCE	
02.	LITERATURE SURVEY	8
03.	EXISTING SYSTEM	11
	2.1 DEMERITS	
04.	PROPOSED SYSTEM	12
	3.1 PREPARING DATASET	
	3.2 MERITS	
05.	SYSTEM STUDY	13
	5.1 OVERVIEW OF SYSTEM	
	5.2 AIM	
	5.3 OBJECTIVE	
	5.4 SCOPE	
	5.5 CONSTRUCTION OF PREDICTIVE MODEL	
06.	PROJECT REQUIREMENTS	15
	6.1 GENERAL	
07.	DESIGN ARCHITECTURE	17
	7.1 SYSTEM ARCHITECTURE	

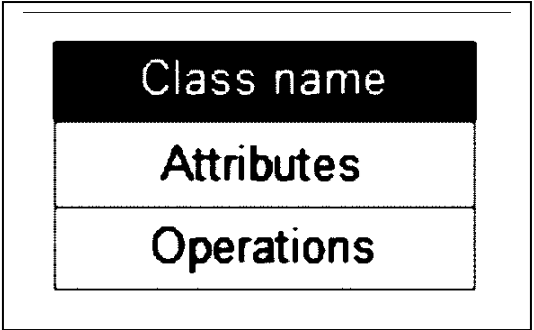
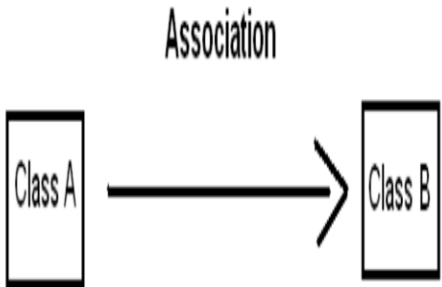
	7.2 WORKFLOW DIAGRAM	
	7.3 USE CASE DIAGRAM	
	7.4 CLASS DIAGRAM	
	7.5 ACTIVITY DIAGRAM	
	7.6 SEQUENCE DIAGRAM	
	7.7 ENTITY RELATIONSHIP DIAGRAM	
	7.8 COLLOBORATION DIAGRAM	
08.	SOFTWARE DESCRIPTION	25
	8.1 ABOUT SOFTWARE	
	8.2 ANACONDA NAVIGATOR	
	8.3 ANACONDA	
	8.4 JUPYTER NOTEBOOK	
	8.5 PYTHON	
09.	PROJECT MODULE	37
	9.1 LIST OF MODULES	
	9.2 MODULE DESCRIPTION	
	9.2.1 DATA PRE PROCESSING	
	9.2.2 EXPLORATION DATA ANALYSIS OF VISUALIZATION	
	9.2.3 RIDGE CLASSIFIER	
	9.2.4 RANDOM FOREST CLASSIFIER	
	9.2.5 BERNOULINNB	
10.	RESULT	64
11.	CONCLUSION & FUTURE WORK	65




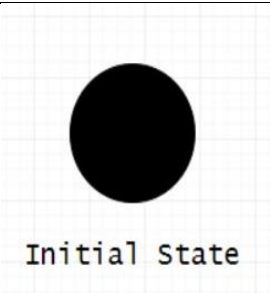
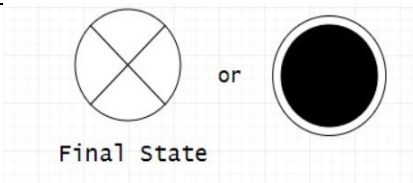
APPENDICES	66
A1. CODING	
A2. SCREEN SHOTS	
REFERENCES	90
PUBLICATIONS	92

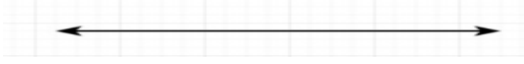
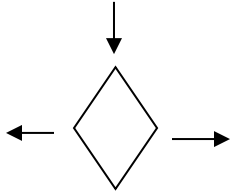
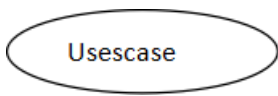
LIST OF FIGURES

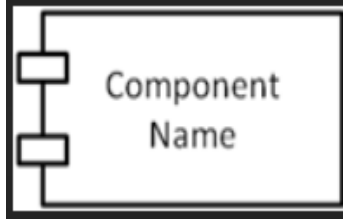
SL.NO	TITLE	PAGE.NO
1.1	PROCESS OF MACHINE LEARNING	7
5.1	PROCESS OF DATA FLOW DIAGRAM	14
7.1	ARCHITECTURE DIAGRAM	17
7.2	WORKFLOW DIAGRAM	18
7.3	USE CASE DIAGRAM	19
7.4	CLASS DIAGRAM	20
7.5	ACTIVITY DIAGRAM	21
7.6	SEQUENCE DIAGRAM	22
7.7	ENTITY RELATIONSHIP DIAGRAM	23
7.8	COLLABORATION DIAGRAM	24
8.1	ANACONDA NAVIGATOR	26
8.2	ANACONDA NAVIGATOR PAGE	27
9.1	DATA PRE-PROCESSING	46
9.2	DATA VISUALIZATION	47
9.3	RIDGE CLASSIFIER	49
9.4	RANDOM FOREST CLASSIFIER	51
9.5	BERNOULLINB	53
10.1	FINAL OUTPUT	64
A2.1	HOME PAGE	87
A2.2	MODEL ANALYSIS PAGE	87
A2.3	DETAILS PAGE	88
A2.4	DETECTION PAGE	88
A2.5	DOMAIN DETAIL PAGE	89

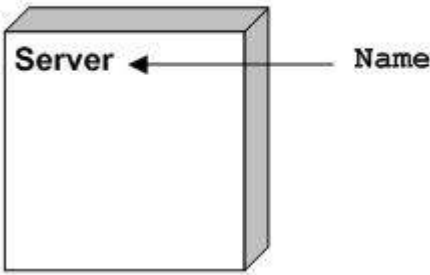
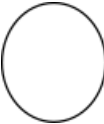
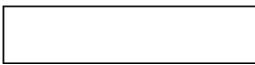

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.

4.	<i>Relation</i> (uses)		Used for additional process communication.
5.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
6.	State		State of the process.
7.	Initial State		Initial state of the object
8.	Final state		Final state of the object

9.	Control flow		Represents various control flow between the states.
10.	Decision box		Represents decision making process from a constraint
11.	Usecase		Interact ion between the system and external environment.

12.	Component		Represents physical modules which is a collection of components.
-----	-----------	---	--

13.	Node	<p style="text-align: center;">Node</p> 	Represents physical modules which are a collection of components.
14.	Data Process/ State		A circle in DFD represents a state or process which has been triggered due to some event or action.
15.	External entity		Represents external entities such as keyboard,sensors,etc.
16.	Transition		Represents communication that occurs between processes.

CHAPTER-1

INTRODUCTION

The most devastating and complicated attack in a wireless sensor network is the Wormhole attack. In this attack, the attacker keeps track of the packets and makes a tunnel with other nodes of different communication networks, and thus the attacker passes the packets through this tunnel. And the outsider attack can be prevented by authentication and encryption techniques by launching a Sybil attack on the sensor network. In WSN the routing protocols in network has a unique identity. The figure demonstrates Sybil attack where an attacker node AD is present with multiple identities.

1.1 DOMAIN OVERVIEW:

1.1.1 DATA SCIENCE:

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data.

Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect.

For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding.

AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal

logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals.

To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence.

These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning.

AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can

perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.



FIGURE 1.1 PROCESS OF MACHINE LEARNING

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$.

CHAPTER-2

LITERATURE SURVEY

[1] Dr. G. Umarani Srikanth, etal, 2021 The networked systems become more and more pervasive and businesses still acquire a lot of sensitive data online, so that the quantity and class of cyber-attacks and network security breaches has risen dramatically. There are also instances that so many volumes of data are hacked even without the knowledge of the people concerned. So far setting an Intrusion Detection System (IDS), it is obvious to set the true working environment to model the possibilities of attacks. Therefore, it is imperative to design a software that will be able to identify network intrusions, in order to protect a computer network from the unknown users. For overcoming this challenge, it is essential to predict whether the connection is targeted or not from KDDCup99 dataset utilizing machine learning techniques. The objective of this work is to investigate machine learning based algorithms for enhancing packet connection transfers forecasting using ensemble learning voting classifier techniques. It is proposed to deploy AI-based technique to precisely anticipate the DOS, R2L, U2R, Probe and large assaults. Results showed that the viability of the proposed AI calculation strategy can be contrasted and the best exactness with accuracy, Recall and F1 Score.

[2] Qinghai Zhou, etal, 2023 Multi-sourced networks naturally appear in many application domains, ranging from bioinformatics, social networks, neuroscience to management. Although state-of-the-art offers rich models and algorithms to find various patterns when input networks are given, it has largely remained nascent on how vulnerable the mining results are due to the adversarial attacks. In this paper, we address the problem of attacking multinetwork mining through the way of deliberately perturbing the networks to alter the mining results. The key idea of the proposed method (ADMIRING) is effective and efficient influence functions on the

Sylvester equation defined over the input networks, which plays a central and unifying role in various multi-network mining tasks. The proposed algorithms bear three main advantages, including (1) effectiveness, being able to accurately quantify the rate of change of the mining results in response to attacks; (2) efficiency, scaling linearly with more than 100 speed-up over the straightforward implementation without any quality loss; and (3) generality, being applicable to a variety of multi-network mining tasks (e.g., graph kernel, network alignment, cross-network node similarity) with different attacking strategies (e.g., edge/node removal, attribute alteration).

[3] Jinyin Chen, etal, 2023 In network link prediction, it is possible to hide a target link from being predicted with a small perturbation on network structure. This observation may be exploited in many real world scenarios, for example, to preserve privacy, or to exploit financial security. There have been many recent studies to generate adversarial examples to mislead deep learning models on graph data. However, none of the previous work has considered the dynamic nature of real-world systems. In this work, we present the first study of adversarial attack on dynamic network link prediction (DNLP). The proposed attack method, namely time-aware gradient attack (TGA), utilizes the gradient information generated by deep dynamic network embedding (DDNE) across different snapshots to rewire a few links, so as to make DDNE fail to predict target links. We implement TGA in two ways: One is based on traversal search, namely TGATra; and the other is simplified with greedy search for efficiency, namely TGA-Gre. We conduct comprehensive experiments which show the outstanding performance of TGA in attacking DNLP algorithms.

[4] Jakub Breier, etal, 2023 Neural network implementations are known to be vulnerable to physical attack vectors such as fault injection attacks. As of now, these

attacks were only utilized during the inference phase with the intention to cause a misclassification. In this work, we explore a novel attack paradigm by injecting faults during the training phase of a neural network in a way that the resulting network can be attacked during deployment without the necessity of further faulting. In particular, we discuss attacks against ReLU activation functions that make it possible to generate a family of malicious inputs, which are called fooling inputs, to be used at inference time to induce controlled misclassifications. Such malicious inputs are obtained by mathematically solving a system of linear equations that would cause a particular behaviour on the attacked activation functions, similar to the one induced in training through faulting. We call such attacks fooling backdoors as the fault attacks at training phase inject backdoors into the network that allow an attacker to produce fooling inputs. We evaluate our approach against multi-layer perceptron networks and convolutional networks on a popular image classification task obtaining high attack success rates (from 60% to 100%) and high classification confidence when as little as 25 neurons are attacked, while preserving high accuracy on the originally intended classification task.

[5] Mohammad Abu Alsheikh, etal, 2015 Wireless sensor networks monitor dynamic environments that change rapidly over time. This dynamic behavior is either caused by external factors or initiated by the system designers themselves. To adapt to such conditions, sensor networks often adopt machine learning techniques to eliminate the need for unnecessary redesign. Machine learning also inspires many practical solutions that maximize resource utilization and prolong the lifespan of the network. In this paper, we present an extensive literature review over the period 2002-2013 of machine learning methods that were used to address common issues in wireless sensor networks (WSNs). The advantages and disadvantages of each proposed algorithm are evaluated against the corresponding problem.

CHAPTER-3

EXISTING SYSTEM

Distributed denial-of-service (DDoS) attacks continue to grow at a rapid rate plaguing Internet Service Providers (ISPs) and individuals in a stealthy way. Thus, intrusion detection systems (IDSs) must evolve to cope with these increasingly sophisticated and challenging security threats. Traditional IDSs are prone to zero-day attacks since they are usually signature-based detection systems. However, the lack of up-to-date labelled training datasets makes these ML/DL based IDSs inefficient. The privacy nature of these datasets and widespread emergence of adversarial attacks makes it difficult for major organizations to share their sensitive data. Federated Learning (FL) is gaining momentum from both academia and industry as a new sub-field of ML that aims to train a global statistical model across multiple distributed users, referred to as collaborators, without sharing their private data. Due to its privacy-preserving nature, FL has the potential to enable privacy-aware learning between a large number of collaborators. This paper presents a novel framework, called MiTFed that allows multiple software defined networks (SDN) domains (i.e., collaborators) to collaboratively build a global intrusion detection model without sharing their sensitive datasets. It a promising framework to cope with the new emerging security threats in SDN.

3.1 DEMERITS:

1. They did not use any machine learning algorithms.
2. Accuracy was low.
3. They did not build a deploy model.

CHAPTER-4

PROPOSED SYSTEM

We proposed a system to develop the project using machine learning algorithm. Recently, Machine learning and Artificial intelligence has played a big role in various industries for their improvement and development. So, we tried to implement machine learning algorithm to make them more securable. The aim of this project is about provide the thread to intimate the security to stop the thread before it impacts huge loss to organization or individuals. We collect the previous record of the attacks that had happened over these times. By collecting these records our machine learning algorithm tried to find out the pattern to that dataset. After finding those patterns the machine is able to predict the instance based on previous records. By doing that with various algorithm we can get high accuracy. We say our model good based on high accuracy values.

4.1 PREPARING DATASETS:

This Dataset contains 374662 records. It is classified into 5 classes.

1. Blackhole
2. Flooding
3. Gray hole
4. Normal
5. TDMA

4.2 MERITS:

1. We build a framework-based user friendly application using django.
2. We use multiple machine learning algorithms for train data.
3. Accuracy was improved.

CHAPTER-5

SYSTEM STUDY

5.1 OVERVIEW OF THE SYSTEM

The most devastating and complicated attack in a wireless sensor network is the Wormhole attack. In this attack, the attacker keeps track of the packets and makes a tunnel with other nodes of different communication networks, and thus the attacker passes the packets through this tunnel. And the outsider attack can be prevented by authentication and encryption techniques by launching a Sybil attack on the sensor network. In WSN the routing protocols in network has a unique identity. The figure demonstrates Sybil attack where an attacker node '__AD' is present with multiple identities.

5.2 AIM

WSN is one of the major factors in our major domain. The most devastating and complicated attack in a wireless sensor network is the Wormhole attack. In this attack, the attacker keeps track of the packets and makes a tunnel with other nodes of different communication networks, and thus the attacker passes the packets through this tunnel. So, this project can easily find out the Attack.

5.3 OBJECTIVE

This analysis aims to observe which features are most helpful in predicting the WSN of BLACKHOLE, FLOODING, GRAYHOLE, NORMAL and SCHEDULING to see the general trends that may help us in model selection and hyper parameter selection. To achieve used machine learning classification methods to fit a function that can predict the discrete class of new input.

5.4 SCOPE

Here the scope of the project is that investigates the Classification of the Wireless Sensor Network Attack with computer-based prediction could reduce errors and improve prediction outcomes. This suggestion is promising as data modelling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment that can help to significantly find out the Classification of Wireless Sensor Network Attack prediction.

5.5 CONSTRUCTION OF PREDICTIVE MODEL

Machine learning needs data gathering have lot of past data. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to preprocess, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving it.

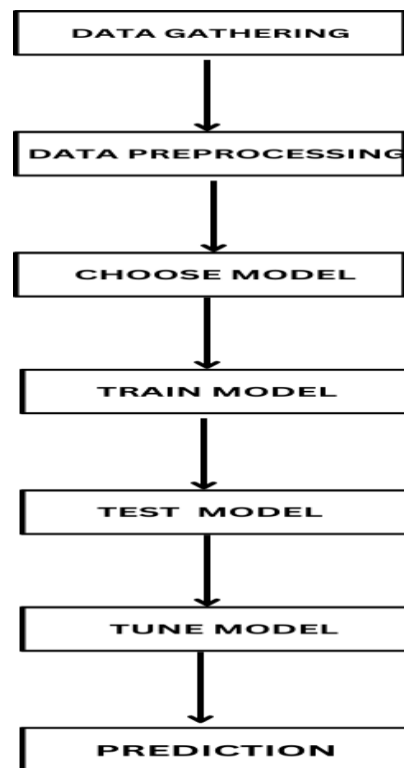


FIGURE 5.1 PROCESS OF DATAFLOW DIAGRAM

CHAPTER-6

PROJECT REQUIREMENTS

6.1 GENERAL

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results

5. Prediction the result

Environment Requirements:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 100 GB

RAM : minimum 4 GB

CHAPTER-7

DESIGN ARCHITECTURE

7.1 SYSTEM ARCHITECTURE

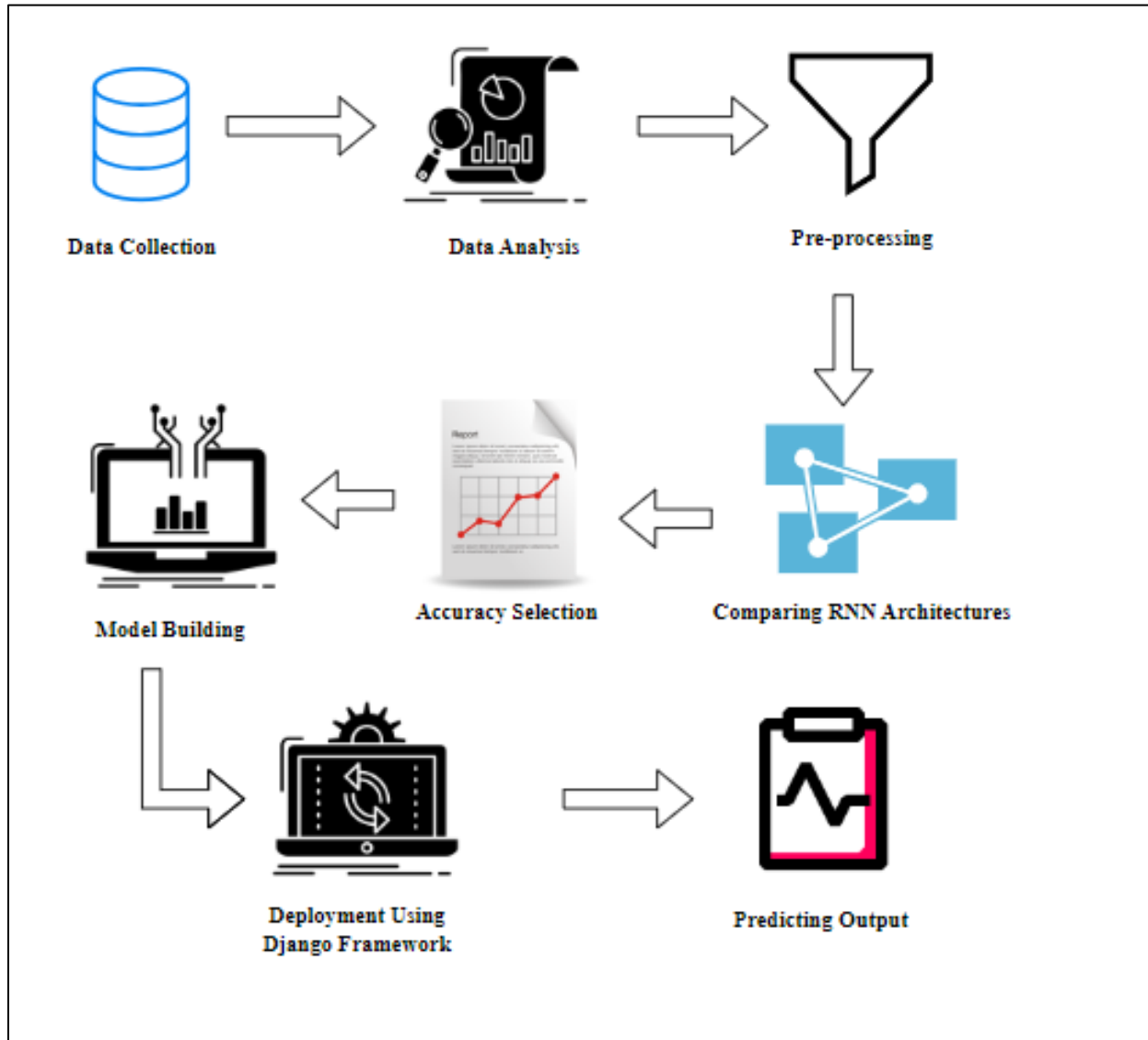


FIGURE 7.1 ARCHITECTURE DIAGRAM

7.2 WORKFLOW DIAGRAM

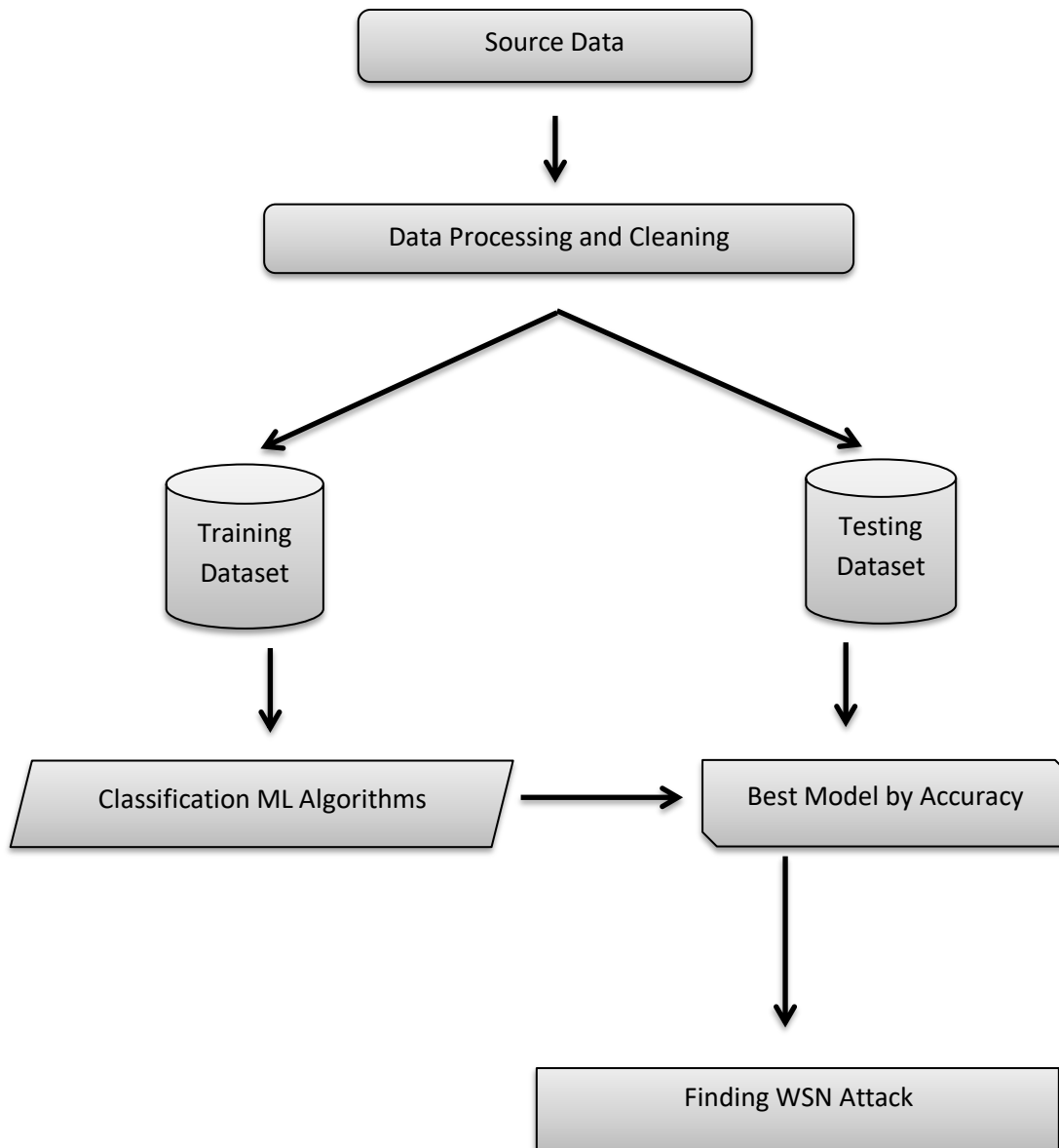


FIGURE 7.2 WORKFLOW DIAGRAM

7.3 USE-CASE DIAGRAM

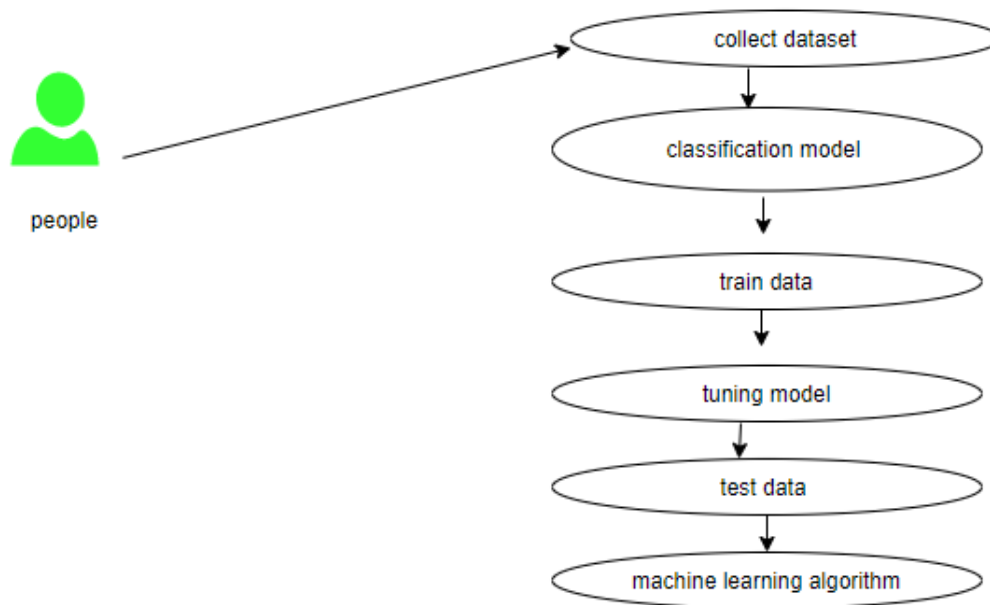


FIGURE 7.3 USE-CASE DIAGRAM

A use case diagram is a visual representation of the interactions between actors (users or external systems) and a system under consideration, depicting the various ways in which users interact with the system to achieve specific goals or tasks. It illustrates the functional requirements of a system by capturing the different use cases or scenarios that can occur, along with the actors involved in each scenario. Use case diagrams to help stakeholders understand the system's behavior, identify its functionalities, and clarify the system's boundaries and scope.

7.4 CLASS DIAGRAM

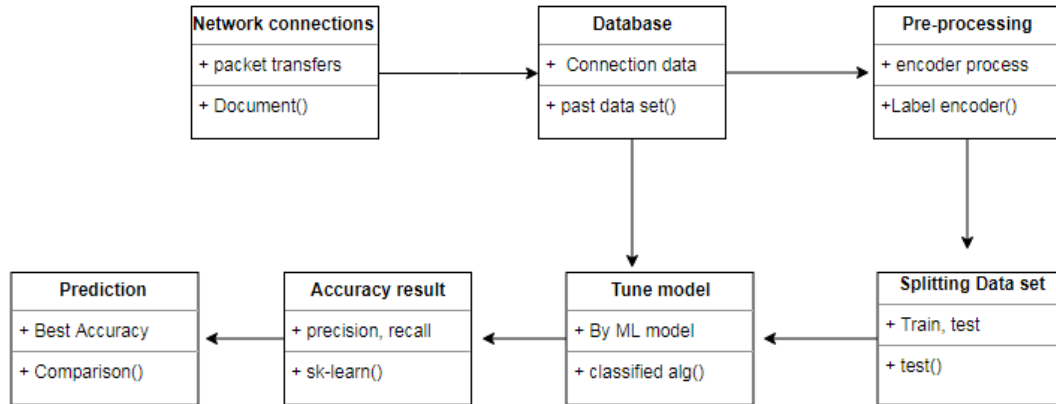


FIGURE 7.4 CLASS DIAGRAM

The class diagram is a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and its relationships should be identified in advance. Responsibility (attributes and methods) of each class should be identified for each class. Minimum number of properties should be specified because unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

7.5 ACTIVITY DIAGRAM

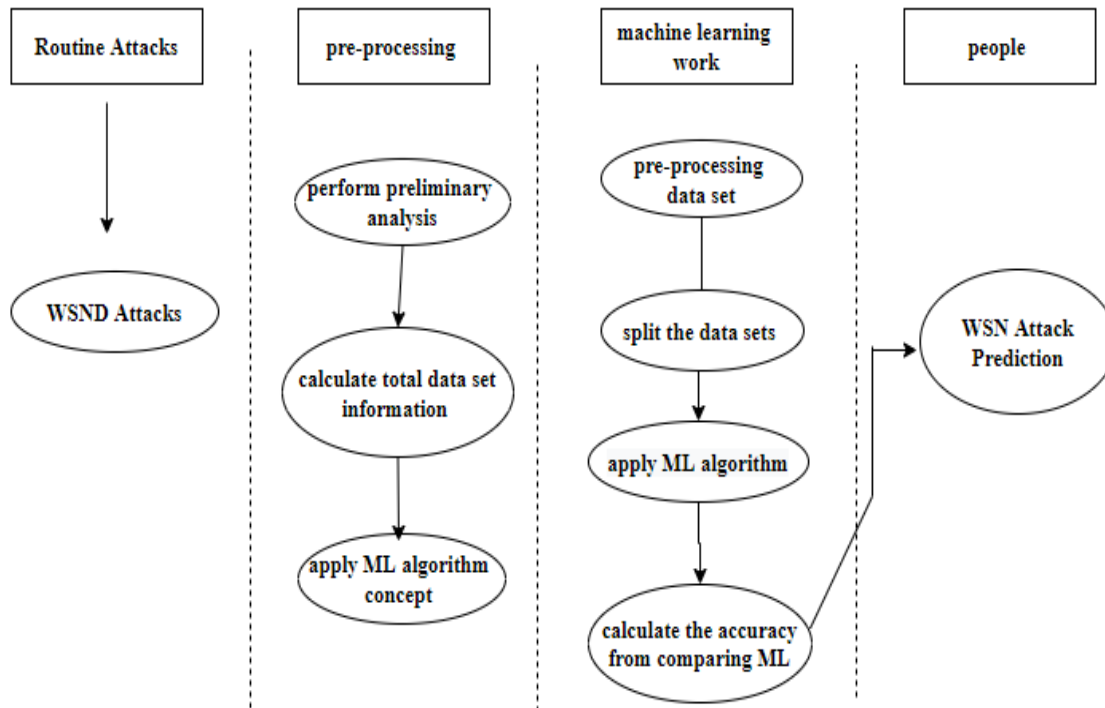


FIGURE 7.5 ACTIVITY DIAGRAM

An activity diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities within a system or process. It provides a graphical representation of the sequence of actions or steps involved in completing a task or achieving a goal, showcasing the transitions between different activities and decision points. Activity diagrams typically use symbols such as circles, rectangles, diamonds, and arrows. They are useful for modeling complex workflows, business processes, software algorithms, and system behaviors, enabling stakeholders to visualize and analyze the logical flow of activities and identify potential bottlenecks or areas for optimization.

7.6 SEQUENCE DIAGRAM

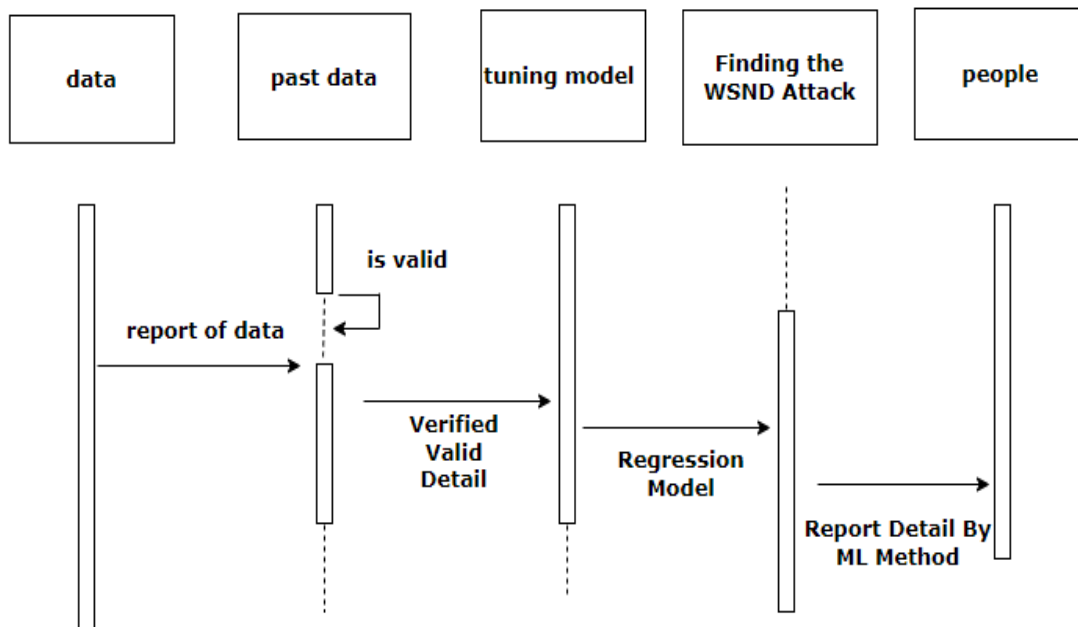


FIGURE 7.6 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behavior within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

7.7 ENTITY RELATIONSHIP DIAGRAM

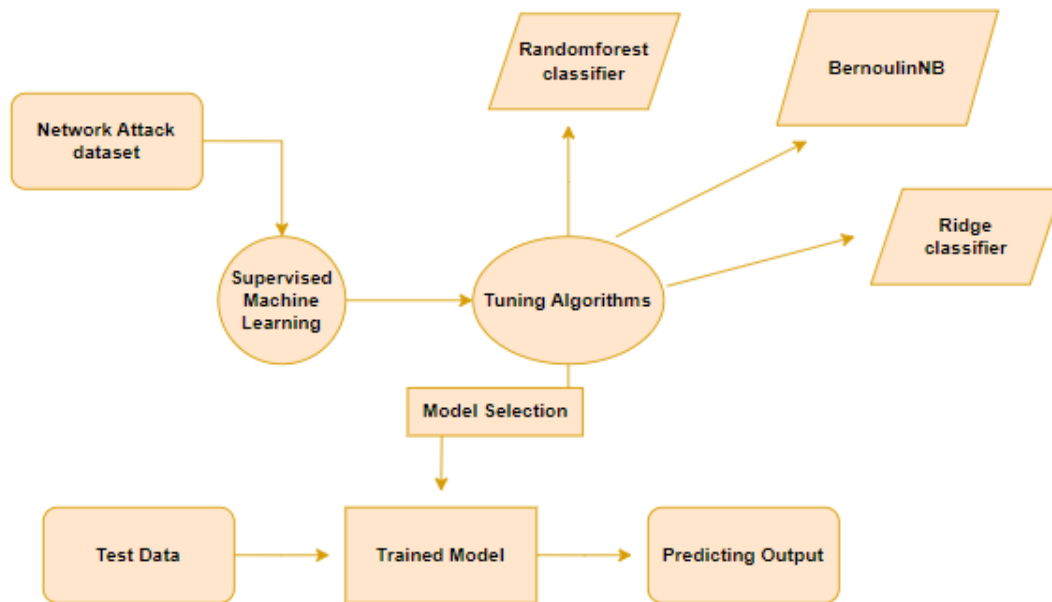


FIGURE 7.7 ENTITY RELATIONSHIP DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts, or events within that system. An ERD is a data modelling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

7.8 COLLOBORATION DIAGRAM

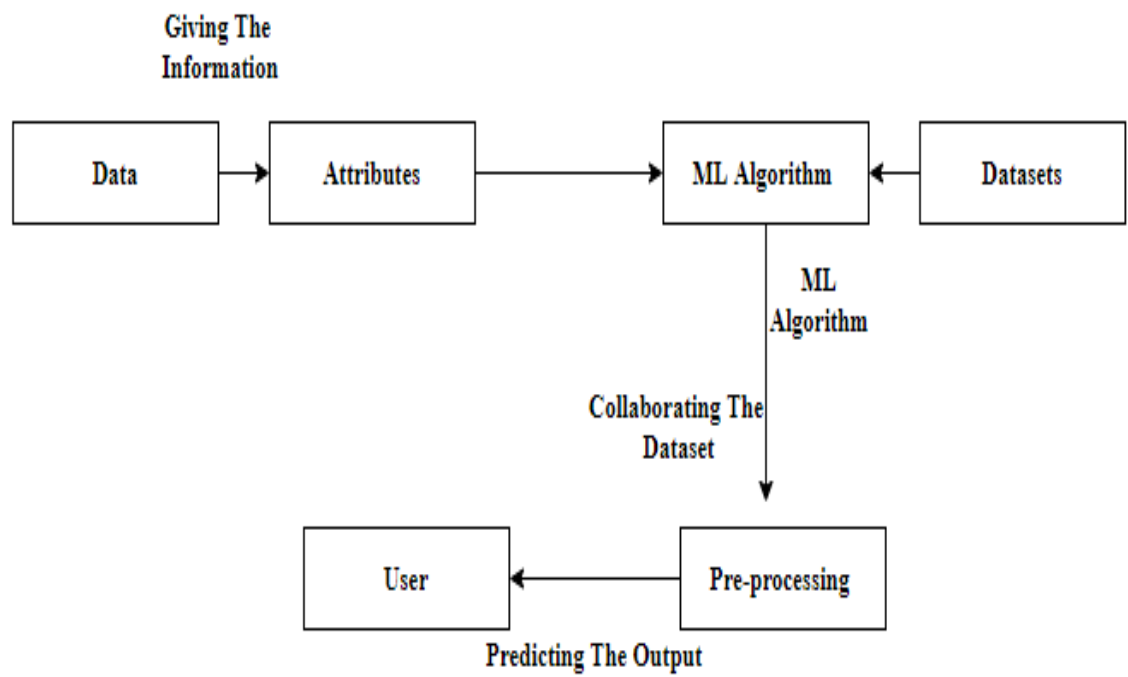


FIGURE 7.8 COLLABORATION DIAGRAM

CHAPTER-8

SOFTWARE DESCRIPTION

8.1 ABOUT SOFTWARE

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

8.2 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage

conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

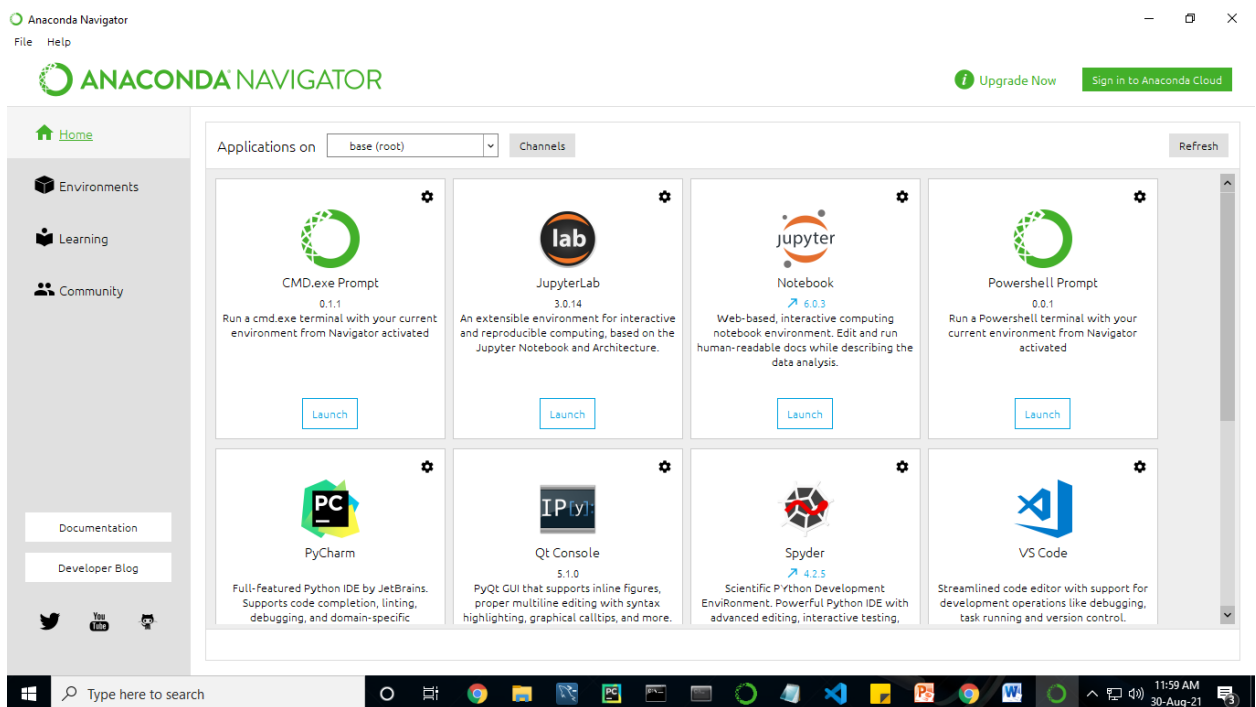


FIGURE 8.1 ANACONDA NAVIGATOR

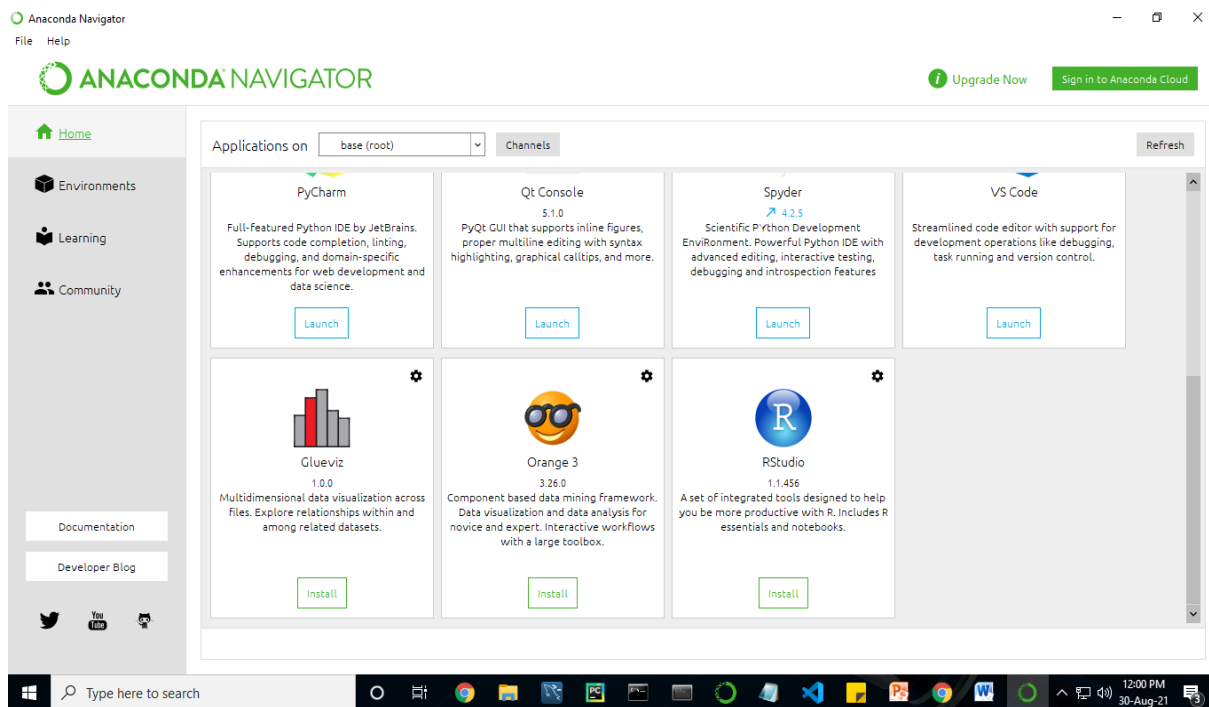


FIGURE 8.2 ANACONDA NAVIGATOR PAGE

8.3 ANACONDA

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user

interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

8.4 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Running the Jupyter Notebook

Launching Jupyter Notebook App: The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook *icon* installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser.

The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu *Cell -> Run All*), the kernel performs the computation and produces the results.

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

8.5 PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC

programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.

- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow:

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).

- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behaviour of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point division. Python also uses the ** operator for exponentiation.
- From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.

- From Python 3.8, the syntax: `=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals ()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` Used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using `lambda` expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).

Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops

- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1: ...` causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

CHAPTER-9

PROJECT MODULES

9.1 LIST OF MODULES

- Data Pre-processing
- Data Analysis and Visualization
- Ridge classifier
- Randomforest classifier
- Bernoulli NB
- Deployment

9.2 MODULE DESCRIPTION

9.2.1.DATA PRE-PROCESSING

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers

uses this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focuses on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of the sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset

- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

Data Validation/ Cleaning/Preparing Process

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

MODULE DIAGRAM

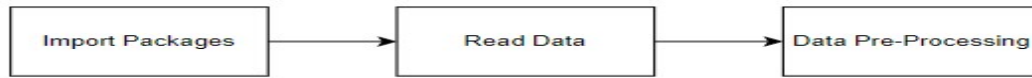


FIGURE 9.1 DATA PRE-PROCESSING

GIVEN INPUT EXPECTED OUTPUT

input: data

output: removing noisy data

9.2.2 EXPLORATION DATA ANALYSIS OF VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning.

It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

MODULE DIAGRAM

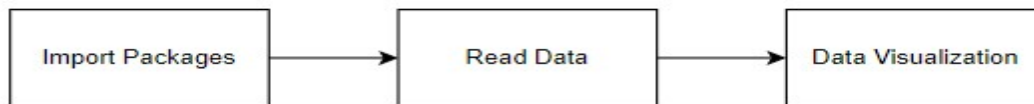


FIGURE 9.2 DATA VISUALIZATION

GIVEN INPUT EXPECTED OUTPUT

input: data

output: visualized data

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have

to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

False Positive (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g., if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g., if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g., if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g., if actual class says this passenger did not survive and predicted class tells you the same thing.

Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different

performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 3 different algorithms are compared:

- Bernoulli NB
- Random forest Classifier
- Ridge Classifier

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done

preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F- Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in NumPy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

MODULE – 3:

9.2.3 RIDGE CLASSIFIER:

The Ridge Classifier is a machine learning algorithm used for classification tasks. It is a variant of the Ridge Regression algorithm, which is primarily used for regression tasks. The Ridge Classifier is specifically designed for multi-class classification problems, where the goal is to assign an input data point to one of several predefined classes or categories.

Here's an explanation of the Ridge Classifier algorithm:

Linear Model: Like many other classification algorithms, the Ridge Classifier is based on a linear model. It assumes that there is a linear relationship between the

input features and the target classes. In other words, it assumes that the decision boundary separating different classes is a linear hyperplane.

Regularization: The term "Ridge" in Ridge Classifier comes from Ridge Regression. It means that the algorithm uses L2 regularization to prevent overfitting. Regularization helps in controlling the complexity of the model and reduces the risk of it fitting noise in the training data.

Objective Function: The Ridge Classifier aims to minimize an objective function that consists of two parts:

Loss Function: The loss function measures the model's error in predicting class labels. Typically, it uses a loss function suitable for classification problems, such as the cross-entropy loss or the log-loss.

Regularization Term: The L2 regularization term is added to the loss function. This term discourages the model from assigning too much importance to any single feature and helps in obtaining a stable and well-behaved model.

Hyperparameter Tuning: The Ridge Classifier has a hyperparameter called "alpha" (α), which controls the strength of the L2 regularization. A higher value of alpha will lead to stronger regularization and, consequently, simpler models. The appropriate value of alpha is typically determined through techniques like cross-validation.

Training: To train the Ridge Classifier, you provide it with a labeled dataset containing input features and corresponding class labels. During training, it adjusts the model's parameters (coefficients) using optimization techniques like gradient descent to minimize the objective function.

Prediction: Once the Ridge Classifier is trained, you can use it to make predictions on new, unseen data. It assigns class labels based on the linear combination of input features and the learned coefficients. The class with the highest score is the predicted class.

Multi-Class Classification: The Ridge Classifier can handle multi-class classification tasks naturally. It typically employs a one-vs-rest (OvR) or one-vs-one (OvO) strategy to extend binary classification to multiple classes.

MODULE DIAGRAM

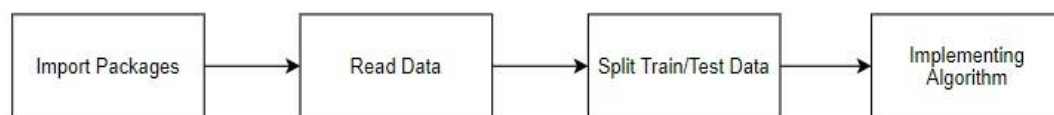


FIGURE 9.3 RIDGE CLASSIFIER

GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

MODULE – 4:

9.2.4 RANDOMFOREST CLASSIFIER:

A Random Forest Classifier is a powerful machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that

combines the predictions of multiple decision trees to produce a more accurate and robust model. Here's an explanation of how the Random Forest Classifier works:

Decision Trees: At the heart of the Random Forest Classifier are decision trees. A decision tree is a simple tree-like structure that makes a sequence of binary decisions to classify data points. Each internal node of the tree represents a decision based on a specific feature, and each leaf node represents a class label or a numerical value (in the case of regression).

Ensemble Learning: A Random Forest consists of a collection of decision trees. Instead of relying on a single decision tree for making predictions, the algorithm builds multiple trees, each with a slightly different subset of the training data and features.

Random Subsampling: When building each decision tree in the forest, a random subset of the training data is used. This process is called "bootstrap sampling" or "bagging." It helps introduce diversity among the individual trees, reducing the risk of overfitting to the training data.

Feature Randomization: Another key aspect of Random Forest is feature randomization. When making decisions at each node of a decision tree, the algorithm doesn't consider all available features. Instead, it randomly selects a subset of features to consider. This introduces randomness and decorrelates the trees, making them less likely to make the same errors.

Voting or Averaging: Once all the individual decision trees are constructed, they can be used to make predictions. For classification tasks, the Random Forest combines the predictions of each tree through a majority vote. The class that receives

the most votes across all trees is the final prediction. For regression tasks, the Random Forest averages the predictions of all trees to produce the final prediction.

Robustness: Random Forests are known for their robustness and ability to handle noisy or complex datasets. They are less prone to overfitting than individual decision trees because the ensemble of trees helps reduce variance.

Hyperparameters: Random Forests have several hyperparameters that can be tuned to optimize their performance, including:

- The number of trees in the forest.
- The maximum depth of each tree.
- The minimum number of samples required to split a node.
- The maximum number of features to consider at each split.
- The criterion for measuring the quality of splits (e.g., Gini impurity or entropy for classification).

Feature Importance: Random Forests can provide information about feature importance, which helps identify which features are most influential in making predictions. This can be useful for feature selection or understanding the importance of different variables in your dataset.

MODULE DIAGRAM

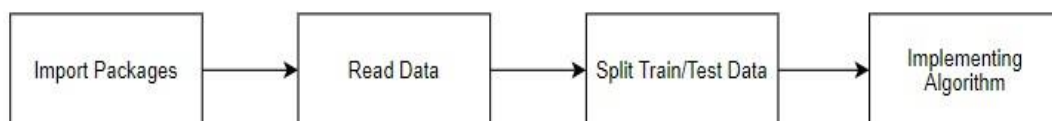


FIGURE 9.4 RANDOM FOREST CLASSIFIER

GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

MODULE – 5

9.2.5 BERNOULLINB:

The Bernoulli Naive Bayes (BernoulliNB) algorithm is a classification algorithm based on the principles of the Naive Bayes theorem. It is specifically designed for binary classification problems, where the target variable has two possible classes (e.g., spam or not spam, positive or negative sentiment).

Here's an explanation of how the Bernoulli Naive Bayes algorithm works:

Naive Bayes Classification: BernoulliNB is a variant of the Naive Bayes algorithm. Naive Bayes methods are based on Bayes' theorem, which is a probabilistic theorem used for classification and other machine learning tasks. It calculates the probability of a data point belonging to a particular class based on its features.

Binary Features: BernoulliNB is well-suited for datasets where the features are binary, meaning they take on values of 0 or 1. These binary features are often used to represent the presence (1) or absence (0) of certain attributes or characteristics in the data.

Assumption of Independence: Like all Naive Bayes algorithms, BernoulliNB makes a "naive" assumption of feature independence. It assumes that the presence

or absence of one feature is independent of the presence or absence of any other feature. This simplifying assumption allows the algorithm to calculate probabilities more efficiently.

Decision Rule: The class with the highest posterior probability is the predicted class for the data point. In binary classification, you choose the class with the higher probability as the predicted class (e.g., if $P(Y=1|X) > P(Y=0|X)$, then the prediction is class 1).

Laplace Smoothing: To avoid zero probabilities in cases where a feature hasn't been observed in a particular class during training, Laplace smoothing (or add-one smoothing) is often applied. This involves adding a small constant to all the probabilities, ensuring that no probability becomes zero.

MODULE DIAGRAM

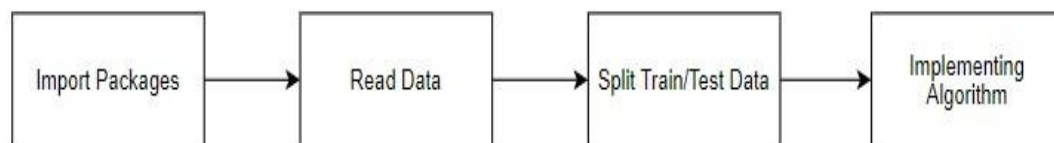


FIGURE 9.5 BERNOULLI NB

GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

Deployment

Deploying the model in Django Framework and predicting output. In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our Django framework for providing better user interface and predicting the output whether the given image is CKD / Not CKD.

Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is

correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and click jacking (see Website security for more details of such attacks).

Scalable

Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g., Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

FEATURES:

Django was designed to be easy to use and extend. The idea behind Django is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also you are free to build your own modules. Django is great for all kinds of projects. It's especially good for prototyping. Django depends on two external libraries: the Jinja2 template engine

Plus Django gives you so much more **CONTROL** on the development stage of your project. It follows the principles of minimalism and let you decide how you will build your application.

- Django has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g., SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Django documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Django.
- It is super easy to deploy Django in production (Django is 100% WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Django is one of the most polished and feature-rich micro frameworks, available. Still young, Django has a thriving community, first-class extensions, and an elegant API. Django comes with all the benefits of fast templates, strong WSGI features, thorough unit testability at the web application and library level, extensive documentation. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Django.

Advantages of Django:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Django is a web framework from Python language. Django provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework Django still doesn't use the Model View Controller (MVC) method.

Django-RESTful is an extension for Django that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Django-Restful is a lightweight abstraction that works with the

existing ORM/libraries. Django-RESTful encourages best practices with minimal setup.

Django Restful is an extension for Django that adds support for building REST APIs in Python using Django as the back-end. It encourages best practices and is very easy to set up. Django restful is very easy to pick up if you're already

MODULE DIAGRAM

GIVEN INPUT EXPECTED OUTPUT

input: data values

output: predicting output

HTML Introduction

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g., HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other at the top of every HTML page that you create.

`<!DOCTYPE html>` — This tag specifies the language you will write on the page. In this case, the language is HTML 5.

`<html>` — This tag signals that from here on we are going to write in HTML code.

`<head>` — This is where all the metadata for the page goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the content of the page goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

`<title>` This is where we insert the page name as it will appear at the top of the browser window or tab.

`<meta>` This is where information about the document is stored: character encoding, name (page context), description.

Head Tag

```
<head><title>My First Webpage</title>
```

```
<meta charset="UTF-8">
```

```
<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">
```

```
<meta name="author" content="Conor Sheils">
```

`</header>` Adding Content

Next, we will make `<body>` tag.

The HTML `<body>` is where we add the content which is designed for viewing by human eyes.

This includes text, images, tables, forms and everything else that we see on the internet each day.

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag `<p>` which creates a new paragraph. We place all of our regular text inside the element `<p>`.

When we write text in HTML, we also have a number of other elements we can use to control the text or make it appear in a certain way

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link takes you to another page within the website you are visiting or to an external site.

Links are included in an attribute opened by the <a> tag. This element is the first that we've met which uses an attribute and so it looks different to previously mentioned tags.

```
<a href=http://www.google.com>Google</a>
```

Image Tag

In today's modern digital world, images are everything. The tag has everything you need to display images on your site. Much like the <a> anchor element, also contains an attribute.

The attribute features information for your computer regarding the source, height, width and alt text of the image

```

```

CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

Cascading: Falling of Styles

Style: Adding designs/Styling our HTML tags

Sheets: Writing our style in different documents

CSS Syntax

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

CSS Comment

Comments don't render on the browser

Helps to understand our code better and makes it readable.

Helps to debug our code

Two ways to comment:

Single line

CSS How-To

There are 3 ways to write CSS in our HTML file.

Inline CSS

Internal CSS

External CSS

Priority order

Inline > Internal > External

Inline CSS

Before CSS this was the only way to apply styles

Not an efficient way to write as it has a lot of redundancy

Self-contained

Uniquely applied on each element

The idea of separation of concerns was lost

GIVEN INPUT EXPECTED OUTPUT

input: data values

output: predicting output

CHAPTER 10

RESULT

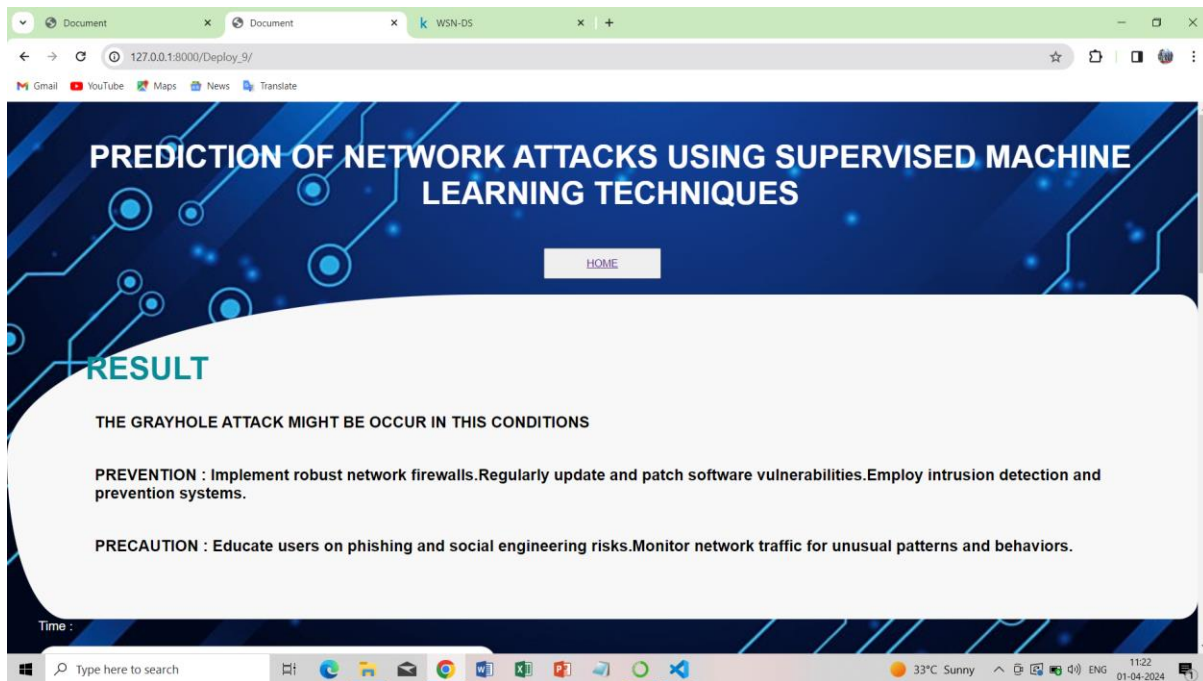


FIGURE 10.1 FINAL OUTPUT

The output of the machine learning model acting as a sentry shield for wireless data packets encompasses packet classifications into categories such as legitimate, suspicious, or malicious, often accompanied by probability scores indicating confidence levels for each classification. Additionally, the model may detect anomalies or intrusions, providing flags or alerts for suspicious patterns or deviations from normal behavior in the wireless data traffic. Furthermore, it might offer actionable recommendations based on the classification results, suggesting actions like blocking or rerouting packets identified as malicious or suspicious, thereby bolstering the security of the wireless network.

CHAPTER-11

CONCLUSION AND FUTURE WORK

11.1 CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be found out by comparing each algorithm with type of all WSN Attacks for future prediction results by finding best connections. This brings some of the following insights about diagnose the network attack of each new connection. To presented a prediction model with the aid of artificial intelligence to improve over human accuracy and provide with the scope of early detection. It can be inferred from this model that; area analysis and use of machine learning technique is useful in developing prediction models that can helps to network sectors reduce the long process of diagnosis and eradicate any human error.

11.2 FUTURE WORK

- Network sector want to automate and detecting the attacks of packet transfers from eligibility process (real time) based on the connection detail.
- To automate this process by show the prediction result in web application or desktop application at cloud.
- To optimize the work to implement in Artificial Intelligence environment.

APPENDIX-I

CODING

MODULE 1:

#DATA PRE-PROCESSING AND DATA CLEANING

```
import pandas as pd
```

```
import numpy as np
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
In [ ]:
```

```
df = pd.read_csv('WSN.csv')
```

```
df.head()
```

```
In [ ]:
```

```
df.shape
```

```
In [ ]:
```

```
df.size
```

```
In [ ]:
```

```
df.columns
```

In []:

```
df.describe()
```

In []:

```
df.info()
```

In []:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
var = ['Attack_type']
```

```
for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

In []:

```
df.tail()
```

In []:

```
df.isnull()
```

In []:

```
df = df.dropna()
```

In []:

```
df["Attack_type"].unique()
```

```
In [ ]:
```

```
df['Attack_type'].unique()
```

```
In [ ]:
```

```
df.corr()
```

```
In [ ]:
```

```
pd.crosstab(df["Rank"], df["Attack_type"])
```

```
In [ ]:
```

```
df.groupby([" send_code ", "Expanded_Energy"]).groups
```

```
In [ ]:
```

```
df["Attack_type"].value_counts()
```

```
In [ ]:
```

```
pd.Categorical(df[" who_CH"]).describe()
```

```
In [ ]:
```

```
df.duplicated()
```

```
In [ ]:
```

```
df = df.drop_duplicates()
```


In []:

```
sum(df.duplicated())
```

In []:

```
df.head(10)
```

MODULE 2:

#DATA VISUALIZATION AND DATA ANALYSIS

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

In []:

```
df = pd.read_csv('WSN.csv')
```

```
df.head()
```

In []:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
var = ['Attack_type']
```

```
for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

In []:

```
df = df.drop_duplicates()
```

In []:

```
plt.figure(figsize=(12,7))
```

```
sns.countplot(x='Attack_type',data=df)
```

In []:

```
plt.figure(figsize=(15,5))
```

```
plt.subplot(1,2,1)
```

```
plt.hist(df[' who CH'],color='magenta')
```

```
plt.subplot(1,2,2)
```

```
plt.hist(df[' Time'],color='grey')
```

In []:

```
df.hist(figsize=(15,55),layout=(15,4), color='cyan')
```

```
plt.show()
```

In []:

```
df[' send_code '].hist(figsize=(10,5),color='pink')
```

In []:

```
sns.scatterplot(df[' Time'], color='blueviolet') # scatter, plot, triplot, stackplot
```

In []:

```
sns.violinplot(df['Rank'], color='coral')
```

In []:

```
df['Attack_type'].plot(kind='density',color="darkgoldenrod")
```

In []:

```
sns.displot(df[' ADV_R'], color='purple')
```

```
# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot,  
kdeplot, pointplot, violinplot, stripplot
```

In []:

```
sns.scatterplot(df['Expanded_Energy'], color='coral') # residplot, scatterplot
```

In []:

```
fig, ax = plt.subplots(figsize=(20,15))
```

```
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
```

In []:

```
def plot(df, variable):
```

```
    dataframe_pie = df[variable].value_counts()
```

```
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
```

```
    ax.set_title(variable + '\n', fontsize = 10)
```

```
    return np.round(dataframe_pie/df.shape[0]*100,2)
```

```
plot(df, 'Attack_type')
```

MODULE 3:

BERNOULLINB ALGORITHM

In []:

```
import pandas as pd
```

```

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

In [ ]:

df = pd.read_csv('WSN.csv')

df.head()

In [ ]:

df=df.dropna()

In [ ]:

df.columns

In [ ]:

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['Attack_type']

for i in var:

    df[i] = le.fit_transform(df[i]).astype(int)

In [ ]:

df = df.drop_duplicates()

In [ ]:

```

```

x1 = df.drop(labels='Attack_type', axis=1)

y1 = df.loc[:, 'Attack_type']

In [ ]:

import imblearn

from imblearn.over_sampling import RandomOverSampler

from collections import Counter

ros =RandomOverSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print("OUR DATASET COUNT      : ", Counter(y1))

print("OVER SAMPLING DATA COUNT : ", Counter(y))

In [ ]:

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)

print("NUMBER OF TRAIN DATASET  : ", len(x_train))

print("NUMBER OF TEST DATASET   : ", len(x_test))

print("TOTAL NUMBER OF DATASET  : ", len(x_train)+len(x_test))

In [ ]:

print("NUMBER OF TRAIN DATASET  : ", len(y_train))

print("NUMBER OF TEST DATASET   : ", len(y_test))

print("TOTAL NUMBER OF DATASET  : ", len(y_train)+len(y_test))

```

In []:

```
from sklearn.naive_bayes import BernoulliNB
```

In []:

```
BNB = BernoulliNB()
```

```
BNB.fit(x_train,y_train)
```

In []:

```
predicted = BNB.predict(x_test)
```

In []:

```
from sklearn.metrics import classification_report
```

```
cr = classification_report(y_test,predicted)
```

```
print('THE CLASSIFICATION REPORT OF BERNOULLINB:\n\n',cr)
```

In []:

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test,predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF BERNOULLINB:\n\n\n',cm)
```

In []:

```
from sklearn.model_selection import cross_val_score
```

```
accuracy = cross_val_score(BNB, x, y, scoring='accuracy')
```

```
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',  
accuracy*100)
```

In []:

```
from sklearn.metrics import accuracy_score
```

```
a = accuracy_score(y_test,predicted)
```

```
print("THE ACCURACY SCORE OF BERNOULLINB IS :",a*100)
```

```
In [ ]:
```

```
from sklearn.metrics import hamming_loss
```

```
hl = hamming_loss(y_test,predicted)
```

```
print("THE HAMMING LOSS OF BERNOULLINB IS :",hl*100)
```

```
In [ ]:
```

```
def plot_confusion_matrix(cm, title="THE CONFUSION MATRIX SCORE OF  
BERNOULLINB\n\n', cmap=plt.cm.cool):
```

```
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
```

```
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF BERNOULLINB:\n\n')
```

```
print(cm)
```

```
plot_confusion_matrix(cm)
```

```
In [ ]:
```

```
import matplotlib.pyplot as plt
```

```
df2 = pd.DataFrame()
```

```
df2["y_test"] = y_test
```

```
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
```

MODULE 4:

RIDGE CLASSIFIER ALGORITHM

In []:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In []:

```
df = pd.read_csv('WSN.csv')
df.head()
```

In []:

```
df=df.dropna()
```

In []:


```
df.columns
```

```
In [ ]:
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
var = ['Attack_type']
```

```
for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
In [ ]:
```

```
df = df.drop_duplicates()
```

```
In [ ]:
```

```
x1 = df.drop(labels='Attack_type', axis=1)
```

```
y1 = df.loc[:, 'Attack_type']
```

```
In [ ]:
```

```
import imblearn
```

```
from imblearn.over_sampling import RandomOverSampler
```

```
from collections import Counter
```

```
ros = RandomOverSampler(random_state=42)
```

```
x,y=ros.fit_resample(x1,y1)
```

```
print("OUR DATASET COUNT      : ", Counter(y1))
```

```
print("OVER SAMPLING DATA COUNT : ", Counter(y))
```

```
In [ ]:
```

```

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)

print("NUMBER OF TRAIN DATASET   :", len(x_train))

print("NUMBER OF TEST DATASET    :", len(x_test))

print("TOTAL NUMBER OF DATASET   :", len(x_train)+len(x_test))

In [ ]:

print("NUMBER OF TRAIN DATASET   :", len(y_train))

print("NUMBER OF TEST DATASET    :", len(y_test))

print("TOTAL NUMBER OF DATASET   :", len(y_train)+len(y_test))

In [ ]:

from sklearn.linear_model import RidgeClassifier

In [ ]:

RC = RidgeClassifier()

RC.fit(x_train,y_train)

In [ ]:

predicted = RC.predict(x_test)

In [ ]:

from sklearn.metrics import classification_report

cr = classification_report(y_test,predicted)

print('THE CLASSIFICATION REPORT OF BERNOULLINB:\n\n',cr)

```

In []:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predicted)

print('THE CONFUSION MATRIX SCORE OF BERNOULLINB:\n\n',cm)
```

In []:

```
from sklearn.model_selection import cross_val_score

accuracy = cross_val_score(RC, x, y, scoring='accuracy')

print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n',
accuracy*100)
```

In []:

```
from sklearn.metrics import accuracy_score

a = accuracy_score(y_test,predicted)

print("THE ACCURACY SCORE OF RIDGE CLASSIFICAION IS :",a*100)
```

In []:

```
from sklearn.metrics import hamming_loss

hl = hamming_loss(y_test,predicted)

print("THE HAMMING LOSS OF RIDGE CLASSIFICAION IS :",hl*100)
```

In []:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, predicted, labels=RC.classes_)

disp =
ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=RC.classes_)
```

```
disp.plot()
```

```
plt.show()
```

```
In [ ]:
```

```
import matplotlib.pyplot as plt
```

```
df2 = pd.DataFrame()
```

```
df2["y_test"] = y_test
```

```
df2["predicted"] = predicted
```

```
df2.reset_index(inplace=True)
```

```
plt.figure(figsize=(20, 5))
```

```
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
```

```
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
```

```
plt.show()
```

MODULE 5:

RANDOM FOREST CLASSIFICATION ALGORITHM

```
In [ ]:
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

In []:

```
df = pd.read_csv('WSN.csv')
```

```
del df['id']
```

```
df.head()
```

In []:

```
df=df.dropna()
```

In []:

```
df = df.rename({'Time': 'Time'}, axis=1)
```

```
df = df.rename({'Is_CH': 'Is_CH'}, axis=1)
```

```
df = df.rename({'who_CH': 'who_CH'}, axis=1)
```

```
df = df.rename({'Dist_To_CH': 'Dist_To_CH'}, axis=1)
```

```
df = df.rename({'ADV_S': 'ADV_S'}, axis=1)
```

```
df = df.rename({'ADV_R': 'ADV_R'}, axis=1)
```

```
df = df.rename({'JOIN_S': 'JOIN_S'}, axis=1)
```

```
df = df.rename({'JOIN_R': 'JOIN_R'}, axis=1)
```

```
df = df.rename({'SCH_S': 'SCH_S'}, axis=1)
```

```
df = df.rename({'SCH_R': 'SCH_R'}, axis=1)
```

```
df = df.rename({'DATA_S': 'DATA_S'}, axis=1)
```

```
df = df.rename({'DATA_R': 'DATA_R'}, axis=1)
```

```
df = df.rename({'Data_Sent_To_BS': 'Data_Sent_To_BS'}, axis=1)
```

```
df = df.rename({'dist_CH_To_BS': 'dist_CH_To_BS'}, axis=1)
```

```
df = df.rename({' send_code ': 'send_code '}, axis=1)
```

```
In [ ]:
```

```
df['Attack_type'].unique()
```

```
In [ ]:
```

```
df.columns
```

```
In [ ]:
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
var = ['Attack_type']
```

```
for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
In [ ]:
```

```
df['Attack_type'].unique()
```

```
In [ ]:
```

```
df = df.drop_duplicates()
```

```
In [ ]:
```

```
x1 = df.drop(labels='Attack_type', axis=1)
```

```
y1 = df.loc[:, 'Attack_type']
```

```
In [ ]:
```

```
import imblearn
```

```
from imblearn.under_sampling import RandomUnderSampler
```

```

from collections import Counter

ros =RandomUnderSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print("OUR DATASET COUNT      : ", Counter(y1))

print("OVER SAMPLING DATA COUNT : ", Counter(y))

In [ ]:

x

In [ ]:

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)

print("NUMBER OF TRAIN DATASET   : ", len(x_train))

print("NUMBER OF TEST DATASET    : ", len(x_test))

print("TOTAL NUMBER OF DATASET   : ", len(x_train)+len(x_test))

In [ ]:

print("NUMBER OF TRAIN DATASET   : ", len(y_train))

print("NUMBER OF TEST DATASET    : ", len(y_test))

print("TOTAL NUMBER OF DATASET   : ", len(y_train)+len(y_test))

In [ ]:

from sklearn.ensemble import RandomForestClassifier

In [ ]:

```

```
RFC = RandomForestClassifier()
```

```
RFC.fit(x_train,y_train)
```

```
In [ ]:
```

```
predicted = RFC.predict(x_test)
```

```
In [ ]:
```

```
from sklearn.metrics import classification_report
```

```
cr = classification_report(y_test,predicted)
```

```
print('THE CLASSIFICATION REPORT OF RANDOM FOREST  
CLASSIFICATION:\n\n',cr)
```

```
In [ ]:
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test,predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST  
CLASSIFICATION:\n\n\n',cm)
```

```
In [ ]:
```

```
from sklearn.model_selection import cross_val_score
```

```
accuracy = cross_val_score(RFC, x, y, scoring='accuracy')
```

```
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY : \n\n\n',  
accuracy*100)
```

```
In [ ]:
```

```
from sklearn.metrics import accuracy_score
```

```
a = accuracy_score(y_test,predicted)
```



```
print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFICATION  
IS :",a*100)
```

In []:

```
from sklearn.metrics import hamming_loss
```

```
hl = hamming_loss(y_test,predicted)
```

```
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFICATION IS  
:",hl*100)
```

In []:

```
def plot_confusion_matrix(cm, title="THE CONFUSION MATRIX SCORE OF  
RANDOM FOREST CLASSIFICATION\n\n', cmap=plt.cm.cool):
```

```
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
```

```
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST  
CLASSIFICATION:\n\n')
```

```
print(cm)
```

```
plot_confusion_matrix(cm)
```

In []:

```
import matplotlib.pyplot as plt
```

```
df2 = pd.DataFrame()
```

```
df2["y_test"] = y_test
```

```
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
```

In []:

```
import joblib
joblib.dump(RFC, 'MODEL2.pkl')
```

In []:

APPENDIX-II

SCREENSHOTS

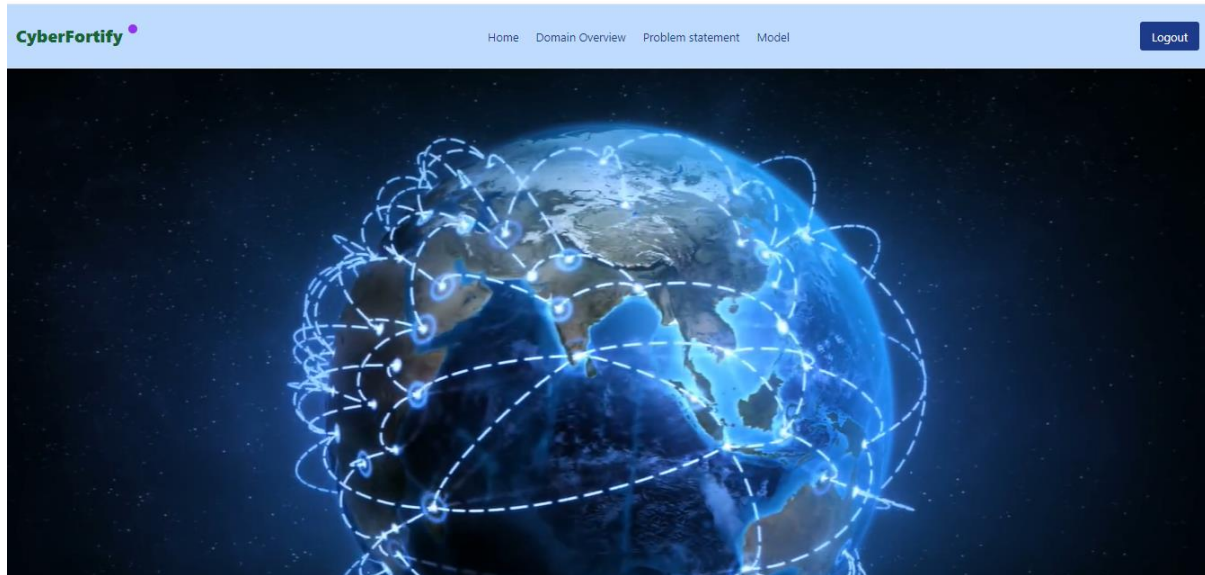


FIGURE A2.1 HOME PAGE

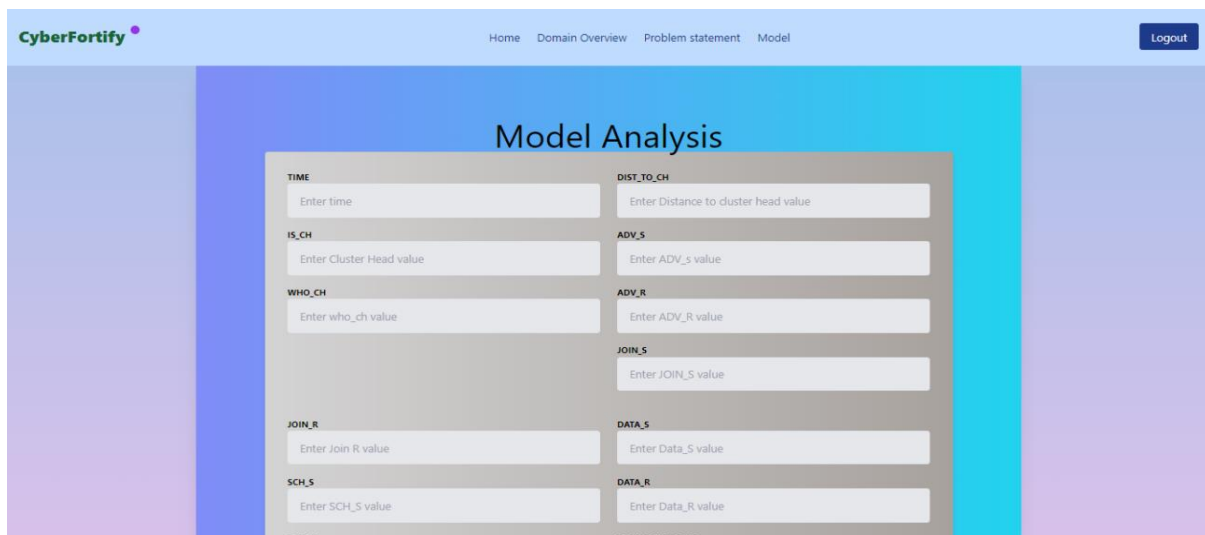
The screenshot displays the 'Model Analysis' page within the CyberFortify application. The page has a light blue header with the same navigation links as the home page. The main content area is titled 'Model Analysis' and contains a large, light gray form with multiple input fields. The form is organized into two columns. The left column includes fields for 'TIME', 'IS_CH', 'WHO_CH', 'JOIN_R', and 'SCH_S'. The right column includes fields for 'DIST_TO_CH', 'ADV_S', 'ADV_R', 'JOIN_S', 'DATA_S', and 'DATA_R'. Each field is accompanied by a placeholder text indicating the required input, such as 'Enter time', 'Enter Distance to cluster head value', 'Enter Cluster Head value', 'Enter ADV_s value', 'Enter ADV_R value', 'Enter JOIN_S value', 'Enter Data_S value', 'Enter Data_R value', 'Enter Join R value', 'Enter SCH_S value', and 'Enter Data_R value'. The form is set against a background of vertical color bands in shades of blue and purple.

FIGURE A2.2 MODEL ANALYSIS PAGE

JOIN_R <input type="text" value="Enter Join_R value"/>	DATA_S <input type="text" value="Enter Data_S value"/>
SCH_S <input type="text" value="Enter SCH_S value"/>	DATA_R <input type="text" value="Enter Data_R value"/>
SCH_R <input type="text" value="Enter SCH_R value"/>	DATA_SENT_TO_BS <input type="text" value="Enter Data_sent_To_BS value"/>
RANK <input type="text" value="Enter Rank value"/>	DIST_CH_TO_BS <input type="text" value="Enter dist_ch_to_bs value"/>
SEND_CODE <input type="text" value="Enter SendCode value"/>	EXPANDED_ENERGY <input type="text" value="Enter Expanded_Energy value"/>
<input type="button" value="Submit"/>	

FIGURE A2.3 DETAILS PAGE

CyberFortify
Home Domain Overview Problem statement Model Logout

The Attack type is Flooding
You have a new message!

ATTACK TYPE

SL Flooding Attack
A flooding attack, often referred to as a network flooding attack or a denial-of-service (DoS) flooding attack, is a type of cyberattack in which an attacker overwhelms a target network or system with an excessive amount of traffic, causing it to become unavailable.

This is Flooding Attack

Welcome to cybersecurity domain Wireless Sensor Networks (WSNs) are integral to cybersecurity by offering real-time threat detection capabilities, safeguarding critical infrastructure, and ensuring data integrity. They enable proactive monitoring across various domains, contributing to early threat mitigation and enhanced security. WSNs play a crucial role in fortifying our digital landscape against evolving cyber threats.

What is Cybersecurity?

Cybersecurity is the practice of protecting computer systems, networks, and data from theft, damage, or unauthorized access. It encompasses a range of technologies, processes, and practices designed to ensure the confidentiality, integrity, and availability of digital information.

FIGURE A2.4 DETECTION PAGE

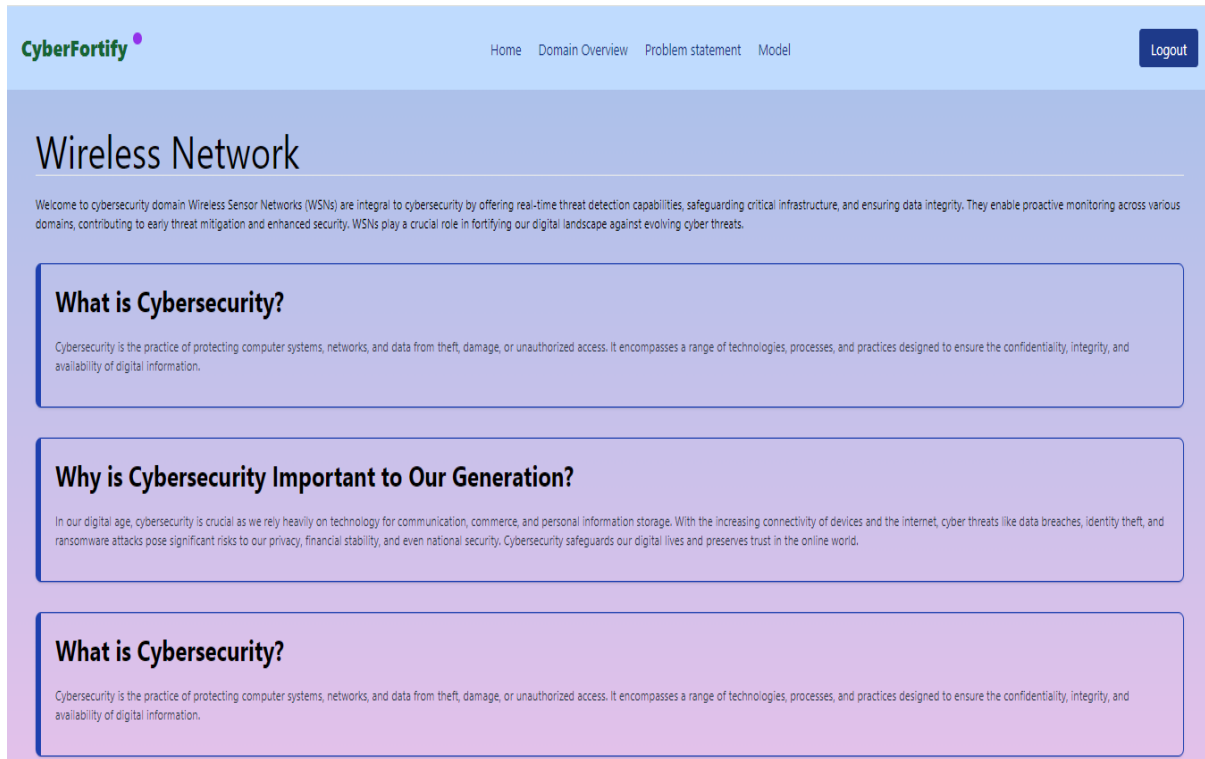


FIGURE A2.5 DOMAIN DETAIL PAGE

REFERENCES

- [1] Dr. G. Umarani Srikanth, Priyadharsini.S, “Prediction of Network Attacks Using Machine Learning Techniques” in Proc IEEE access, 2021, Vol. 5.
- [2] Qinghai Zhou , Liangyue Li , Nan Cao , Lei Ying , and Hanghang Tong, “Adversarial Attacks on Multi-Network Mining: Problem Definition and Fast Solutions” in Proc IEEE access,2023, VOL. 35.
- [3] Jinyin Chen , Jian Zhang, Zhi Chen, Min Du, and Qi Xuan, ” Time-Aware Gradient Attack on Dynamic Network Link Prediction” in Proc IEEE access,2023, Vol.35.
- [4] Jakub Breier , Xiaolu Hou , Martí'n Ochoa , and Jesus Solano, “FooBaR: Fault Fooling Backdoor Attack on Neural Network Training” in Proc IEEE access, 2023, Vol.20.
- [5] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato and Hwee-Pink Tan, “Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications” in Proc IEEE access, 2015.
- [6] P. Amudha, S. Karthik and S. Sivakumari, "Classification techniques for intrusion detection-an overview", *International Journal of Computer Applications*, vol. 76, no. 16, 2013.
- [7] F. Haddadi, S. Khanchi, M. Shetabi and V. Derhami, "Intrusion detection and attack classification using feed-forward neural network", *Computer and Network Technology (ICCNT) 2010 Second International Conference on. IEEE*, pp. 262-266, 2010.

- [8] M. Alkasassbeh, G. Al-Naymat, A. B. Hassanat and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques", *International Journal of Advanced Computer Science & Applications*, vol. 1, no. 7, pp. 436-445.
- [9] W. Alsharafat, "Applying artificial neural network and extended classifier system for network intrusion detection", *International Arab Journal of Information Technology (IAJIT)*, vol. 10, no. 3, 2013.
- [10] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson and J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification", *Proc. IEEE Workshop on Information Assurance and Security*, pp. 85-90, 2001.

PUBLICATIONS

Gopinath V, Dharun Krithik K, Bhuvaneswari S, 2024, “Sentry Shield For Wireless Data Packets Using Machine Learning”, National Conference on Intelligence Computing & Data Science organized by Anand Institute of Higher Technology on 07th March 2024, pp. 36.



ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR, OMR, KAZHIPATTUR, CHENNAI-603103



Department of Artificial Intelligence and Data Science

&

Department of Information Technology



Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE


NCICDS'24

07th March 2024

CERTIFICATE

of Achievement

This is to certify that Prof/Dr./Mr./Ms GOPINATH.V
of Prince Abi Venkateshwarra Padmanavathy Engineering College has participated/ presented a
paper entitled Sentry Shield for Wireless data Packets using Machine Learning
in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)**
organized by Department of Artificial Intelligence and Data Science & Department of Information
Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu,
held on 07th March 2024.


Mr. P. Sachuthanandam
Organizing Secretary
NCICDS'24 - AI & DS


Mr. N. Manikandan
Organizing Secretary
NCICDS'24 - IT


Dr. K. Karnavel
Convener
HOD - AI & DS / IT


Dr. P. Suresh Mohan Kumar
PRINCIPAL



POORVIKA



ANAND INSTITUTE OF HIGHER TECHNOLOGY

ACCREDITED BY NBA | APPROVED BY AICTE & AFFILIATED TO ANNA UNIVERSITY
IT CORRIDOR, OMR, KAZHIPATTUR, CHENNAI-603103



Department of Artificial Intelligence and Data Science

&

Department of Information Technology



Jointly organizing

NATIONAL CONFERENCE ON INTELLIGENCE COMPUTING & DATA SCIENCE

NCICDS'24

07th March 2024


CERTIFICATE
of Achievement

This is to certify that Prof/Dr./Mr./Ms DHARUN KRITHIK K
of Prince Shri Venkateshwara Padmanabha Engineering College has participated/ presented a
paper entitled Sentry Shield for Wireless data packets using machine learning
in the **National Conference on Emerging Trends in Intelligence Computing (NCICDS'24)**
organized by Department of Artificial Intelligence and Data Science & Department of Information
Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai - 603 103, Tamilnadu,
held on 07th March 2024.


Mr. P. Sachuthanandam
Organizing Secretary
NCICDS'24 - AI & DS


Mr. N. Manikandan
Organizing Secretary
NCICDS'24 - IT


Dr. K. Karnavel
Convener
HOD - AI & DS / IT

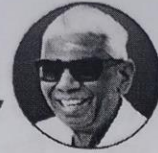

Dr. P. Suresh Mohan Kumar
PRINCIPAL



POORVIKA



ANAND INSTITUTE OF HIGHER TECHNOLOGY



APPROVED BY AICTE, NEW DELHI. AFFILIATED TO ANNA UNIVERSITY, CHENNAI AN ISO 9001:2008 CERTIFIED INSTITUTION & ACCREDITED BY NBA

(A KALASALINGAM GROUP OF INSTITUTIONS)

SENTRY SHIELD FOR WIRELESS DATA PACKETS USING MACHINE LEARNING

Gopinath V¹, Dharun Krithik K²

^{1,2} UG Student, Prince Shri Venkateshwara Padmavathy Engineering College, Chennai.

ABSTRACT

Network attacks pose a significant threat to the security and integrity of computer networks. The ability to predict and prevent these attacks is crucial for maintaining a secure network environment. Supervised machine learning techniques have emerged as effective tools for network attack prediction due to their ability to analyze large amounts of network data and identify patterns indicative of malicious activity. We present a comprehensive analysis of supervised machine learning techniques for the prediction of network attacks. We collect and pre-process the data, extracting relevant features and transforming them into a suitable format for machine learning algorithms. We evaluate the performance of these algorithms. We investigate the interpretability of the trained models to gain insights into the underlying patterns and characteristics of network attacks. This allows network administrators to understand the nature of attacks and develop appropriate defenses strategies. Additionally, we discuss the challenges and limitations associated with the application of supervised machine learning techniques in the domain of network attack prediction, such as the need for real-time analysis and the emergence of sophisticated evasion techniques.

QR CODE

