A

Major Project

On

Deep Learning Approach for Intelligent Intrusion Detection System

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

BY

U. Srikanth	(177R1A05N8)
S. Chandana	(177R1A05P9)
T. Nithin	(177R1A05P3)

Under the Guidance of

D. Mounica

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

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 UGC Act.1956,
Kandlakoya (V), Medchal Road, Hyderabad-501401.

2017-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"DEEP LEARNING APPROACH FOR INTELLIGENT INTRUSION DETECTION SYSTEM"** being submitted by **U.SRIKANTH(177R1A05N8), S.CHANDANA(177R1A05P9), T.NITHIN(177R1A05P3)** in the partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance supervision during the year 2020-2021.

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

INTERNAL GUIDE
D. Mounica
Assistant Professor

DIRECTOR
Dr. A. Raji Reddy

HOD
Dr. K. Srujan Raju

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

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

D. Mounica, Assistant 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 Project Review Committee (PRC) Coordinators: **Mr. B. Ramji and Mr. J. Narasimha rao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to the Head of the Department **Dr. K. Srujan Raju** for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

We are obliged to our Director **Dr. A. Raji Reddy** for being cooperative throughout the course of this project. We would like to express our sincere gratitude to our Chairman Sri.

Ch. Gopal Reddy for his encouragement throughout the course of this project

The guidance and support received from all the members of **CMR TECHNICAL CAMPUS** who contributed and who are contributing to this project, was vital for the success of the project. We are grateful for their constant support and help.

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

U.SRIKANTH (177R1A05N8)
S.CHANDANA (177R1A05P9)
T.NITHIN (177R1A05P3)

ABSTRACT

Machine learning techniques are being widely used to develop an intrusion detection system (IDS) for detecting and classifying cyber-attacks at the network-level and host-level in a timely and automatic manner. However, no existing study has shown the detailed analysis of the performance of various machine learning algorithms on various publicly available datasets. In this paper, deep neural network (DNN), a type of deep learning model is explored to develop flexible and effective IDS to detect and classify unforeseen and unpredictable cyber-attacks. The continuous change in network behavior and rapid evolution of attacks makes it necessary to evaluate various datasets which are generated over the years through static and dynamic approaches. This type of study facilitates to identify the best algorithm which can effectively work in detecting future cyber-attacks. A comprehensive evaluation of experiments of DNNs and other classical machine learning classifiers are shown on various publicly available benchmark malware datasets. Our DNN model learns the abstract and high dimensional feature representation of the IDS data by passing them into many hidden layers. Through a rigorous experimental testing it is confirmed that DNNs perform well in comparison to the classical machine learning classifiers. Which can be used in real time to effectively monitor the network traffic and host-level events to proactively alert possible cyber-attacks.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

An intrusion detection system (IDS) for detecting and classifying cyber-attacks at the network-level and host-level in a timely and automatic manner. A type of deep learning model is explored to develop flexible and effective IDS to detect and classify unforeseen and unpredictable cyber-attacks. Our DNN model learns the abstract and high dimensional feature representation of the IDS data by passing them into many hidden layers. DNNs perform well in comparison to the classical machine learning classifiers which can be used in real time to effectively monitor the network traffic and host-level events to proactively alert possible cyber-attacks.

1.2 PROJECT PURPOSE

Malicious cyber-attacks pose serious security issues that demand the need for a novel, flexible and more reliable intrusion detection system (IDS). An IDS is a proactive intrusion detection tool used to detect and classify intrusions, attacks, or violations of the security policies automatically. Based on intrusive behaviors, intrusion detection is classified into network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS). The network behaviors are collected using network equipment via mirroring by networking devices, such as switches, routers, and network taps and analyzed in order to identify attacks and possible threats concealed within in network traffic.

1.3 PROJECT FEATURES

The main feature of this project is the security of computer networks which plays a strategic role in modern computer systems. In order to enforce high protection levels against threats, a number of software tools are currently developed. Intrusion Detection Systems aim at detecting intruder who eluded the "first line" protection. In this project, a pattern recognition approach to network intrusion detection based on ensemble learning paradigms is proposed. The potentialities of such an approach for data fusion and some open issues are outlined.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 PROBLEM DEFINITION

Machine Learning (ML) and Deep Learning (DL) methods for network analysis of intrusion detection and provides a brief tutorial description of each ML/DL method. Project representing each method were indexed, read, and summarized based on their temporal or thermal correlations. Because data are so important in ML/DL methods, we describe some of the commonly used network datasets used in ML/DL, discuss the challenges of using ML/DL for cyber security and provide suggestions for research directions.

2.2 EXISTING SYSTEM

Many challenges arise since malicious attacks are continually changing and are occurring in very large volumes requiring a scalable solution. There are different malware datasets available publicly for further research by cyber security community. However, no existing study has shown the detailed analysis of the performance of various machine learning algorithms on various publicly available datasets. Due to the dynamic nature of malware with continuously changing attacking methods, the malware datasets available publicly are to be updated systematically and benchmarked.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Malicious cyber-attacks pose serious security issues.
- Data theft is the serious issue faced by cyber-attack.
- Test & Prediction accuracy is low.
- Takes more time.

To avoid all these limitations and make the working more accurately the system needs to be implemented efficiently.

2.3 PROPOSED SYSTEM

We proposed a hybrid intrusion detection alert system using a highly scalable framework on commodity hardware server which has the capability to analyze the network and host-level activities. The framework employed distributed deep learning model with DNNs for handling and analyzing very large scale data in real time. The DNN model was chosen by comprehensively evaluating their performance in comparison to classical machine learning classifiers on various benchmark IDS datasets. In addition, we collected host-based and network-based features in real-time and employed the proposed DNN model for detecting attacks and intrusions. In all the cases, we observed that DNNs exceeded in performance when compared to the classical machine learning classifiers. Our proposed architecture is able to perform better than previously implemented classical machine learning classifiers in both HIDS and NIDS. To the best of our knowledge this is the only framework which has the capability to collect network-level and host-level activities in a distributed manner using DNNs to detect attack more accurately.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

- This can effectively work in detecting cyber-attacks or malicious attacks.
- Detects and gives more accuracy.
- HIDS are easy to deploy and it doesn't affect current architecture.
- NIDS will not affect the performance of hosts.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Operational Feasibility
- Technical Feasibility

2.4.1 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.

2.4.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. 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.

2.4.3 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.

2.5 HARDWARE AND SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. For developing the application, the following are the Hardware Requirements:

- Processor - CORE I7
- Speed - 1.1 GHz
- RAM - more than 8 GB
- Hard Disk - 50 GB

2.5.2 SOFTWARE REQUIREMENTS

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating System : Windows 10
- Programming Language : Python

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

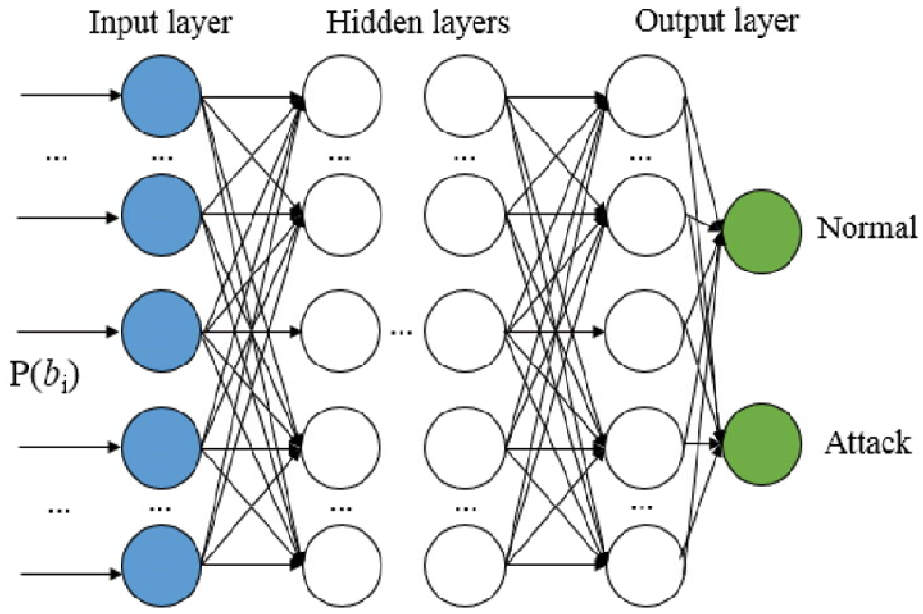


Figure 3.1: Project Architecture of DNN Model.

3.2 DESCRIPTION

Input Layer: The input layer is responsible for receiving the inputs. These inputs can be loaded from an external source such as a web service or a csv file. There must always be one input layer in a neural network. The input layer takes in the inputs, performs the calculations via its neurons and then the output is transmitted onto the subsequent layers.

Hidden Layer: A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. It is a typical part of nearly any neural network in which engineers simulate the types of activity that go on in the human brain.

Output Layer: The output layer is responsible for producing the final result. There must always be one output layer in a neural network. The output layer takes in the inputs which are passed in from the layers before it, performs the calculations via its neurons and then the output is computed.

3.3 MODULES

3.3.1 UPLOADING DATASET:

The KDD data set is a standard data set used for the research on intrusion detection systems. It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records. There is no duplicate records in the proposed test sets therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

3.3.2 SVM ALGORITHM:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. It is preferred over other classification algorithms because it uses less computation and gives notable accuracy. It is good because it gives reliable results even if there is less data.

3.3.3 RANDOM FOREST ALGORITHM:

Random Forest is a supervised machine learning algorithm made up of decision trees. Random Forest is used for both classification and regression for example, classifying whether an email is “spam” or “not”. The most convenient benefit of using random forest is its default ability to correct for decision trees’ habit of overfitting to their training set.

3.3.4 DNN (DEEP NEURAL NETWORKS) ALGORITHM:

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers. The main purpose of a neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like

classification. Neural networks are widely used in supervised learning and reinforcement learning problems.

3.4 USECASE DIAGRAM

A usecase diagram is a simple portrayal of a client's association with the framework and describing the details of a utilization case. These usecase diagrams can depict various sorts of clients of any framework and also different ways how they interface with framework. These kinds of diagrams are utilized connection with textual usecase and will frequently be joined by different sorts of diagrams too.

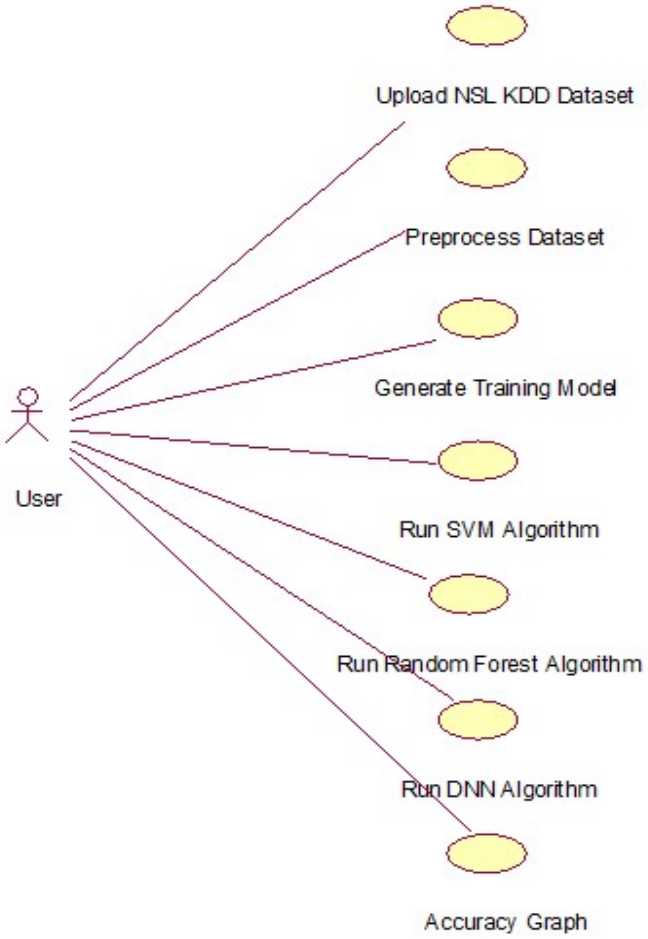


Figure 3.4: Usecase Diagram For IDS

3.5 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.

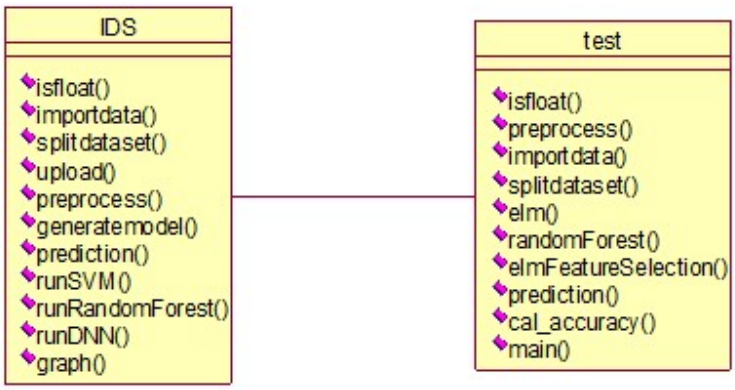


Figure 3.5: Class Diagram for IDS

3.6 SEQUENCE DIAGRAM

Sequence Diagram is a sort of interaction-diagram that displays how the processes work with each other and the order. Sequence diagram displays object collaborations in their specified allocated time. It characterizes the classes and objects inculpated with situation and arrangement of messages traded between the items expected to complete the particular functions of the situation. These diagrams are connected with use case acknowledge in the Logical View of structure worked on. This diagrams sometimes are called timing diagrams, event -diagrams, event scenarios.

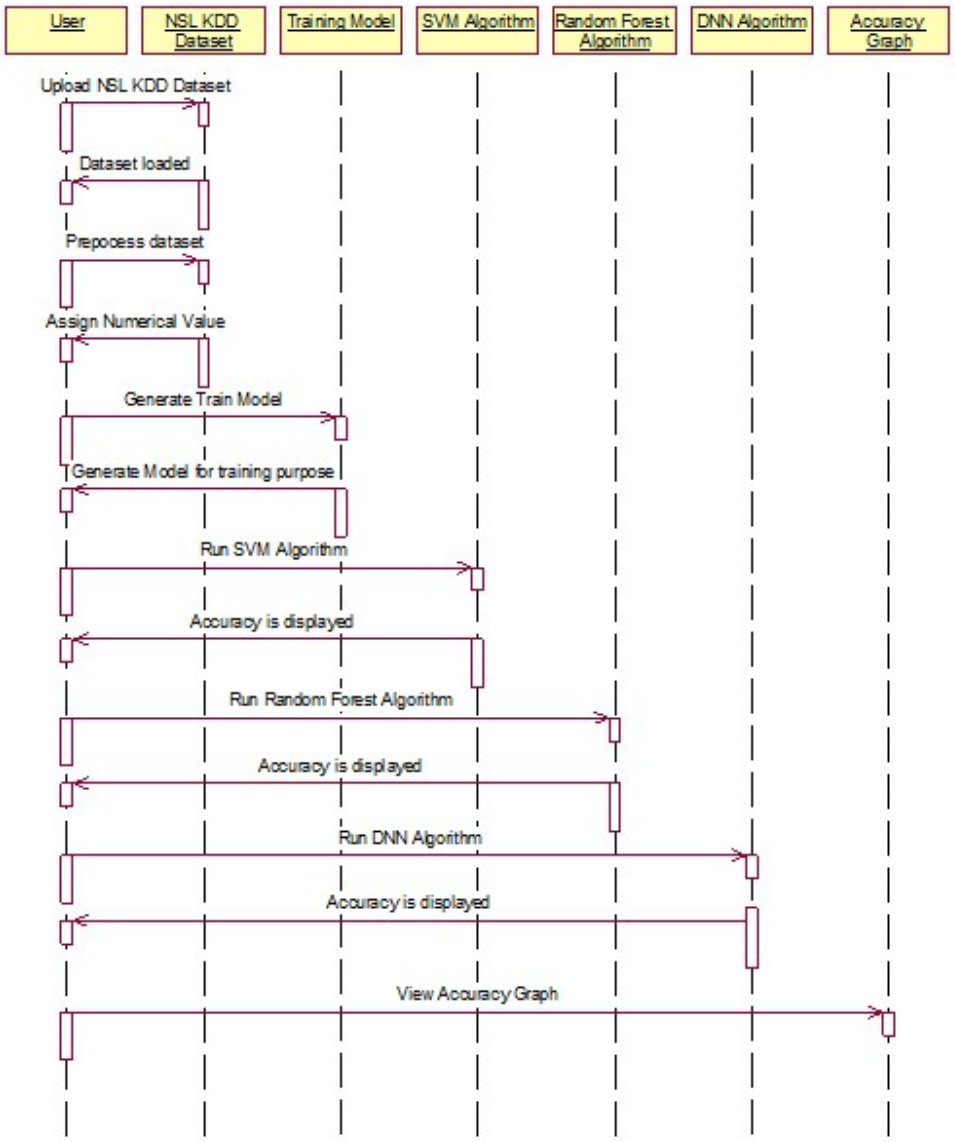


Figure 3.6: Sequence Diagram For IDS

3.7 ACTIVITY DIAGRAM

An activity diagram is a significant diagram in Unified Modeling Language (UML) to depict dynamic attributes of system. It is generally a flowchart diagram to signify the flow of activities. Operation of the framework is described as an activity. Therefore, control flow is taken from the operations. This pattern can be concurrent, sequential or branched.

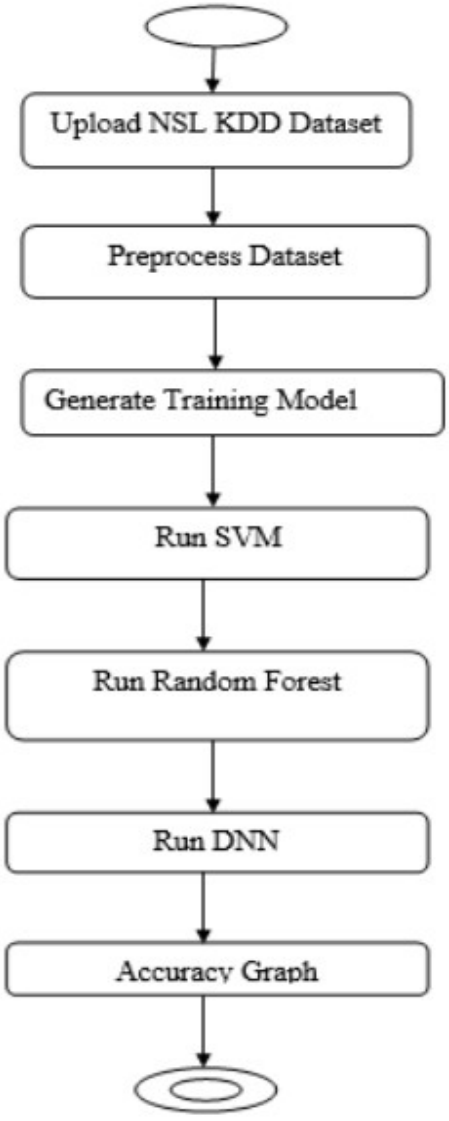


Figure 3.7: Activity Diagram For IDS

4. IMPLEMENTATION

4. IMPLEMENTATION

IDS.py:

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from imutils import paths

import matplotlib.pyplot as plt

import numpy as np

from tkinter.filedialog import askopenfilename

import numpy as np

import pandas as pd

from sklearn import *

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

from sklearn.ensemble import RandomForestClassifier

from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier

from sklearn_extensions.extreme_learning_machines.random_layer import RBFRandomLayer,
MLPRandomLayer

from sklearn.feature_selection import SelectFromModel
```

```
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

main = tkinter.Tk()
main.title("Deep Learning")
main.geometry("1300x1200")

global filename
global labels
global columns
global balance_data
global data
global X, Y, X_train, X_test, y_train, y_test
global svm_acc, random_acc, dnn_acc

def isfloat(value):
  try:
    float(value)
    return True
  except ValueError:
    return False

def importdata():
    global balance_data
```

```
    balance_data = pd.read_csv("clean.txt")
    return balance_data


def splitdataset(balance_data):
    X = balance_data.values[:, 0:37]
    Y = balance_data.values[:, 38]
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.2, random_state = 0)
    return X, Y, X_train, X_test, y_train, y_test


def upload():
    global filename
    text.delete('1.0', END)
    filename = askopenfilename(initialdir = "dataset")
    pathlabel.config(text=filename)
    text.insert(END,"Dataset loaded\n\n")


def preprocess():
    global labels
    global columns
    global filename


    text.delete('1.0', END)
    columns =
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","u
rgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
```

"num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate","label"]

```
    labels =
{"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"satan":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptunnel":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":35,"xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39}

    balance_data = pd.read_csv(filename)

    dataset = ''

    index = 0

    cols = ''

    for index, row in balance_data.iterrows():

      for i in range(0,42):

        if(isfloat(row[i])):

          dataset+=str(row[i])+','

          if index == 0:

            cols+=columns[i]+','

      dataset+=str(labels.get(row[41]))

      if index == 0:

        cols+='Label'

      dataset+='\n'

      index = 1;
```

```
    f = open("clean.txt", "w")
    f.write(cols+"\n"+dataset)
    f.close()

    text.insert(END,"Removed non numeric characters from dataset and saved inside clean.txt file\n\n")
    text.insert(END,"Dataset Information\n\n")
    text.insert(END,dataset+"\n\n")

def generateModel():
    global data
    global X, Y, X_train, X_test, y_train, y_test

    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    text.delete('1.0', END)
    text.insert(END,"Training model generated\n\n")

def prediction(X_test, cls):
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
```

```
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
    cm = confusion_matrix(y_test, y_pred)
    accuracy = accuracy_score(y_test,y_pred)*100
    text.insert(END,details+"\n\n")
    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
    text.insert(END,"Report : "+str(classification_report(y_test, y_pred))+"\n")
    text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n\n")
    return accuracy

def runSVM():
    global svm_acc
    global X, Y, X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)
    cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test, cls)
    svm_acc = cal_accuracy(y_test, prediction_data,'SVM Accuracy, Classification Report &
Confusion Matrix')

def runRandomForest():
    global random_acc
    global X, Y, X_train, X_test, y_train, y_test
```

```
    text.delete('1.0', END)

    cls = RandomForestClassifier(n_estimators=1,max_depth=0.9,random_state=None)

    cls.fit(X_train, y_train)

    text.insert(END,"Prediction Results\n\n")

    prediction_data = prediction(X_test, cls)

    random_acc = cal_accuracy(y_test, prediction_data,'Random Forest Algorithm Accuracy, Classification Report & Confusion Matrix')


def runDNN():

    global dnn_acc

    global X, Y, X_train, X_test, y_train, y_test

    text.delete('1.0', END)

    srhl_tanh = MLPRandomLayer(n_hidden=8, activation_func='tanh')

    cls = GenELMClassifier(hidden_layer=srhl_tanh)

    cls.fit(X_train, y_train)

    text.insert(END,"Prediction Results\n\n")

    prediction_data = prediction(X_test, cls)

    dnn_acc = cal_accuracy(y_test, prediction_data,'DNN Algorithm Accuracy, Classification Report & Confusion Matrix')


def graph():

    height = [svm_acc,random_acc,dnn_acc]

    bars = ('SVM Accuracy', 'Random Forest Accuracy','DNN Accuracy')

    y_pos = np.arange(len(bars))

    plt.bar(y_pos, height)

    plt.xticks(y_pos, bars)
```

```
    plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Deep Learning Approach for Intelligent Intrusion Detection System')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload NSL KDD Dataset", command=upload)
upload.place(x=50,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=300,y=100)

preprocess = Button(main, text="Preprocess Dataset", command=preprocess)
preprocess.place(x=50,y=150)
preprocess.config(font=font1)

model = Button(main, text="Generate Training Model", command=generateModel)
model.place(x=330,y=150)
model.config(font=font1)
```

```
runsvm = Button(main, text="Run SVM Algorithm", command=runSVM)

runsvm.place(x=610,y=150)

runsvm.config(font=font1)


runrandomforest = Button(main, text="Run Random Forest Algorithm",
command=runRandomForest)

runrandomforest.place(x=870,y=150)

runrandomforest.config(font=font1)


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

runeml.place(x=50,y=200)

runeml.config(font=font1)


graph = Button(main, text="Accuracy Graph", command=graph)

graph.place(x=330,y=200)

graph.config(font=font1)


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

text=Text(main,height=30,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)
```

```
main.config(bg='brown')

main.mainloop()
```

Test.py:

```
import numpy as np

import pandas as pd

from sklearn import *

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

from sklearn.ensemble import RandomForestClassifier

from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier

from sklearn_extensions.extreme_learning_machines.random_layer import RBFRandomLayer,
MLPRandomLayer

from sklearn.feature_selection import SelectFromModel

from sklearn.linear_model import Lasso

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import chi2


global labels

global columns

global balance_data
```

```
def isfloat(value):
  try:
    float(value)
    return True
  except ValueError:
    return False


def preprocess():
    global labels
    global columns
    columns =
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","u
rgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
"num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_
host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","s
rv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host
_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","
dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror
_rate","dst_host_srv_rerror_rate","label"]


    labels =
{"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"sata
n":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,
"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21
,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptunnel
":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":35,
"xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39}
    balance_data = pd.read_csv("dataset.txt")
    dataset = ''
```

```
    index = 0
    cols = ''
    for index, row in balance_data.iterrows():
      for i in range(0,42):
        if(isfloat(row[i])):
          dataset+=str(row[i])+','
          if index == 0:
            cols+=columns[i]+','
      dataset+=str(labels.get(row[41]))
      if index == 0:
        cols+='Label'
      dataset+='\n'
      index = 1;

    f = open("clean.txt", "w")
    f.write(cols+"\n"+dataset)
    f.close()

def importdata():
    global balance_data
    balance_data = pd.read_csv("clean.txt")
    # Printing the dataswet shape
    print ("Dataset Lenght: ", len(balance_data))
    print ("Dataset Shape: ", balance_data.shape)
```

```
    # Printing the dataset obseravtions
    print ("Dataset: ",balance_data.head())
    return balance_data

def splitdataset(balance_data):
    X = balance_data.values[:, 0:37]
    Y = balance_data.values[:, 38]
    # Spliting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.2, random_state = 0)
    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):

        # Creating the classifier object
        clf_gini = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)

        # Performing training
        clf_gini.fit(X_train, y_train)
        return clf_gini

def elm(X_train, X_test, y_train):
    srhl_tanh = MLPRandomLayer(n_hidden=8, activation_func='tanh')
```

```
    cls = GenELMClassifier(hidden_layer=srhl_tanh)
    cls.fit(X_train, y_train)
    return cls

def randomForest(X_train, X_test, y_train):
    cls = RandomForestClassifier(n_estimators=1,max_depth=0.9,random_state=None)
    cls.fit(X_train, y_train)
    return cls

def elmFeatureSelection(X_train, X_test, y_train):
    srhl_tanh = MLPRandomLayer(n_hidden=15, activation_func='tanh')
    cls = GenELMClassifier(hidden_layer=srhl_tanh)
    print('Original features:', X_train.shape[1])
    total = X_train.shape[1];
    X_train = SelectKBest(chi2, k=1).fit_transform(X_train, y_train)
    print('features set reduce after applying features concept:', (total - X_train.shape[1]))
    cls.fit(X_train, y_train)
    return cls

# Function to make predictions
def prediction(X_test, clf_object):
    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
```

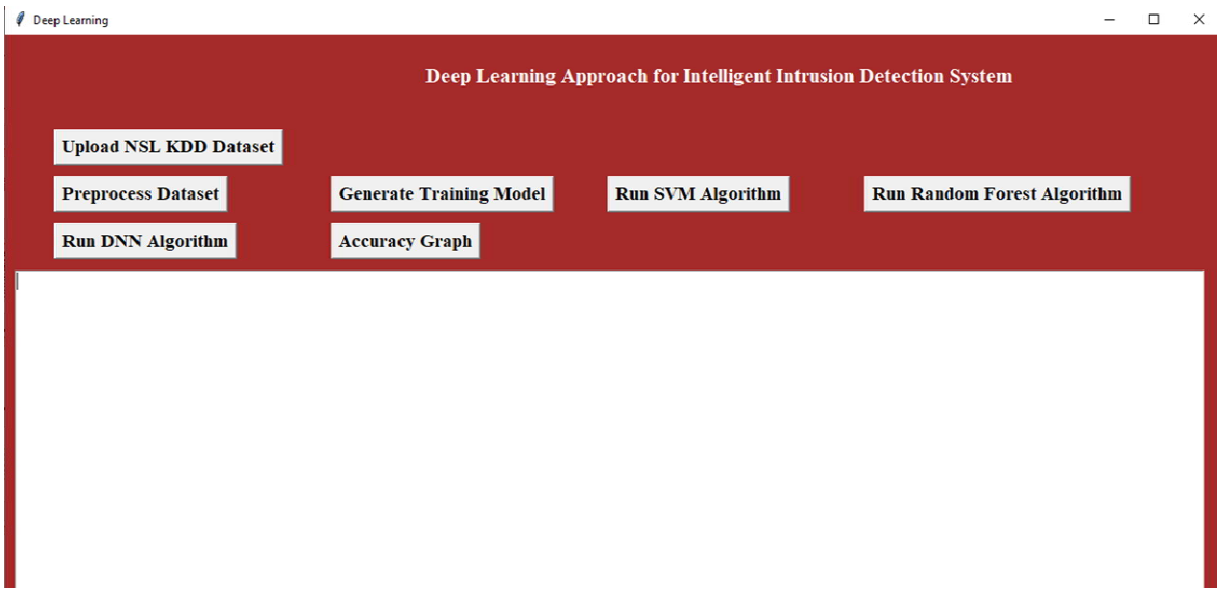
```
    #for i in range(len(X_test)):
      #print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print ("Accuracy : ", accuracy_score(y_test,y_pred)*100)
    print("Report : ", classification_report(y_test, y_pred))


def main():
    #preprocess()
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)

    #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)
    clf_gini = elm(X_train, X_test, y_train)

    #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)
```

```
    clf_gini = randomForest(X_train, X_test, y_train)
    #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    clf_gini = elmFeatureSelection(X_train, X_test, y_train)
    #print("Results Using Gini Index:")
    X_test = SelectKBest(chi2, k=1).fit_transform(X_test,y_test)

    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

# Calling main function
if __name__=="__main__":
        main()
```

5. SCREENSHOTS

5. SCREENSHOTS

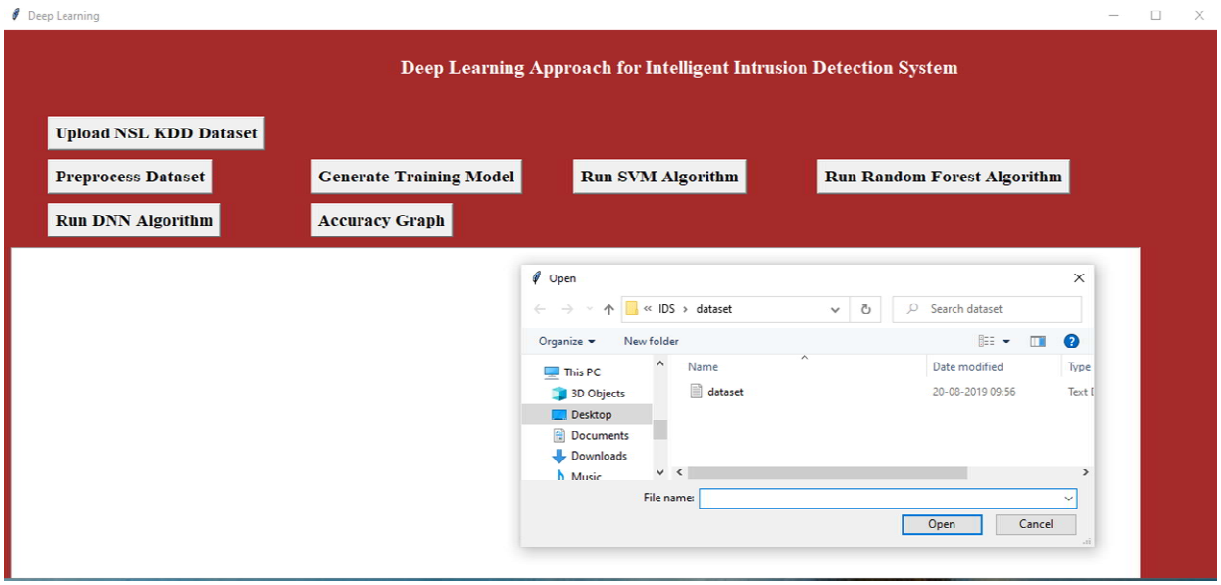
5.1 UPLOAD NSL KDD DATASET



Screenshot 5.1: Upload NSL KDD Dataset

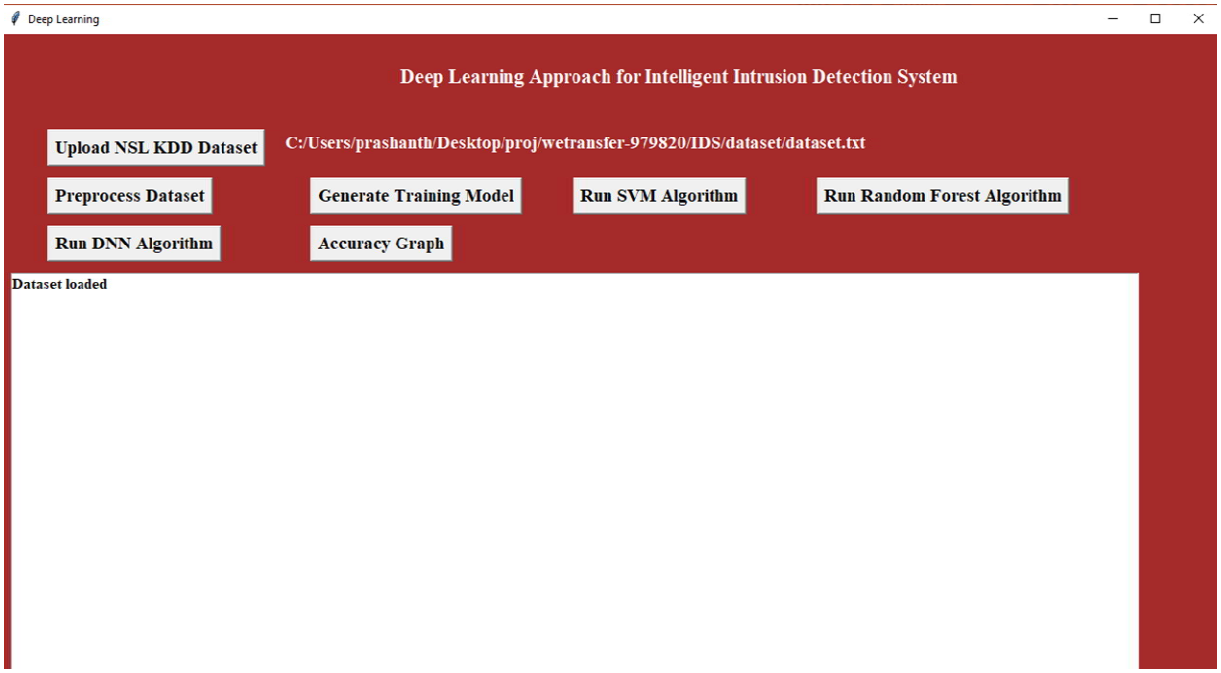
In above screen click on 'Upload NSL KDD Dataset' button to upload dataset.

5.2 SELECTING DATASET



Screenshot 5.2: Selecting Dataset

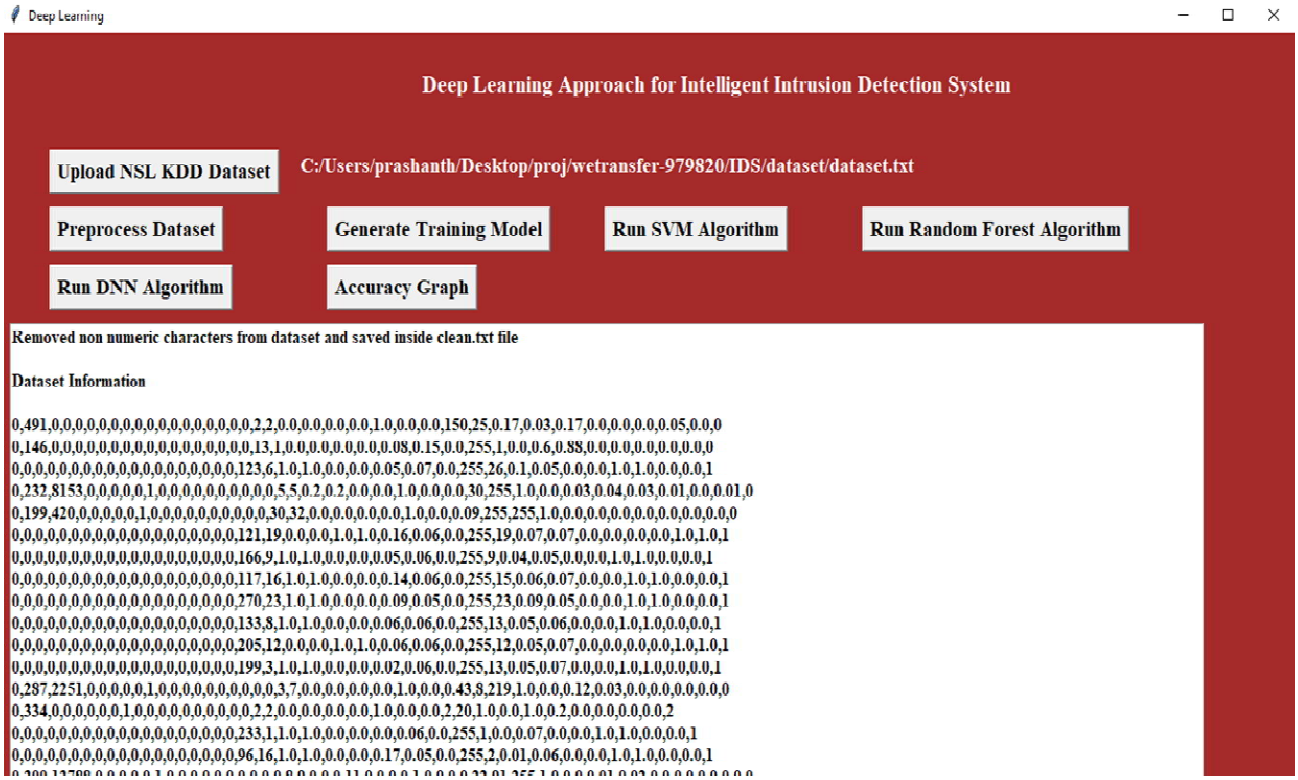
5.3 PREPROCESS DATASET



Screenshot 5.3: Preprocess Dataset

Now click on 'Preprocess Dataset' button to assign numeric values to each attack names as algorithms will not understand string names.

5.4 ASSIGNING NUMERIC ID'S



Screenshot 5.4: Assigning Numeric Id's

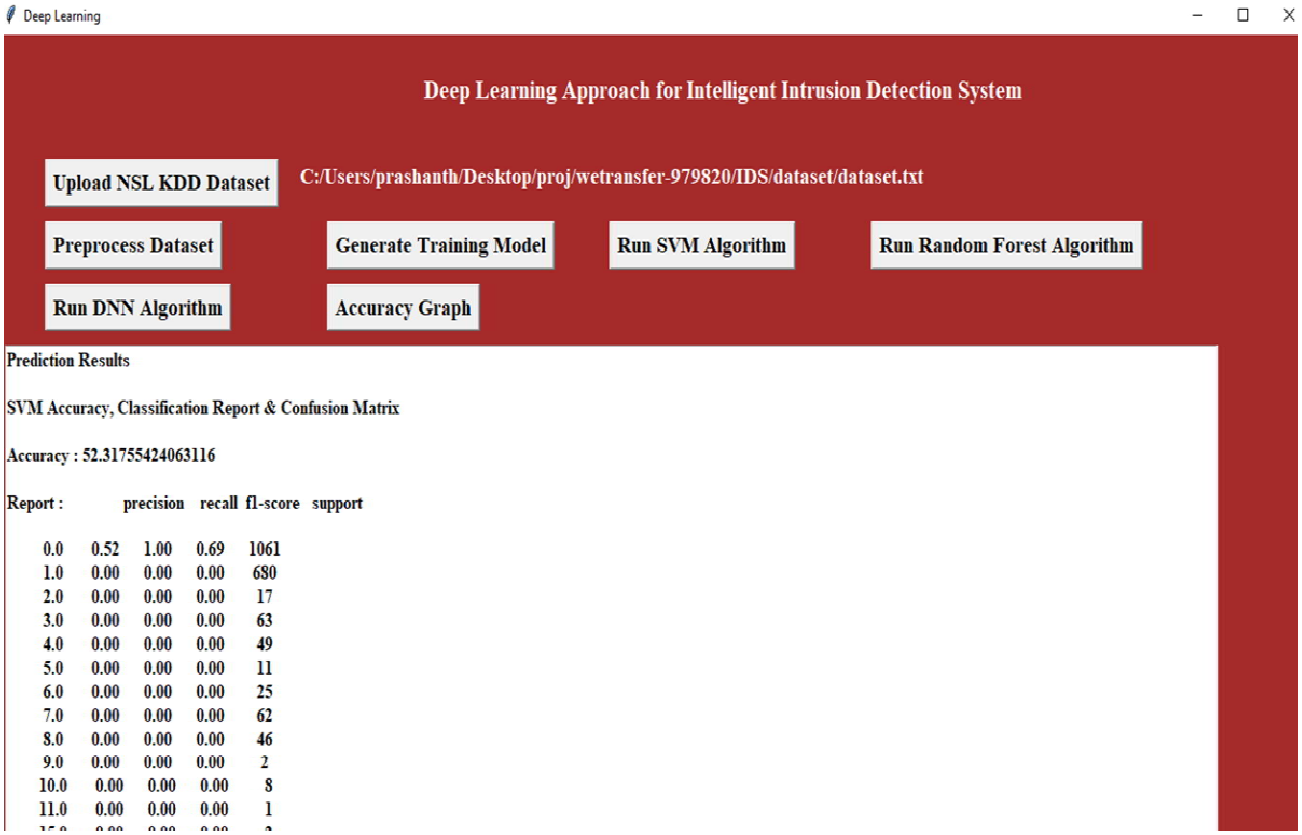
In above screen we can see we assign numeric id to each attack. Now click on 'Generate Training Model' button to generate model for training purpose.

5.5 GENERATING TRAINING MODEL

Screenshot 5.5: Generating Training Model

In above screen we can see dataset arrange in such a format so algorithms can build training and test set for prediction and accuracy result. Now click on 'Run SVM Algorithm' to get its prediction accuracy.

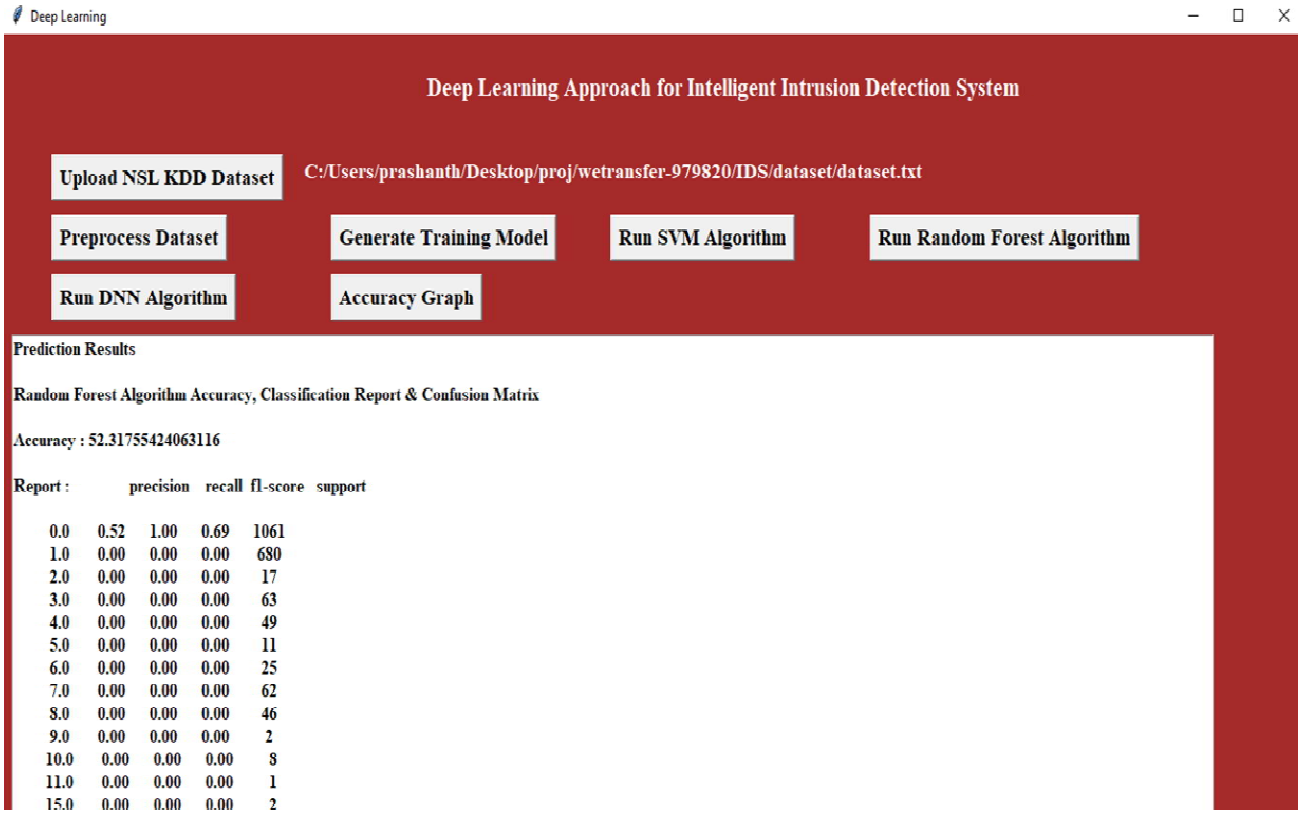
5.6 SVM ALGORITHM ACCURACY



Screenshot 5.6: SVM Algorithm Accuracy

In above screen we can see SVM (Support Vector Machine) prediction accuracy is 52%. Now click on 'Run Random Forest Algorithm' button to get its accuracy.

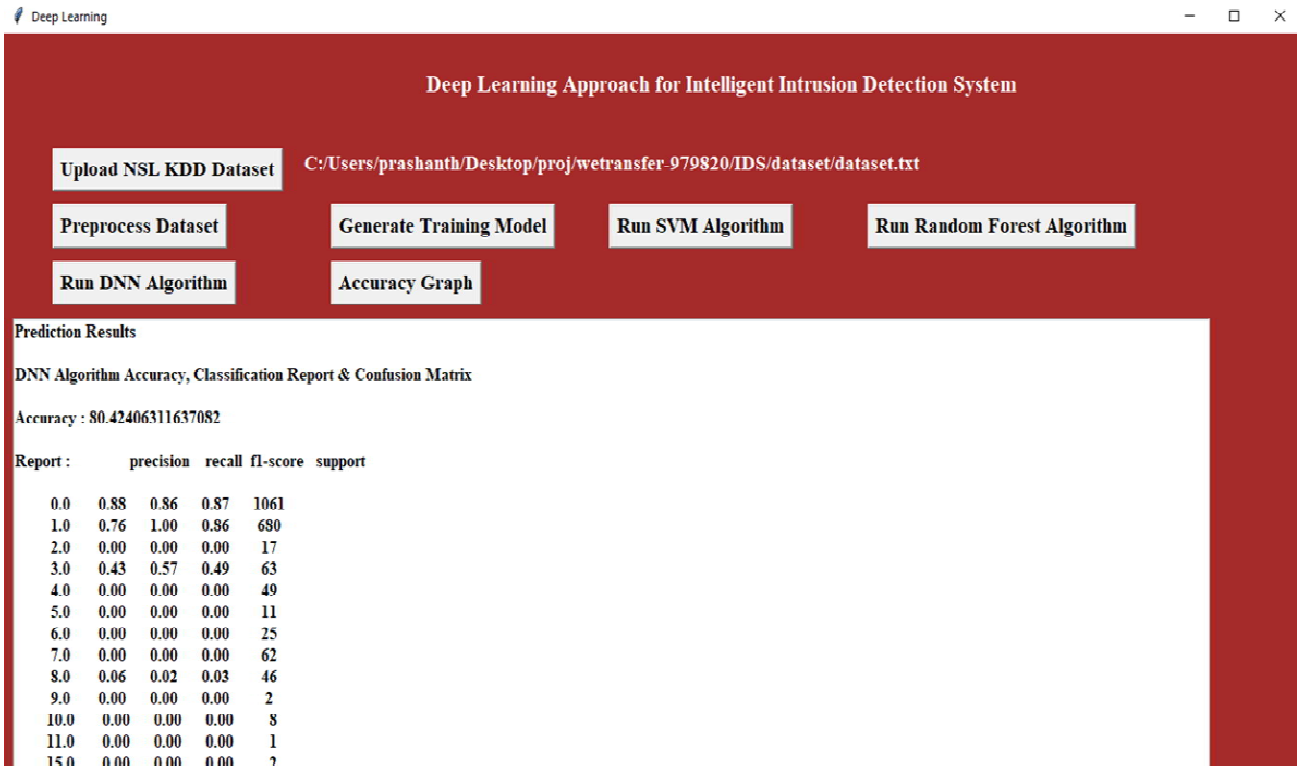
5.7 RANDOM FOREST ALGORITHM ACCURACY



Screenshot 5.7: Random Forest Algorithm Accuracy

In above screen we can see random forest also got same accuracy. Now run DNN(Deep Neural Network) Algorithm.

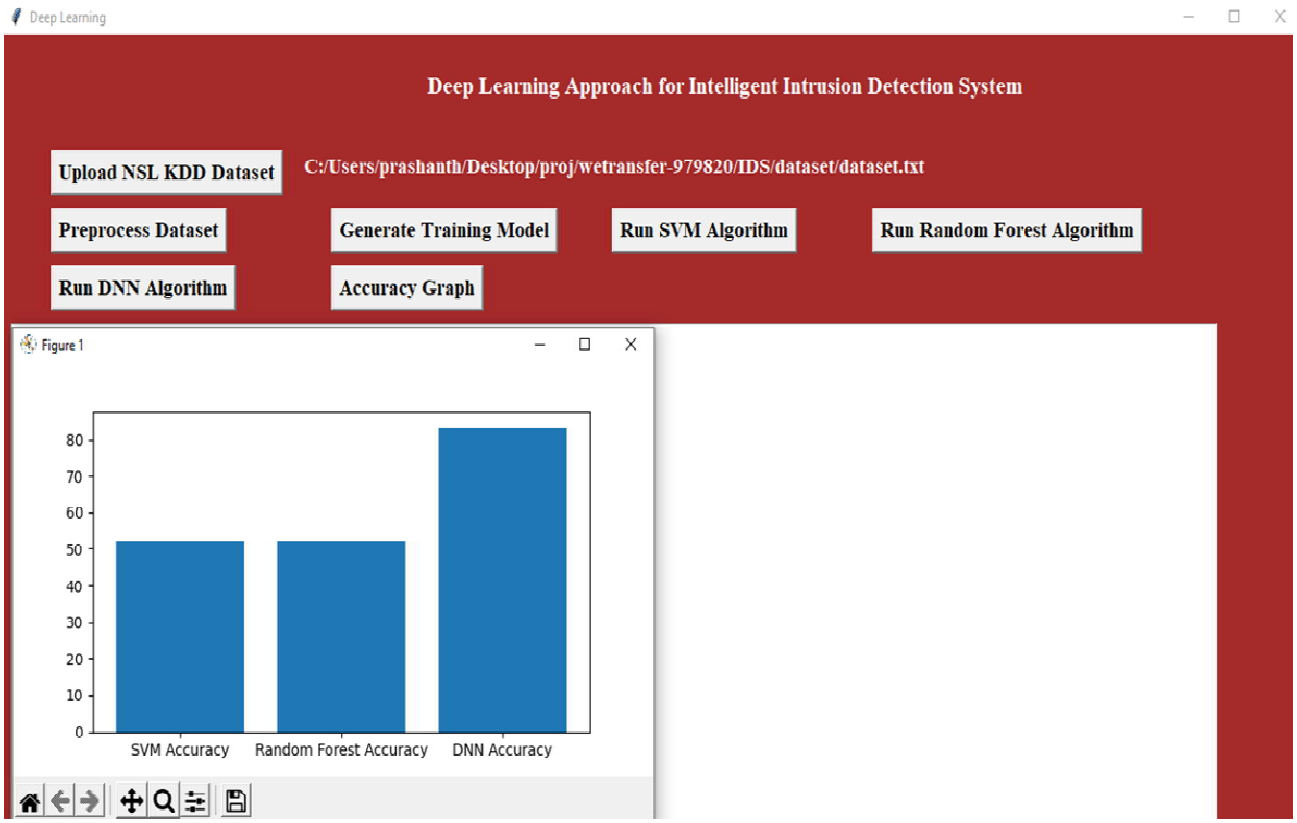
5.8 DNN ALGORITHM ACCURACY



Screenshot 5.8: DNN Algorithm Accuracy

In above screen we can see DNN accuracy is better than other two algorithms. DNN algorithm accuracy may be vary different times as it hidden layer will be chosen randomly from dataset. Now click on 'Accuracy Graph' button to get below graph.

5.9 ACCURACY COMPARISON GRAPH



Screenshot 5.9: Accuracy Comparison Graph

In above graph x-axis represents algorithm name and y-axis represents accuracy and DNN is the propose technique.

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 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. 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.

6.2.2 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. 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.

6.2.3 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.

6.2.4 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.

6.2.5 FUNCTIONAL TESTING:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Upload NSL KDD Dataset	Test whether the Train Dataset is uploaded or not into the system	If Train Dataset may not uploaded	We cannot do the further operations	Train Dataset is uploaded	High	High
02	Preprocess Dataset	Verify the either preprocessing is performed or not	Without loading the Train Dataset	We cannot perform preprocess	We can perform preprocess	High	High
03	Generate Training Model	Verify the either Training Model is Generated or not	Without preprocessing	We cannot generate the model for training purpose	We can generate the model for training purpose	High	High
04	Run SVM	Verify either the SVM algorithm is processed or not	Without having Training Model	We cannot Run the SVM	We can Run the SVM	High	High

05	Run Random Forest	Verify either the Random Forest algorithm is processed or not.	Without having Training Model	We cannot Run the Random Forest	We can Run the Random Forest	High	High
06	Run DNN	Verify either the DNN algorithm is processed or not	Without having Training Model	We cannot Run the DNN	We can Run the DNN	High	High
07	Accuracy Compariso n Graph	Verify either the Accuracy Graph is displayed or not	Without saving the accuracy values of the SVM, Random Forest and DNN algorithm	The Accuracy Comparison Graph is not displayed	The Accuracy Comparison Graph is displayed	High	High

7. CONCLUSION

7. CONCLUSION AND FUTURE SCOPE

7.1 PROJECT CONCLUSION:

This project work claims that a hybridized intrusion detecting alert system using a highly scalable framework on a server which has the capacity to scrutinize the host and network level actions. The framework employed distributed deep learning model with DNNs for handling and analyzing very large-scale data. The proposed system can perform way better than the previous NIDS and HIDS classifiers. This framework has the ability to gather both the host and network level activities in a dispersed way utilizing DNNs to identify attacks precisely. In general, performance could improve due to training of complex DNNs structures on cutting edge equipment through dispersed methodology. Because of broad computational expense related with complex DNNs designs, they weren't trained utilizing benchmark IDS datasets.

7.2 FUTURE SCOPE:

The execution time of the proposed system can be enhanced by adding more nodes to the existing cluster. In addition, the proposed system does not give detailed information on the structure and characteristics of the malware. Overall, the performance can be further improved by training complex DNNs architectures on advanced hardware through distributed approach. Due to extensive computational cost associated with complex DNNs architectures, they were not trained in this research using the benchmark IDS datasets. This will be an important task in an adversarial environment and is considered as one of the significant directions for future work.

8. BIBILOGRAPHY

8.1 REFERENCES

[1] Azab, A., Alazab, M. & Aiash, M. (2016) "Machine Learning Based Botnet Identification Traffic" The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (Trustcom 2016), Tianjin, China, 23-26 August, pp. 1788-1794.

[2] Larson, D. (2016). Distributed denial of service attacks-holding back the flood. Network Security, 2016(3), 5-7.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[4] Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials.

[5] Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. IEEE network, 8(3), 26-41.

[6] Staudemeyer, R. C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal, 56(1), 136-154.

[7] Tang, M., Alazab, M., Luo, Y., Donlon, M. (2018) Disclosure of cyber security vulnerabilities: time series modelling, International Journal of Electronic Security and Digital Forensics. Vol. 10, No.3, pp 255 - 275.

[8] Venkatraman, S., Alazab, M. "Use of Data Visualisation for Zero-Day Malware Detection," Security and Communication Networks, vol. 2018, Article ID 1728303, 13 pages, 2018. https://doi.org/10.1155/2018/1728303

[9] Vinayakumar R. (2019, January 19). vinayakumarr/Intrusion-detection v1 (Version v1). Zenodo. http://doi.org/10.5281/zenodo.2544036

[10] V. Paxson. Bro: A system for detecting network intruders in realtime. Computer networks, vol. 31, no. 23, pp. 24352463, 1999. DOI http://dx.doi. org/10.1016/S1389-1286(99)00112-7