IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Impact Factor: 7.301
Volume 3, Issue 3, April 2023

Toward Discovery and Attribution of Digital Assaults in IoT-Enabled Digital Actual Frameworks

L Sankara Rao[1], Ch. Srinivas Goutham[2], Bevera Nikhil[3], Chammallamudi Sritrivedh[4], Ch Sudhir Kumar[5]
Assistant Professor, Department of Computer Science and Engineering[1]
Students, Department of Computer Science and Engineering[2,3,4,5]
Raghu Institute of Technology, Visakhapatnam, India

Abstract: *Cyber-physical systems (CPS) that are enabled by the Internet of Things (IoT) can be difficult to protect because security measures designed for general information/operational technology (IT/OT) systems may not be as effective in a CPS setting. As a result, an industrial control system (ICS)-specific, two-level ensemble attack detection and attribution framework is presented in this article. A decision tree and a novel ensemble deep representation-learning model are used to detect attacks in imbalanced ICS environments at the first level. An ensemble deep neural network is made to make attack attribution easier at the second level. Gas pipeline and water treatment system data sets are used to evaluate the proposed model. The results show that, despite having a similar computational complexity, the proposed model performs better than other competing methods.*

Keywords: Securing Internet-of-Things (IoT); Enabled cyber– physical systems (CPS); Industrial control system (ICS)

I. INTRODUCTION

Cyber-physical systems (CPS) are increasingly incorporating Internet of Things (IoT) devices, including critical infrastructure sectors like dams and utility plants. In these settings, IoT gadgets [also alluded to as Modern IoT (IIoT)] are in many cases part of a modern control framework (ICS), entrusted with the solid effort of the foundation. Systems with programmable logic controllers (PLCs) and Modbus protocols, distributed control systems (DCS), and supervisory control and data acquisition (SCADA) systems are all examples of ICS. However, the connection of ICS or IIoT-based systems to public networks increases their vulnerability to cyberattacks and their attack surface. The Stuxnet campaign, which reportedly in 2010 targeted Iranian centrifuges for nuclear enrichment and severely damaged the equipment [1], [2], is one prominent example. Another example is the incident in 2011 that targeted a pump and caused a water plant in Illinois to fail [3]. Another campaign, BlackEnergy3, targeted Ukraine's power grids in 2015, causing approximately 230 000 people to lose power [4]. There were also reports in April 2018 of successful cyberattacks against three gas pipeline companies in the United States, which caused electronic customer communication systems to be unavailable for several days [1]. In spite of the fact that security arrangements created for data innovation (IT) and functional innovation (OT) frameworks are moderately experienced, they may not be straightforwardly pertinent to ICS. This could be the case, for instance, because the cyber systems and the controlled physical environment are so tightly integrated. As a result, physical behavior analysis and system availability maintenance necessitate system-level security measures [1]. ICS security objectives are focused on in the request for accessibility, trustworthiness, and privacy, in contrast to most IT/OT frameworks (by and large focused on in the request for classification, uprightness, and accessibility) [5]. Because of close coupling between factors of the criticism control circle and actual cycles, (effective) digital assaults on ICS can bring about extreme and possibly lethal ramifications for the general public and our current circumstance. This supports the significance of planning very vigorous wellbeing and security estimations to recognize and forestall interruptions focusing on ICS [1]. Signature and anomaly-based methods for attack detection and attribution are common. There have been attempts to introduce hybrid-based approaches to alleviate the known limitations of signature-based and anomaly-based detection and attribution methods [6]. Due to frequent network upgrades, hybrid-based approaches, which are effective at detecting unusual activations, are unreliable, resulting in various intrusion detection system (IDS) typologies. Beyond this, network metadata analysis (such as IP addresses, transmission ports, traffic duration, and packet intervals) is the primary component of conventional attack detection and attribution methods. As a result, machine learning (ML) or deep neural network (DNN)-based attack detection and attribution solutions have recently reawakened interest. Additionally, there are network-based and host-based approaches to attack detection. For the purpose of detecting attacks in network traffic, the techniques of supervised clustering, single-class or multiclass support vector machine (SVM), fuzzy logic, artificial neural network (ANN), and DNN are frequently utilized. These strategies dissect continuous traffic information to recognize noxious assaults on time. However, sophisticated attacks and insider attacks may be missed by attack detection that only takes into account the host and network data. Because they do not require in-depth knowledge of the cyberthreats, unsupervised models that incorporate process/physical data can enhance a system's monitoring. In general, robust security measures can be circumvented by a sophisticated attacker with sufficient knowledge and time, such as a nation-state advanced persistent threat actor. By modeling only a system's normal behavior and reporting deviations from normal behavior as anomalies, the majority of existing methods also ignore the imbalanced property of ICS data. This may be because there aren't many attack samples in real-world scenarios or data sets. Even though using majority class samples is a good way to avoid problems caused by unbalanced data sets, the trained model won't be able to see the patterns in the attack samples. To put it another way, this method has a high rate of false positives and is unable to identify unseen attacks [7]. As a result, efforts have been made to make use of DL methods, such as to make it easier for automated feature (representation) learning to model complex concepts from simpler ones [8] without relying on features created by humans [9]. This article presents our novel two-stage ensemble deep-learning-based attack detection and attribution framework for imbalanced ICS data sets, inspired by the aforementioned observations. In the main stage, a group portrayal learning model joined with a choice tree (DT) is intended to identify assaults in an imbalanced climate. Several one-versus-all classifiers will join together to form a larger DNN during the second stage to classify the attack attributes with a confidence interval. Additionally, the proposed framework is able to identify previously unseen attack samples. A rundown of our methodology in this study is as per the following. 1) We foster a clever two-stage gathering ICS assault identification strategy fit for distinguishing both recently seen and concealed assaults. In addition, we will show that, in terms of accuracy and f-measure, the proposed method performs better than other competing methods. This approach is tolerant of data with imbalances thanks to the proposed deep representation learning. 2) We propose a clever self-tuning two-stage assault attribution technique that outfits a few profound one-versusall classifiers involving a DNN engineering for diminishing phony problem rates. Attacks with high similarity can be accurately attributed using the proposed method. At the time of this study, this is the first ICS/IIoT attack attribution method that is based on machine learning. 3) We examine the computational intricacy of the proposed assault identification and assault attribution system, showing that notwithstanding its predominant exhibition, its computational intricacy is like that of other DNN-based techniques in the writing. The following is how the rest of this article will be laid out. The relevant background and related literature will be discussed in Section II. Area III will portray the proposed structure, trailed by the exploratory arrangement in Segment IV. The evaluation results, which are based on two real-world ICS data sets, demonstrate in Section V that the proposed framework performs better than several other systems.

II. CONVENTIONAL MACHINE LEARNING

AIn [11], the effectiveness of ML algorithms like K-Nearest Neighbor (KNN), Random Forest (RF), DT, Logistic Regression (LR), ANN, Naive Bayes (NB), and SVM in detecting SQL injection, backdoor, and command attacks in water storage systems was compared. With a recall of 0.9744, the comparative summary suggested that the RF algorithm has the best attack detection; With a recall of 0.8718, the ANN is the fifth best algorithm; Additionally, the LR algorithm has the worst performance, with a recall of 0.4744. Additionally, the authors reported that the ANN considered 0.03% of the normal samples to be attacks and could not detect 12.82 percent of the attacks. Additionally, these ML algorithms are sensitive to imbalanced data because LR, SVM, and KNN considered many attack samples to be normal samples. As such, they arenot reasonable for assault location in ICS. In [12], the creatorsintroduced a KNN calculation to recognize digital assaults on gaspipelines. They oversampled the dataset to achieve balance in order to minimize the effect of using an imbalanced dataset in the algorithm. They found an f-measure of 0.95, an accuracy of 97%, a precision of 0.98, and a recall of 0.92 when they used the KNN on the balanced dataset. A Logical Analysis of

Data (LAD) approach was presented by the authors in [13] to design a two-step anomaly detection system by extracting patterns and rules from sensor data. In the initial step,a framework is named steady or temperamental, and in the secondone, the presence of a not entirely set in stone. They compared the proposed LAD method's performance to that of the DNN, SVM, and CNN methods. The precision metric revealed that the DNN method performed better than the LAD method in these tests; However, recall and the f-measure were better with the LAD.

The DNN algorithm was used by the authors in [14] to identify power system false data injection attacks. Their evaluation of two datasets revealed an accuracy of 91.80%.An autoencoder-based approach to cleaning false data injection attacks with denoising autoencoders was proposed in [15]. Their tests demonstrated that these approaches performed better than the SVM-based approach. They skipped over attack data when training the autoencoder to deal with the effect of unbalanced data on the algorithm. An approach for CPS attack detection that is based on Extreme Learning Machine (ELM) was presented in [16]. To address the imbalanced testof brain organizations, preparing was directed utilizing just ordinary information. The proposed ELM-based method outperformed the SVM attack detection method, as demonstrated by these tests. The majority of existing machine learning algorithms are plagued by the dreadful problem of dimensionality as a result of the enormous volume of data generated by real-world ICS, despite the promising outcomes of conventional and deep learning-based methods. To reduce computational overhead, feature engineering must therefore either generate a new feature representation or reduce the number of features. Additionally, an imbalanced dataset of the ICS is another test that ought to be thought of. Oversampling and under sampling, as well as ignoring attack samples and building algorithms with normal samples, have been used by researchers to try to solve this problem. The purpose of attack attribution is to respond to the question, "What kind of attack was it?" furthermore, this is by and large seriously testing to reply in ICS than in commonplace IT/OT frameworks due to the different organization structures, industry-explicit conventions, andso forward [17], [18]. Designing robust and efficient ML-based attack attribution for ICS and IIoT systems appears to be understudied, despite the limited number of ML-based malware attack attributions [19, 20]. As a result, a two-stage ensemble deep learning-based framework for ICS attack detection and attribution is proposed in this paper. In order to solve the imbalanced data issue without resorting to either oversampling or subsampling, our method makes use of both physical and process data. For attack detection, the proposed framework makes use of an unsupervised ensemble of learned representations from both normal and attack instances. After that, it forms a two-part DNN to attribute the samples into their corresponding attack attributes by employing an ensemble of several one-versus-all classifiers that have been trained on each attack attribute.

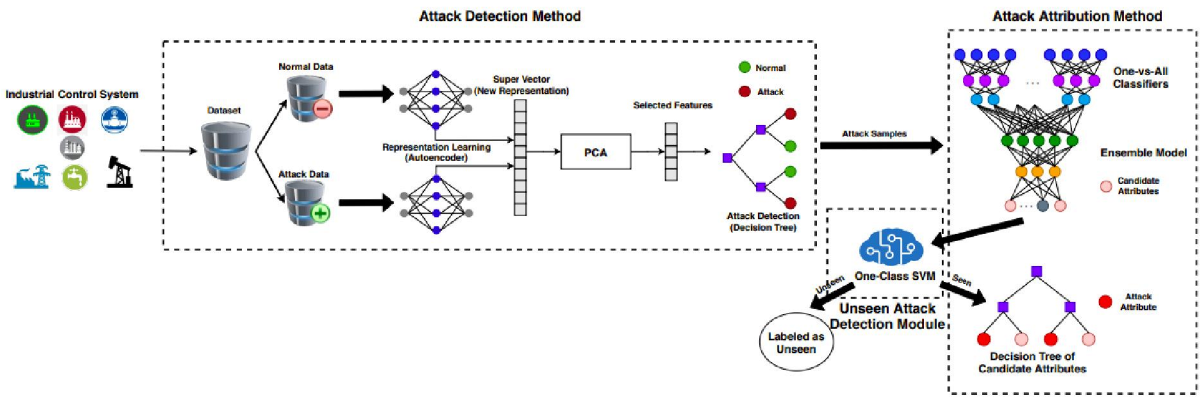

Attack Detection Method
Attack Attribution Method
Industrial Control System
Dataset
Normal Data
Attack Data
Super Vector (New Representation)
Representation Learning (Autoencoder)
PCA
Selected Features
Normal
Attack
Attack Detection (Decision Tree)
Attack Samples
One-vs-All Classifiers
Ensemble Model
Candidate Attributes
One-Class SVM
Unseen
Seen
Unseen Attack Detection Module
Labeled as Unseen
Attack Attribute
Decision Tree of Candidate Attributes

Fig. 1. Proposed attack detection and attribution framework

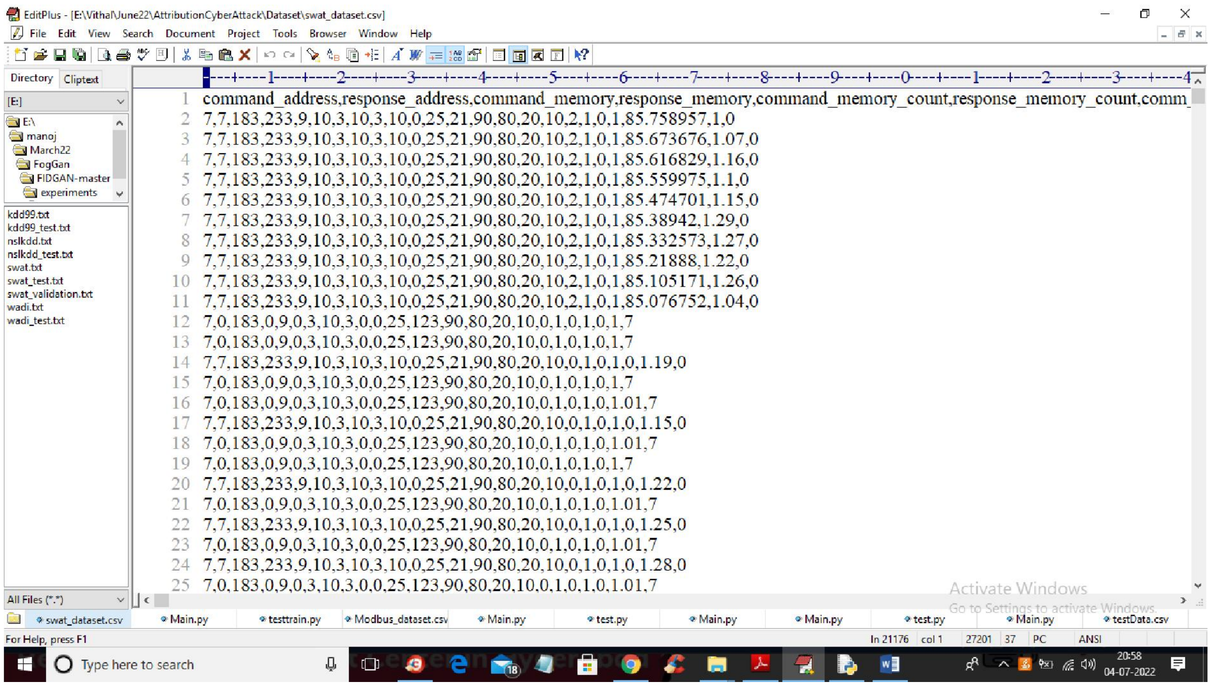
III. EVALUATION METRICS

Cyber-physical systems like industrial equipment and operational IT were able to send and receive data over the internet thanks to the Internet of Things. This hardware's will have sensors to detect gear condition and report to concentrated server utilizing web association. These sensors may be attacked or hacked by malicious users at times, modifying their data before reporting false data to a centralized server and initiating false actions. Many nations' equipment failed as a result of false data, and numerous algorithms were developed to detect attacks. However, all of these algorithms suffer from data imbalance because one class contains a large number of records (such as NORMAL records) and another class, such as attack, may contain fewer records, resulting in an imbalance problem and detection algorithms failing to accurately predict. The current algorithms used OVER and UNDER sampling to correct for the imbalance in the data. This will produce new records for a smaller class, but it improves accuracy but is not sufficient.

We has developed a novel approach to address this issue that does not employ any undersampling or oversampling algorithms. The approach consists of two parts: 1) Auto Encoder: Auto encoder deep learning will be trained on an imbalanced dataset before extracting features from it and using the DECISION TREE algorithm to predict a label for known or unknown attacks. A smaller number of features from the PCA (principal component analysis) algorithm are used to train the decision tree.

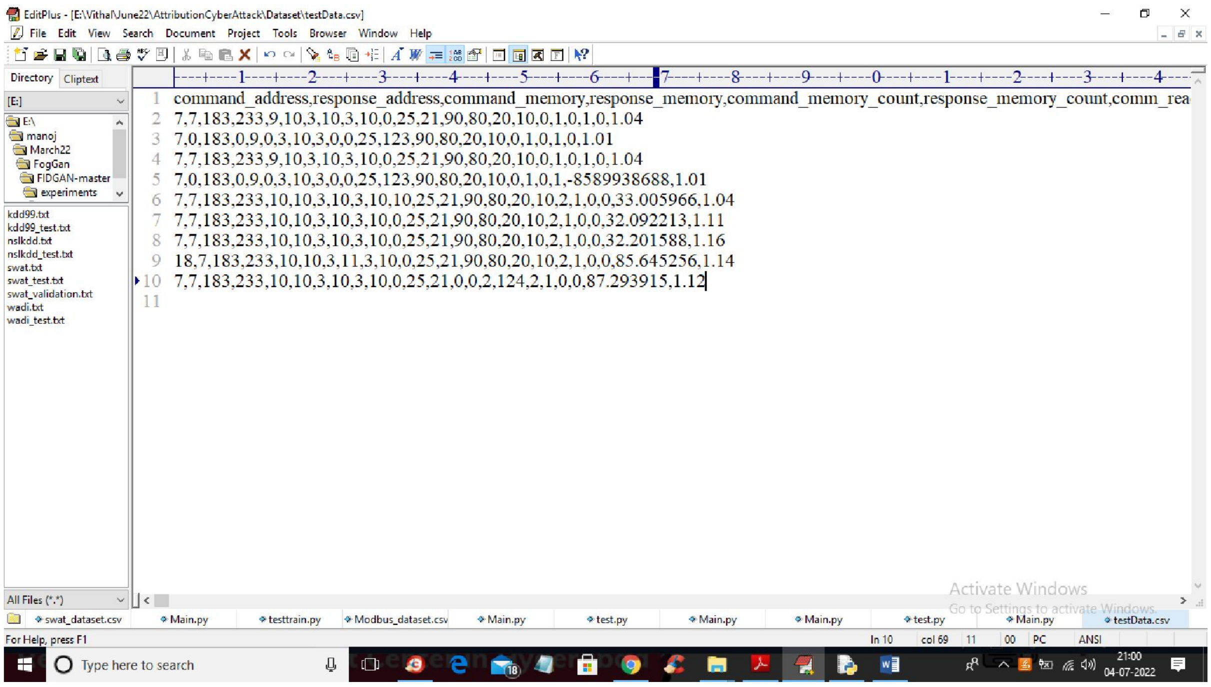
2) DNN, or deep neural network: The DNN algorithm is trained on both known and unknown attacks at this level. In the event that any records contains assault signature, DNN will recognize assault name or class and characteristic them.

The author used SWAT (secure water treatment) to implement this project. The dataset includes IOT request and response signatures and associates each dataset with a unique attack label. The following cyber-attack labels can be found in the dataset: "Normal," "Naive Malicious Response Injection (NMRI)," "Complex Malicious," "Response Injection (CMRI)," "Malicious State Command Injection (MSCI)," "Malicious Parameter Command Injection The details of the dataset are displayed on the screen below.



```
command_address,response_address,command_memory,response_memory,command_memory_count,response_memory_count,comm
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.758957,1,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.673676,1.07,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.616829,1.16,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.559975,1.1,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.474701,1.15,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.38942,1.29,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.332573,1.27,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.21888,1.22,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.105171,1.26,0
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,1,85.076752,1.04,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1,7
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1,7
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.19,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1,7
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01,7
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.15,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01,7
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1,7
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.22,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01,7
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.25,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01,7
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.28,0
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01,7
```

The names of the dataset columns are in the first row of the above dataset screen; the values of the dataset are in the rows that follow, and the attack type ranges from label 0 to label 7. We will utilized above dataset to prepare propose Auto Encoder, choice tree and DNN calculations. In the following screen, we are employing NEW test data that only consists of a signature and does not include any class labels; the proposed algorithm will then identify and assign class labels.



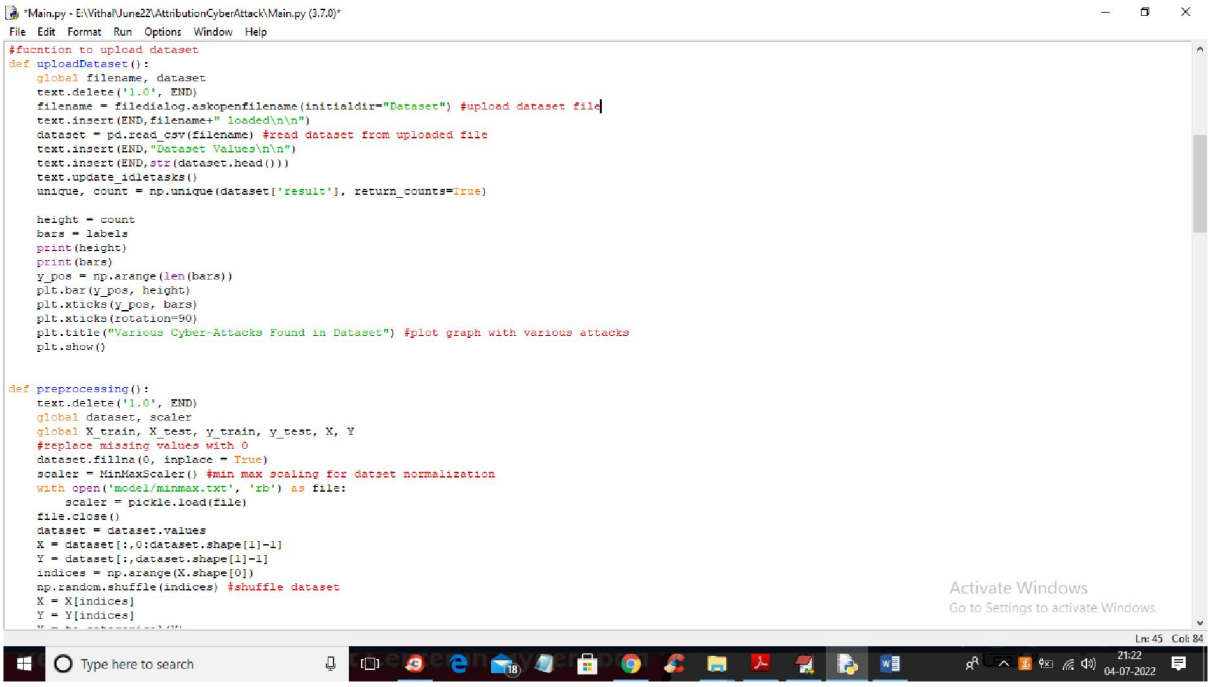
```
command_address,response_address,command_memory,response_memory,command_memory_count,response_memory_count,comm_rea
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.04
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,0,1.01
7,7,183,233,9,10,3,10,3,10,0,25,21,90,80,20,10,0,1,0,1,0,1.04
7,0,183,0,9,0,3,10,3,0,0,25,123,90,80,20,10,0,1,0,1,-8589938688,1.01
7,7,183,233,10,10,3,10,3,10,10,25,21,90,80,20,10,2,1,0,0,33.005966,1.04
7,7,183,233,10,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,0,32.092213,1.11
7,7,183,233,10,10,3,10,3,10,0,25,21,90,80,20,10,2,1,0,0,32.201588,1.16
18,7,183,233,10,10,3,11,3,10,0,25,21,90,80,20,10,2,1,0,0,85.645256,1.14
7,7,183,233,10,10,3,10,3,10,0,25,21,0,0,2,124,2,1,0,0,87.293915,1.12
```

We observe an IOT request signature without class labels in the preceding test data.

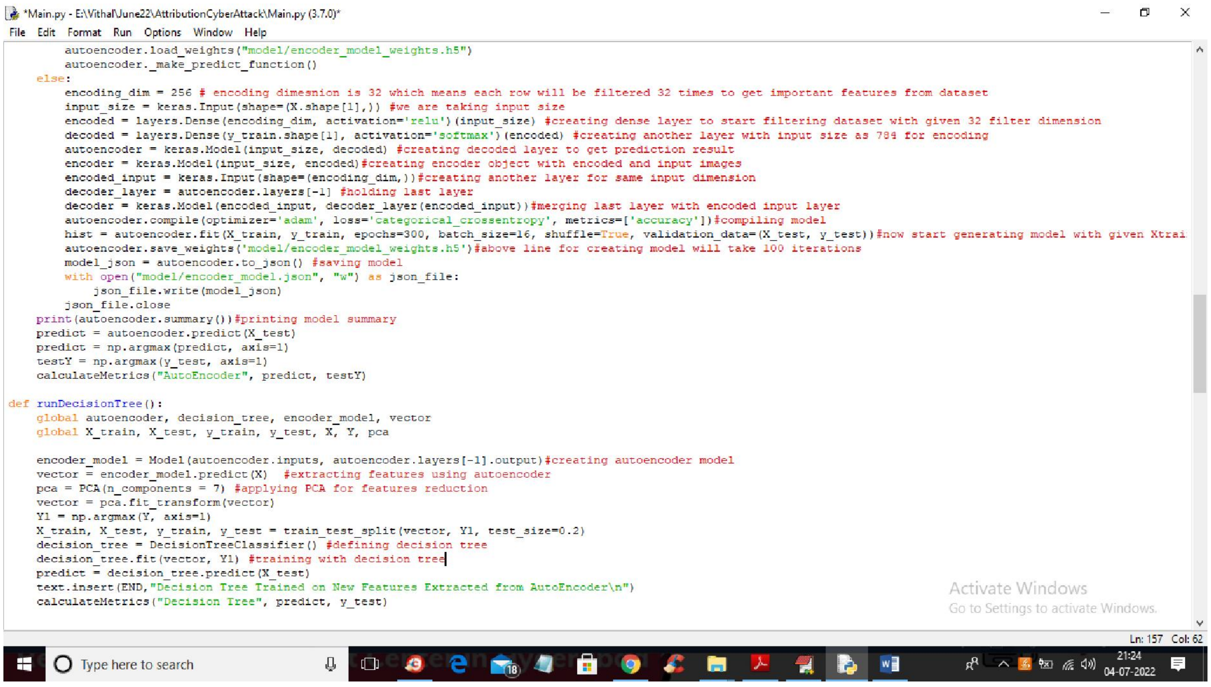
We created the following modules to put this project into action:

1. Upload SWAT Water Dataset: We will upload the dataset to the application using this module, read it, and then identify various attacks in it.
2. Preprocess the dataset: We will use this module to replace all missing values with 0, apply the MIN-MAX scaling algorithm to normalized features, divide the dataset into train and test, using 80 percent of the dataset for training and 20 percent for testing.
3. Run the AutoEncoder Algorithm: We will train the AutoEncoder deep learning algorithm with this module, and then we will extract features from that model.
4. Use PCA to run the decision tree: PCA will be used to reduce the size of the extracted features from AutoEncoder, and then Decision tree will be used to retrain them. Based on the signatures of the dataset, Decision Tree will determine the label for each record.
5. Run the DNN Algorithm: The predicted decision tree label will be further trained using the DNN (deep neural network) algorithm to identify and attribute attacks. utilizing this module we will transfer obscure or un-name TEST Information and afterward DNN will anticipate assault type
6. Comparison Chart: We will plot a comparison graph between all algorithms using this module. 8) Comparison Table: We will use this module to display a comparison table containing metrics like accuracy, precision, recall, and FSCORE for each algorithm.

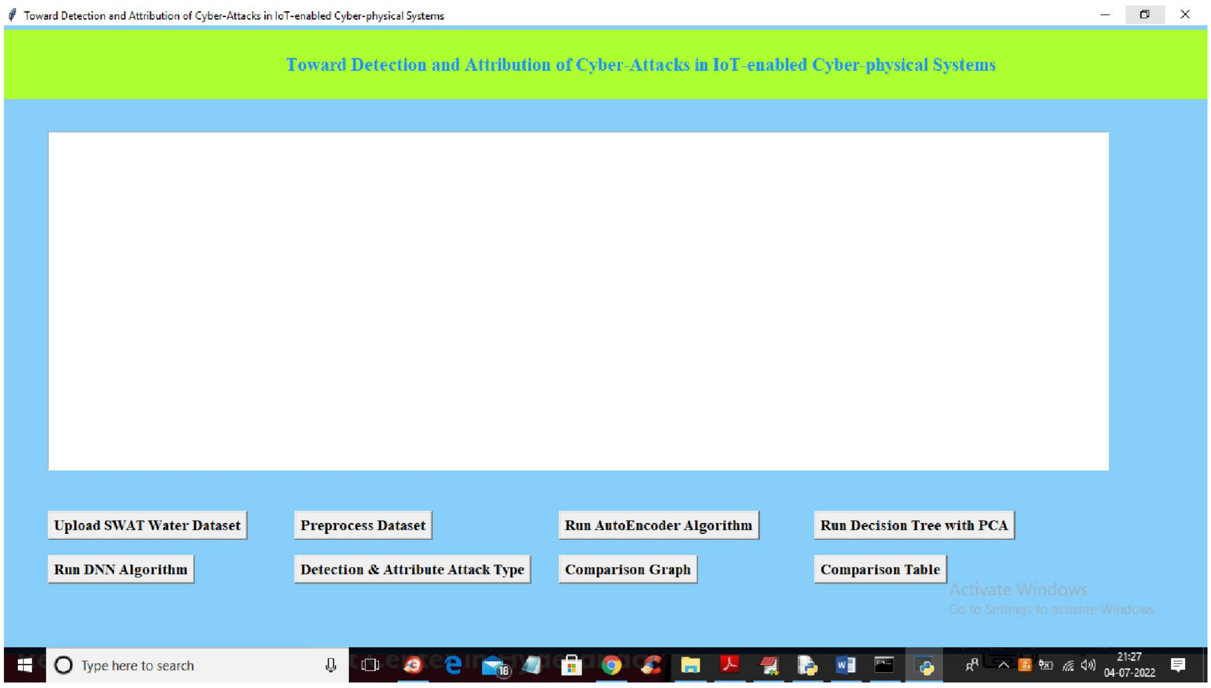
Read the comments in red on the screen below to learn more about how the algorithms are implemented.


Main.py - E:\Vithal\June22\AttributionCyberAttack\Main.py (3.7.0)

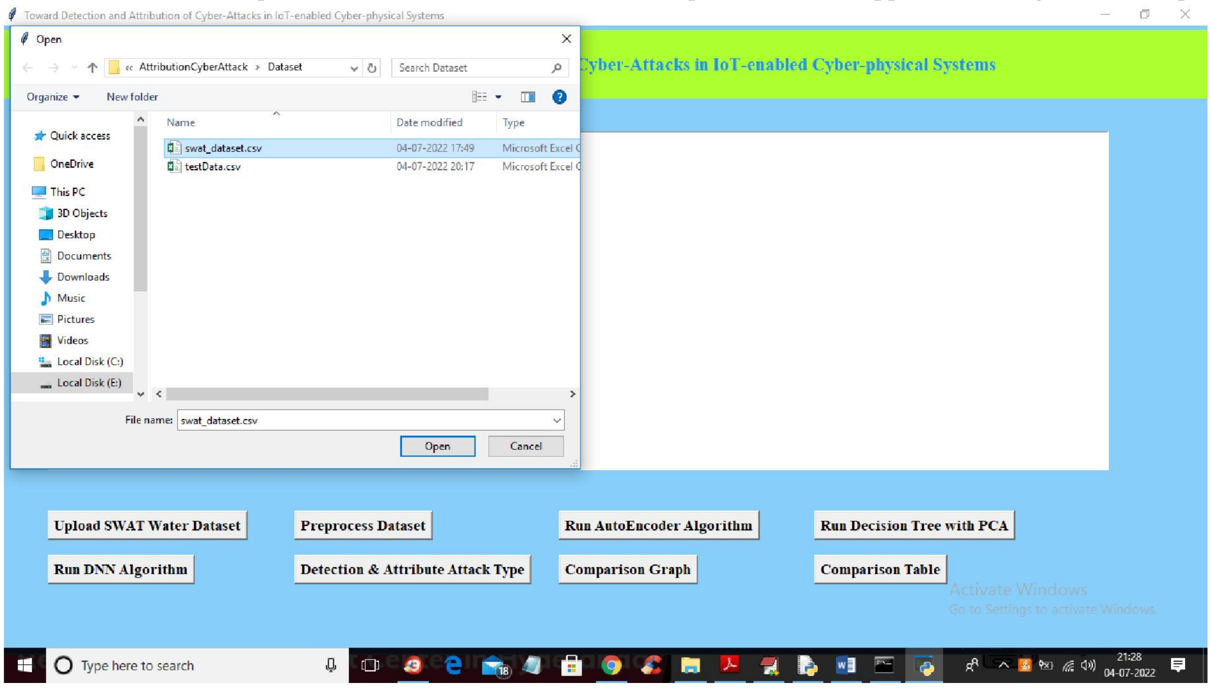
```
#fucntion to upload dataset
def uploadDataset():
    global filename, dataset
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="Dataset") #upload dataset file
    text.insert(END,filename+" loaded\n\n")
    dataset = pd.read_csv(filename) #read dataset from uploaded file
    text.insert(END,"Dataset Values\n\n")
    text.insert(END,str(dataset.head()))
    text.update_idletasks()
    unique, count = np.unique(dataset['result'], return_counts=True)

    height = count
    bars = labels
    print(height)
    print(bars)
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.xticks(rotation=90)
    plt.title("Various Cyber-Attacks Found in Dataset") #plot graph with various attacks
    plt.show()


def preprocessing():
    text.delete('1.0', END)
    global dataset, scaler
    global X_train, X_test, y_train, y_test, X, Y
    #replace missing values with 0
    dataset.fillna(0, inplace = True)
    scaler = MinMaxScaler() #min max scaling for datset normalization
    with open('model/minmax.txt', 'rb') as file:
        scaler = pickle.load(file)
    file.close()
    dataset = dataset.values
    X = dataset[:,0:dataset.shape[1]-1]
    Y = dataset[:,dataset.shape[1]-1]
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices) #shuffle dataset
    X = X[indices]
    Y = Y[indices]
```


Main.py - E:\Vithal\June22\AttributionCyberAttack\Main.py (3.7.0)

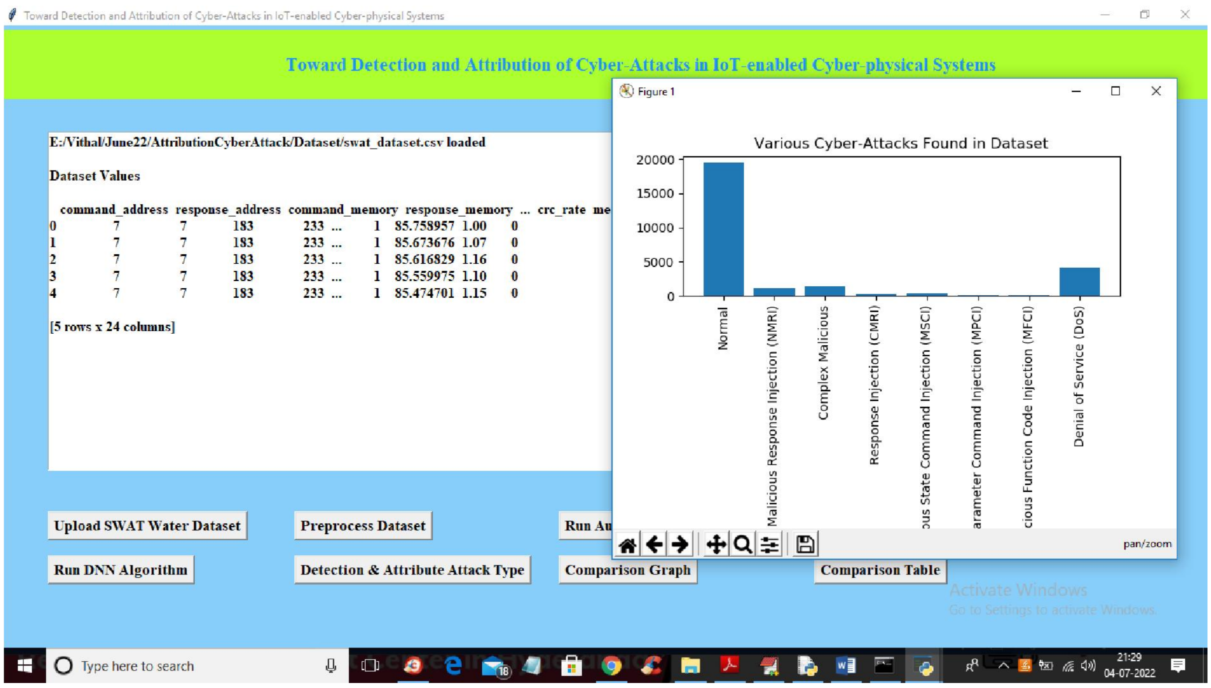
```
        autoencoder.load_weights("model/encoder_model_weights.h5")
        autoencoder._make_predict_function()
    else:
        encoding_dim = 256 # encoding dimesnion is 32 which means each row will be filtered 32 times to get important features from dataset
        input_size = keras.Input(shape=(X.shape[1],)) #we are taking input size
        encoded = layers.Dense(encoding_dim, activation='relu')(input_size) #creating dense layer to start filtering dataset with given 32 filter dimension
        decoded = layers.Dense(y_train.shape[1], activation='softmax')(encoded) #creating another layer with input size as 784 for encoding
        autoencoder = keras.Model(input_size, decoded) #creating decoded layer to get prediction result
        encoder = keras.Model(input_size, encoded)#creating encoder object with encoded and input images
        encoded_input = keras.Input(shape=(encoding_dim,))#creating another layer for same input dimension
        decoder_layer = autoencoder.layers[-1] #holding last layer
        decoder = keras.Model(encoded_input, decoder_layer(encoded_input))#merging last layer with encoded input layer
        autoencoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])#compiling model
        hist = autoencoder.fit(X_train, y_train, epochs=300, batch_size=16, shuffle=True, validation_data=(X_test, y_test))#now start generating model with given Xtrai
        autoencoder.save_weights('model/encoder_model_weights.h5')#above line for creating model will take 100 iterations
        model_json = autoencoder.to_json() #saving model
        with open("model/encoder_model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close
    print(autoencoder.summary())#printing model summary
    predict = autoencoder.predict(X_test)
    predict = np.argmax(predict, axis=1)
    testY = np.argmax(y_test, axis=1)
    calculateMetrics("AutoEncoder", predict, testY)

def runDecisionTree():
    global autoencoder, decision_tree, encoder_model, vector
    global X_train, X_test, y_train, y_test, X, Y, pca

    encoder_model = Model(autoencoder.inputs, autoencoder.layers[-1].output)#creating autoencoder model
    vector = encoder_model.predict(X)  #extracting features using autoencoder
    pca = PCA(n_components = 7) #applying PCA for features reduction
    vector = pca.fit_transform(vector)
    Y1 = np.argmax(Y, axis=1)
    X_train, X_test, y_train, y_test = train_test_split(vector, Y1, test_size=0.2)
    decision_tree = DecisionTreeClassifier() #defining decision tree
    decision_tree.fit(vector, Y1) #training with decision tree
    predict = decision_tree.predict(X_test)
    text.insert(END,"Decision Tree Trained on New Features Extracted from AutoEncoder\n")
    calculateMetrics("Decision Tree", predict, y_test)
```


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

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

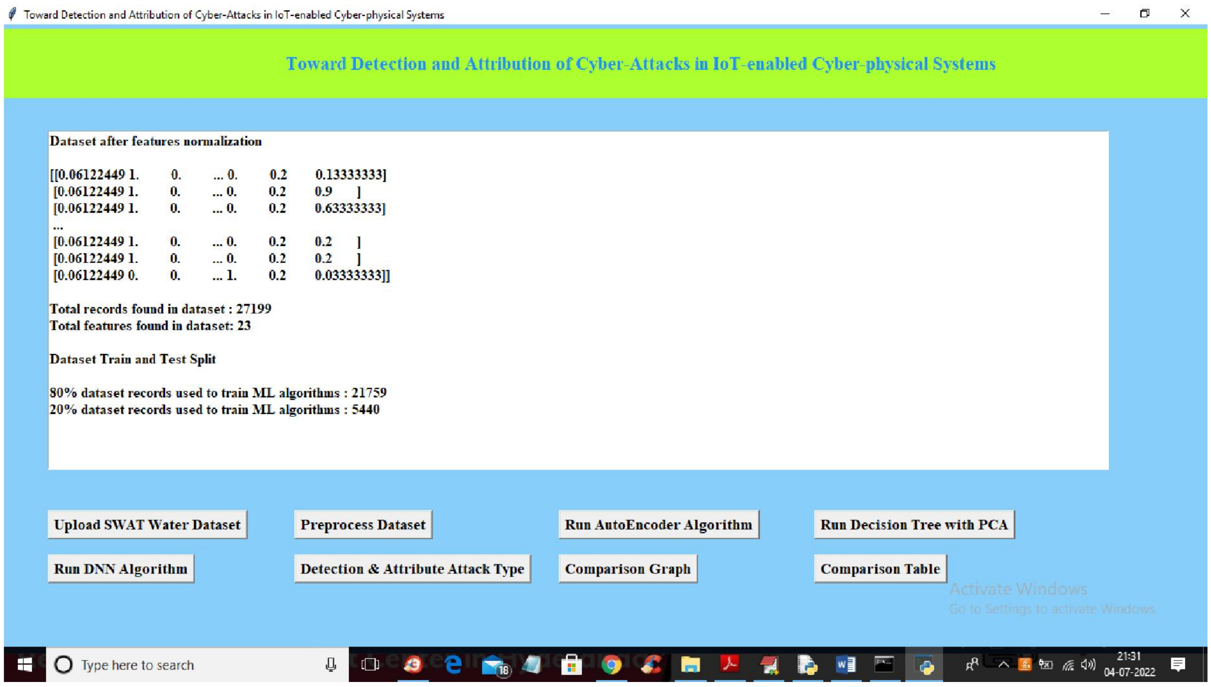

Open
AttributionCyberAttack › Dataset
swat_dataset.csv
testData.csv
File name: swat_dataset.csv
Open
Cancel
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

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


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Figure 1
Various Cyber-Attacks Found in Dataset
20000
15000
10000
5000
0
Normal
Malicious Response Injection (NMRI)
Complex Malicious
Response Injection (CMRI)
ous State Command Injection (MSCI)
arameter Command Injection (MPCI)
cious Function Code Injection (MFCI)
Denial of Service (DoS)
pan/zoom
Upload SWAT Water Dataset
Preprocess Dataset
Run Au
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

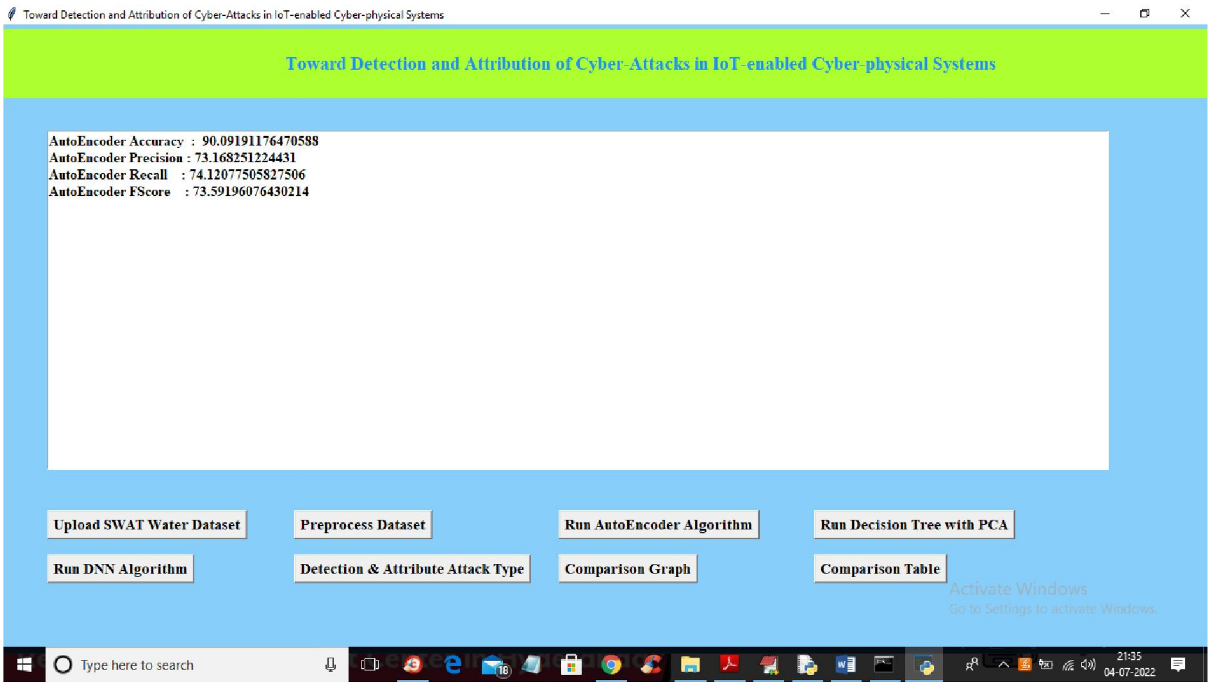
```
E:/Vithal/June22/AttributionCyberAttack/Dataset/swat_dataset.csv loaded

Dataset Values

  command_address  response_address  command_memory  response_memory  ...  crc_rate  me
0          7              7          183         233  ...      1   85.758957  1.00     0
1          7              7          183         233  ...      1   85.673676  1.07     0
2          7              7          183         233  ...      1   85.616829  1.16     0
3          7              7          183         233  ...      1   85.559975  1.10     0
4          7              7          183         233  ...      1   85.474701  1.15     0

[5 rows x 24 columns]
```

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


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

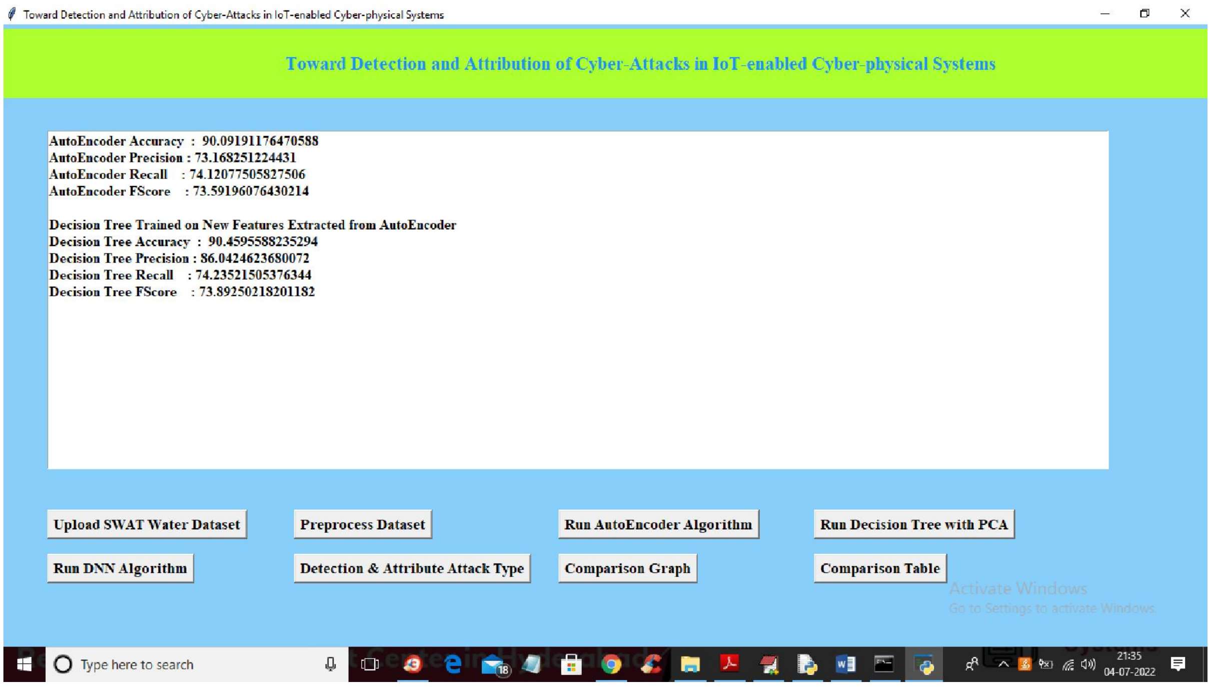
```
Dataset after features normalization

[[0.06122449 1.        0.        ... 0.        0.2        0.13333333]
 [0.06122449 1.        0.        ... 0.        0.2        0.9       ]
 [0.06122449 1.        0.        ... 0.        0.2        0.63333333]
 ...
 [0.06122449 1.        0.        ... 0.        0.2        0.2       ]
 [0.06122449 1.        0.        ... 0.        0.2        0.2       ]
 [0.06122449 0.        0.        ... 1.        0.2        0.03333333]]

Total records found in dataset : 27199
Total features found in dataset: 23

Dataset Train and Test Split

80% dataset records used to train ML algorithms : 21759
20% dataset records used to train ML algorithms : 5440
```

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


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

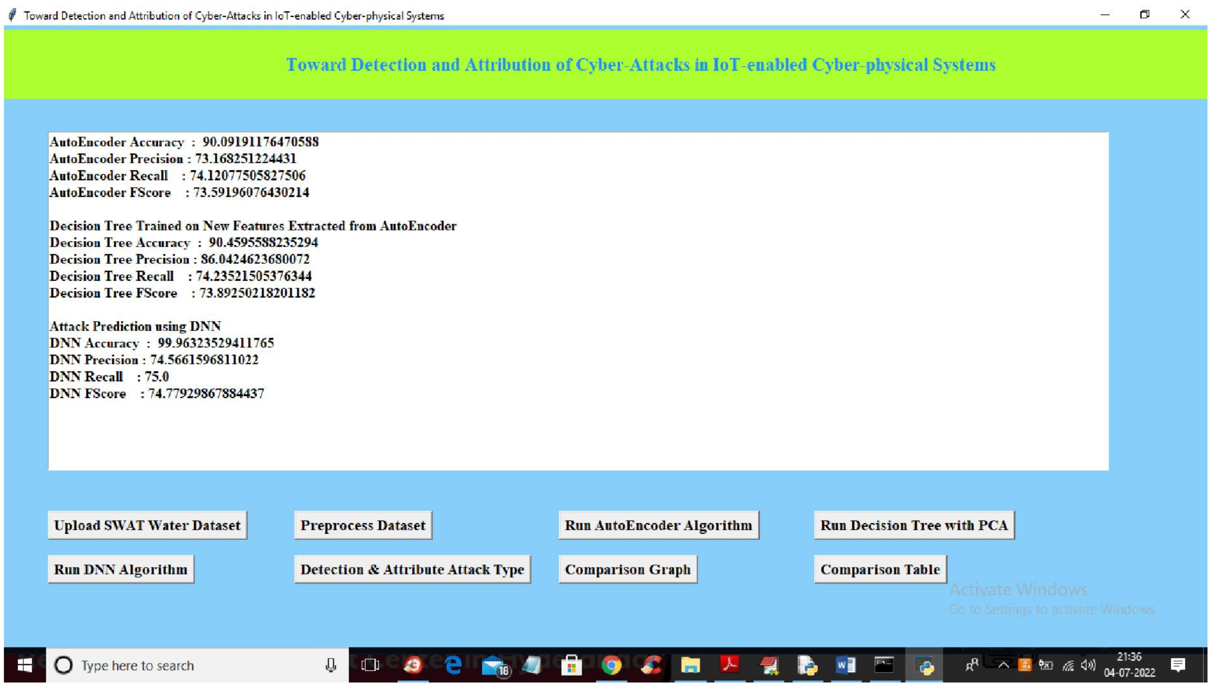
```
AutoEncoder Accuracy  :  90.09191176470588
AutoEncoder Precision : 73.168251224431
AutoEncoder Recall    : 74.12077505827506
AutoEncoder FScore    : 73.59196076430214
```

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


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

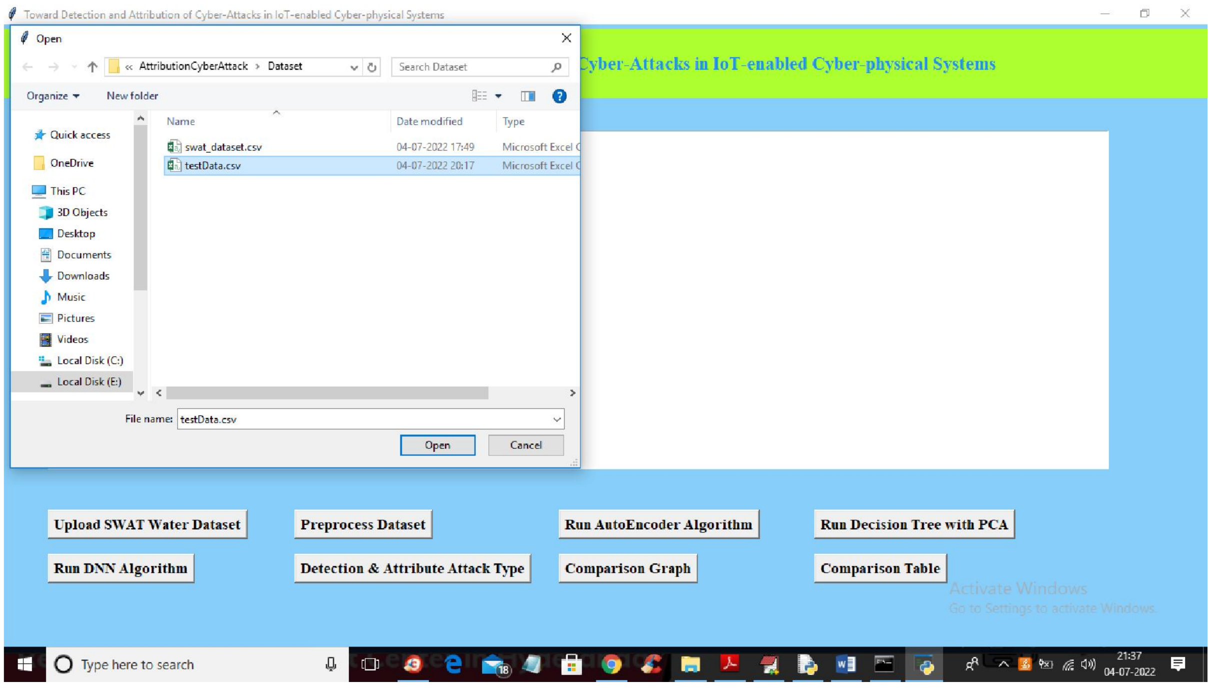
```
AutoEncoder Accuracy  :  90.09191176470588
AutoEncoder Precision : 73.168251224431
AutoEncoder Recall    : 74.12077505827506
AutoEncoder FScore    : 73.59196076430214

Decision Tree Trained on New Features Extracted from AutoEncoder
Decision Tree Accuracy  :  90.4595588235294
Decision Tree Precision : 86.0424623680072
Decision Tree Recall    : 74.23521505376344
Decision Tree FScore    : 73.89250218201182
```

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

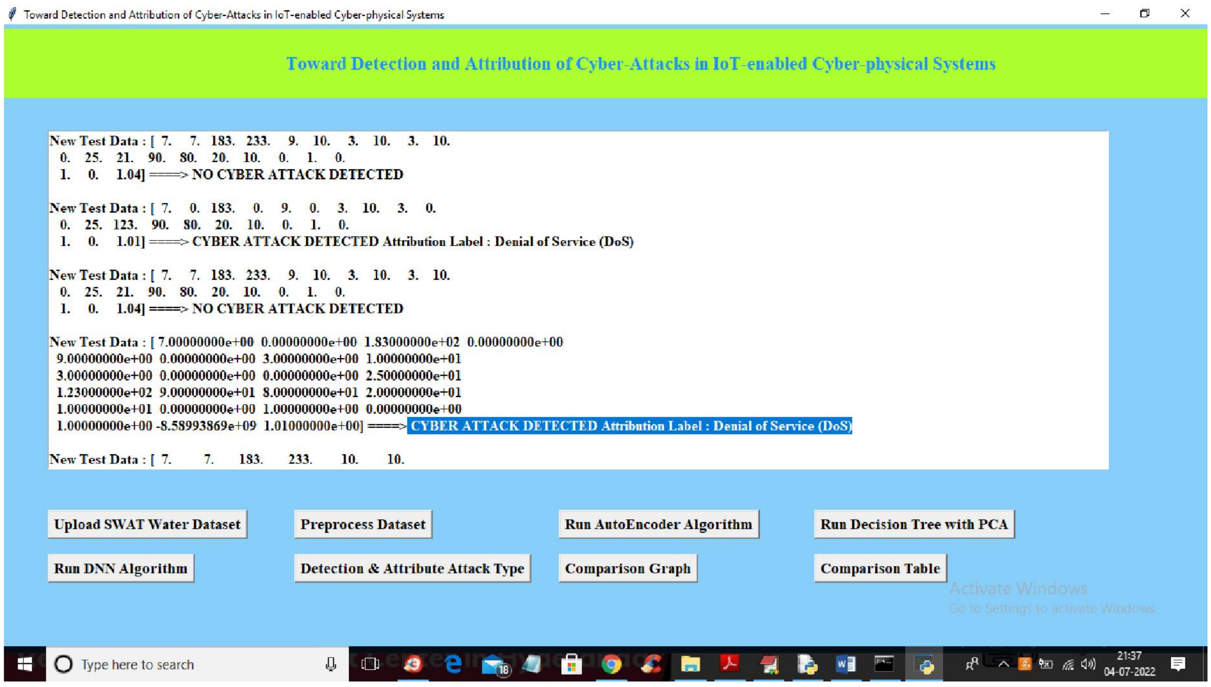

Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

```
AutoEncoder Accuracy  :  90.09191176470588
AutoEncoder Precision : 73.168251224431
AutoEncoder Recall    : 74.12077505827506
AutoEncoder FScore    : 73.59196076430214

Decision Tree Trained on New Features Extracted from AutoEncoder
Decision Tree Accuracy  :  90.4595588235294
Decision Tree Precision : 86.0424623680072
Decision Tree Recall    : 74.23521505376344
Decision Tree FScore    : 73.89250218201182

Attack Prediction using DNN
DNN Accuracy  :  99.96323529411765
DNN Precision : 74.5661596811022
DNN Recall    : 75.0
DNN FScore    : 74.77929867884437
```

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


Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems
Open
AttributionCyberAttack > Dataset
swat_dataset.csv
testData.csv
File name: testData.csv
Open
Cancel
Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

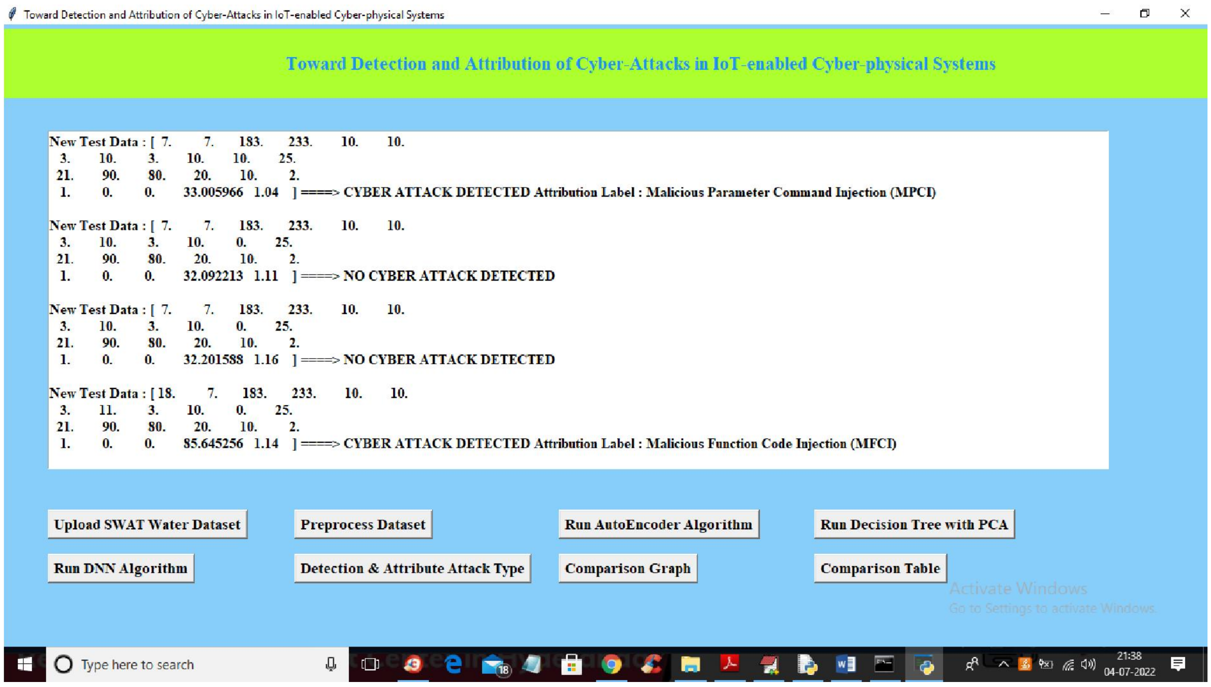
In above screen selecting and uploading ‘TEST DATA’ file and then click on ‘Open’ button to get below output


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

```
New Test Data : [ 7.   7.  183.  233.   9.  10.   3.  10.   3.  10.
  0.  25.  21.  90.  80.  20.  10.   0.   1.   0.
  1.   0.   1.04] ====> NO CYBER ATTACK DETECTED

New Test Data : [ 7.   0.  183.   0.   9.   0.   3.  10.   3.   0.
  0.  25.  123.  90.  80.  20.  10.   0.   1.   0.
  1.   0.   1.01] ====> CYBER ATTACK DETECTED Attribution Label : Denial of Service (DoS)

New Test Data : [ 7.   7.  183.  233.   9.  10.   3.  10.   3.  10.
  0.  25.  21.  90.  80.  20.  10.   0.   1.   0.
  1.   0.   1.04] ====> NO CYBER ATTACK DETECTED

New Test Data : [ 7.00000000e+00  0.00000000e+00  1.83000000e+02  0.00000000e+00
  9.00000000e+00  0.00000000e+00  3.00000000e+00  1.00000000e+01
  3.00000000e+00  0.00000000e+00  0.00000000e+00  2.50000000e+01
  1.23000000e+02  9.00000000e+01  8.00000000e+01  2.00000000e+01
  1.00000000e+01  0.00000000e+00  1.00000000e+00  0.00000000e+00
  1.00000000e+00 -8.58993869e+09  1.01000000e+00] ====> CYBER ATTACK DETECTED Attribution Label : Denial of Service (DoS)

New Test Data : [ 7.      7.      183.      233.      10.      10.
```

Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

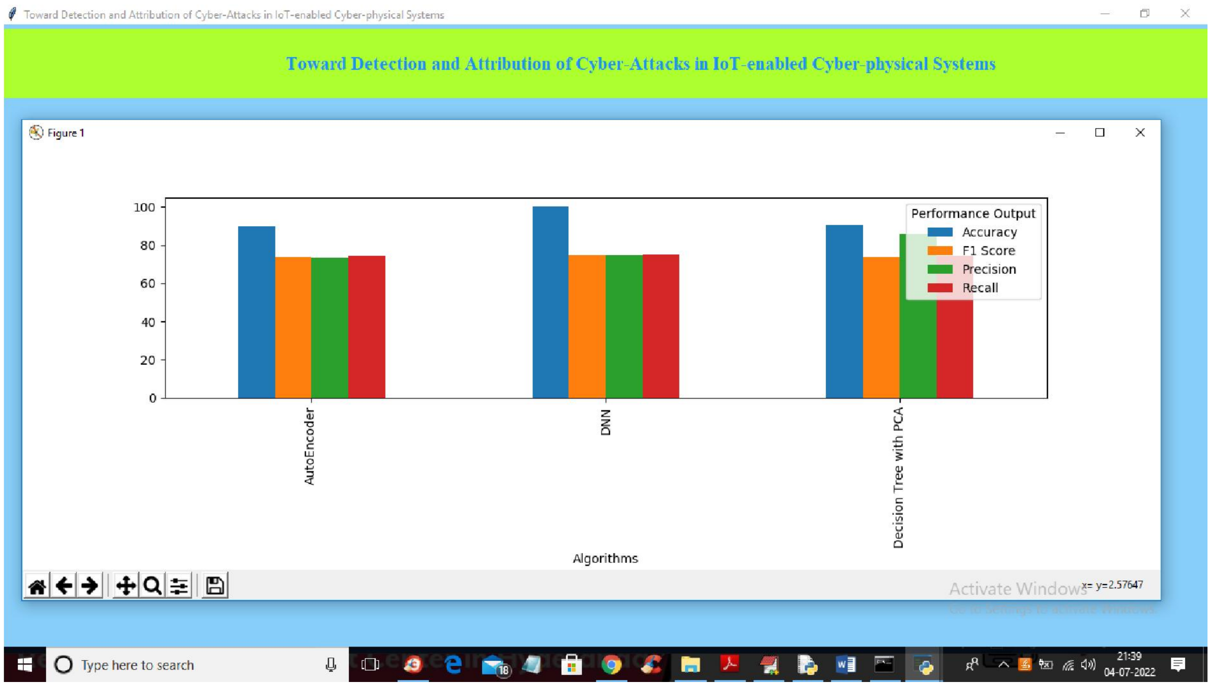
In above screen in square bracket we can see TEST data values and after arrow =➔ symbol we can see detected ATTACK TYPE and scroll down above text area to view all detection


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

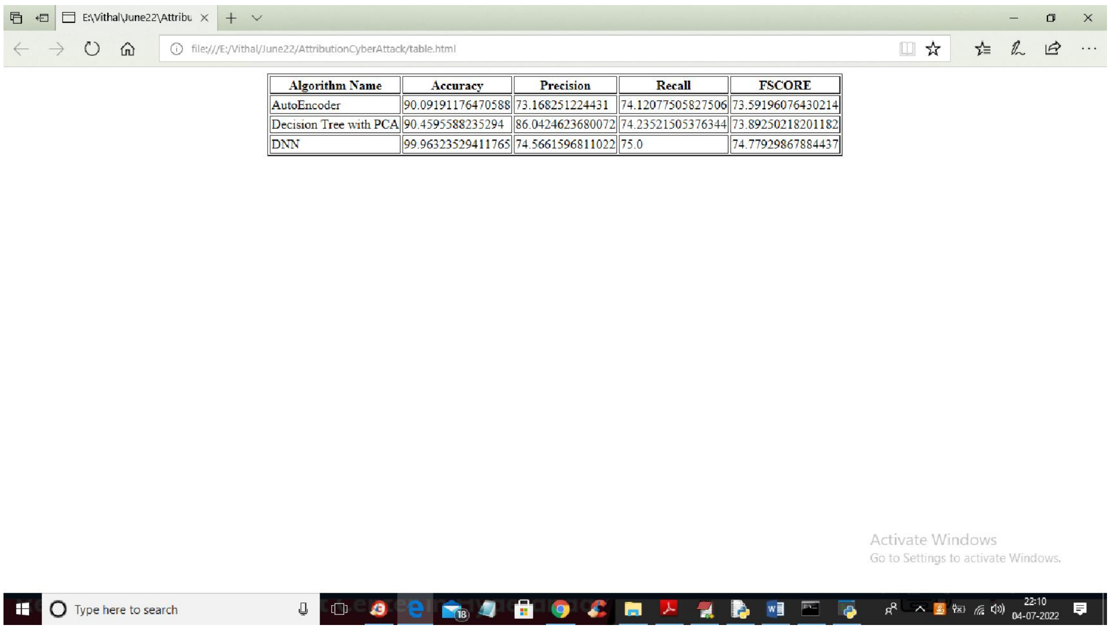
```
New Test Data : [ 7.      7.      183.      233.      10.      10.
  3.      10.      3.      10.      10.      25.
 21.      90.      80.      20.      10.      2.
  1.      0.      0.      33.005966  1.04   ] ====> CYBER ATTACK DETECTED Attribution Label : Malicious Parameter Command Injection (MPCI)

New Test Data : [ 7.      7.      183.      233.      10.      10.
  3.      10.      3.      10.      0.      25.
 21.      90.      80.      20.      10.      2.
  1.      0.      0.      32.092213  1.11   ] ====> NO CYBER ATTACK DETECTED

New Test Data : [ 7.      7.      183.      233.      10.      10.
  3.      10.      3.      10.      0.      25.
 21.      90.      80.      20.      10.      2.
  1.      0.      0.      32.201588  1.16   ] ====> NO CYBER ATTACK DETECTED

New Test Data : [18.      7.      183.      233.      10.      10.
  3.      11.      3.      10.      0.      25.
 21.      90.      80.      20.      10.      2.
  1.      0.      0.      85.645256  1.14   ] ====> CYBER ATTACK DETECTED Attribution Label : Malicious Function Code Injection (MFCI)
```

Upload SWAT Water Dataset
Preprocess Dataset
Run AutoEncoder Algorithm
Run Decision Tree with PCA
Run DNN Algorithm
Detection & Attribute Attack Type
Comparison Graph
Comparison Table

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



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



Algorithm Name	Accuracy	Precision	Recall	FSCORE
AutoEncoder	90.09191176470588	73.168251224431	74.12077505827506	73.59196076430214
Decision Tree with PCA	90.4595588235294	86.0424623680072	74.23521505376344	73.89250218201182
DNN	99.96323529411765	74.5661596811022	75.0	74.77929867884437

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

IV. CONCLUSION

For unbalanced ICS data, this article proposed a novel two-stage ensemble deep learning-based attack detection and attribution framework. In the attack detection stage, a DT is used to identify the attack samples and deep representation learning is used to map the samples to the new higher dimensional space. This stage is able to detect attacks that have not been seen before and is resistant to data sets with imbalances. The assault attribution stage is a group of a few one-versus-all classifiers, each prepared on a particular assault quality. As demonstrated, the model as a whole is a complex DNN with a component that is both partially and fully connected and can accurately attribute cyberattacks. Even though the proposed framework has a complicated architecture, the training and testing phases have computational complexity of O(n4) and O(n2), where n is the number of training samples, which is comparable to other DNN-based techniques described in the literature. In addition, the proposed framework outperforms previous efforts in terms of recall and f-measure in its ability to accurately identify and attribute samples. The future expansion incorporates the plan of a digital danger hunting part to work with the distinguishing proof of irregularities imperceptible to the discovery part for instance by building an ordinary profile over the whole framework and the resources.

REFERENCES

[1]. K. Graves, Ceh: Official certified ethical hacker review guide: Exam 312-50. John Wiley & Sons, 2007.

[2]. R. Christopher, “Port scanning techniques and the defense against them,” SANS Institute, 2001.

[3]. M. Baykara, R. Das¸, and I. Karado ˘gan, “Bilgi g ¨uvenli ˘gisistemlerindekullanilanarac¸larinincelenmesi,” in 1st International Symposium on Digital Forensics and Security (ISDFS13), 2013, pp. 231–239.

[4]. S. Staniford, J. A. Hoagland, and J. M. McAlerney, “Practical automated detection of stealthy portscans,” Journal of Computer Security, vol. 10, no. 1-2, pp. 105–136, 2002.

[5]. S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo, “Surveillance detection in high bandwidth environments,” in DARPA Information Survivability Conference and Exposition, 2003. Proceedings, vol. 1. IEEE, 2003, pp. 130–138.

[6]. K. Ibrahimi and M. Ouaddane, “Management of intrusion detection systems based-kdd99: Analysis with lda and pca,” in Wireless Networks and Mobile Communications (WINCOM), 2017 International Conference on. IEEE, 2017, pp. 1–6.

[7]. N. Moustafa and J. Slay, “The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems,” in Building Analysis Datasets and Gathering

[8]. Experience Returns for Security (BADGERS), 2015 4th International Workshop on. IEEE, 2015, pp. 25–31.

[9]. L. Sun, T. Anthony, H. Z. Xia, J. Chen, X. Huang, and Y. Zhang, “Detection and classification of malicious patterns in network traffic using benford’s law,” in Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017. IEEE, 2017, pp. 864–872.

[10]. S. M. Almansob and S. S. Lomte, “Addressing challenges for intrusion detection system using naive bayes and pca algorithm,” in Convergence in Technology (I2CT), 2017 2nd International Conference for. IEEE, 2017, pp. 565–568.

[11]. M. C. Raja and M. M. A. Rabbani, “Combined analysis of support vector machine and principle component analysis for ids,” in IEEE International Conference on Communication and Electronics Systems, 2016, pp. 1–5.

[12]. S. Aljawarneh, M. Aldwairi, and M. B. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” Journal of Computational Science, vol. 25, pp. 152–160, 2018.

[13]. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” in ICISSP, 2018, pp. 108–116.

[14]. D. Aksu, S. Ustebay, M. A. Aydin, and T. Atmaca, “Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm,” in International Symposium on Computer and Information Sciences. Springer, 2018, pp. 141–149.

[15]. N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, “Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark,” IEEE Access, 2018.

[16]. P. A. A. Resende and A. C. Drummond, “Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling,” Security and Privacy, vol. 1, no. 4, p. e36, 2018.

[17]. C. Cortes and V. Vapnik, “Support-vector networks,” Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[18]. R. Shouval, O. Bondi, H. Mishan, A. Shimoni, R. Unger, and A. Nagler, “Application of machine learning algorithms for clinical predictive modeling: a data-mining approach in sct,” Bone marrow transplantation, vol. 49, no. 3, p. 332, 2014.