CHAPTER -4

IMPLEMENTATION

4.1 SOURCE CODE :

```
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.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy score
from sklearn. feature_selection import SelectFromModel
from sklearn.linear_model import Lasso
from sklearn. feature selection import SelectKBest
from sklearn.feature_selection import chi2
from metamodels import Sequential
from keras.layers import Dense
main = tkinter.Tk()
main.title("Network Intrusion Detection")
main.geometry("1300x1200")

global filename
```

```
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, ann_acc, classifier

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

def splitdataset(balance_data):
  X = balance_data.values[:, 0:38]
  Y = balance_data.values[:, 38]
  print(X)
  print(Y)
  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 = "NSL-KDD-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","urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell",
"su_attempted","num_root","num_file_creations","num_shells","num_access_files","num_ou
tbound_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","d
st_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_sr
v_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,"nm
ap":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,"loadmodu
le":20,"spy":21,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtab
le":27,"httptunnel":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xt
erm":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]+','
        if row[41] == 'normal':
          dataset+='0'
        if row[41] == 'anomaly':
          dataset+='1'
        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():
    text.delete('1.0', END)
    global X, Y, X_train, X_test, y_train, y_test
    global balance_data
    balance_data = pd.read_csv("clean.txt")
    X, Y, X_train, X_test, y_train, y_test = splitdataset(balance_data)
    text.insert(END,"Train & Test Model Generated\n\n")
    text.insert(END,"Total Dataset Size : "+str(len(balance_data))+"\n")
    text.insert(END,"Split Training Size : "+str(len(X_train))+"\n")
    text.insert(END,"Split Test Size : "+str(len(X_test))+"\n")

def prediction(X_test, cls):
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
```

```
            cols+=columns[i]+','
        if row[41] == 'normal':
          dataset+='0'
        if row[41] == 'anomaly':
          dataset+='1'
        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():
    text.delete('1.0', END)
    global X, Y, X_train, X_test, y_train, y_test
    global balance_data
    balance_data = pd.read_csv("clean.txt")
    X, Y, X_train, X_test, y_train, y_test = splitdataset(balance_data)
    text.insert(END,"Train & Test Model Generated\n\n")
    text.insert(END,"Total Dataset Size : "+str(len(balance_data))+"\n")
    text.insert(END,"Split Training Size : "+str(len(X_train))+"\n")
    text.insert(END,"Split Test Size : "+str(len(X_test))+"\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):
  accuracy = accuracy_score(y_test,y_pred)*100
  text.insert(END,details+"\n\n")
  text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
  return accuracy

def runSVM():
  text.delete('1.0', END)
  global  svm_acc
  global classifier
  global X, Y, X_train, X_test, y_train, y_test
  total = X_train.shape[1];
  X_train1 = SelectKBest(chi2,15).fit_transform(X_train, y_train)
  X_test1 = SelectKBest(chi2,15).fit_transform(X_test,y_test)
  text.insert(END,"Total Features: "+str(total)+"\n")
  text.insert(END,"Features set reduce after applying features selection concept : "+str((total
- X_train.shape[1]))+"\n\n")
  cls = svm.SVC(kernel='rbf', class_weight='balanced', probability=True)
  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')
  classifier = cls
 def runANN():
  text.delete('1.0', END)
  global ann_acc
  global X, Y, X_train, X_test, y_train, y_test
  total = X_train.shape[1];
```

```
    X_train = SelectKBest(chi2,25).fit_transform(X_train, y_train)
    X_test = SelectKBest(chi2,25).fit_transform(X_test,y_test)
    text.insert(END,"Total Features : "+str(total)+"\n")
    text.insert(END,"Features set reduce after applying features selection concept :
    "+str((total -X_train.shape[1]))+"\n\n")
    model = Sequential()
    model.add(Dense(30, input_dim=25, activation='relu'))
    model.add(Dense(25, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=100, batch_size=32)
    _, ann_acc = model.evaluate(X_train, y_train)
    ann_acc = ann_acc*100
    text.insert(END,"ANN Accuracy : "+str(ann_acc)+"\n\n")
  def detectAttack():
    text.delete('1.0', END)
    global X, Y, X_train, X_test, y_train, y_test
    filename = filedialog.askopenfilename(initialdir="NSL-KDD-Dataset")
    test = pd.read_csv(filename)
    text.insert(END,filename+" test file loaded\n");
    y_pred = classifier.predict(test)
    print(y_pred)
    for i in range(len(test)):
      if str(y_pred[i]) == '1.0':
        text.insert(END,"X=%s, Predicted=%s" % (X_test[i], ' Infected. Detected Anamoly
Signatures')+"\n\n")
      else:
        text.insert(END,"X=%s, Predicted=%s" % (X_test[i], 'Normal Signatures')+"\n\n")

def graph():
    height = [svm_acc,ann_acc]
    bars = ('SVM Accuracy', 'ANN 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='Network Intrusion Detection using Supervised Machine Learning
Technique with Feature Selection')
title.config(bg='PaleGreen2', fg='Khaki4')
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=700,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='DarkOrange1', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=700,y=150)

preprocess = Button(main, text="Preprocess Dataset", command=preprocess)
preprocess.place(x=700,y=200)
preprocess.config(font=font1)

model = Button(main, text="Generate Training Model", command=generateModel)
model.place(x=700,y=250)
model.config(font=font1)

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

```
runsvm.place(x=700,y=300)
runsvm.config(font=font1)

annButton = Button(main, text="Run ANN Algorithm", command=runANN)
annButton.place(x=700,y=350)
annButton.config(font=font1)

attackButton = Button(main, text="Upload Test Data & Detect Attack",
command=detectAttack)
attackButton.place(x=700,y=400)
attackButton.config(font=font1)

graphButton = Button(main, text="Accuracy Graph", command=graph)
graphButton.place(x=700,y=450)
graphButton.config(font=font1)
font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=80)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=100)
text.config(font=font1)
main.config(bg='PeachPuff2')
main.mainloop()
```