A

Mini Project On

**Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles**

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

By

JIYA GARG (207R1A6688)

ABHINAV LAKKARAJU (207R1A6698)

SANJAY KATUKOJWALA (207R1A6695)

Under the Guidance of

**Mr. Sk. Sharif**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

**CMR TECHNICAL CAMPUS UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**2020-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)**



**CERTIFICATE**

This is to certify that this Project entitled **“Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles”** submitted by **JIYA GARG (207R1A6688), ABHINAV LAKKARAJU (207R1A6698) & SANJAY KATUKOJWALA (207R1A6695)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (AI&ML) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mr.SK, Sharif Dr. S Rao Chintalapudi**

INTERNAL GUIDE HOD CSE(AI&ML)

**EXTERNAL EXAMINER**

**Submitted for Viva voce Examination held on**

**ACKNOWLEDGMENT**

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. Sk. Sharif** Associate Professor for his exemplary guidance, monitoring, and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. G. Vinoda Reddy, Dr. K. Mahesh, Md. Hafeena & D. Babu Rao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. S Rao Chintalapudi,** Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**JIYA GARG (207R1A6688)**

**ABHINAV LAKKARAJU (207R1A6698)**

**SANJAY KATUKOJWALA (207R1A6695)**

**ABSTRACT**

One of the major challenges in cybersecurity is the provision of an automated and effective cyber-threats detection technique. In this paper, we present an AI technique for cyber-threats detection, based on artificial neural networks. The proposed technique converts multitude of collected security events to individual event profiles and use a deep learning-based detection method for enhanced cyber- threat detection. For this work, we developed an AI-SIEM system based on a combination of event profiling for data preprocessing and different artificial neural network methods, including FCNN, CNN, and LSTM. The system focuses on discriminating between true positive and false positive alerts, thus helping security analysts to rapidly respond to cyber threats. All experiments in this study are performed by authors using two benchmark datasets (NSLKDD and CICIDS2017) and two datasets collected in the real world. To evaluate the performance comparison with existing methods, we conducted experiments using the five conventional machine-learning methods (SVM, k-NN, RF, NB, and DT). Consequently, the experimental results of this study ensure that our proposed methods are capable of being employed as learning-based models for network intrusion-detection, and show that although it is employed in the real world, the performance outperforms the conventional machine-learning methods.

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| Figure 3.1 | Project Architecture for | 7 |
|  |  |  |
| Figure 3.2 | Use Case Diagram | 8 |
| Figure 3.3 | Class Diagram | 9 |
|  |  |  |
| Figure 3.4 | Sequence diagram | 10 |
| Figure 3.5 | Activity diagram | 11 |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| Screenshot 5.1 | User Interface . | 16 |
| Screenshot 5.2 | Upload Train Dataset | 16 |
| Screenshot 5.3 | Model Execution | 17 |
| Screenshot 5.4 | Algorithm results | 17 |
| Screenshot 5.5 | Resultant Graphs. | 18 |
| Screenshot 5.6 | Display all Datasets | 18 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

**CHAPTHERS TITLE PAGE**

**NO NO**

1. INTRODUCTION 1
   1. Outline of Project 1
   2. [Domain Introduction 2](#_TOC_250036)
   3. [Benefits of Deep Neural Network 3](#_TOC_250035)
2. LITERATURE SURVEY 4
   1. Enhanced Network Anomaly Detection 4
   2. Network Intrusion Detection 5
   3. Learning hierarchical spatial-temporal 6
3. AIM AND SCOPE OF THE PROJECT 7
   1. [Aim 7](#_TOC_250034)
   2. [Scope of the project 7](#_TOC_250033)
   3. Objective 7
   4. System requirement 8
   5. [Hardware requirements 8](#_TOC_250032)
      1. Software requirements 9
   6. [Software features 9](#_TOC_250031)
      1. [Python 9](#_TOC_250030)
      2. Kara’s 10
      3. [IDE Setup 10](#_TOC_250029)
   7. [DJANGO 11](#_TOC_250028)
   8. [Algorithm 12](#_TOC_250027)
4. METHODOLOGY 13
   1. [Existing System 13](#_TOC_250026)
   2. [PROPOSED SYSTEM 13](#_TOC_250025)
   3. [SYSTEM ARCHITECTURE 14](#_TOC_250024)
   4. [MODULES 14](#_TOC_250023)
   5. [Module Description 14](#_TOC_250022)
      1. [UPLOAD DATASETS 15](#_TOC_250021)
      2. [TF-IDF ALGORITHM 16](#_TOC_250020)
      3. [GENRATE EVENT VECTOR 17](#_TOC_250019)
      4. [NEURAL NETWORKS PROFRING 17](#_TOC_250018)
      5. [SVM ALGORITHM 18](#_TOC_250017)
      6. [KNN ALGORITHM 19](#_TOC_250016)
      7. [RANDOM FOREST ALGORITHM 20](#_TOC_250015)
      8. [NAÏVE BAYES ALGORITHM 21](#_TOC_250014)
      9. [DECISION THREE ALGORITHM 21](#_TOC_250013)
   6. [SYSTEM STUDY 23](#_TOC_250012)
      1. [ECONOMICAL FEASIBILITY 23](#_TOC_250011)
      2. [TECHNICAL FEASIBILITY 23](#_TOC_250010)
      3. [SOCIAL FEASIBILITY 24](#_TOC_250009)
   7. [INPUT AND OUTPUT DESIGN 24](#_TOC_250008)
      1. [INPUT DESIGN 24](#_TOC_250007)
      2. [OUTPUT DESIGN 25](#_TOC_250006)
5. RESULTS AND DISCUSSIONS 26
   1. [DATASETS 26](#_TOC_250005)
   2. [DATA VISUALIZATION 27](#_TOC_250004)
   3. [Experimental Results 28](#_TOC_250003)
6. CONCLUSION AND FUTURE WORK 29
   1. [SUMMARY 29](#_TOC_250002)
   2. [CONCLUSION 30](#_TOC_250001)
   3. [FUTURE WORK 31](#_TOC_250000)
7. **REFERENCES**
   1. REFERENCES 32
   2. GITHUB LINK 32

**CHAPTER 1 INTRODUCTION**

* 1. **OUTLINE OF THE PROJECT**

With the emergence of artificial intelligence (AI) techniques, learning-based approaches for detecting cyberattacks, have become further improved, and they have achieved significant results in many studies. However, owing to constantly evolving cyberattacks, it is still highly challenging to protect IT systems against threats and malicious behaviors in networks. Because of various network intrusions and malicious activities, effective defenses and security considerations were given high priority for finding reliable solutions [1], [2], [3], [4].

Traditionally, there are two primary systems for detecting cyber-threats and network intrusions. An intrusion prevention system (IPS) is installed in the enterprise network, and can examine the network protocols and flows with signature-based methods primarily. It generates appropriate intrusion alerts, called the security events, and reports the generating alerts to another system, such as SIEM. The security information and event management (SIEM) has been focusing on collecting and managing the alerts of IPSs. The SIEM is the most common and dependable solution among various security operations solutions to analyze the collected security events and logs [5]. Moreover, security analysts make an effort to investigate suspicious alerts by policies and threshold, and to discover malicious behavior by analyzing correlations among events, using knowledge related to attacks.

A learning-based method geared toward determining whether an attack occurred in a large amount of data can be useful to analysts who need to instantly analyze numerous events. According to [10], information security solutions generally fall into two categories: analyst-driven and machine learning-driven solutions. Analyst- driven solutions rely on rules determined by security experts called analysts. Meanwhile, machine learning-driven solutions used to detect rare or anomalous patterns can improve detection of new cyber threats [10]. Nevertheless, while

networks, we observed that existing learning-based approaches have four main limitations.

First, learning-based detection methods require labeled data, which enable the training of the model and evaluation of generated learning models. Furthermore, it is not straightforward to obtain such labeled data at a scale that allow accurate training of a model. Despite the need for labeled data, many commercial SIEM solutions do not maintain labeled data that can be applied to supervised learning models [10].

Second, most of the learning features that are theoretically used in each study are not generalized features in the real world, because they are not contained in common network security systems [3]. Hence, it makes difficult to utilize to practical cases. Recent efforts on intrusion detection research have considered an automation approach with deep learning technologies, and performance has been evaluated using well known datasets like NSLKDD [11], CICIDS2017 [12], and Kyoto-Honeypot [13]. However, many previous studies used benchmark dataset, which, though accurate, are not generalizable to the real world because of the insufficient features. To overcome these limitations, an employed learning model requires to evaluate with datasets that are collected in the real world.

# DOMAIN INTRODUCTION

The Deep Neural Network (DNN) is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modelling to process data in complex ways.

A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy.” One of the keys uses of these sophisticated neural networks is dealing with un labelled or unstructured data.

learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

# BENEFITS OF DEEP NEURAL NETWORK

Neural networks use randomness by design to ensure they effectively learn the function being approximated for the problem. Randomness is used because this class of machine learning algorithm performs better with it than without.

The most common form of randomness used in neural networks is the random initialization of the network weights. Although randomness can be used in other areas, here is just a short list:

* + - Randomness in Initialization, such as weights.
    - Randomness in Regularization, such as dropout.
    - Randomness in Layers, such as word embedding.

**CHAPTER 2 LITERATURE SURVEY**

* 1. **Enhanced Network Anomaly Detection Based on Deep Neural Networks**

Due to the monumental growth of Internet applications in the last decade, the need for security of information network has increased manifolds. As a primary defense of network infrastructure, an intrusion detection system is expected to adapt to dynamically changing threat landscape. Many supervised and unsupervised techniques have been devised by researchers from the discipline of machine learning and data mining to achieve reliable detection of anomalies. Deep learning is an area of machine learning which applies neuron-like structure for learning tasks. Deep learning has profoundly changed the way we approach learning tasks by delivering monumental progress in different disciplines like speech processing, computer vision, and natural language processing to name a few. It is only relevant that this new technology must be investigated for information security applications. The aim of this paper is to investigate the suitability of deep learning approaches for anomaly-based intrusion detection system. For this research, we developed anomaly detection models based on different deep neural network structures, including convolutional neural networks, autoencoders, and recurrent neural networks. These deep models were trained on NSLKDD training data set and evaluated on both test data sets provided by NSLKDD, namely NSLKDD Test+ and NSLKDDTest21. All experiments in this paper are performed by authors on a GPU- based test bed. Conventional machine learning-based intrusion detection models were implemented using well-known classification techniques, including extreme learning machine, nearest neighbor, decision-tree, random-forest, support vector machine, naive-bays, and quadratic discriminant analysis. Both deep and conventional machine learning models were evaluated using well-known classification metrics, including receiver operating characteristics, area under curve, precision-recall curve, mean average precision and accuracy of classification.

Experimental results of deep IDS models showed promising results for real-world application in anomaly detection systems.

* 1. **Network Intrusion Detection Based on Directed Acyclic Graph and Belief Rule Base**

Intrusion detection is very important for network situation awareness. While a few methods have been proposed to detect network intrusion, they cannot directly and effectively utilize semi‐quantitative information consisting of expert knowledge and quantitative data. Hence, this paper proposes a new detection model based on a directed acyclic graph (DAG) and a belief rule base (BRB). In the proposed model, called DAG‐BRB, the DAG is employed to construct a multi‐layered BRB model that can avoid explosion of combinations of rule number because of a large number of types of intrusion. To obtain the optimal parameters of the DAG‐BRB model, an improved constraint covariance matrix adaption evolution strategy (CMA‐ES) is developed that can effectively solve the constraint problem in the BRB. A case study was used to test the efficiency of the proposed DAG‐BRB. The results showed that compared with other detection models, the DAG‐BRB model has a higher detection rate and can be used in real networks.

* 1. **HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks**

The development of an anomaly-based intrusion detection system (IDS) is a primary research direction in the field of intrusion detection. An IDS learns normal and anomalous behavior by analyzing network traffic and can detect unknown and new attacks. However, the performance of an IDS is highly dependent on feature design and designing a feature set that can accurately characterize network traffic is still an ongoing research issue. Anomaly-based IDSs also have the problem of a high false alarm rate (FAR), which seriously restricts their practical applications. In this paper, we propose a novel IDS called the hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS), which first learns the low-level spatial features of network traffic using deep convolutional neural networks (CNNs) and then learns high-level temporal features using long short-term memory networks.

automatically; no feature engineering techniques are required. The automatically learned traffic features effectively reduce the FAR. The standard DARPA1998 and ISCX2012 data sets are used to evaluate the performance of the proposed system. The experimental results show that the HAST-IDS outperforms other published approaches in terms of accuracy, detection rate, and FAR, which successfully demonstrates its effectiveness in both feature learning and FAR reduction.

**CHAPTER 3**

**AIM AND SCOPE OF THE PROJECT**

# AIM

The aim of the project is Threat detection and response is the most important aspect of cybersecurity for IT organizations that depend on cloud infrastructure. Threat detection, therefore, describes the ability of IT organizations to quickly and accurately identify threats to the network or to applications or other assets within the network.

# SCOPE OF THE PROJECT

A learning-based method geared toward determining whether an attack occurred in a large amount of data can be useful to analysts who need to instantly analyze numerous events. According to [10], information security solutions generally fall into two categories: analyst-driven and machine learning-driven solutions.

* 1. **OBJECTIVES**

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant Thus the objective of input design is to create an input layout that is easy to follow

* 1. **SYSTEM REQUIREMENTS**

For developing the application, the following are the Software Requirements:

* + 1. Python
    2. Django

**Operating Systems supported**

1. Windows 7
2. Windows 8
3. Windows 10

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser
  1. **Hardware Requirements**

For developing application, the following are the Hardware Requirements:

* Processor: Pentium IV or higher
* RAM: 256 MB
* Space on Hard Disk: minimum 512MB

# SOFTWARE FEATURES

# Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that code [readability](https://en.wikipedia.org/wiki/Readability) using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [C](https://en.wikipedia.org/wiki/CPython) [Python](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit [Python Software](https://en.wikipedia.org/wiki/Python_Software_Foundation) [Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation) Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory](https://en.wikipedia.org/wiki/Memory_management) [management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-](https://en.wikipedia.org/wiki/Object-oriented_programming) [oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

* + 1. **Kera**

Kera is Open-source Neural Network library written in Python that runs on top of Theano or Tensor flow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Kera doesn't handle low- level computation. Instead, it uses another library to do it, called the "Backend.

So Kera is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Kara’s high-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Kera also compiles our model with loss and optimizer functions, training process with fit function. Kera doesn't handle Low- Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

# IDE Setup

The Machine Learning Concepts can be well implemented through PYTHON We have a numerous Python Tools but for the DNN implementation Anaconda can meet our needs. Spyder in Anaconda is chosen as the IDE setup.

The latest Anaconda Spyder has Python 3.6 version. Python 3.6 is unstable to hold the KERAS back end Apart from the inability to build over Python 3.6 KERAS requires two additional library packages THEANO and TENSORFLOW. Tensor flow can be implemented over Python3.5 only.

A cloning environment is built using the Anaconda Prompt the Python 3.5 is cloned using the command.

**Condi create -n py35 python=3.5 anaconda**

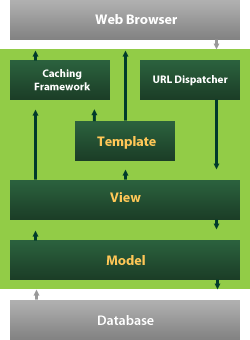
After activating the cloned environment, the Spyder is installed using the command So the Python3.5 has its own Spyder version which does not interfere with the base version of Anaconda.

# DJANGO

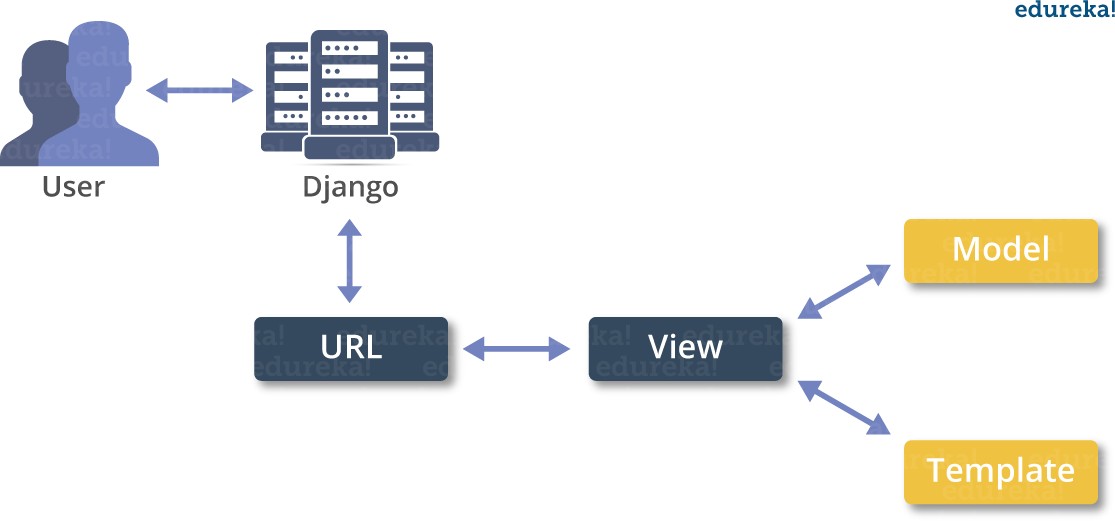
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes

care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and](https://en.wikipedia.org/wiki/Create%2C_read%2C_update_and_delete) [delete](https://en.wikipedia.org/wiki/Create%2C_read%2C_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models.



# ALGORITHM

In this paper author is describing concept to detect threats using AI-SIEM (Artificial Intelligence-Security Information and Event Management) technique which is a combination of deep learning algorithms such as FCNN, CNN (Convolution Neural Networks) and LSTM (long short-term memory) and this technique works based on events profiling such as attack signatures. Author evaluating propose work performance with conventional algorithms such as SVM, Decision Tree, Random Forest, KNN and Naïve Bayes. Here I am implementing CNN and LSTM algorithms.

* Data Parsing
* TF-IDF
* Event Profiling Stage
* Deep Learning Neural Network Model

**CHAPTER 4 METHODOLOGY**

# Existing System

As there is no staff accessible in automated cafés, it is hard for the eatery the board to appraise how the idea and the food is capable by the clients. Existing Rating frameworks, like Google and Trip Advisor, just in part tackle this issue, as they just cover a piece of the client's conclusions. These rating frameworks are just utilized by a subset of the clients who rate the café on free evaluating stages on their own drive. This applies basically to clients who experience their visit as certain or negative.

# PROPOSED SYSTEM

In order to solve the above problem, all customers must be motivated to give a rating. This paper introduces an approach for a restaurant rating system that asks every customer for a rating after their visit to increase the number of ratings as much as possible. This system can be used unmanned restaurants; the scoring system is based on facial expression detection using pretrained convolutional neural network (CNN) models. It allows the customer to rate the food by taking or capturing a picture of his face that reflects the corresponding feelings. Compared to text-based rating system, there is much less information and no individual experience reports collected. However, this simple fast and playful rating system should give a wider range of opinions about the experiences of the customers with the restaurant concept.

# SYSTEM ARCHITECTURE

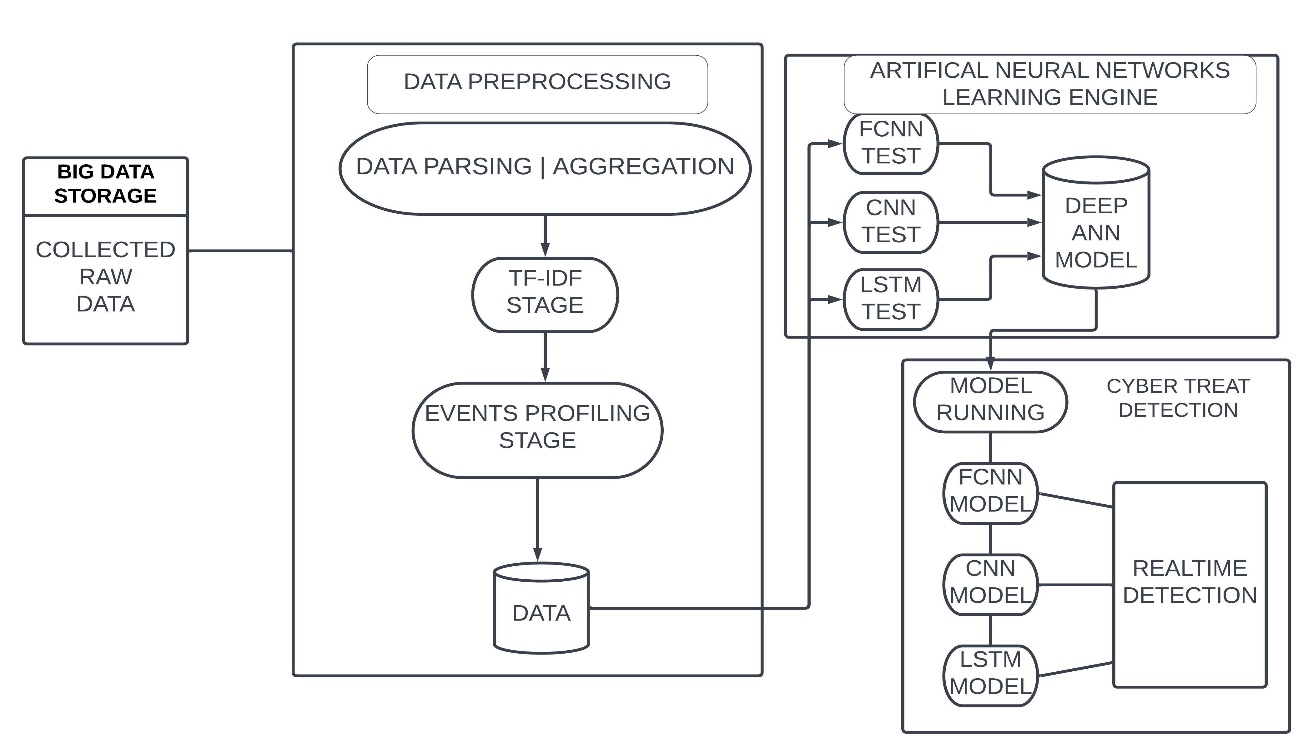


Fig 4.1 Architectural design

* 1. **USE CASE DIAGRAM**

In the use case diagram, we have basically one actor who is the user of the code editor.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

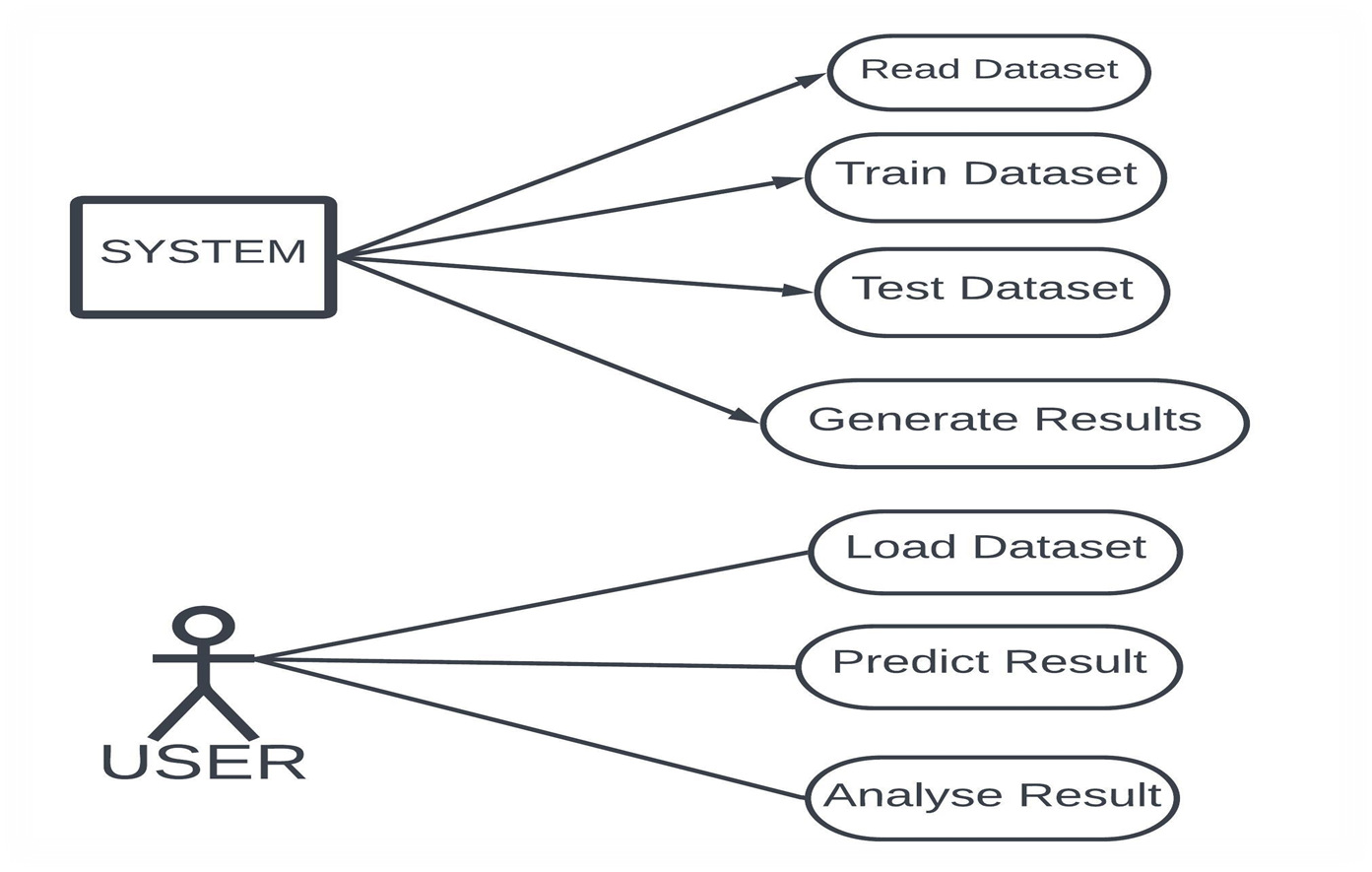
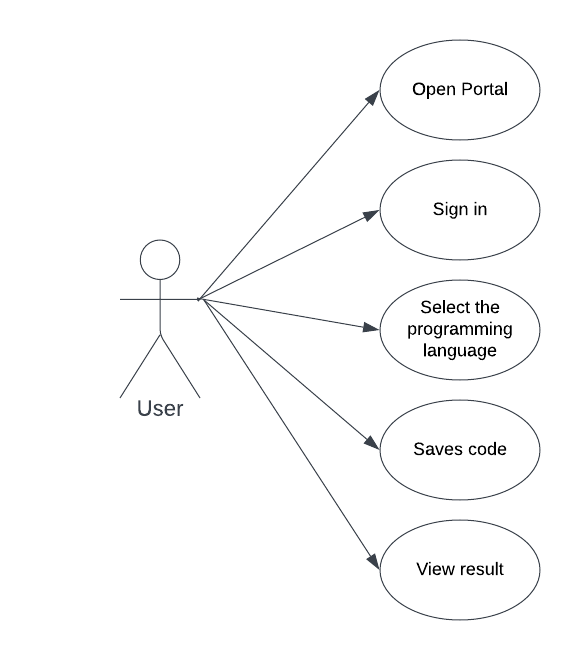
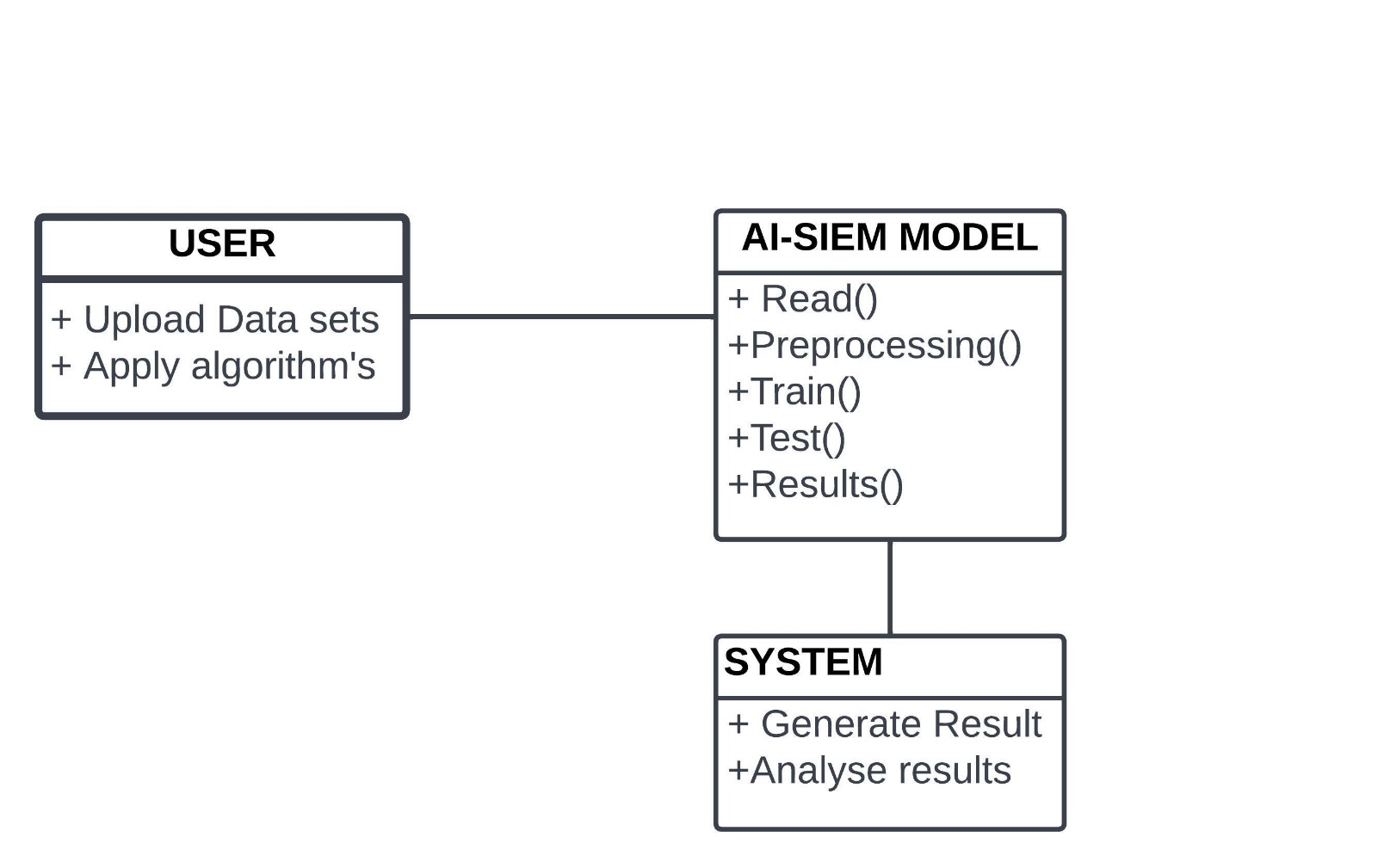
**

Figure 3.2: Case Diagram

* 1. **CLASS DIAGRAM**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations(or methods), and the relationships among objects.



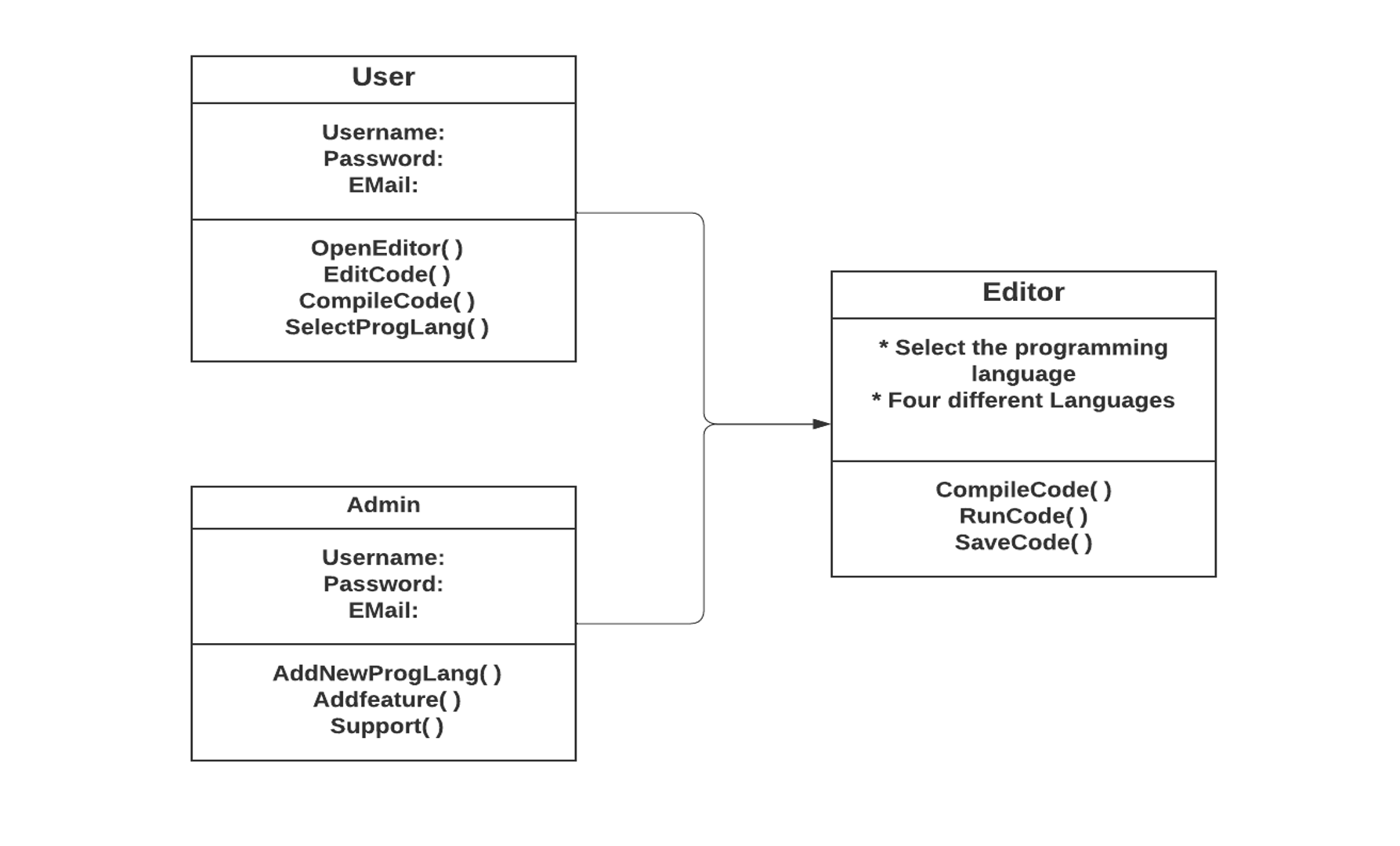
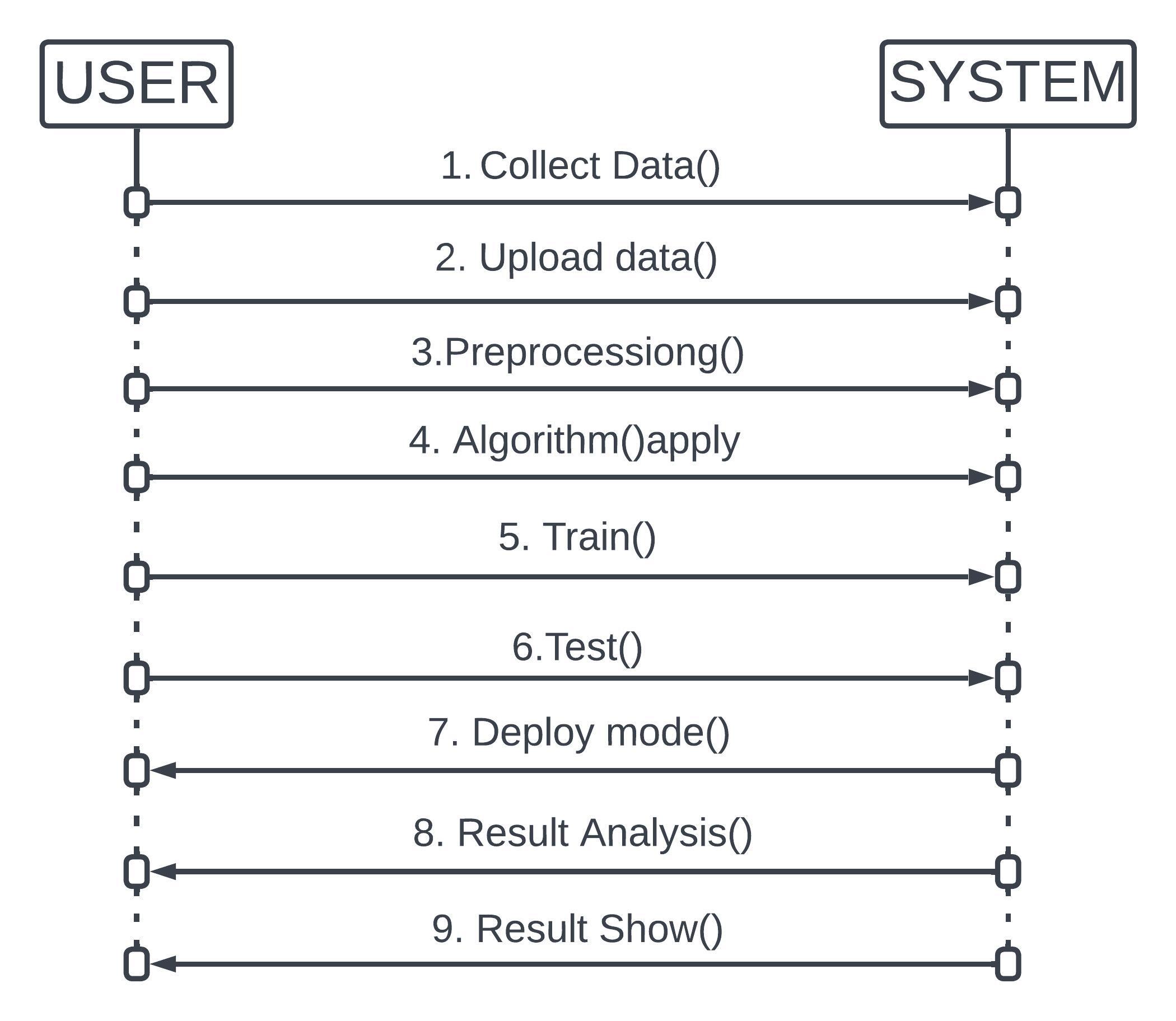


Figure 3.2: Class Diagram for NJAC-Online Code Editor

* 1. **SEQUENCE DIAGRAM**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects 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.





* 1. **SCOUCE CODE**

**from tinker import message box from tinker import \***

**from tinker import simple dialog import tinker**

**from tinker import file dialog import matplotlib. Pylon as Plt import NumPy as np**

**from interfile dialog import ask open file name import OS**

**import pandas as pd**

**from skarn import preprocessing from Sklearn. feature extraction. Text**

**import Count Vectorizer, Tf-idf Vectorizer from sklearn import Svm**

**from sklearn.metrics import accuracy score**

**from sklearn.model selection import train test slit from keras.models import Sequential**

**from keras.layers import Flatten**

**from keras.layers import Dense, Activation, Dropout from sklearn. pre-processing import One Hot Encoder import skleras. Layers**

**from keras.layers import Convolution2D from keras.layers import MaxPooling2D from keras.layers import Flatten**

**from keras.layers**

**import Dense, Activation, Batch Normalization, Dropout**

**from sklearn.metrics import precision score from sklearn.metrics import recall score from sklearn.metrics import f1 score**

**from sklearn.naive bayes import Bernoulli**

**from sklearn.neighbors import K Neighbors Classifier from sklearn.tree import Decision Tree Classifier**

**from sklearn.ensemble import Random Forest Classifier**

**main = tkinter.Tk()**

**main. Title ("Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles") #designing main screen**

**main. Geometry("1300x1200")**

**le = pre-processing. Label Encoder () global filename**

**global feature extraction**

**global X, Y global doc**

**global label names**

**global Train, X test, y train, ytest**

**global lstm acc,cnn acc,svm acc,knn acc,dt acc,random acc,nb acc globallstm\_precision,cnn\_precision,svm\_precision,knn\_precisio dt\_precision,random\_precision,nb\_precision**

**globallstm\_recall,cnn\_recall,svm\_recall,knn\_recall,dt\_acc,do\_real l,nb\_recall**

**global lstm\_fm,cnn\_fm,svm\_fm,knn\_fm,dt\_fm,random\_fm,nb\_fm**

**def upload (): global filename global X, Y global doc**

**global label names**

**filename=file dialog. Ask open filename(initial dirt="datasets") dataset = pd.read\_csv(filename)**

**label names = dataset.labels.unique() dataset['labels'] = le.fit\_transform(dataset['labels']) cols = dataset. Shape [1]**

**cols = cols - 1**

**X = dataset. Values [: 0: cols] Y = dataset. Values [: cols]**

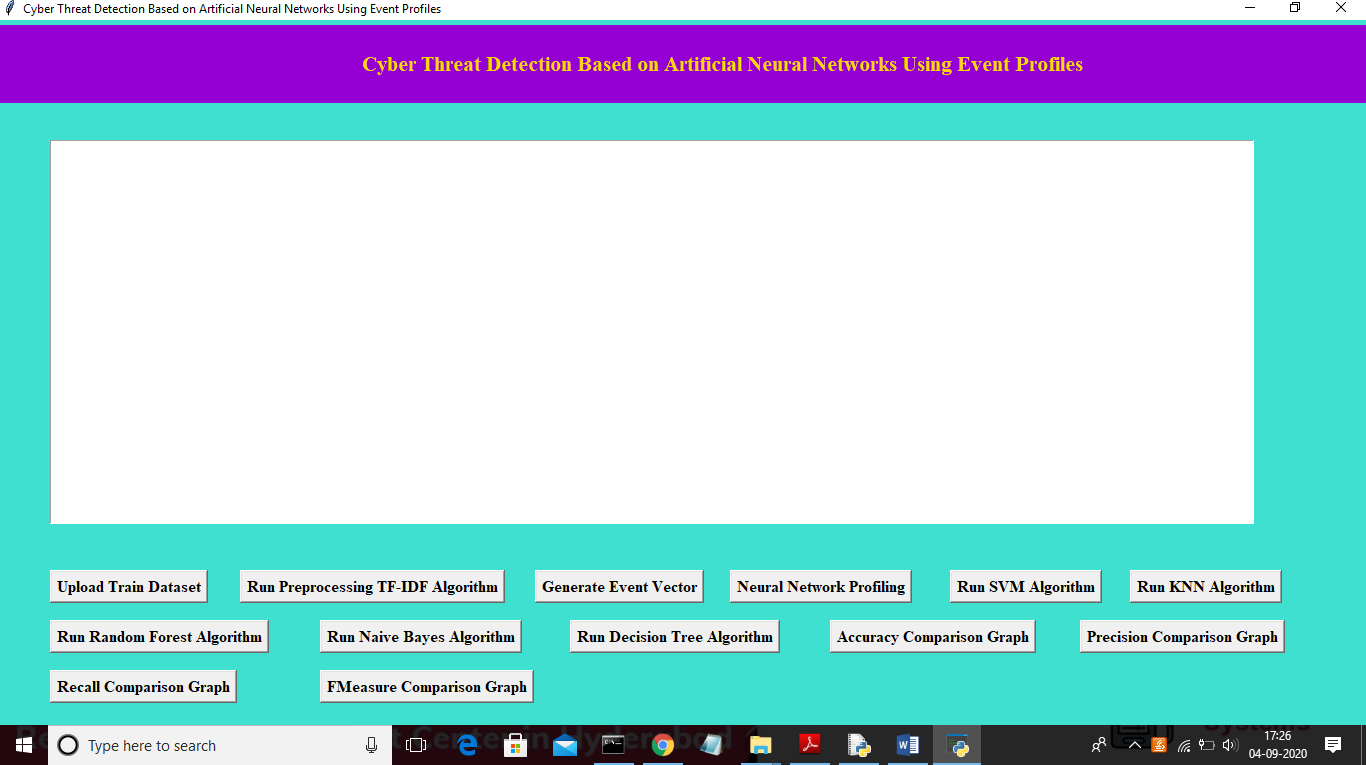
**Y = Y.astype('int') doc = []**

**for I in range (Len(X)):**

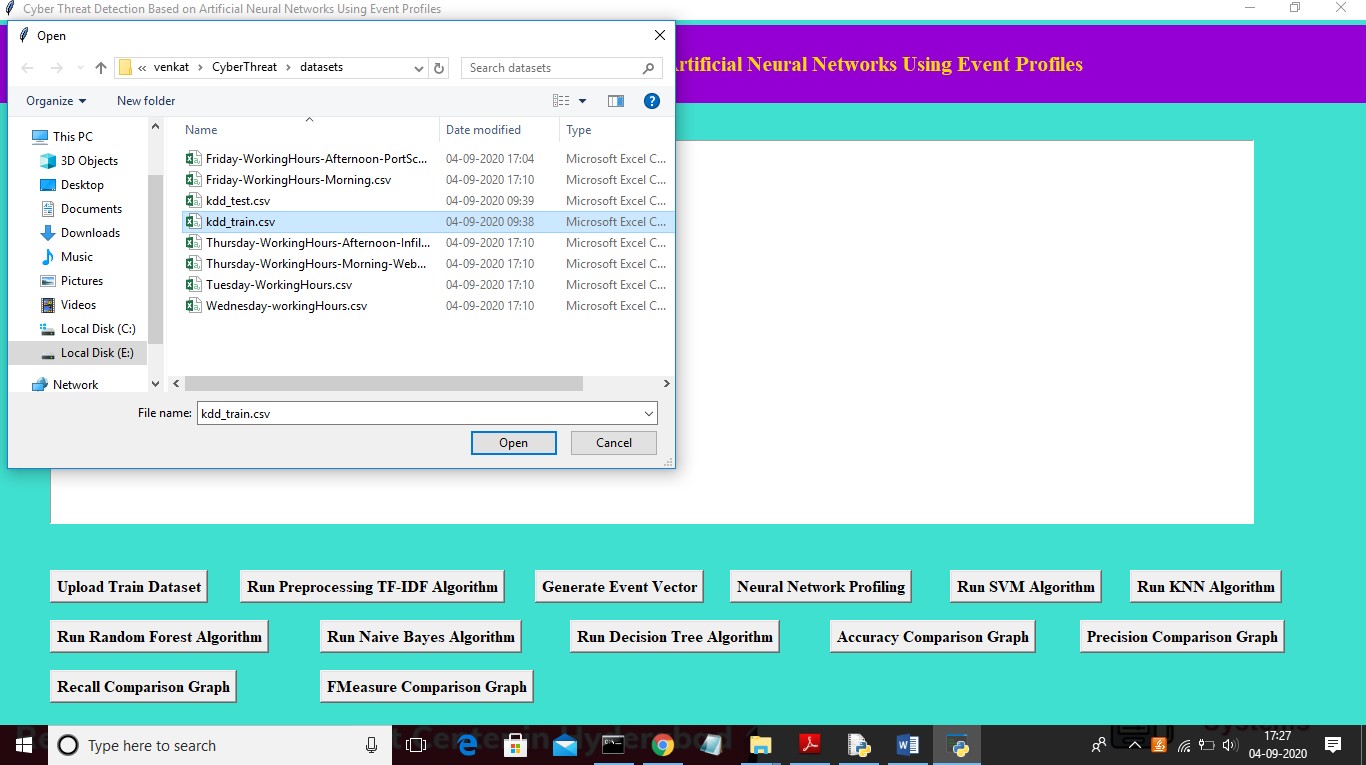
**strs = ''for j in range (Len(X[i])): strs+=str(X[i,j])+" " doc.append(strs.strip**

# \*

# UPLOAD DATASETS

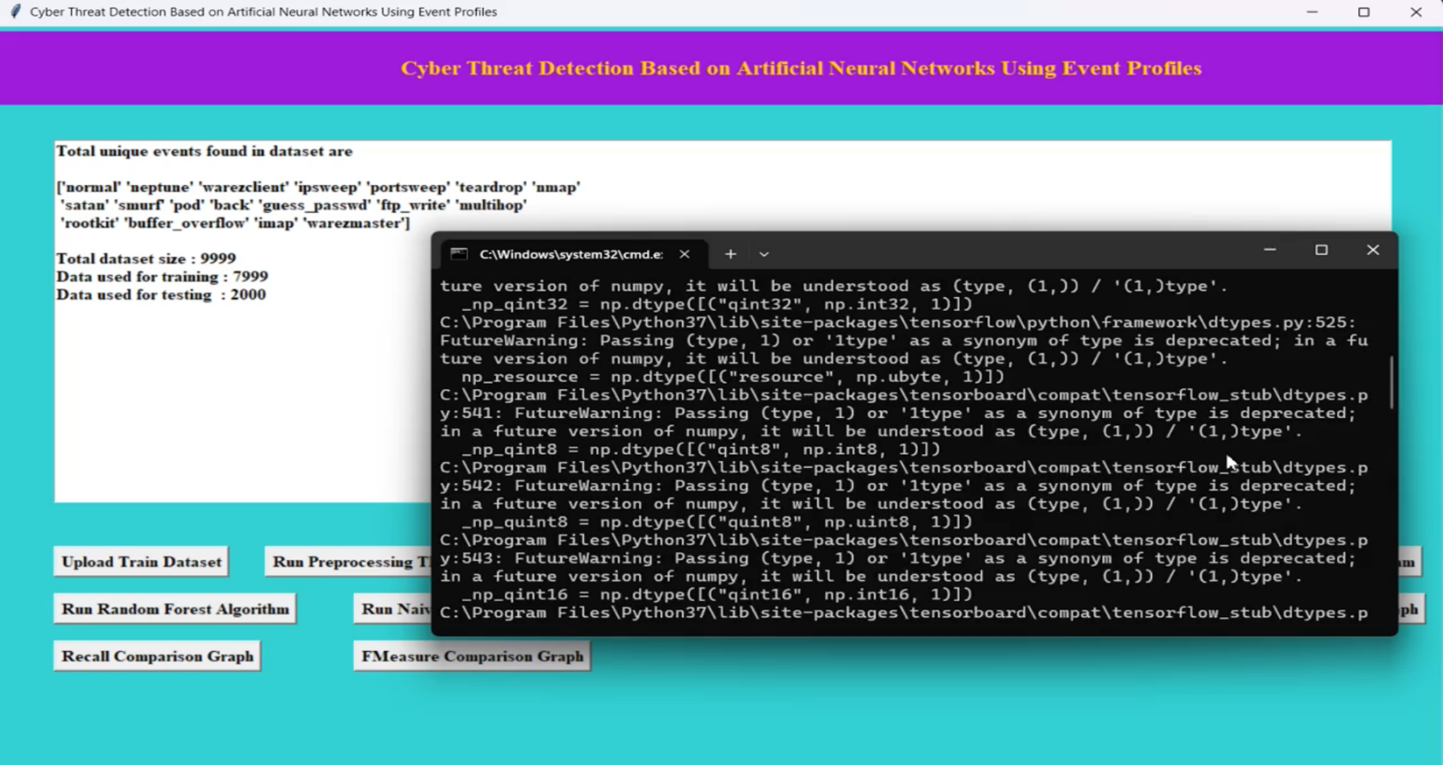
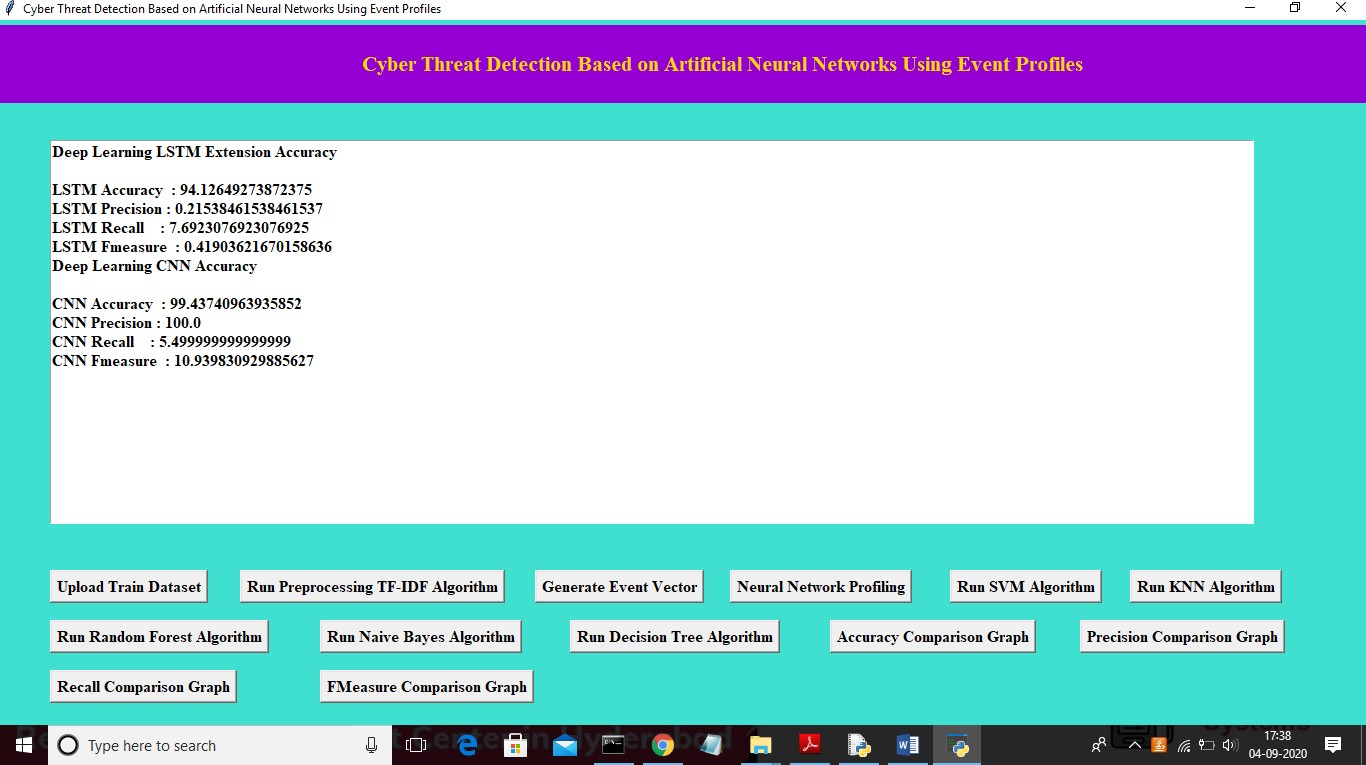


In above screen click on ‘Upload Train Dataset’ button and upload dataset



In above screen uploading ‘kdd\_train.csv’ dataset and after upload will get below screen.

# TF-IDF ALGORITHM



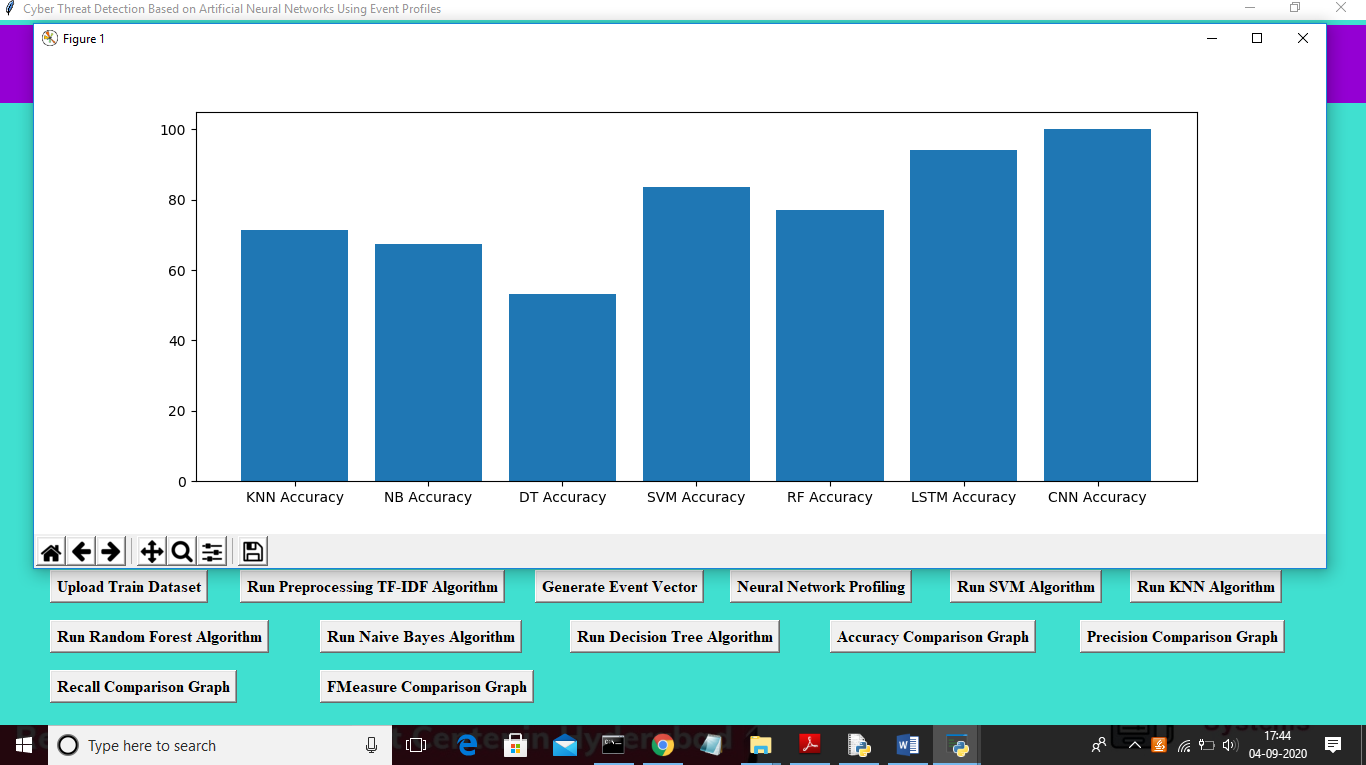
4.5.1 KNN ALGORITHM

In above screen we can see both algorithms accuracy, precision, recall and FMEA sure values. Now click on ‘Run SVM Algorithm’ button to run existing SVM algorithm.

In above screen CNN also starts first iteration with accuracy as 0.72 and after completing all iterations 10 we got filtered improved accuracy as 0.99 and multiply by 100 will give us 99% accuracy. So, CNN is giving better accuracy compare to LSTM and now see below GUI screen with all details.

In above screen we can see SVM algorithm output values and now click on ‘Run KNN Algorithm’ to run KNN algorithm.

Now click on ‘Accuracy Comparison Graph’ button to get accuracy of all algorithms



In above graph x-axis represents algorithm name and y-axis represents accuracy of those algorithms and from above graph we can conclude that LSTM and CNN perform well. Now click on Precision Comparison Graph’ to get below graph

# INPUT AND OUTPUT DESIGN

# INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

* + 1. **OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision- making.

* + - 1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
      2. Select methods for presenting information.
      3. Create document, report, or other formats that contain information produced by the system.

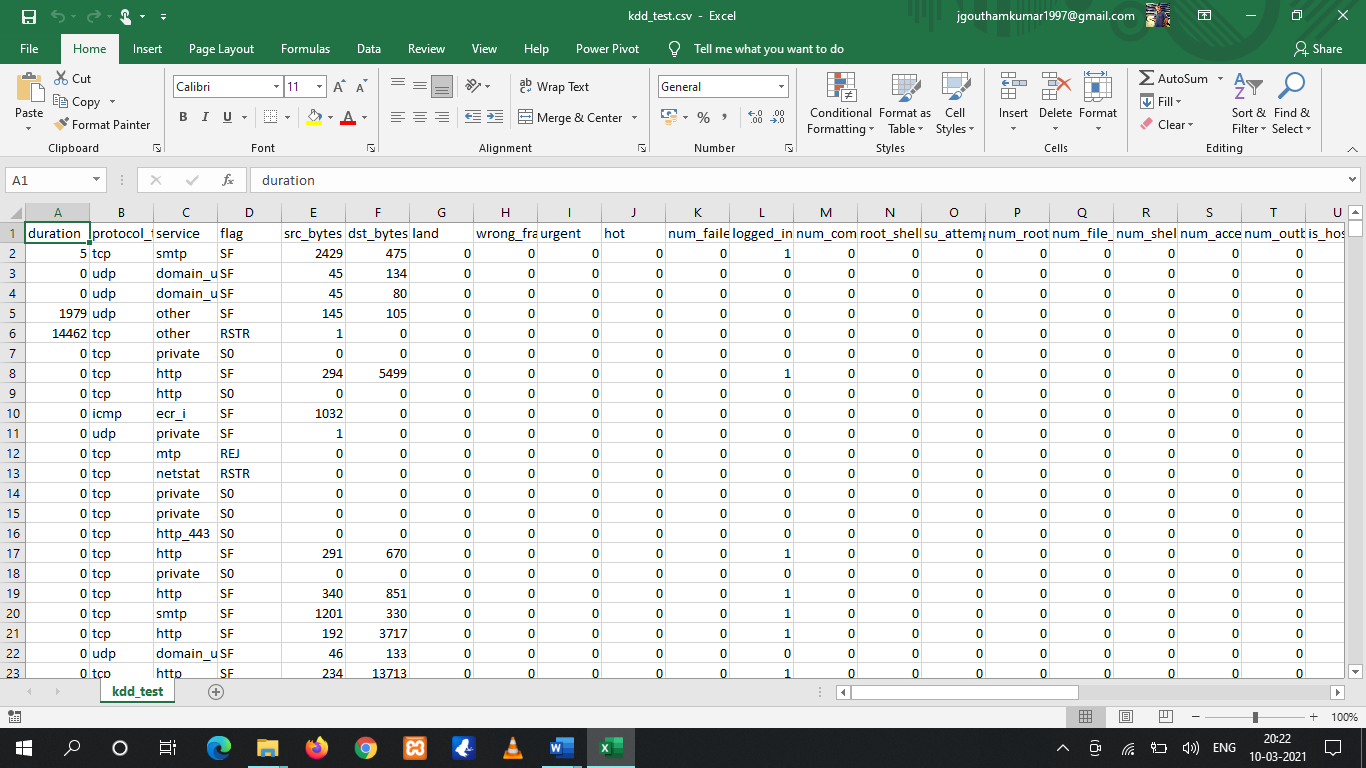
The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

# DATASETS



Our dataset has been collected from two large enterprise sys-terms, namedESX- 1andESX-2. The security raw events were collected over 5 months for ESX-1, over 30 days for ESX-2, respectively, in which the detecting threat information was separately recorded by the SOC security analysts whenever network intrusion occurred. The list of threat detection information contains threat occurrence time, related attacks, category of attack, respond contents, attack IP address, and victim network information**.**

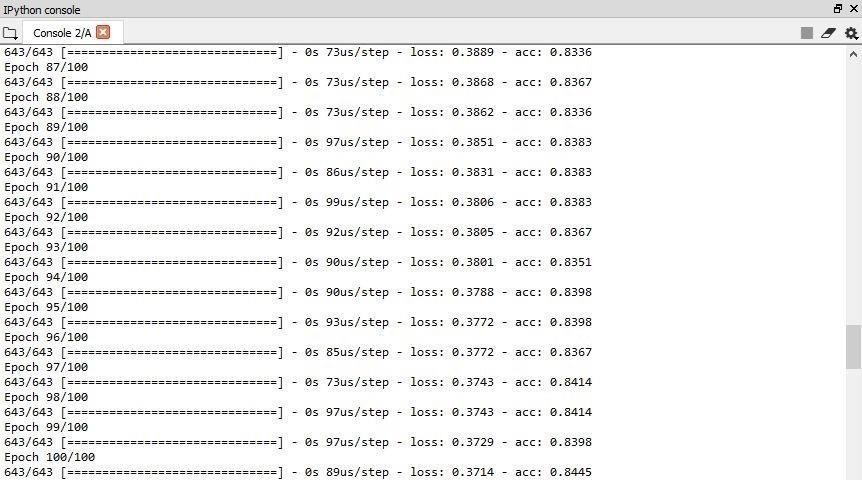
In our datasets, we investigated 798 detecting cyber threats in ESX-1, which are dispersed across the entire collection period. Looking at the type of occurred attacks in recorded cyber threats, there are 240 scanning, 547 system hack-king, and 11 worm attacks. Similarly, in ESX-2 there are941 scanning, 3,077 system hacking, and 51 worm attacks. This categorizing of attack type was manually performed by SOC analysts. By category, the system hacking attack includes a cross site script, DDoS, brute force attack, and injection attack. A trojan and backdoor attack belongs scanning attack. Overall, the number of attacks were found 4,079 cyber-threats.

# DATA VISUALIZATION

The t-SNE is not only commonly utilized for vector data visualization but also considered as embedding tools to visualize high-dimensional data. The t-SNE is able to visual-zed high-dimensional data into two-dimensional maps by learning two- dimensional embedding vectors that preserves neighbour structures among high- dimensional data. The N data rows in dataset are randomly selected, which are visualized by performing analysis in t-SNE represent the maps that are visualized by t-SNE for CICIDS 2017 and ESX-2, respectively. The t-SNE plots in the figure show that the normal and attack data points located nearby in the same space, which makes it very hard to classify them into either normal or attack. Although the t-SNE plots of normal and attack data are clustered, it clearly finds out that those are not linearly separated. In general, it is known that deep learning is then effective at dealing with high-dimensional data with non-linearity [50], which is one of the reasons we employ deep learning approaches to detect cyber threats.

# EXPERIMENTAL RESULTS

Based on the results of this experiment, we are able to arrive at two meaningful conclusions. First, our mech-amiss are capable of being employed as learning- based models for network intrusion detection. When the performance evaluations were conducted using two well-known benchmark datasets such as NSLKDD and CICIDS2017, the result proved as capable as the conventional machine-learning models. This means that our proposed methods, employed in the AI-SIEM system, have applicability for learning-based network intrusion detection. Second, when the conventional learning-based methods, which accomplish good result by bench mark dataset, are employed in the real world, the performance of overall accuracy is not as reliable as those of benchmark datasets. Never the less, the accuracy performance of our three EP-ANN models were not significantly degraded, despite the large amount of data and a lack of benchmark dataset features, such as seen in the result for ESX-2. By contrast, the accuracy of conventional methods had degraded from approximately 0.90 to 0.85.



**Epoch**

In neural networks generally, an epoch is a single pass through a full dataset. It is the iterations constituting one forward pass and one backward pass. A confusion matrix is a technique for summarizing the performance of a classification algorithm. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. Classification accuracy is the ratio of correct cyber threats. Computers do not generally store arbitrarily large numbers. Instead, each number stored by a computer is allotted a fixed amount of space. Therefore, when the number of time units that have elapsed since a system's epoch exceeds the largest number that can fit in the space allotted to the time representation, the time representation [overflows](https://en.wikipedia.org/wiki/Arithmetic_overflow), and problems can occur. While a system's behavior after overflow occurs is not necessarily predictable, in most systems the number representing the time will reset to zero, and the computer system will think that the current time is the epoch time again.

**CHAPTER 6**

**Conclusion and Future Work**

# SUMMARY

With Intrusion Detection Systems and trained network security auditors, organizations have a reliable means to prioritize and isolate the most critical threats in real time. They are tasked with identifying the significant elements of the attack and translating them into IDS signatures threat detection and response is the most important aspect of cybersecurity for IT organizations that depend on cloud infrastructure. Threat detection, therefore, describes the ability of IT organizations to quickly and accurately identify threats to the network or to applications or other assets within the network.

The proposed procedure changes large number of gathered security occasions over to singular occasion profiles and utilize a profound learning-based discovery strategy for upgraded digital danger identification. For this work, we built up an AI- SIEM framework dependent on a blend of occasion profiling for information preprocessing and distinctive counterfeit neural organization techniques, including FCNN, CNN, and LSTM. The framework centers around separating between obvious positive and bogus positive cautions, consequently causing security examiners to quickly react to digital dangers.

# CONCLUSION:

In this paper, we have proposed the AI-SIEM system using event profiles and artificial neural networks. The novelty of our work lies in condensing very large-scale data into event profiles and using the deep learning-based detection methods for enhanced cyber-threat detection ability. The AI-SIEM system enables the security analysts to deal with significant security alerts promptly and efficiently by comparing long term security data. By reducing false positive alerts, it can also help the security analysts to rapidly respond to cyber threats dispersed across a large number of security events.

For the evaluation of performance, we performed a performance comparison using two benchmark datasets (NSLKDD, CICIDS2017) and two datasets collected in the real world. First, based on the comparison experiment with other methods, using widely known benchmark datasets, we showed that our mechanisms can be applied as one of the learning-based models for network intrusion detection. Second, through the evaluation using two real datasets, we presented promising results that our technology also outperformed conventional machine learning methods in terms of accurate classifications.

# FUTURE WORK

In the future, to address the evolving problem of cyber-attacks, we will focus on enhancing earlier threat predictions through the multiple deep learning approach to discovering the long-term patterns in history data. In addition, to improve the precision of labeled dataset for supervised-learning and construct good learning datasets, many SOC analysts will make efforts directly to record labels of raw security events one by one over several months. For testing, we constructed the purpose-built test bed where for conducting performance evaluations. This test bed consists of the big data platform and the AI-SIEM system. Moreover, in the SOC, we also had collected real-world IPS data over several months.

**References**

1. S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, K. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," *IEEE Access*, vol. 6, pp. 48231-48246, 2018.
2. B. Zhang, G. Hu, Z. Zhou, Y. Zhang, P. Qian, L. Chang, "Network Intrusion Detection Based on Directed Acyclic Graph and Belief Rule Base", *ETRI Journal*, vol. 39, no. 4, pp. 592-604, Aug. 2017

**GITHUB LINK**