**Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems**

**1. INTRODUCTION:**

Internet of Things (IoT) devices are increasingly integrated in cyber-physical systems (CPS), including in critical infrastructure sectors such as dams and utility plants. In these settings, IoT devices (also referred to as Industrial IoT or IIoT) are often part of an Industrial Control System (ICS), tasked with the reliable operation of the infrastructure. ICS can be broadly defined to include supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and systems that comprise programmable logic controllers (PLC) and Modbus protocols. The connection between ICS or IIoT-based systems with public networks, however, increases their attack surfaces and risks of being targeted by cyber criminals. One high-profile example is the Stuxnet campaign, which reportedly targeted Iranian centrifuges for nuclear enrichment in 2010, causing severe damage to the equipment Another example is that of the incident targeting a pump that resulted in the failure of an Illinois water plant in 2011. Therefore, system-level security methods are necessary to analyze physical behaviour and maintain system operation availability ICS security goals are prioritized in the order of availability, integrity, and confidentiality, unlike most IT/OT systems (generally prioritized in the order of confidentiality, integrity, and availability) Due to close coupling between variables of the feedback control loop and physical processes, (successful) cyber-attacks on ICS can result in severe and potentially fatal consequences for the society and our environment. This reinforces the importance of designing extremely robust safety and security measurements to detect and prevent intrusions targeting ICS. Popular attack detection and attribution approaches include those based on signatures and anomalies. To mitigate the known limitations in both signature-based and anomaly-based detection and attribution approaches, there have been attempts to introduce hybrid-based approaches Although hybridbased approaches are effective at detecting unusual activates, they are not reliable due to frequent network upgrades, resulting in different Intrusion Detection System (IDS) typologies Beyond this, conventional attack detection and attribution techniques mainly rely on network metadata analysis (e.g. IP addresses, transmission ports, traffic duration, and packet intervals). Therefore, there has been renewed interest in utilizing attack detection and attribution solutions based on Machine Learning (ML) or Deep Neural Networks (DNN) in recent times.

**1.1 Objective of the project:**

Securing Internet of Things (IoT)-enabled cyberphysical systems (CPS) can be challenging, as security solutions developed for general information / operational technology (IT / OT) systems may not be as effective in a CPS setting. Thus, this paper presents a two-level ensemble attack detection and attribution framework designed for CPS, and more specifically in an industrial control system (ICS). At the first level, a decision tree combined with a novel ensemble deep representation learning model is developed for detecting attacks imbalanced ICS environments. At the second level, an ensemble deep neural network is designed for attack attribution. The proposed model is evaluated using real-world datasets in gas pipeline and water treatment system. Findings demonstrate that the proposed model outperforms other competing approaches with similar computational complexity

**2. LITERATURE SURVEY:**

**Multilayer Data-Driven Cyber-Attack Detection System for Industrial Control Systems Based on Network, System, and Process Data**

The growing number of attacks against cyber-physical systems in recent years elevates the concern for cyber security of industrial control systems (ICSs). The current efforts of ICS cyber security are mainly based on firewalls, data diodes, and other methods of intrusion prevention, which may not be sufficient for growing cyber threats from motivated attackers. To enhance the cyber security of ICS, a cyber-attack detection system built on the concept of defense-in-depth is developed utilizing network traffic data, host system data, and measured process parameters. This attack detection system provides multiple-layer defense in order to gain the defenders precious time before unrecoverable consequences occur in the physical system. The data used for demonstrating the proposed detection system are from a real-time ICS testbed. Five attacks, including man in the middle (MITM), denial of service (DoS), data exfiltration, data tampering, and false data injection, are carried out to simulate the consequences of cyber attack and generate data for building data-driven detection models. Four classical classification models based on network data and host system data are studied, including k-nearest neighbor (KNN), decision tree, bootstrap aggregating (bagging), and random forest (RF), to provide a secondary line of defense of cyber-attack detection in the event that the intrusion prevention layer fails. Intrusion detection results suggest that KNN, bagging, and RF have low missed alarm and false alarm rates for MITM and DoS attacks, providing accurate and reliable detection of these cyber attacks. Cyber attacks that may not be detectable by monitoring network and host system data, such as command tampering and false data injection attacks by an insider, are monitored for by traditional process monitoring protocols. In the proposed detection system, an auto-associative kernel regression model is studied to strengthen early attack detection.

**Stealthy Attack Against Redundant Controller Architecture of Industrial Cyber Physical System**

In an industrial cyber-physical system (iCPS), the controller plays a critical role in guaranteeing reliability and stability. Therefore, redundant controller architecture is a well-adopted approach by distributed control systems (DCS), supervisory control and data acquisition (SCADA), and other typical iCPSs. They monitor and control the critical industrial process, such as power generation, chemical industry, water treatment plant, etc. Redundant controller architecture has been designed and largely implemented in response to unpredictable mechanical failures. However, this structure initially proposed for guaranteeing reliability and safety may expand the cyber-attack surface, posing the risk that an attacker may take advantage of this architecture for stealthy attacks. In this article, we analyze the vulnerability arising from the redundant controller architecture and propose a combined attack methodology against these redundant controller architecture systems in a stealthy manner. We find several 0-day vulnerabilities of the real-world devices from three manufacturers and further implement the combined attack over these devices. Our experimental results over various types of real-world devices show that the redundant controller architecture can be exploited to compromise all tested systems stealthily. We also present guidelines for mitigating this risk.

**IIoT Cybersecurity Risk Modeling for SCADA Systems**

Urban critical infrastructure such as electric grids, water networks, and transportation systems are prime targets for cyber attacks. These systems are composed of connected devices which we call the Industrial Internet of Things (IIoT). An attack on urban critical infrastructure IIoT would cause considerable disruption to society. Supervisory control and data acquisition (SCADA) systems are typically used to control IIoT for urban critical infrastructure. Despite the clear need to understand the cyber risk to urban critical infrastructure, there is no data-driven model for evaluating SCADA software risk for IIoT devices. In this paper, we compare non-SCADA and SCADA systems and establish, using cosine similarity tests, that SCADA as a software subclass holds unique risk attributes for IIoT. We then disprove the commonly accepted notion that the common vulnerability scoring system risk metrics of exploitability and impact are not correlated with attack for the SCADA subclass of software. A series of statistical models are developed to identify SCADA risk metrics that can be used to evaluate the risk that a SCADA-related vulnerability is exploited. Based on our findings, we build a customizable SCADA risk prioritization schema that can be used by the security community to better understand SCADA-specific risk. Considering the distinct properties of SCADA systems, a data-driven prioritization schema will help researchers identify security gaps specific to this software subclass that is essential to our society’s operations.

**Anomaly Detection Based on Zone Partition for Security Protection of Industrial Cyber-Physical Systems**

A developing trend of traditional industrial systems is the integration of the cyber and physical domain to improve flexibility and the efficiency of supervision, management, and control. But, the deep integration of these industrial cyber-physical systems (ICPSs), increases the potential for security threats. Attack detection, which forms initial protective barrier, plays an important role in overall security protection. However, most traditional methods focused on cyber information and ignored any limitations that might arise from the characteristics of the physical domain. In this paper, an anomaly detection approach based on zone partition is designed for ICPSs. In detail, initially an automated zone partition method, ensuring crucial system states can be observed in more than one zone, is designed. Then, methods of building zone function model, which do not require any prior knowledge of the physical system are presented before analyzing the anomaly based on zone information. Finally, an experimental rig is constructed to verify the effectiveness of the proposed approach. The results demonstrate that the approach presents a high-accuracy solution, which also performs effectively in real time.

**Industrial control system network intrusion detection by telemetry analysis**

Until recently, industrial control systems (ICSs) used “air-gap” security measures, where every node of the ICS network was isolated from other networks, including the Internet, by a physical disconnect. Attaching ICS networks to the Internet benefits companies and engineers who use them. However, as these systems were designed for use in the air-gapped security environment, protocols used by ICSs contain little to no security features and are vulnerable to various attacks. This paper proposes an approach to detect the intrusions into network attached ICSs by measuring and verifying data that is transmitted through the network but is not inherently the data used by the transmission protocol-network telemetry. Using simulated PLC units, the developed IDS was able to achieve 94.3 percent accuracy when differentiating between machines of an attacker and engineer on the same network, and 99.5 percent accuracy when differentiating between attacker and engineer on the Internet.

**A dataset to support research in the design of secure water treatment systems**

This paper presents a dataset to support research in the design of secure Cyber Physical Systems (CPS). The data collection process was implemented on a six-stage Secure Water Treatment (SWaT) testbed. SWaT represents a scaled down version of a real-world industrial water treatment plant producing 5 gallons per minute of water filtered via membrane based ultrafiltration and reverse osmosis units. This plant allowed data collection under two behavioral modes: normal and attacked. SWaT was run non-stop from its “empty” state to fully operational state for a total of 11-days. During this period, the first seven days the system operated normally i.e. without any attacks or faults. During the remaining days certain cyber and physical attacks were launched on SWaT while data collection continued. The dataset reported here contains the physical properties related to the plant and the water treatment process, as well as network traffic in the testbed. The data of both physical properties and network traffic contains attacks that were created and generated by our research team.

**Anomaly detection for a water treatment system using unsupervised machine learning,**

In this paper, we propose and evaluate the application of unsupervised machine learning to anomaly detection for a Cyber-Physical System (CPS). We compare two methods: Deep Neural Networks (DNN) adapted to time series data generated by a CPS, and one-class Support Vector Machines (SVM). These methods are evaluated against data from the Secure Water Treatment (SWaT) testbed, a scaled-down but fully operational raw water purification plant. For both methods, we first train detectors using a log generated by SWaT operating under normal conditions. Then, we evaluate the performance of both methods using a log generated by SWaT operating under 36 different attack scenarios. We find that our DNN generates fewer false positives than our one-class SVM while our SVM detects slightly more anomalies. Overall, our DNN has a slightly better F measure than our SVM. We discuss the characteristics of the DNN and one-class SVM used in this experiment, and compare the advantages and disadvantages of the two methods.

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**:

In existing system, ML-based attack detection techniques are generally designed to detect moving targets that constantly evolve by learning new vulnerabilities and not relying on known attack signatures or normal network patterns, K-Nearest Neighbor (KNN), Random Forest (RF), DT, Logistic Regression (LR), ANN, Na¨ıve Bayes (NB), and SVM were compared in terms of their effectiveness in detecting backdoor. they performed oversampling on the dataset to achieve balance. They compared the performance of the proposed LAD method with the DNN, SVM, and CNN methods. Based on these experiments, the DNN outperformed the LAD method in the precision metric; however, the LAD performed better in recall and f-measure.

**Disadvantages:**

* + Low accuracy
  + Time taking process
  + Less effectiveness

**3.2 PROPOSED SYSTEM:**

This paper proposed a novel two-stage ensemble deep learning-based attack detection and attack attribution framework for imbalanced ICS data. The attack detection stage uses deep representation learning to map the samples to the new higher dimensional space and applies a DT to detect the attack samples. This stage is robust to imbalanced datasets and capable of detecting previously unseen attacks. The attack attribution stage is an ensemble of several one all classifiers, each trained on a specific attack attribute. The entire model forms a complex DNN with a partially connected and fully connected component that can accurately attribute cyber attacks, as demonstrated. To avoid the above mentioned problems in handling imbalanced datasets, this study proposed a new deep representation learning method to make the DNN able to handle imbalanced datasets without changing, generating, or removing samples.Despite the complex architecture of the proposed framework, (n is the number of training samples), which are similar to those of other DNN-based techniques in the literature. Moreover, the proposed framework can detect and attribute the samples timely with a better recall and f-measure than previous works.

**ADVANTAGES:**

* High accuracy

**Modules Information:**

Internet of Things enabled cyber physical systems such as Industrial equipment’s and operational IT to send and receive data over internet. This equipment’s will have sensors to sense equipment condition and report to centralized server using internet connection. Sometime some malicious users may attack or hack such sensors and then alter their data and this false data will be report to centralized server and false action will be taken. Due to false data many countries equipment got failed and many algorithms was developed to detect attack but all this algorithms suffers from data imbalance (one class my contains huge records (for example NORMAL records and other class like attack may contains few records which lead to imbalance problem and detection algorithms may failed to predict accurately). To deal with data imbalance existing algorithms were using OVER and UNDER sampling which will generate new records for FEWER class but this technique improve accuracy but not up to the mark.

To overcome from this issue author is introducing novel technique without using any under or oversampling algorithms and this technique consists of two parts

This project consists of two modules

To overcome from this issue author is introducing novel technique without using any under or oversampling algorithms and this technique consists of two parts

1. Auto Encoder: auto encoder deep learning will get trained on imbalanced dataset and then extract features from it and this extracted featured will get trained with DECISION TREE algorithm to predict label for known or unknown attacks. Decision tree get trained on reduced number of features obtained from PCA (principal component analysis) algorithm.
2. Deep Neural Network (DNN): in this level DNN algorithm get trained on known and unknown attacks. If any records contains attack signature then DNN will identify attack label or class and attribute them.

To implement this project author has used SWAT (secure water treatment) and this dataset contains IOT request and response signature and associate each dataset with unique attack label and dataset contains below cyber-attack labels

**'Normal', 'Naive Malicious Response Injection (NMRI)', 'Complex Malicious', 'Response Injection (CMRI)', 'Malicious State Command Injection (MSCI)', 'Malicious Parameter Command Injection (MPCI)', 'Malicious Function Code Injection (MFCI)', 'Denial of Service (DoS)'**

**FUNCTIONAL REQUIREMENTS:**

**SOFTWARE REQIREMENTS:**

**System Atributes:**

1. filename, autoencoder, decision\_tree, dnn, encoder\_model, pca
2. X,Y
3. dataset
4. accuracy, precision, recall, fscore, vector
5. X\_train, X\_test, y\_train, y\_test, scaler

**Data base Requirements:**

No need

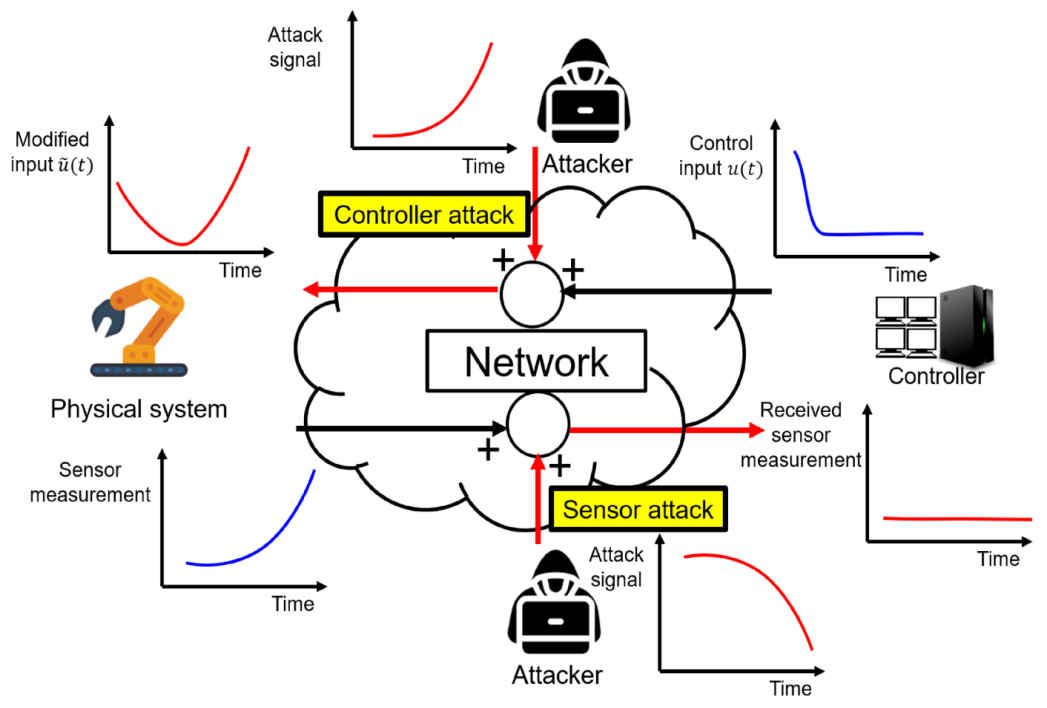
**USECASE:**

Use cases - Use cases describe the interaction between the system and external users that leads to achieving particular goals.

* Upload swat water dataset
* pre-process dataset
* run auto encoder algorithm
* run decision tree with PCA
* run DNN algorithm
* detection and attack attribute type
* comparison graph
* comparison table

**User Stories:** User can see algorithm names and its metrics values such as accuracy and precision and other. different metric values such as precision, recall, accuracy and FSCORE with different colour bars and in all algorithms DNN

**Work down Structure:**



**Prototype:**

python 3.7.0 or 3.7.4

opencv-python==4.5.1.48

keras==2.3.1

tensorflow==1.14.0

protobuf==3.16.0

h5py==2.10.0

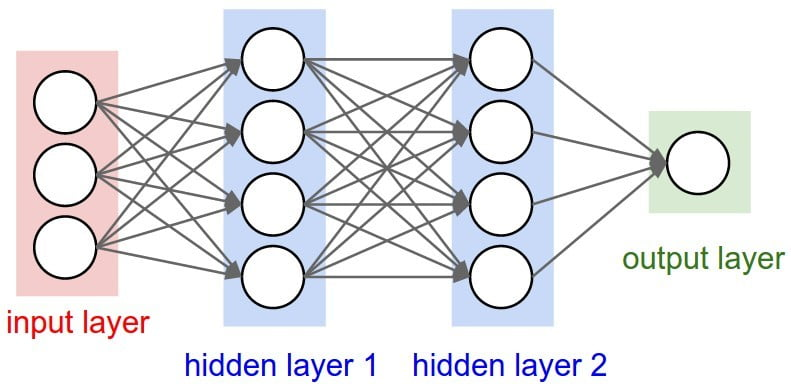
sklearn-extensions==0.0.2

scikit-learn==0.22.2.post1

Numpy

Pandas

**Models and Diagrams:**



**NON-FUNCTIONAL REQUIREMENT:**

**Usability:**  Usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process.(how it was handle entire project easy)

**Security:** the quality or state of being secure: such as. a : freedom from danger : safety. b : freedom from fear or anxiety. c : freedom from the prospect of being laid off job security.

**Readability:** Readability is the ease with which a reader can understand a written text.

**Performance**: the execution of an action. : something accomplished : deed, feat. : the fulfillment of a claim, promise, or request : implementation. 3. : the action of representing a character in a play.

**Availability**: the quality or state of being available trying to improve the availability of affordable housing. 2 : an available person or thing.

**Scalability**: Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

**ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**TECHNICAL FEASIBILITY**

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

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 Ghz
* RAM - 4GB(min)
* Hard Disk - 500 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**SEQUENCE DIAGRAM:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.

****

**COMPONENT DIAGRAM:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Upload swat water dataset

preprocess dataset

run auto encoder algorithm

run decision tree with PCA

run DNN algorithm

Detection and attack attribute type

camparsion graph

camparsion table

**Data flow diagram :**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. Uload swat water dataset 2. Successfully Uload swat water dataset

3. preprocess dataset 4. Successfully Deep Neural Network (DNN)

5. run auto encoder algorithm 6. Successfully run auto encoder algorithm

6.run decision tree with PCA 7.sucessfully run decision tree with PCA tree with PCA

8.run DNN algorithm 9. Successfully run DNN algorithm

10. detection and attack attribute type 11 sucessfully detection and attack attribute type

12.camparsion graph 13. sucessfully camparsion graph

14.camparsion table 15. sucessfully camparsion table

**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

import os

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

import webbrowser

import pickle

from sklearn.decomposition import PCA

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import MinMaxScaler

import keras

from keras import layers

from keras.models import model\_from\_json

from keras.utils.np\_utils import to\_categorical

from keras.models import Model

from sklearn.neural\_network import MLPClassifier

global filename, autoencoder, decision\_tree, dnn, encoder\_model, pca

global X,Y

global dataset

global accuracy, precision, recall, fscore, vector

global X\_train, X\_test, y\_train, y\_test, scaler

labels = ['Normal', 'Naive Malicious Response Injection (NMRI)', 'Complex Malicious', 'Response Injection (CMRI)', 'Malicious State Command Injection (MSCI)',

'Malicious Parameter Command Injection (MPCI)', 'Malicious Function Code Injection (MFCI)', 'Denial of Service (DoS)']

main = tkinter.Tk()

main.title("Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems") #designing main screen

main.geometry("1300x1200")

#fucntion to upload dataset

def uploadDataset():

global filename, dataset

text.delete('1.0', END)

filename = filedialog.askopenfilename(initialdir="Dataset") #upload dataset file

text.insert(END,filename+" loaded\n\n")

dataset = pd.read\_csv(filename) #read dataset from uploaded file

text.insert(END,"Dataset Values\n\n")

text.insert(END,str(dataset.head()))

text.update\_idletasks()

unique, count = np.unique(dataset['result'], return\_counts=True)

height = count

bars = labels

print(height)

print(bars)

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xticks(rotation=90)

plt.title("Various Cyber-Attacks Found in Dataset") #plot graph with various attacks

plt.show()

def preprocessing():

text.delete('1.0', END)

global dataset, scaler

global X\_train, X\_test, y\_train, y\_test, X, Y

#replace missing values with 0

dataset.fillna(0, inplace = True)

scaler = MinMaxScaler() #min max scaling for datset normalization

with open('model/minmax.txt', 'rb') as file:

scaler = pickle.load(file)

file.close()

dataset = dataset.values

X = dataset[:,0:dataset.shape[1]-1]

Y = dataset[:,dataset.shape[1]-1]

indices = np.arange(X.shape[0])

np.random.shuffle(indices) #shuffle dataset

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

X = scaler.transform(X)

text.insert(END,"Dataset after features normalization\n\n")

text.insert(END,str(X)+"\n\n")

text.insert(END,"Total records found in dataset : "+str(X.shape[0])+"\n")

text.insert(END,"Total features found in dataset: "+str(X.shape[1])+"\n\n")

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

text.insert(END,"Dataset Train and Test Split\n\n")

text.insert(END,"80% dataset records used to train ML algorithms : "+str(X\_train.shape[0])+"\n")

text.insert(END,"20% dataset records used to train ML algorithms : "+str(X\_test.shape[0])+"\n")

def calculateMetrics(algorithm, predict, y\_test):

a = accuracy\_score(y\_test,predict)\*100

p = precision\_score(y\_test, predict,average='macro') \* 100

r = recall\_score(y\_test, predict,average='macro') \* 100

f = f1\_score(y\_test, predict,average='macro') \* 100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

text.insert(END,algorithm+" Accuracy : "+str(a)+"\n")

text.insert(END,algorithm+" Precision : "+str(p)+"\n")

text.insert(END,algorithm+" Recall : "+str(r)+"\n")

text.insert(END,algorithm+" FScore : "+str(f)+"\n\n")

def runAutoEncoder():

text.delete('1.0', END)

global X\_train, X\_test, y\_train, y\_test, X, Y

global autoencoder

global accuracy, precision, recall, fscore

accuracy = []

precision = []

recall = []

fscore = []

if os.path.exists("model/encoder\_model.json"):

with open('model/encoder\_model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

autoencoder = model\_from\_json(loaded\_model\_json)

json\_file.close()

autoencoder.load\_weights("model/encoder\_model\_weights.h5")

autoencoder.\_make\_predict\_function()

else:

encoding\_dim = 256 # encoding dimesnion is 32 which means each row will be filtered 32 times to get important features from dataset

input\_size = keras.Input(shape=(X.shape[1],)) #we are taking input size

encoded = layers.Dense(encoding\_dim, activation='relu')(input\_size) #creating dense layer to start filtering dataset with given 32 filter dimension

decoded = layers.Dense(y\_train.shape[1], activation='softmax')(encoded) #creating another layer with input size as 784 for encoding

autoencoder = keras.Model(input\_size, decoded) #creating decoded layer to get prediction result

encoder = keras.Model(input\_size, encoded)#creating encoder object with encoded and input images

encoded\_input = keras.Input(shape=(encoding\_dim,))#creating another layer for same input dimension

decoder\_layer = autoencoder.layers[-1] #holding last layer

decoder = keras.Model(encoded\_input, decoder\_layer(encoded\_input))#merging last layer with encoded input layer

autoencoder.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])#compiling model

hist = autoencoder.fit(X\_train, y\_train, epochs=300, batch\_size=16, shuffle=True, validation\_data=(X\_test, y\_test))#now start generating model with given Xtrain as input

autoencoder.save\_weights('model/encoder\_model\_weights.h5')#above line for creating model will take 100 iterations

model\_json = autoencoder.to\_json() #saving model

with open("model/encoder\_model.json", "w") as json\_file:

json\_file.write(model\_json)

json\_file.close

print(autoencoder.summary())#printing model summary

predict = autoencoder.predict(X\_test)

predict = np.argmax(predict, axis=1)

testY = np.argmax(y\_test, axis=1)

calculateMetrics("AutoEncoder", predict, testY)

def runDecisionTree():

global autoencoder, decision\_tree, encoder\_model, vector

global X\_train, X\_test, y\_train, y\_test, X, Y, pca

encoder\_model = Model(autoencoder.inputs, autoencoder.layers[-1].output)#creating autoencoder model

vector = encoder\_model.predict(X) #extracting features using autoencoder

pca = PCA(n\_components = 7) #applying PCA for features reduction

vector = pca.fit\_transform(vector)

Y1 = np.argmax(Y, axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(vector, Y1, test\_size=0.2)

decision\_tree = DecisionTreeClassifier() #defining decision tree

decision\_tree.fit(vector, Y1) #training with decision tree

predict = decision\_tree.predict(X\_test)

text.insert(END,"Decision Tree Trained on New Features Extracted from AutoEncoder\n")

calculateMetrics("Decision Tree", predict, y\_test)

def runDNN():

global autoencoder, decision\_tree, encoder\_model, dnn, vector

global X\_train, X\_test, y\_train, y\_test, X, Y

attack\_type = []

for i in range(len(vector)):

temp = []

temp.append(vector[i])

attack = decision\_tree.predict(np.asarray(temp)) #using decision tree we are predicting attack type

attack\_type.append(attack[0])

attack\_type = np.asarray(attack\_type)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(vector, attack\_type, test\_size=0.2)

dnn = MLPClassifier() #defining DNN algorithm

dnn.fit(vector, attack\_type) #train DNN with various attack type

predict = dnn.predict(X\_test) #predict label forr unknown attack

text.insert(END,"Attack Prediction using DNN\n")

calculateMetrics("DNN", predict, y\_test)

def attackAttributeDetection():

text.delete('1.0', END)

global autoencoder, decision\_tree, encoder\_model, dnn, pca

filename = filedialog.askopenfilename(initialdir="Dataset")

dataset = pd.read\_csv(filename)

dataset.fillna(0, inplace = True)

values = dataset.values

temp = dataset.values

temp = scaler.transform(temp)

test\_vector = encoder\_model.predict(temp) #extracting features using autoencoder

test\_vector = pca.transform(test\_vector)

print(test\_vector.shape)

predict = dnn.predict(test\_vector)

for i in range(len(predict)):

if predict[i] == 0:

text.insert(END,"New Test Data : "+str(values[i])+" ====> NO CYBER ATTACK DETECTED\n\n")

else:

text.insert(END,"New Test Data : "+str(values[i])+" ====> CYBER ATTACK DETECTED Attribution Label : "+str(labels[predict[i]])+"\n\n")

def graph():

df = pd.DataFrame([['AutoEncoder','Precision',precision[0]],['AutoEncoder','Recall',recall[0]],['AutoEncoder','F1 Score',fscore[0]],['AutoEncoder','Accuracy',accuracy[0]],

['Decision Tree with PCA','Precision',precision[1]],['Decision Tree with PCA','Recall',recall[1]],['Decision Tree with PCA','F1 Score',fscore[1]],['Decision Tree with PCA','Accuracy',accuracy[1]],

['DNN','Precision',precision[2]],['DNN','Recall',recall[2]],['DNN','F1 Score',fscore[2]],['DNN','Accuracy',accuracy[2]],

],columns=['Algorithms','Performance Output','Value'])

df.pivot("Algorithms", "Performance Output", "Value").plot(kind='bar')

plt.show()

def comparisonTable():

output = "<html><body><table align=center border=1><tr><th>Algorithm Name</th><th>Accuracy</th><th>Precision</th><th>Recall</th>"

output+="<th>FSCORE</th></tr>"

output+="<tr><td>AutoEncoder</td><td>"+str(accuracy[0])+"</td><td>"+str(precision[0])+"</td><td>"+str(recall[0])+"</td><td>"+str(fscore[0])+"</td></tr>"

output+="<tr><td>Decision Tree with PCA</td><td>"+str(accuracy[1])+"</td><td>"+str(precision[1])+"</td><td>"+str(recall[1])+"</td><td>"+str(fscore[1])+"</td></tr>"

output+="<tr><td>DNN</td><td>"+str(accuracy[2])+"</td><td>"+str(precision[2])+"</td><td>"+str(recall[2])+"</td><td>"+str(fscore[2])+"</td></tr>"

output+="</table></body></html>"

f = open("table.html", "w")

f.write(output)

f.close()

webbrowser.open("table.html",new=2)

font = ('times', 16, 'bold')

title = Label(main, text='Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems')

title.config(bg='greenyellow', fg='dodger blue')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=50,y=120)

text.config(font=font1)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload SWAT Water Dataset", command=uploadDataset)

uploadButton.place(x=50,y=550)

uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess Dataset", command=preprocessing)

processButton.place(x=330,y=550)

processButton.config(font=font1)

autoButton = Button(main, text="Run AutoEncoder Algorithm", command=runAutoEncoder)

autoButton.place(x=630,y=550)

autoButton.config(font=font1)

dtButton = Button(main, text="Run Decision Tree with PCA", command=runDecisionTree)

dtButton.place(x=920,y=550)

dtButton.config(font=font1)

dnnButton = Button(main, text="Run DNN Algorithm", command=runDNN)

dnnButton.place(x=50,y=600)

dnnButton.config(font=font1)

attributeButton = Button(main, text="Detection & Attribute Attack Type", command=attackAttributeDetection)

attributeButton.place(x=330,y=600)

attributeButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=630,y=600)

graphButton.config(font=font1)

tableButton = Button(main, text="Comparison Table", command=comparisonTable)

tableButton.place(x=920,y=600)

tableButton.config(font=font1)

main.config(bg='LightSkyBlue')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

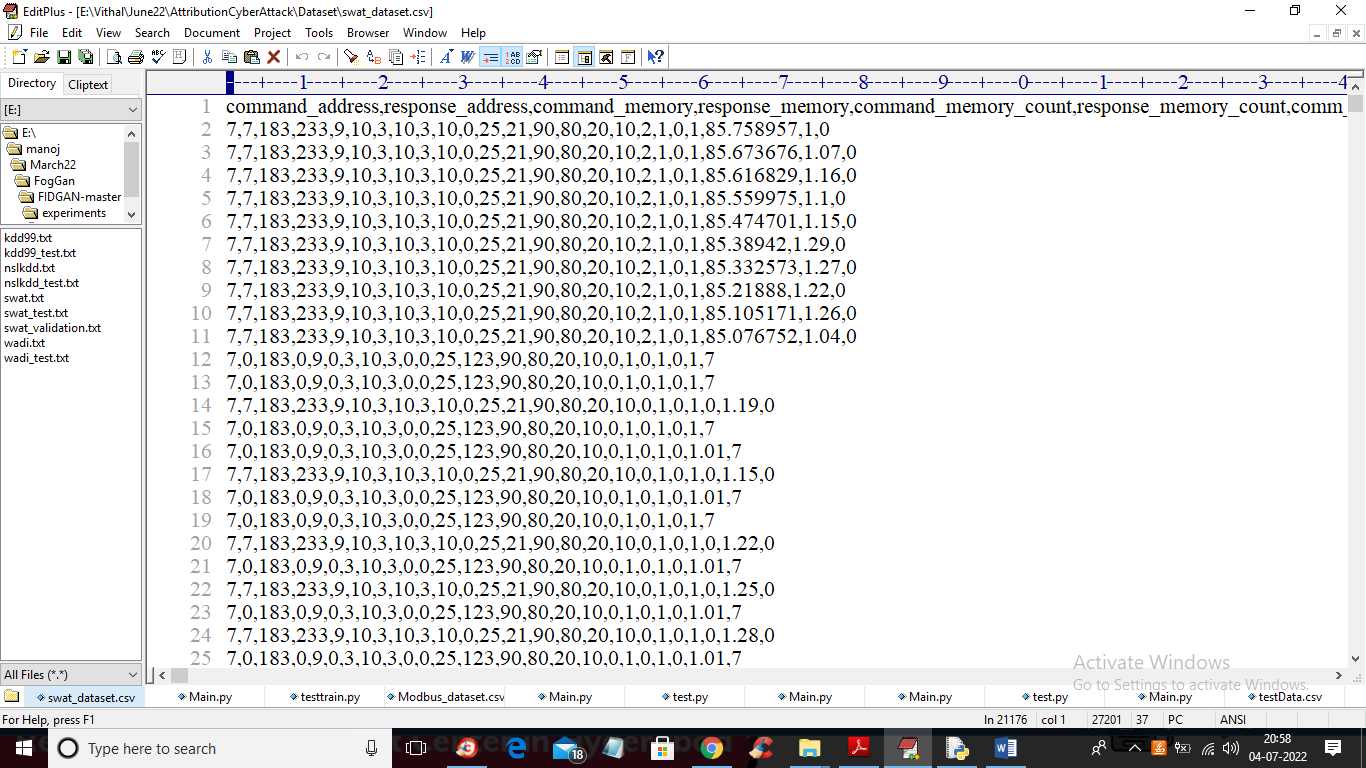
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload swat water dataset | Test whether the Dataset swat is uploaded or not into the system | the Personality Dataset may not uploaded | We cannot do further operations | Dataset uploaded we will do further operations | High | High |
| 02 | preprocess dataset | Test whether the preprocess Dataset uploaded not into the system | the dataset may not uploaded | we cannot do  pre process dataset | Up Load dataset  we will do further | High | High |
| 03 | run auto encoder algorithm | Test whether run auto encoder algorithm run successful or not | If the encoder is  Not sent | We cannot run algorithm | operations we will do further | High | High |
| 04 | run decision tree with PCA | run successfully decision tree or not | Without  Run pca | we cannot  do extended pca | operations we will do further | High | High |
| 05 | run DNN algorithm | Verify run Rule biased cusum detector | Without detected | we cannot run dnn | operations we will do further | High | High |
| 06 | detection and attack attribute type | Verify attack detion graph successfull or not | Without detected attribute | We cannot do attribute | operations we will do further | high | high |
| 07 | camparsion graph | Verify run Rule biased cusum detector | Without camparsion | we cannot do graph | operations we will do further | High | High |
| 08 | camparsion table | Verify attack detion graph successfull or not | Without camparsion table | We cannot do table | operations we will do further | high | high |

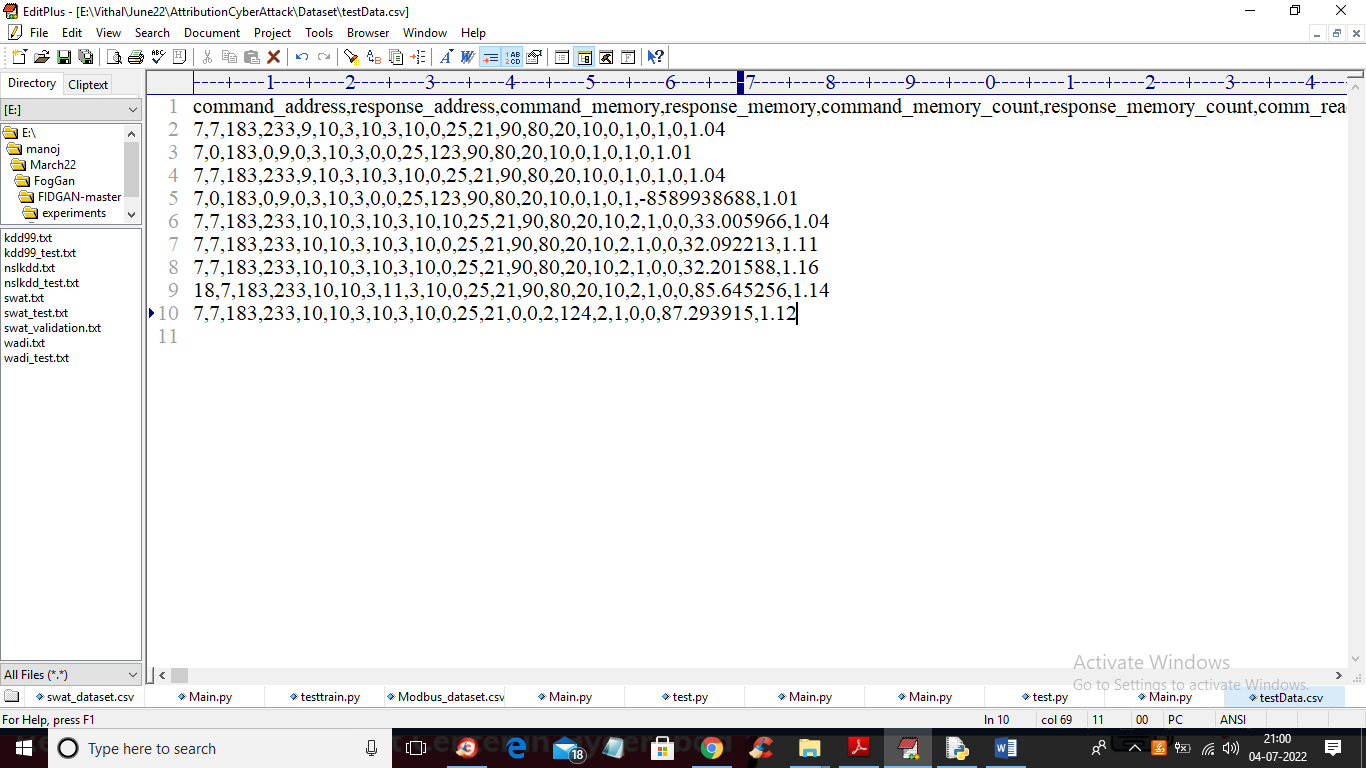
**7. SCREENSHOTS:**

Above are the attacks found in dataset and dataset contains above labels as integer value of its index for example NORMAL label index will be 0 and continues up to 8 class labels. Below screen showing dataset details



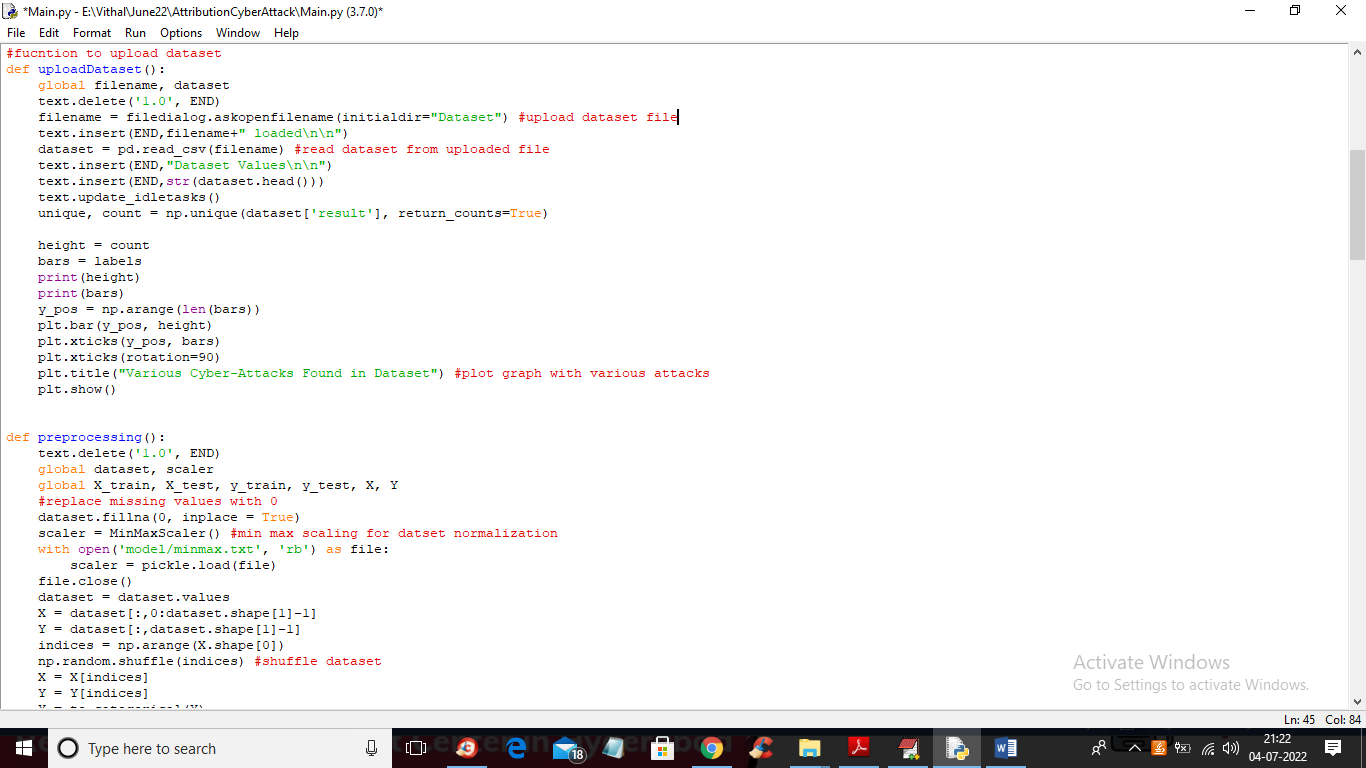
In above dataset screen first row contains dataset column names and remaining rows contains dataset values and in last column we have attack type from label 0 to 7. We will used above dataset to train propose Auto Encoder, decision tree and DNN algorithms.

In below screen we are using NEW test data which contains only signature and there is no class label and propose algorithm will detect and attribute class labels.

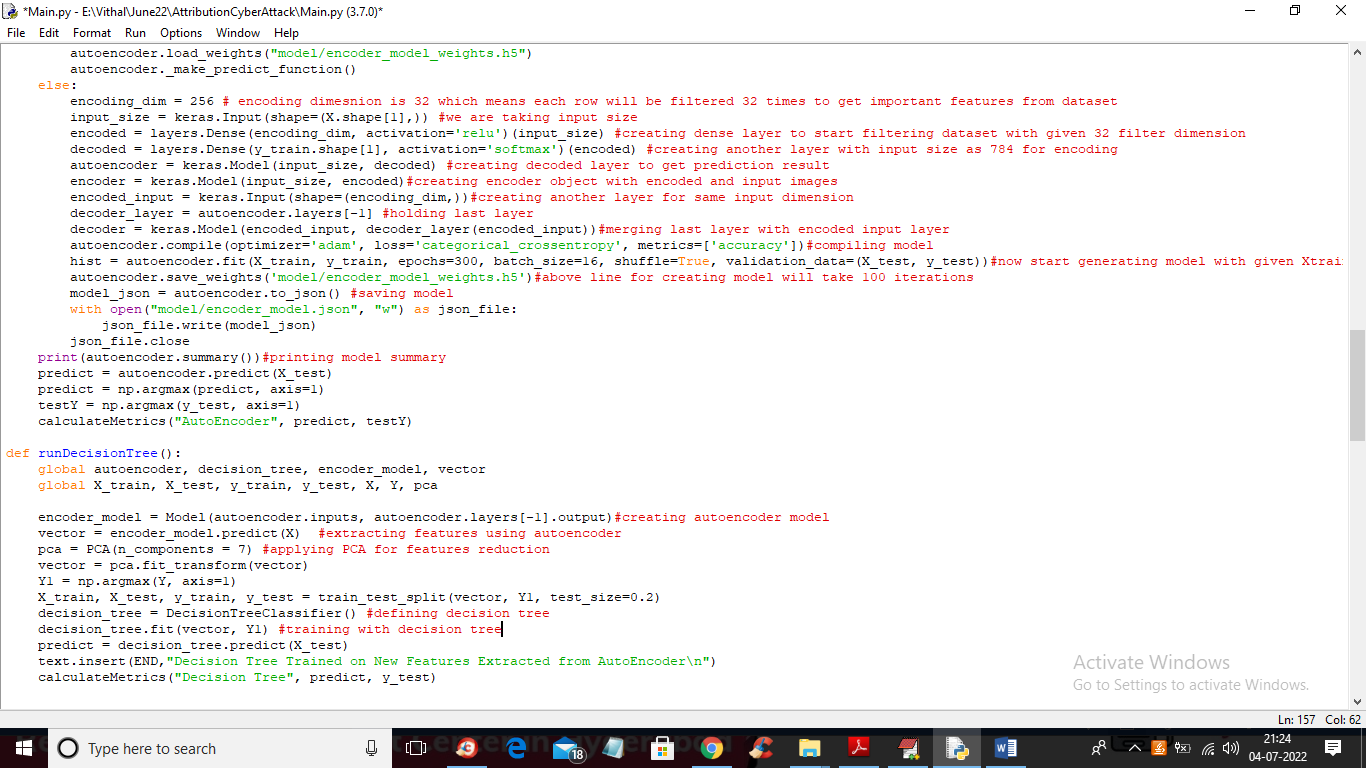


In above test data we have IOT request signature without class labels.

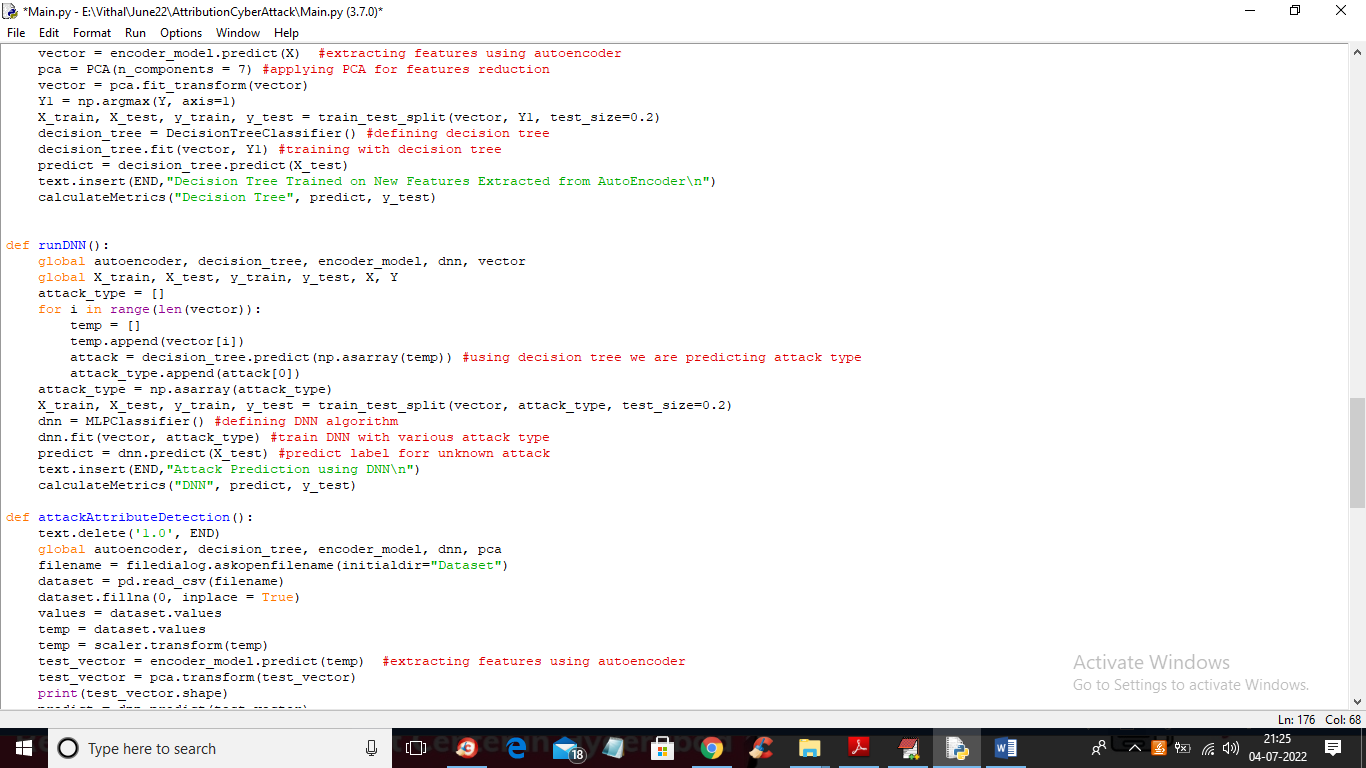
In below screen you can read red color comments to know about algorithms implementation



In above screen read red colour comments to know about dataset loading and min-max normalization



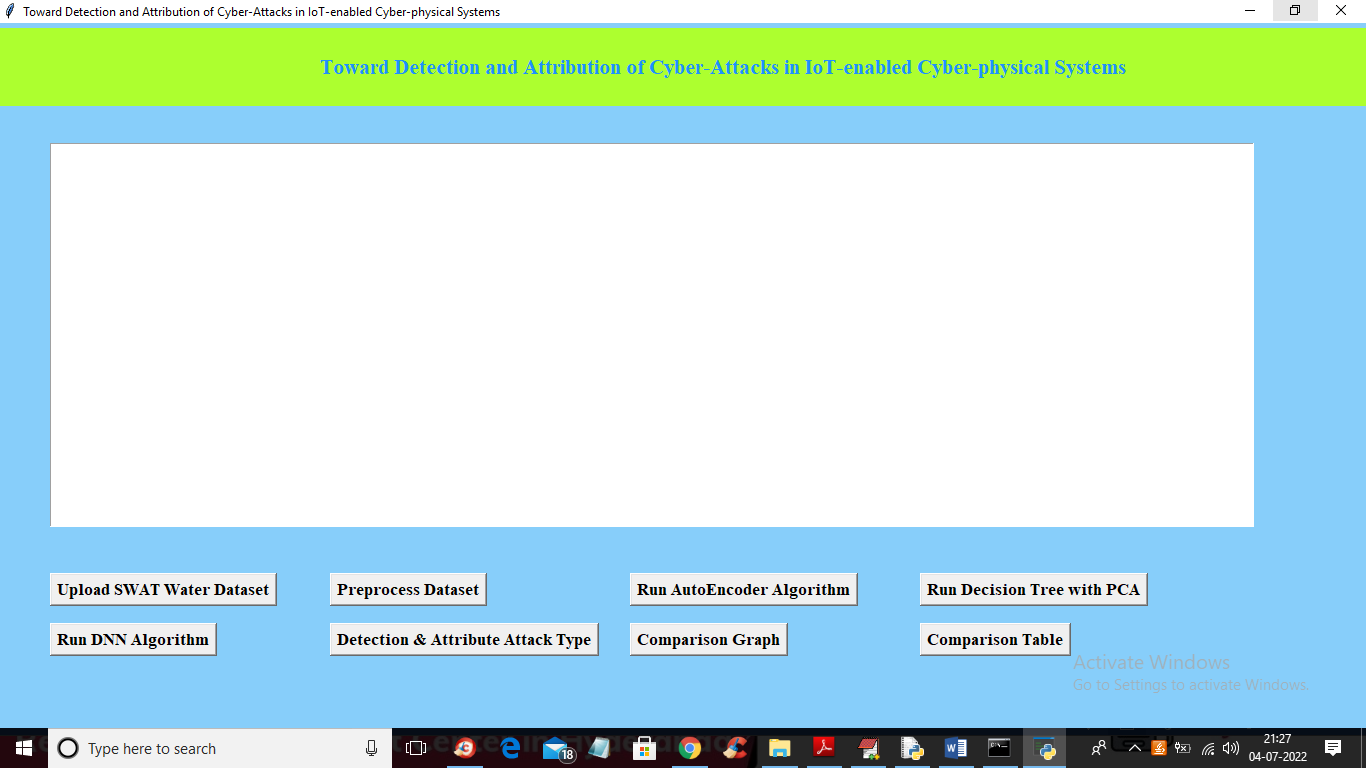
In above screen you can see we are using Auto Encoder, PCA and decision tree to train dataset and in below screen we are using DNN algorithms to train dataset with predicted labels from Decision Tree.



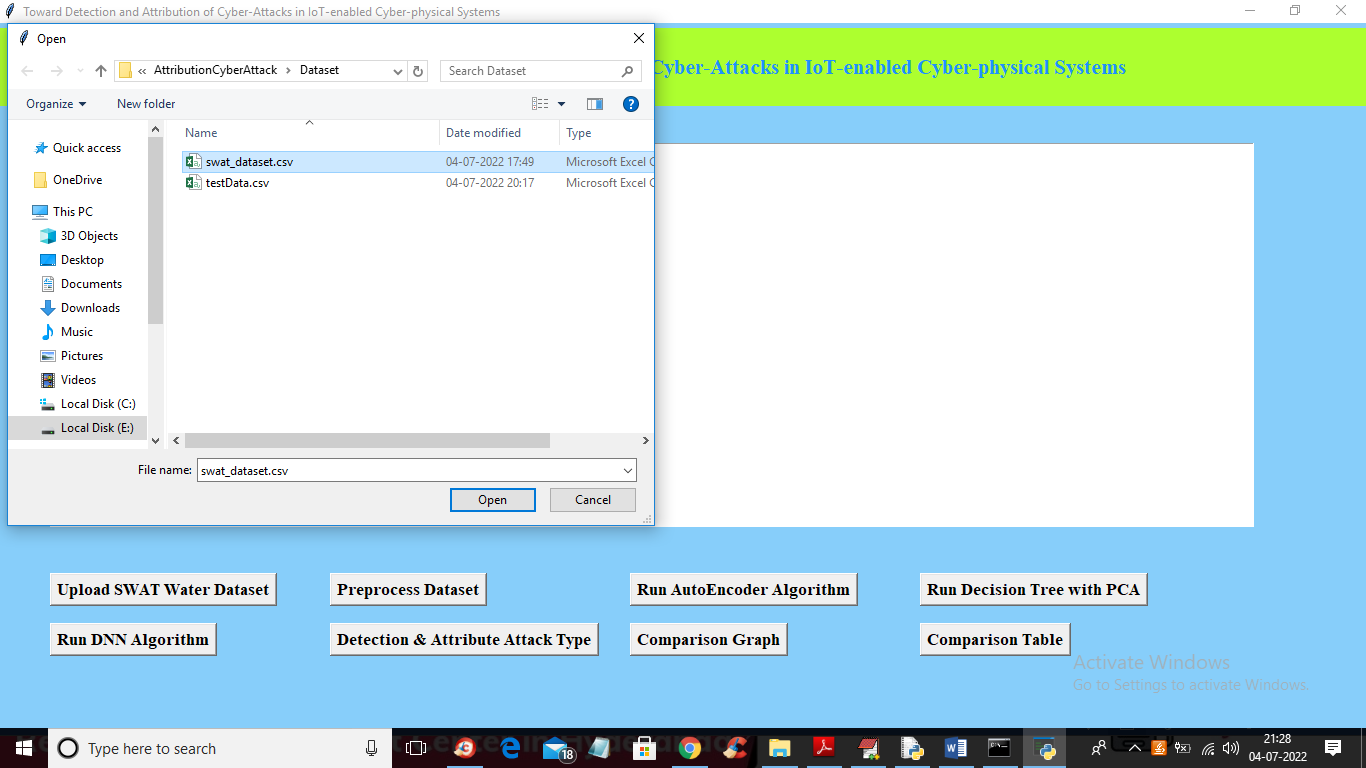
In above screen we are training dataset with DNN algorithms

SCREEN SHOTS

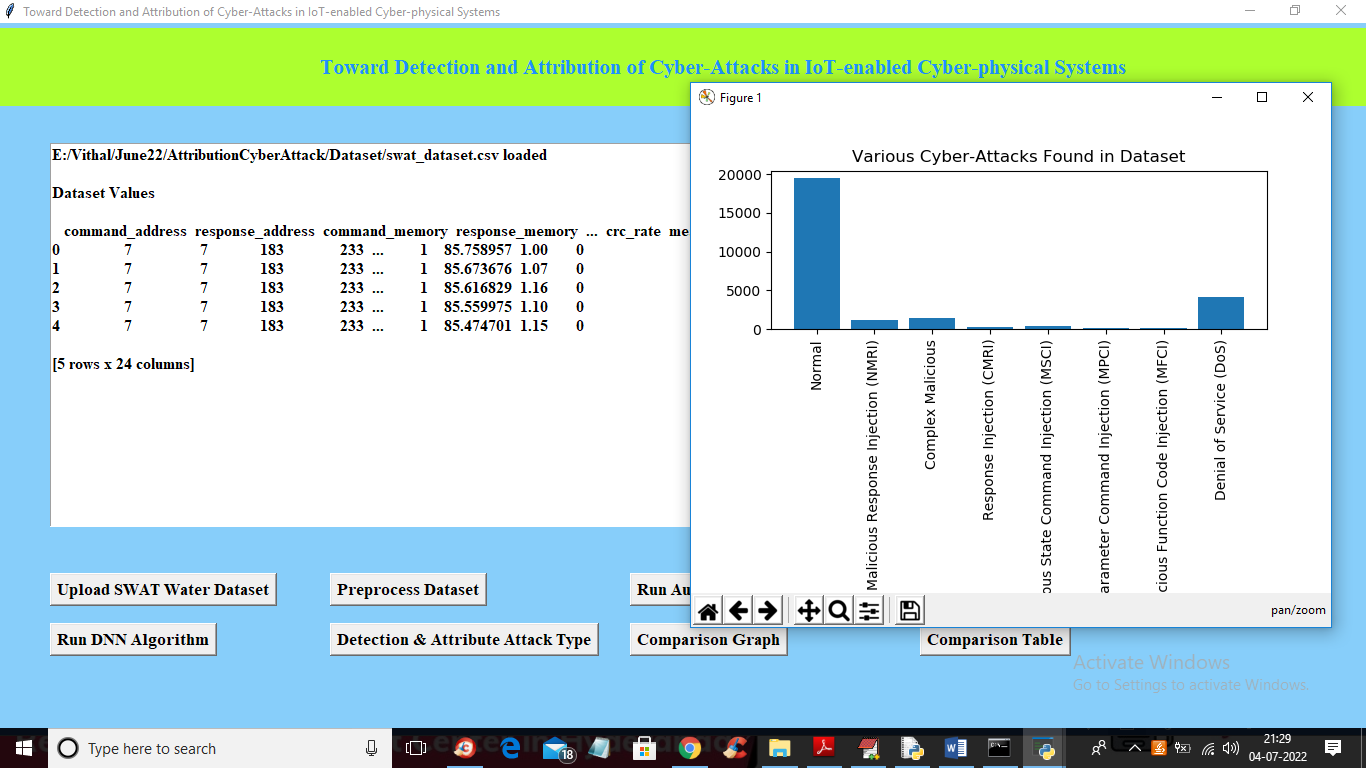
To run project double click on ‘run.bat’ file to get below screen



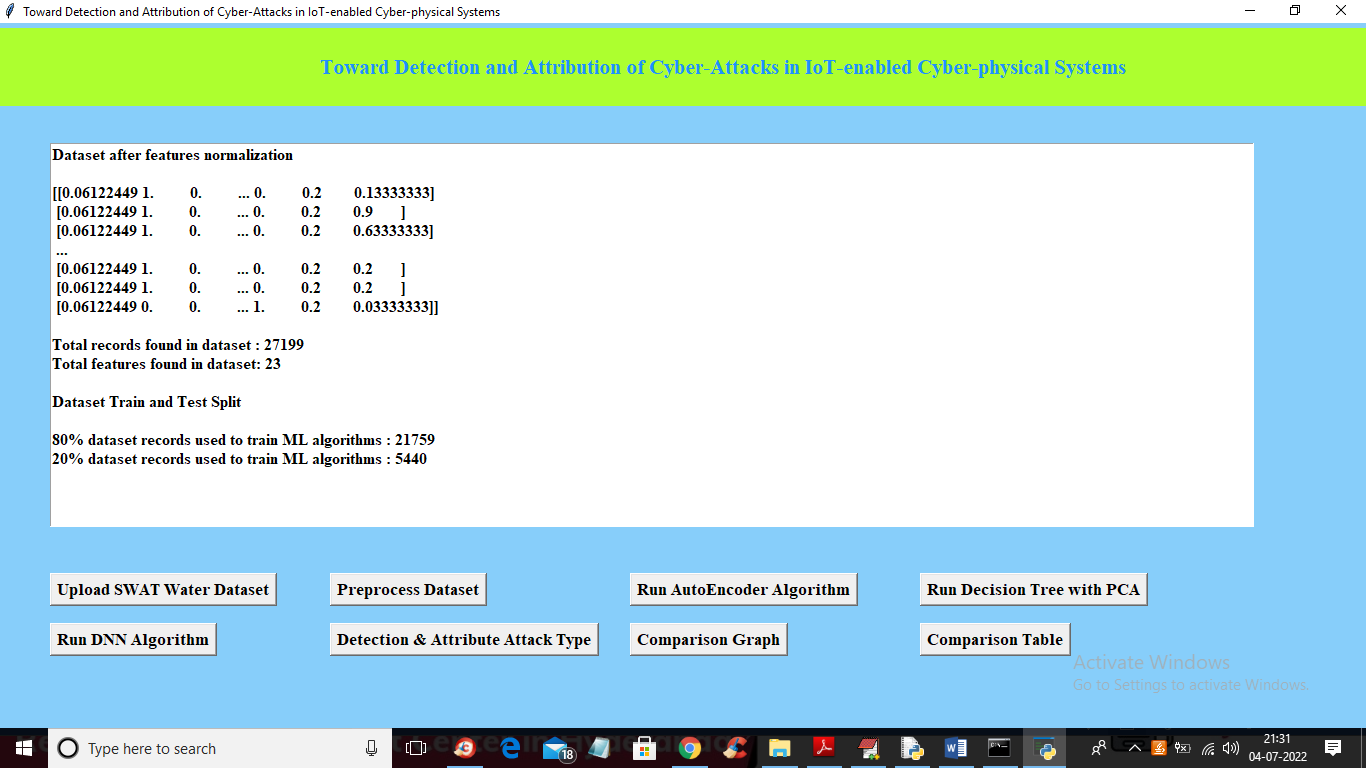
In above screen click on ‘Upload SWAT Water Dataset’ button to upload dataset to application and get below output



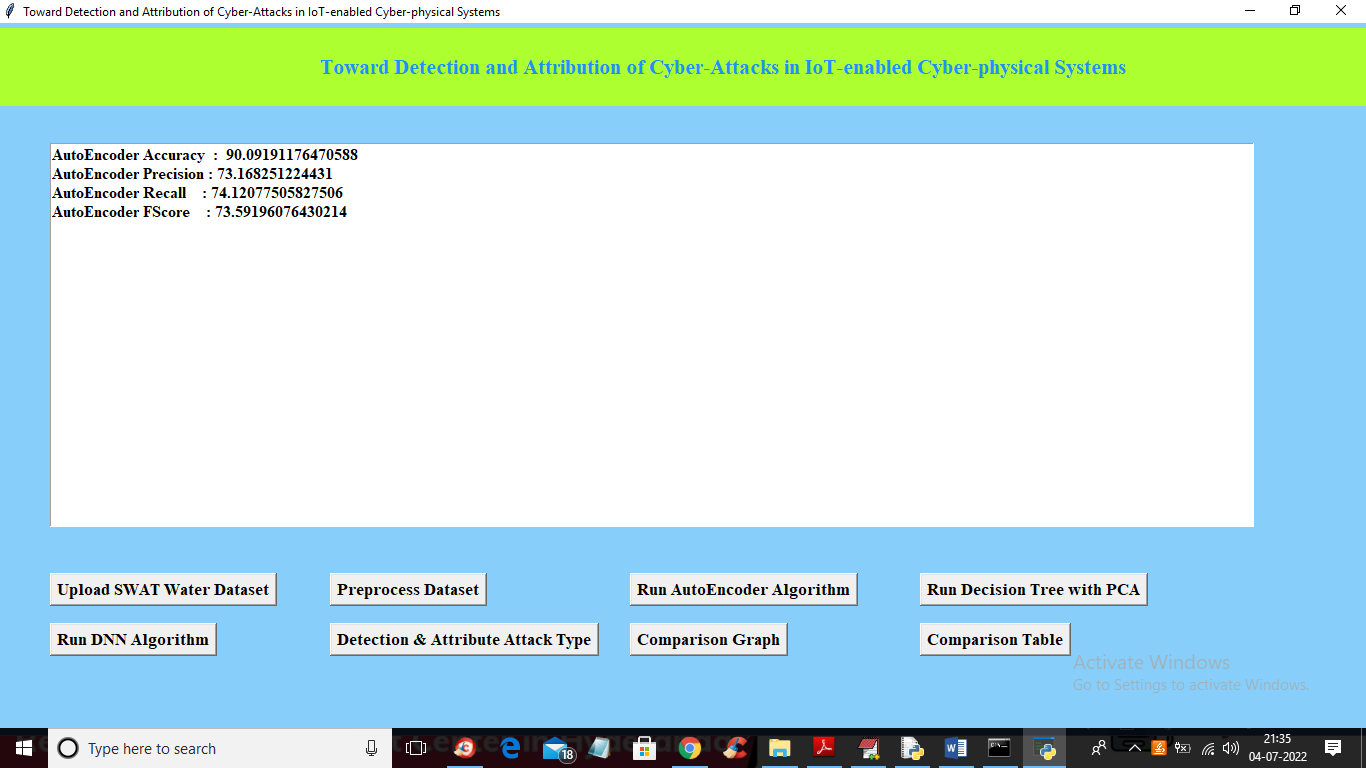
In above screen selecting and uploading SWAT dataset file and then click on ‘Open’ button to load dataset and get below output



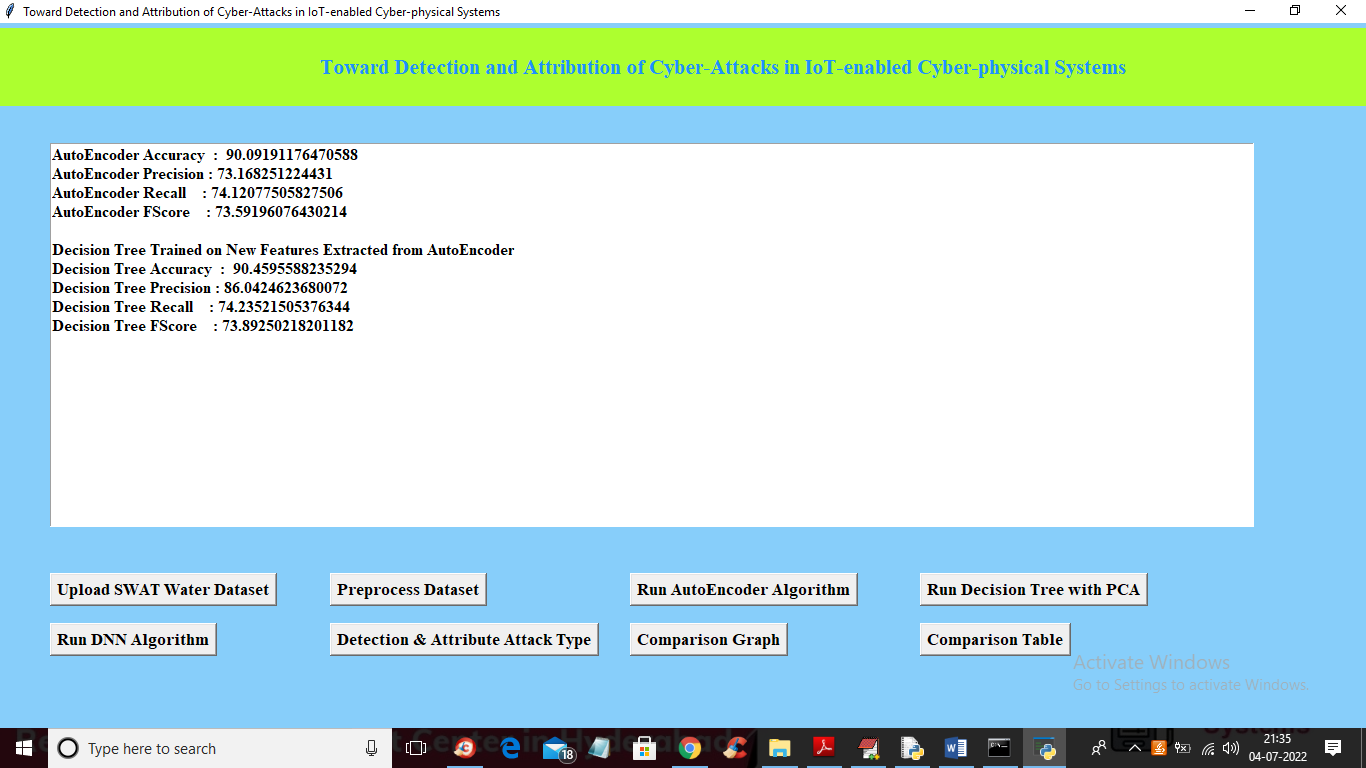
In above screen dataset loaded and in graph x-axis contains ATTACK NAME and y-axis contains count of those attacks found in dataset and we can see ‘NORMAL’ class contains so many records and other attacks contains very few records so it will raise data imbalance problem which can be solved using AutoEncoder, Decision Tree and DNN. Now close above graph and then click on ‘Preprocess Dataset’ button to remove missing values and then normalized values with MIN-MAX algorithm



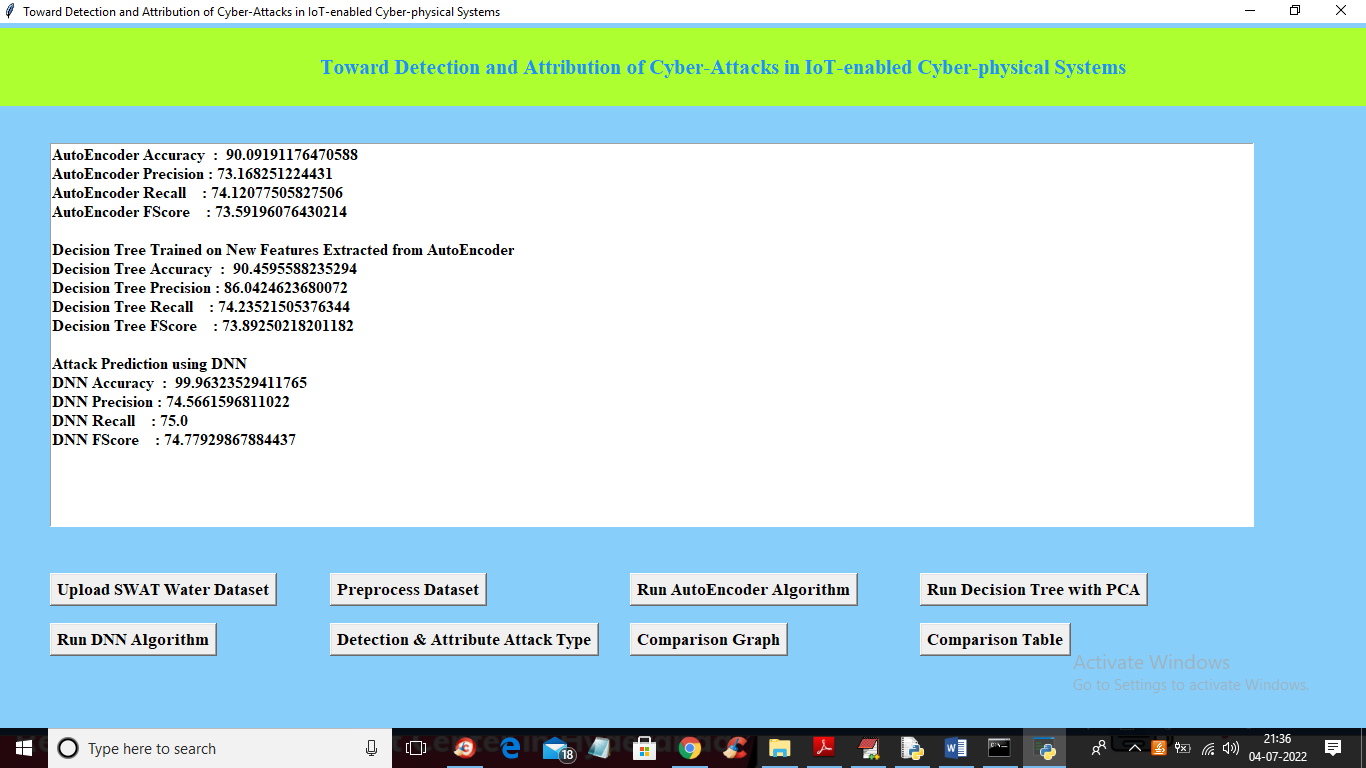
In above screen all values are normalized ( converting data between 0 and 1 called as normalization) and then we can see total records in dataset and then dataset train and test split records count also displaying. Now dataset is ready and now click on ‘Run AutoEncoder Algorithm’ button to train dataset with AutoEncoder and get below accuracy



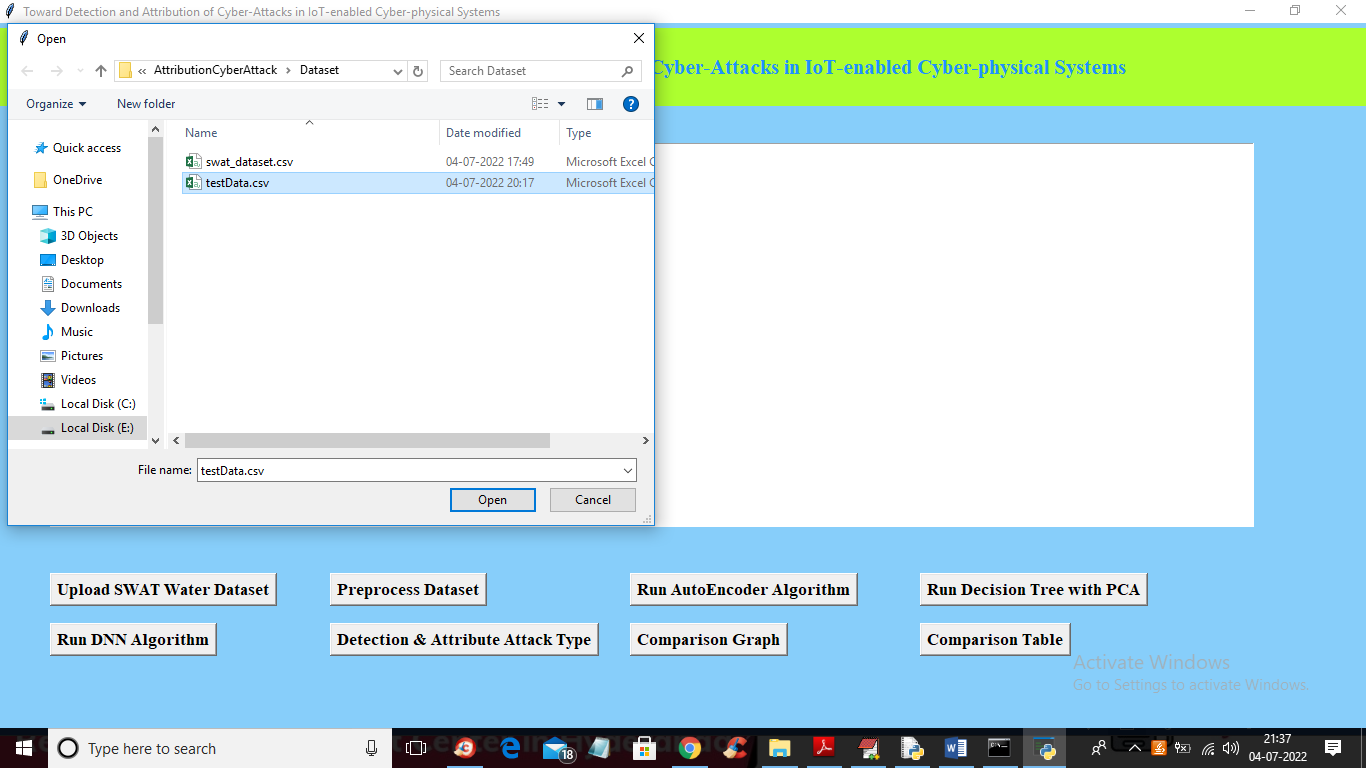
In above screen with AutoEncoder we got 90% accuracy and this accuracy can be enhance by implementing Decision Tree with PCA algorithm and now click on ‘Run Decision Tree with PCA’ button to get below output



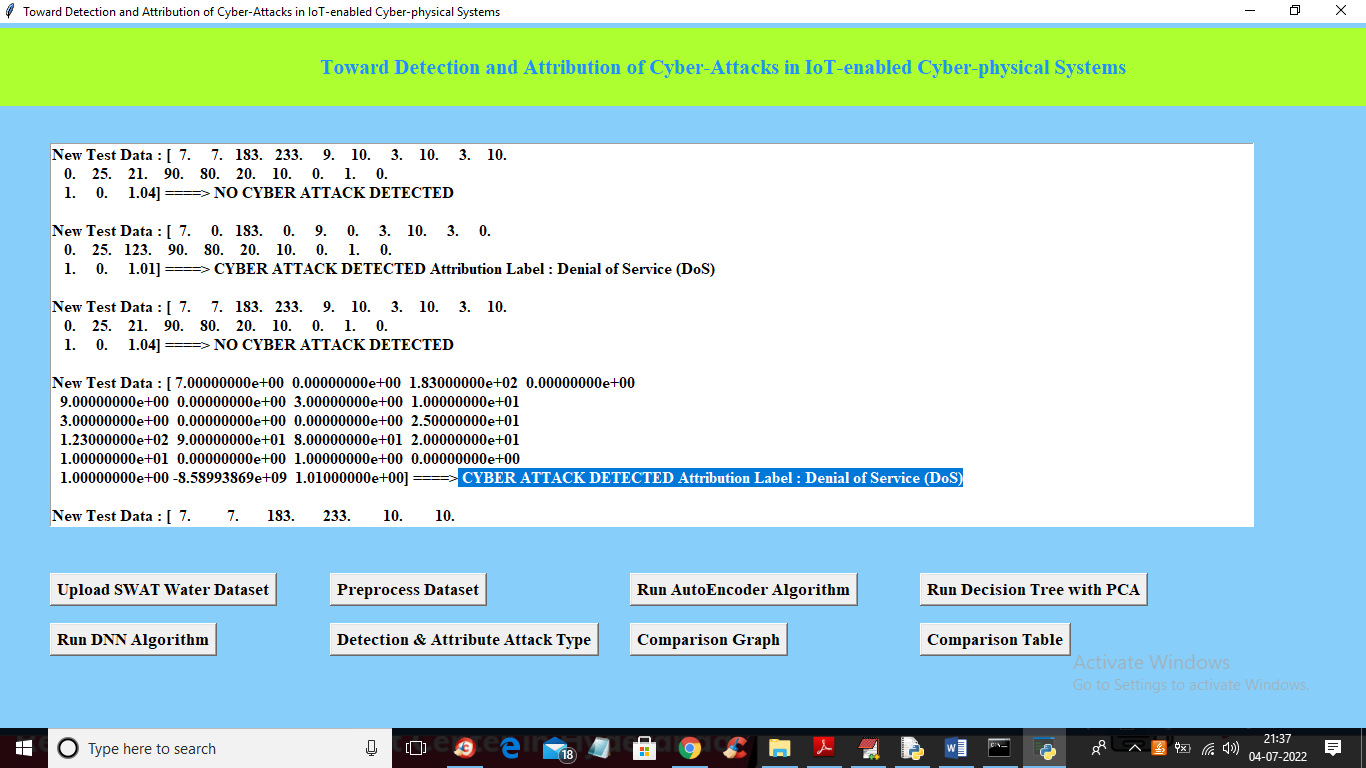
In above screen we can see with decision tree accuracy and precision value is enhanced and now click on ‘Run DNN Algorithm’ button to further enhance accuracy and get below output



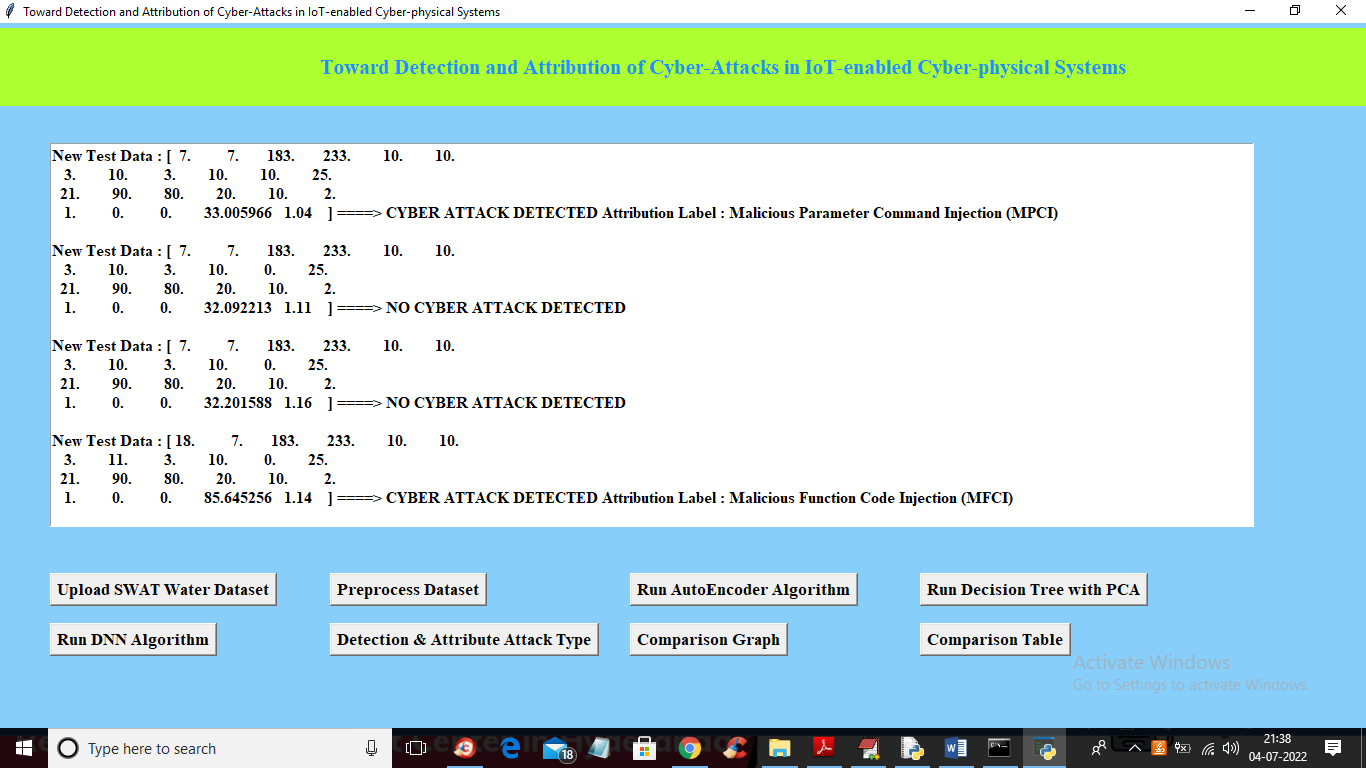
In above screen with DNN we got 99% accuracy and now click on ‘Detection & Attribute Attack Type’ button to upload test DATA and detect attack attributes



In above screen selecting and uploading ‘TEST DATA’ file and then click on ‘Open’ button to get below output



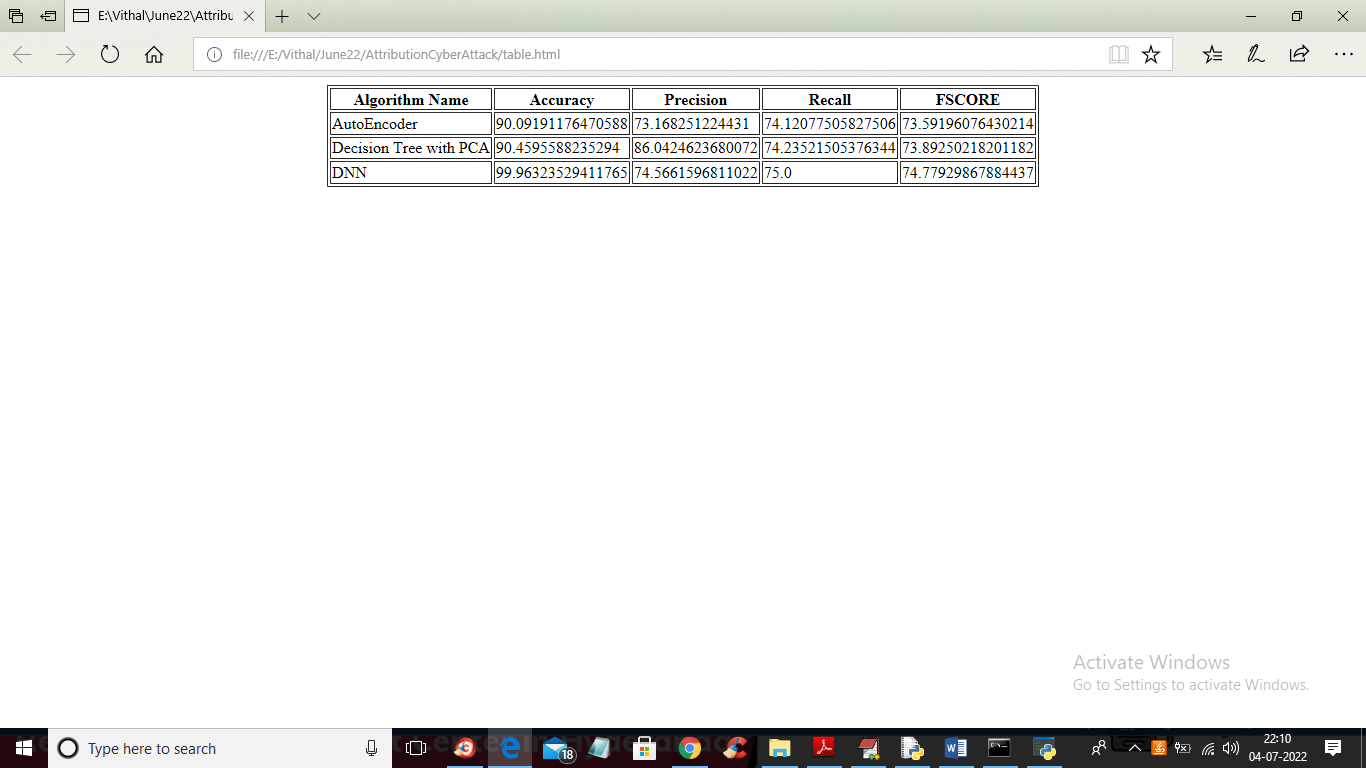
In above screen in square bracket we can see TEST data values and after arrow =🡺 symbol we can see detected ATTACK TYPE and scroll down above text area to view all detection



In above screen we can see detected various attacks and now click on ‘Comparison Graph’ button to get below graph



In above graph x-axis represents algorithms names and y-axis represents different metric values such as precision, recall, accuracy and FSCORE with different colour bars and in all algorithms DNN got high accuracy and now close above graph and then click on ‘Comparison Table’ to get below comparison table of all algorithms



In above table we can see algorithm names and its metrics values such as accuracy and precision and other.

**8. CONCLUSION:**

This paper proposed a novel two-stage ensemble deep learning-based attack detection and attack attribution framework for imbalanced ICS data. The attack detection stage uses deep representation learning to map the samples to the new higher dimensional space and applies a DT to detect the attack samples. This stage is robust to imbalanced datasets and capable of detecting previously unseen attacks. The attack attribution stage is an ensemble of several one-vs-all classifiers, each trained on a specific attack attribute. The entire model forms a complex DNN with a partially connected and fully connected component that can accurately attribute cyber attacks, as demonstrated. Despite the complex architecture of the proposed framework, the computational complexity of the training and testing phases are respectively (n is the number of training samples), which are similar to those of other DNN-based techniques in the literature. Moreover, the proposed framework can detect and attribute the samples timely with a better recall and f-measure than previous works.

**9. REFERENCES:**

[1] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble, “Multilayer Data-Driven Cyber-Attack Detection System for Industrial Control Systems Based on Network, System, and Process Data,” IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4362–4369, 2019.

[2] R. Ma, P. Cheng, Z. Zhang, W. Liu, Q. Wang, and Q. Wei, “Stealthy Attack Against Redundant Controller Architecture of Industrial CyberPhysical System,” IEEE Internet of Things Journal, vol. 6, no. 6, pp. 9783–9793, 2019.

[3] E. Nakashima, “Foreign hackers targeted U.S. water plant in apparent malicious cyber attack, expert says.” [Online]. Available: https://www.washingtonpost.com/blogs/checkpointwashington/post/foreign-hackers-broke-into-illinois-water-plant-controlsystem-industry-expert-says/2011/11/18/gIQAgmTZYN blog.html

[4] G. Falco, C. Caldera, and H. Shrobe, “IIoT Cybersecurity Risk Modeling for SCADA Systems,” IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4486–4495, 2018.

[5] J. Yang, C. Zhou, S. Yang, H. Xu, and B. Hu, “Anomaly Detection Based on Zone Partition for Security Protection of Industrial Cyber-Physical Systems,” IEEE Transactions on Industrial Electronics, vol. 65, no. 5, pp. 4257–4267, 2018.

[6] S. Ponomarev and T. Atkison, “Industrial control system network intrusion detection by telemetry analysis,” IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 2, pp. 252–260, 2016.

[7] J. F. Clemente, “No cyber security for critical energy infrastructure,” Ph.D. dissertation, Naval Postgraduate School, 2018. [8] C. Bellinger, S. Sharma, and N. Japkowicz, “One-class versus binary classification: Which and when?” in 2012 11th International Conference on Machine Learning and Applications, vol. 2, 2012, pp. 102–106.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org>

[10] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, “Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things,” IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6822–6834, 2019.

[12] I. A. Khan, D. Pi, Z. U. Khan, Y. Hussain, and A. Nawaz, “HML-IDS: A hybrid-multilevel anomaly prediction approach for intrusion detection in SCADA systems,” IEEE Access, vol. 7, pp. 89 507–89 521, 2019. [13] T. K. Das, S. Adepu, and J. Zhou, “Anomaly detection in industrial control systems using logical analysis of data,” Computers & Security, vol. 96, p. 101935, 2020.

[14] J. J. Q. Yu, Y. Hou, and V. O. K. Li, “Online False Data Injection Attack Detection With Wavelet Transform and Deep Neural Networks,” IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3271–3280, 2018. [15] M. M. N. Aboelwafa, K. G. Seddik, M. H. Eldefrawy, Y. Gadallah, and M. Gidlund, “A machine-learning-based technique for false data injection attacks detection in industrial iot,” IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8462–8471, 2020.

[16] W. Yan, L. K. Mestha, and M. Abbaszadeh, “Attack detection for securing cyber physical systems,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8471–8481, 2019.

[17] A. Cook, A. Nicholson, H. Janicke, L. Maglaras, and R. Smith, “Attribution of Cyber Attacks on Industrial Control Systems,” EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, vol. 3, no. 7, p. 151158, 2016.

[18] L. Maglaras, M. Ferrag, A. Derhab, M. Mukherjee, H. Janicke, and S. Rallis, “Threats, Countermeasures and Attribution of Cyber Attacks on Critical Infrastructures,” ICST Transactions on Security and Safety, vol. 5, no. 16, p. 155856, 2018.

[19] M. Alaeiyan, A. Dehghantanha, T. Dargahi, M. Conti, and S. Parsa, “A Multilabel Fuzzy Relevance Clustering System for Malware Attack Attribution in the Edge Layer of Cyber-Physical Networks,” ACM Transactions on Cyber-Physical Systems, vol. 4, no. 3, pp. 1–22, 2020.

[20] U. Noor, Z. Anwar, T. Amjad, and K.-K. R. Choo, “A machine learning-based FinTech cyber threat attribution framework using highlevel indicators of compromise,” Future Generation Computer Systems, vol. 96, pp. 227–242, 2019.