SDN based DDoS Detection using Machine Learning and Deep Learning Techniques

We are in the era of exponentially growing digitalization. The role of emerging information and communication technologies in our day-to-day life I undeniable. Moreover, it creates new security concerns. These systems are vulnerable to various cyber threats and attacks. The Software-Defined Network (SDN) is a new network paradigm promises more dynamic and efficiently manageable network architecture. The network intelligence is logically decoupled and centralized into the Control layer. The major benefit of SDN is that it separates the control and data planes, making the network more versatile and easier to manage. The entire system can be managed by a single distant computer known as the controller. Many business-related and industrialized enterprises are implementing SDN technology in their network environments.

The centralized structure of SDN brings new vulnerabilities to the system. Distributed Denial of Service (DDoS) are the most prevalent and sophisticated attack. DDoS attack tries to overwhelm the available resources of a victim from providing service for normal users, by sending massive malicious requests from a huge number of hijacked machines. DDoS attacks are easy to initiate, there have been a certain number of easily obtainable DDoS attack tools. DDoS attacks are hard to defend and it has strong destructiveness. Many companies such as Amazon, Facebook, Twitter, GitHub suffered DDoS and get big losses. So it is necessary to accurately detect DDoS attack and protect the system.

The first line of defense against these attacks are Network Intrusion detection System (NIDS). Artificial Intelligence is now commonly used in defensive measure. The Machine Learning algorithms such as SVM, Naïve bayes, Decision Tree, Random Forest are used for the DDoS detection. The Deep Learning Techniques such as DNN, Auto Encoders, CNN, LSTM , TCN can significantly improve the performance of DDoS detection systems.

Proposed Techniques:-

- **Supervised Machine Learning Algorithms (Logistic Regression, Gaussian Naïve Bayes, SVM, Decision Tree, Random Forest, Gradient Boosting)**
- **Unsupervised Machine Learning Algorithm(K Means Clustering)**

- **Deep Neural Network (DNN)**
- **Auto Encoders (AE):**
- **Auto Encoder and XGBOOST Classifier:**
- **Convolutional Neural Network (CNN):**
- **Long Short Term Memory (LSTM):**
- **Temporal Convolutional Network (TCN):**

Dataset :

▪ DDoS attack SDN dataset– The SDN specific dataset created using the SDN architecture and includes UpToDate SDN DDoS traffic data.

Literature Survey :

NO	TITLE	TECHNIQUES USED	ADVANTAGES	DISADVANTAGES
1.	Machine Learning Approach Equipped with Neighborhood Component Analysis for DDoS Attack Detection in Software-Defined Networking [2021]	NCA KNN Decision Tree ANN SVM	NCA gives most relevant features by feature selection, UpToDate dataset	Does not give optimal number of features to be selected, The performance depends on the selected features
2.	Clustering based semi-supervised machine learning for DDoS attack classification [2019]	Agglomerative clustering K means clustering KNN, SVM, Random Forest	Optimizing and validating the model improves the performance	Clustering approach leads to high false positive values

3.	Detection of DDoS attacks with feed forward based deep neural network model [2021]	DNN	High accuracy	Failed to detect adversarial attack
4	A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks [2020]	RNN LSTM CNN RNN+LSTM	Improved accuracy Minimal computational complexity	Failed to detect adversarial DDoS attacks
5	DDoSNet: A Deep-Learning Model for Detecting Network Attacks [2020]	RNN AutoEncoder	DDoSNet gives the highest evaluation metrics in terms of recall, precision, F-score, and accuracy compared to the existing well known classical ML techniques	Vanishing gradient problem

Design Steps :

- ❑ Preprocessing of Dataset
- ❑ Implement Supervised Machine Learning models for DDoS detection
- ❑ Implement Clustering Algorithm (K MEANS) for DDoS detection
- ❑ Implement Deep Neural Network for DDoS detection
- ❑ Implement Auto Encoder for the DDoS detection
- ❑ Implement Auto Encoder for feature extraction and XGBOOST classifier for DDoS detection

- ❑ Implement CNN for DDoS detection
- ❑ Implement LSTM for the DDoS detection
- ❑ Implement TCN for DDoS detection
- ❑ Performance evaluation

Framework : Keras and tensorflow

Detailed Design Steps:-

- **Dataset Preprocessing:**

- **DDoS Attack SDN Dataset:**

This is a SDN specific data set generated by using mininet emulator and used for traffic classification by machine learning and deep learning algorithms. The project start by creating ten topologies in mininet in which switches are connected to single Ryu controller. Network simulation runs for benign TCP, UDP and ICMP traffic and malicious traffic which is the collection of TCP Syn attack, UDP Flood attack, ICMP attack. Total 23 features are available in the data set in which some are extracted from the switches and others are calculated. Extracted features include Switch-id, Packet_count, byte_count, duration_sec, duration_nsec which is duration in nano-seconds, total duration is sum of duration_sec and durstaion_nsec, Source IP, Destination IP, Port number, tx_bytes is the number of bytes transferred from the switch port, rx_bytes is the number of bytes received on the switch port. dt field show the date and time which has been converted into number and a flow is monitored at a monitoring interval of 30 second. Calculated features include Packet per flow which is packet count during a single flow, Byte per flow is byte count during a single flow, Packet Rate is number of packets send per second and calculated by dividing the packet per flow by monitoring interval, number of Packet_ins messages, total flow entries in the switch, tx_kbps, rx_kbps are data transfer and receiving rate and Port Bandwidth is the sum of tx_kbps and rx_kbps. Last column indicates the class label which indicates

whether the traffic type is benign or malicious. Benign traffic has label 0 and malicious traffic has label 1. Network simulation is run for 250 minutes and 1,04,345 rows of data is collected.

Feature Name	**Description**
dt	Transmission moment of packets over the network device
switch	Switch ID
src	IP address of the sender of the packets
dst	IP address to which the packets was sent
pktcount	Number of packets
bytecount	Number of bytes
dur	Duration
dur_nsec	Duration in nano seconds
tot_dur	Total duration of network flow
flows	Number of flow packets
packetins	Total flow entries in the switch
pktperflow	Packet count during a single flow
byteperflow	Byte count during a single flow
pktrate	Number of packets per sec
Pairflow	Number of flow packets per second

Protocol	Types of communications internet protocols
port_no	Port number of the sender of the packets
tx_bytes	Number of bytes transferred from the switch port
rx_bytes	Number of bytes received on the switch port
tx_kbps	Data transfer rate
rx_kbps	Data Receiving rate
tot_kbps	Sum of tx_kbps and rx_kbps

- **DDoS Detection Using Supervised Machine Learning Algorithms**
  - Load the dataset
  - Splitting the dataset into features and label
  - Splitting the dataset in to training and testing (40% for testing and 60% for training)
  - First create a Machine Learning model using **Logistic Regression** for DDoS detection
  - Evaluate the logistic Regression model
  - Create a Machine Learning model using **Gaussian Naïve bayes** algorithm
  - The Gaussian Naïve bayes model is trained and tested in the dataset
  - Evaluate the performance of Gaussian Naïve bayes model
  - Create a Machine Learning model using **Support Vector Machine (SVM)** algorithm
  - Train the SVM model on the dataset and test the model
  - Evaluate the performance of SVM model for DDoS detection
  - Create a Machine Learning model using **Decision Tree** algorithm
  - Train the Decision Tree model in the dataset and test the model
  - Evaluate the performance of Decision Tree model for DDoS detection
  - Create a Machine Learning model using **Random Forest** algorithm

- ❖ Train the Random Forest model in the dataset and test the model
- ❖ Evaluate the performance of Random Forest model for DDoS detection
- ❖ Create a Machine Learning model using **Gradient Boosting** algorithm
- ❖ Train the Random Forest model in the dataset and test the model
- ❖ Evaluate the performance of Random Forest model for DDoS detection
  - ❑ Results:

Evaluation Metrics	LR	GNB	SVM	DT	RF	GB
Accuracy (%)	77.11	67.35	78.44	99.95	100	98.52
Precision (%)	72.68	57.54	76.16	99.93	100	100
Recall (%)	66.29	62.41	65.17	99.95	100	100
F1 score (%)	75.54	66.17	76.67	99.95	99.95	98.46

- **K Means Clustering for DDoS Detection:**
  - **K Means Clustering with PCA & TSNE**
    - ❖ K Means attempts to organize the data into a specified number of clusters. The goal of K Means is to identify similar data points and cluster them together while trying to distance each cluster as far as possible. Its "similarity" calculation is determined via Euclidean distance or an ordinary straight line between two points. K Means is very sensitive to scale and requires all features to be on the same scale. K Means will put more weight or emphasis on features with larger variances and those features will impose more influence on the final cluster shape. Clustering algorithms such as K Means have a difficult time accurately clustering data of high dimensionality (ie. too many features).
    - ❖ Compare PCA and t-SNE data reduction techniques prior to running K-Means clustering algorithm
    - ❖ **Principal component analysis or (PCA)** is a classic method we can use to reduce high-dimensional data to a low-dimensional space. The beauty behind PCA is the fact that despite the reduction into a lower-dimensional space we still retain most (+90%) of the variance or information from our original high-dimensional dataset. The information or variance from our original features is "squeezed" into what PCA calls principal components (PC). The first PC will contain the majority of the information from the original features. The second PC will contain the

next largest amount of information, the 3rd PC the third largest amount of info and so on and so on. The PC are not correlated (ie. orthogonal) which means they all contain unique pieces of information.

- **T-Distributed Stochastic Neighbor Embedding (t-SNE) :** t-SNE takes high-dimensional data and reduces it to a low-dimensional graph (2-D typically). It is also a great dimensionality reduction technique. Unlike PCA, t-SNE can reduce dimensions with non-linear relationships. In other words, if our data had this "Swiss Roll" non-linear distribution where the change in X or Y does not correspond with a constant change in the other variable. PCA would not be able to accurately distill this data into principal components. This is because PCA would attempt to draw the best fitting line through the distribution. T-SNE would be a better solution in this case because it calculates a similarity measure based on the distance between points instead of trying to maximize variance.
- T-SNE looks at the similarity between local or nearby points by observing the distance (Euclidean distance). Points that are nearby each other are considered similar. t-SNE then converts this similarity distance for each pair of points into a probability for each pair of points. If two points are close to each other in the high-dimensional space they will have a high probability value and vice versa. This way the probability of picking a set of points is proportional to their similarity. Then each point gets randomly projected into a low dimensional space. t-SNE is a visualization tool first and a dimensionality reduction tool second.
- We need to understand how well K-Means managed to cluster our data. **Silhouette Method:** This technique measures the separability between clusters. First, an average distance is found between each point and all other points in a cluster. Then it measures the distance between each point and each point in other clusters. We subtract the two average measures and divide by whichever average is larger. We ultimately want a high (ie. closest to 1) score which would indicate that there is a small intra-cluster average distance (tight clusters) and a large inter-cluster average distance (clusters well separated).

❖ Standardize the data : The standardization of data will ultimately bring all features to the same scale and bringing the mean to zero and the standard deviation to 1.

❖ Applying K Means on the original dataset : apply K Means on the original dataset requesting 2 clusters. We achieved a silhouette score of 0.196 which is on the low end.

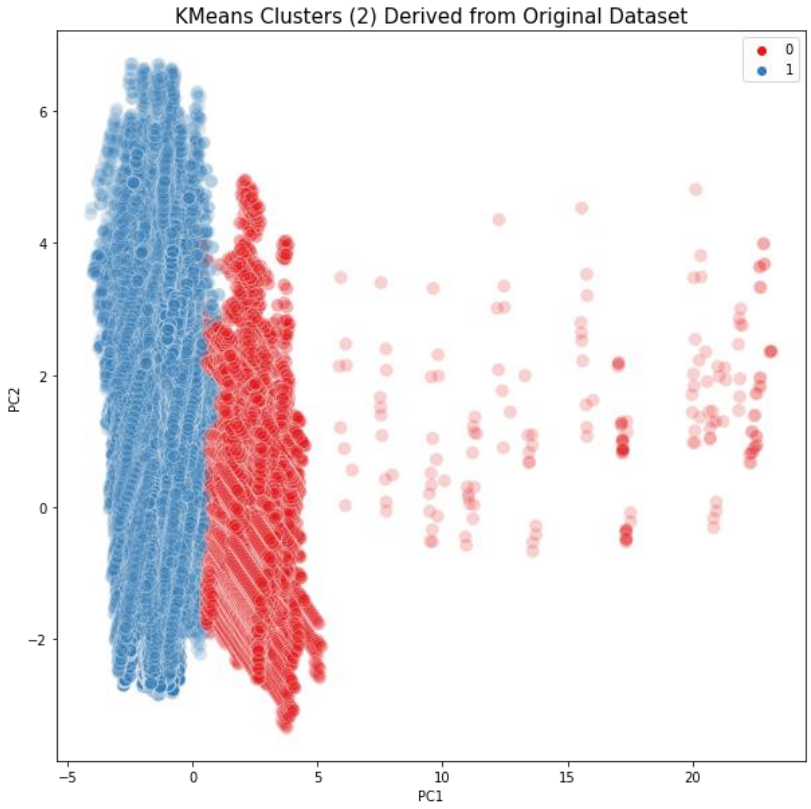
❖ Feature Reduction via PCA : Using PCA to reduce the dataset into 2 principal components we can plot the K Means derived clusters

❖ Applying K Means to PCA principal components : Now that we have reduced the original dataset of 19 features to just 2 principal components let's apply the K Means algorithm. We can see a definite improvement in K Means ability to cluster our data when we reduce the number of dimensions to 2 principal components. K Means PCA Scaled Silhouette Score: 0.4613677492070967

❖ Feature Reduction via t-SNE : In this section we will reduce our data once again using t-SNE and compare K Means results to that of PCA K Means. We will reduce down to 2 t-SNE components. t-SNE is a computationally heavy algorithm. Computational time can be reduced using the 'n_iter' parameter.

❖ Applying K Means to t-SNE clusters : Applying KMeans to our 2 t-SNE derived components we were able to obtain a Silhouette score of 0.3413. If we recall the Silhouette score obtained from K Means on PCA's 2 principal components was 0.4613

❖Comparing PCA and t-SNE K Means derived clusters : First merge the K Means clusters with the original unscaled features. We'll create two separate data frames. One for the PCA derives K Means clusters and one for the t-SNE K Means clusters. Obtain univariate review of the clusters by comparing the clusters based on each individual feature.



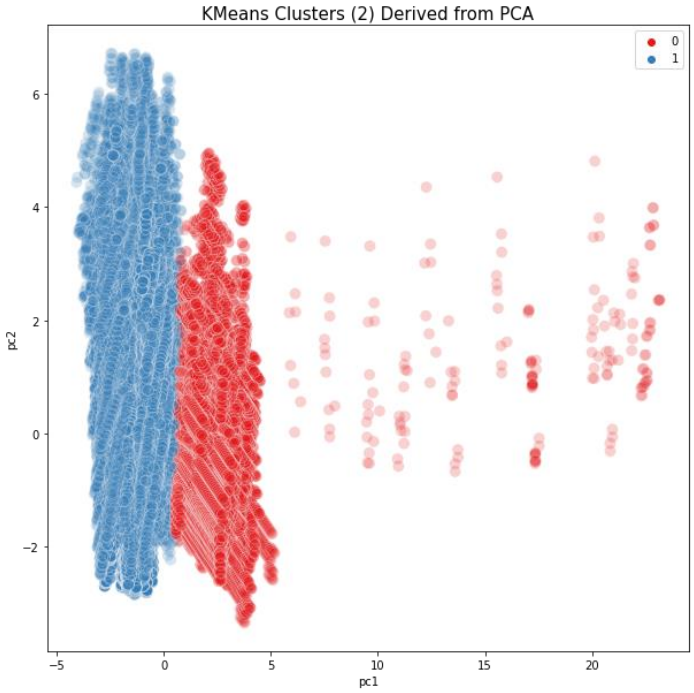

```
[ ]  df_scale2 = df_scale.copy()
     kmeans_scale = KMeans(n_clusters=2, n_init=100, max_iter=400, init='k-means++', random_state=42).fit(df_scale2)
     print('KMeans Scaled Silhouette Score: {}'.format(silhouette_score(df_scale2, kmeans_scale.labels_, metric='euclidean')))
     labels_scale = kmeans_scale.labels_
     clusters_scale = pd.concat([df_scale2, pd.DataFrame({'cluster_scaled':labels_scale})], axis=1)

     KMeans Scaled Silhouette Score: 0.1961007339513506
```

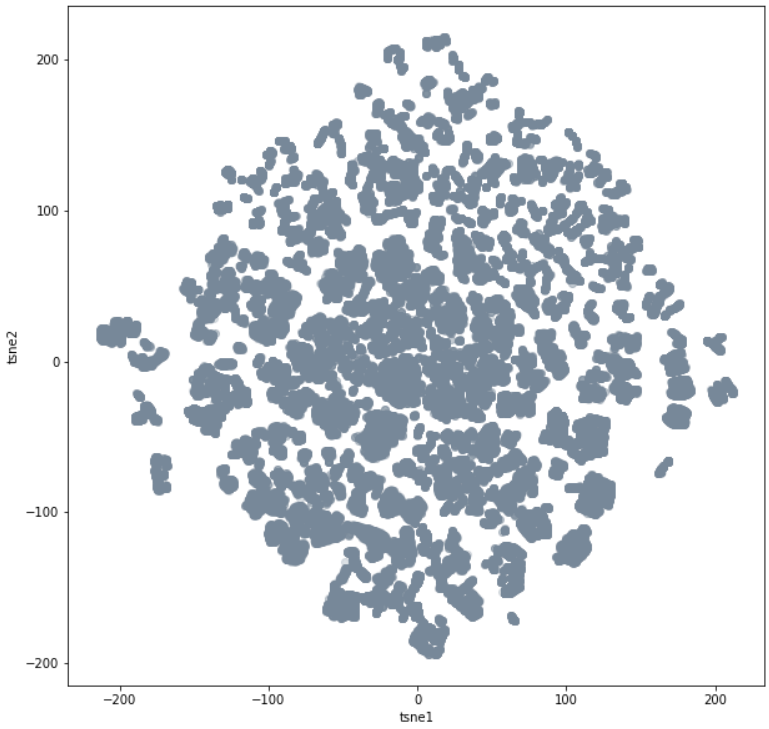
```
[ ]  pca = PCA(n_components=2)
     pca_scale = pca.fit_transform(df_scale)
     pca_df_scale = pd.DataFrame(pca_scale, columns=['pc1','pc2'])
     print(pca.explained_variance_ratio_)
```

```
[0.23682893 0.13383368]
```

```
[ ]  kmeans_pca_scale = KMeans(n_clusters=2, n_init=100, max_iter=400, init='k-means++', random_state=42).fit(pca_df_scale)
     print('KMeans PCA Scaled Silhouette Score: {}'.format(silhouette_score(pca_df_scale, kmeans_pca_scale.labels_, metric='euclidean')))
     labels_pca_scale = kmeans_pca_scale.labels_
     clusters_pca_scale = pd.concat([pca_df_scale, pd.DataFrame({'pca_clusters':labels_pca_scale})], axis=1)
```

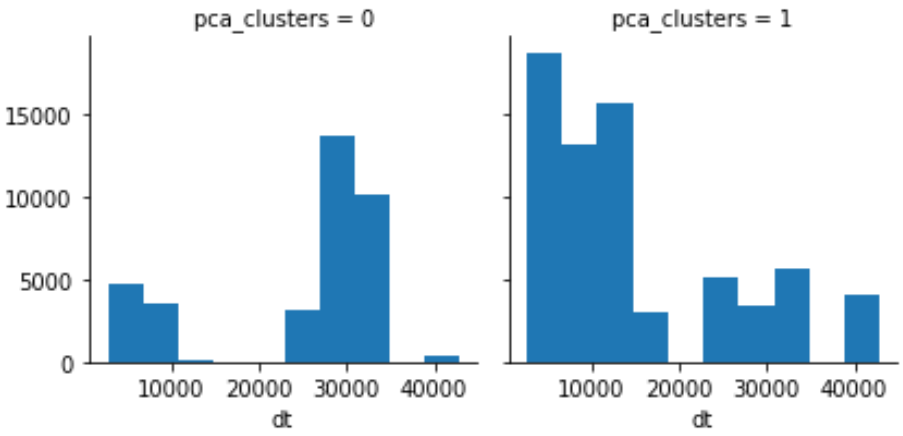
```
KMeans PCA Scaled Silhouette Score: 0.4613677492070967
```

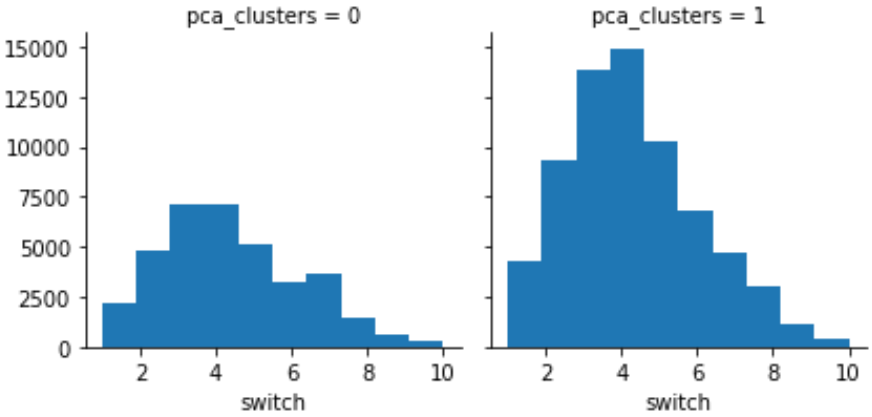


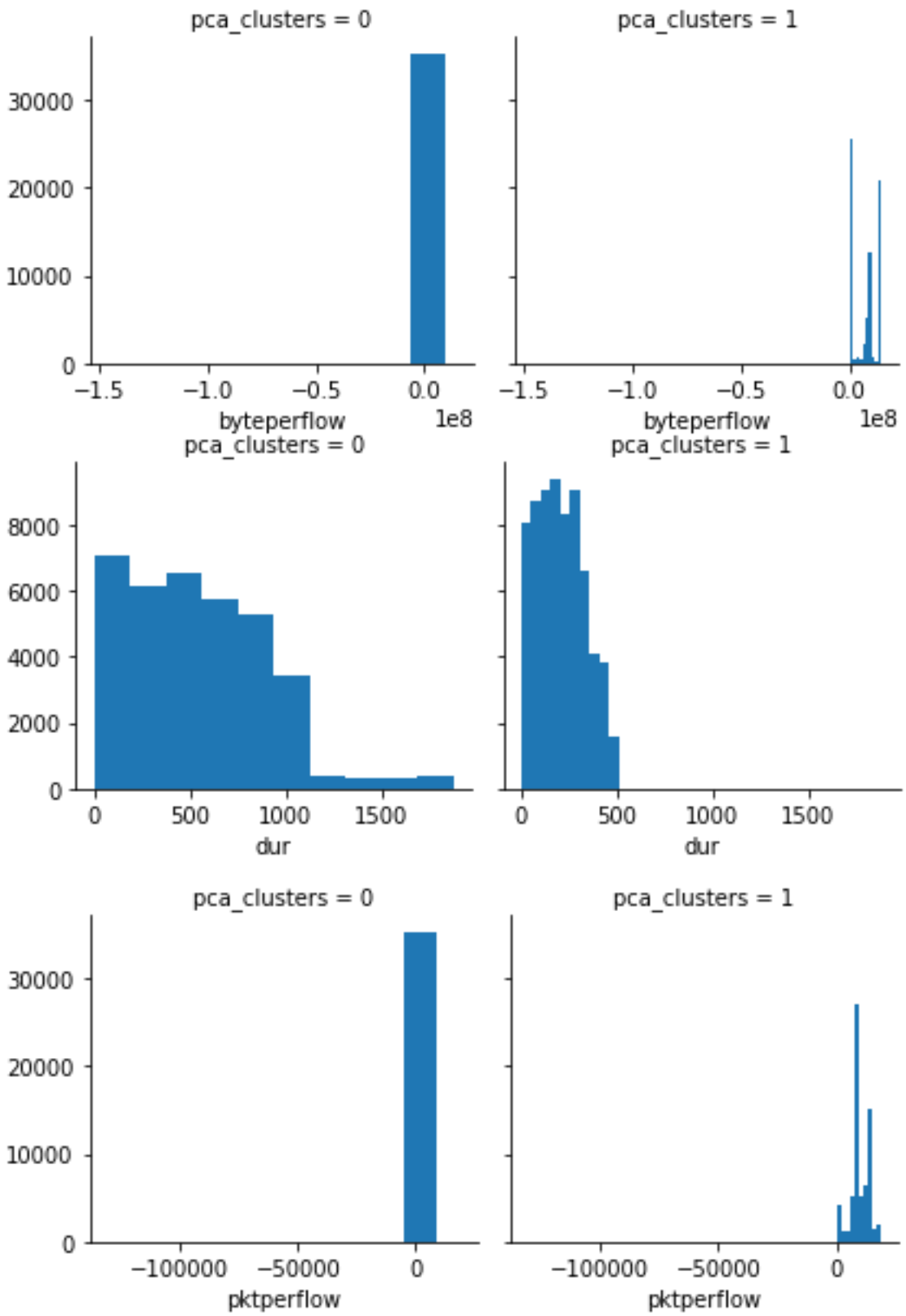


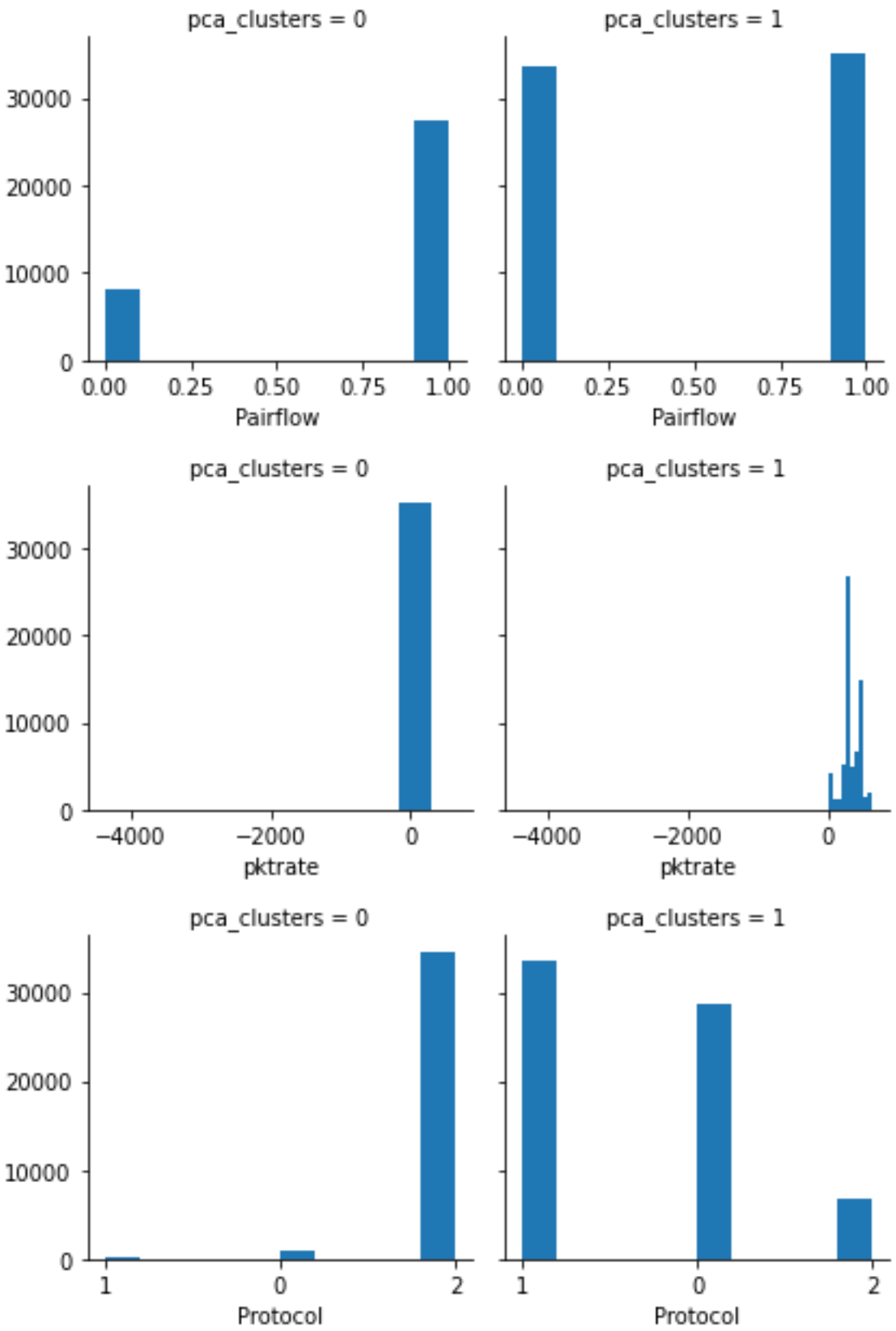

```
[ ]  kmeans_tsne_scale = KMeans(n_clusters=2, n_init=100, max_iter=400, init='k-means++', random_state=42).fit(tsne_df_scale)
     print('KMeans tSNE Scaled Silhouette Score: {}'.format(silhouette_score(tsne_df_scale, kmeans_tsne_scale.labels_, metric='euclidean')))
     labels_tsne_scale = kmeans_tsne_scale.labels_
     clusters_tsne_scale = pd.concat([tsne_df_scale, pd.DataFrame({'tsne_clusters':labels_tsne_scale})], axis=1)
```

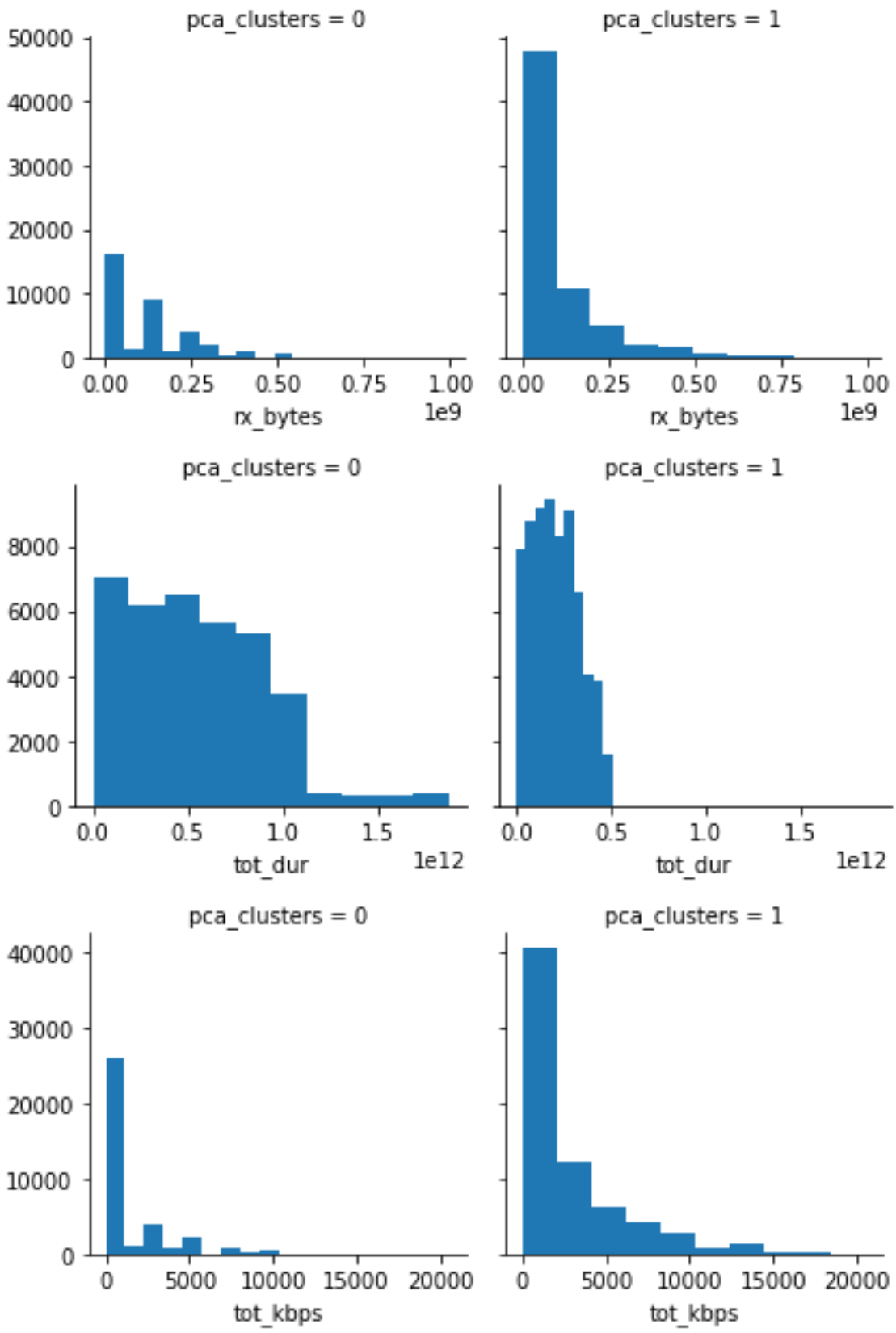
```
KMeans tSNE Scaled Silhouette Score: 0.34130406379699707
```

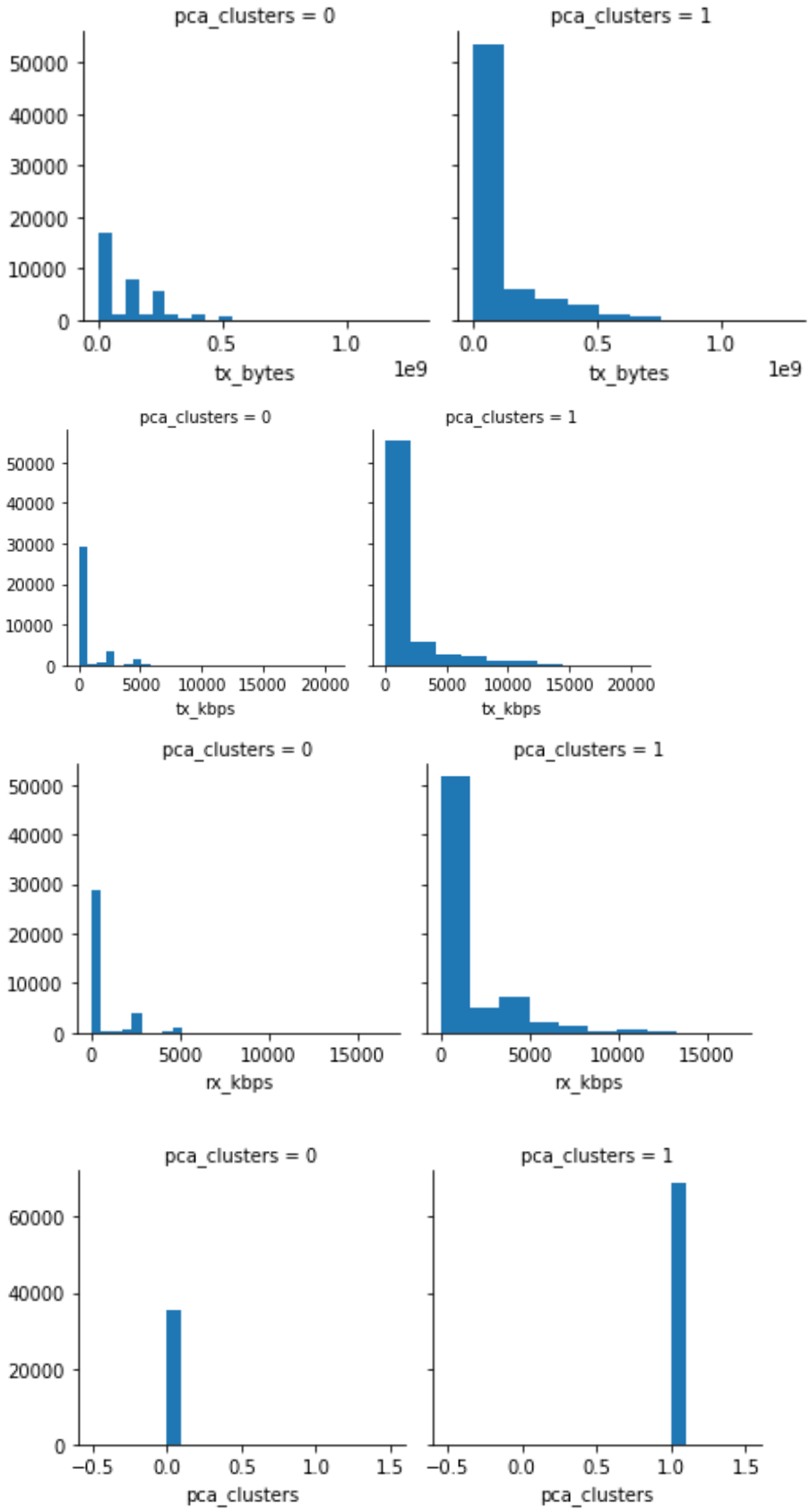

pca_clusters = 0
pca_clusters = 1
15000
10000
5000
0
10000
20000
30000
40000
dt

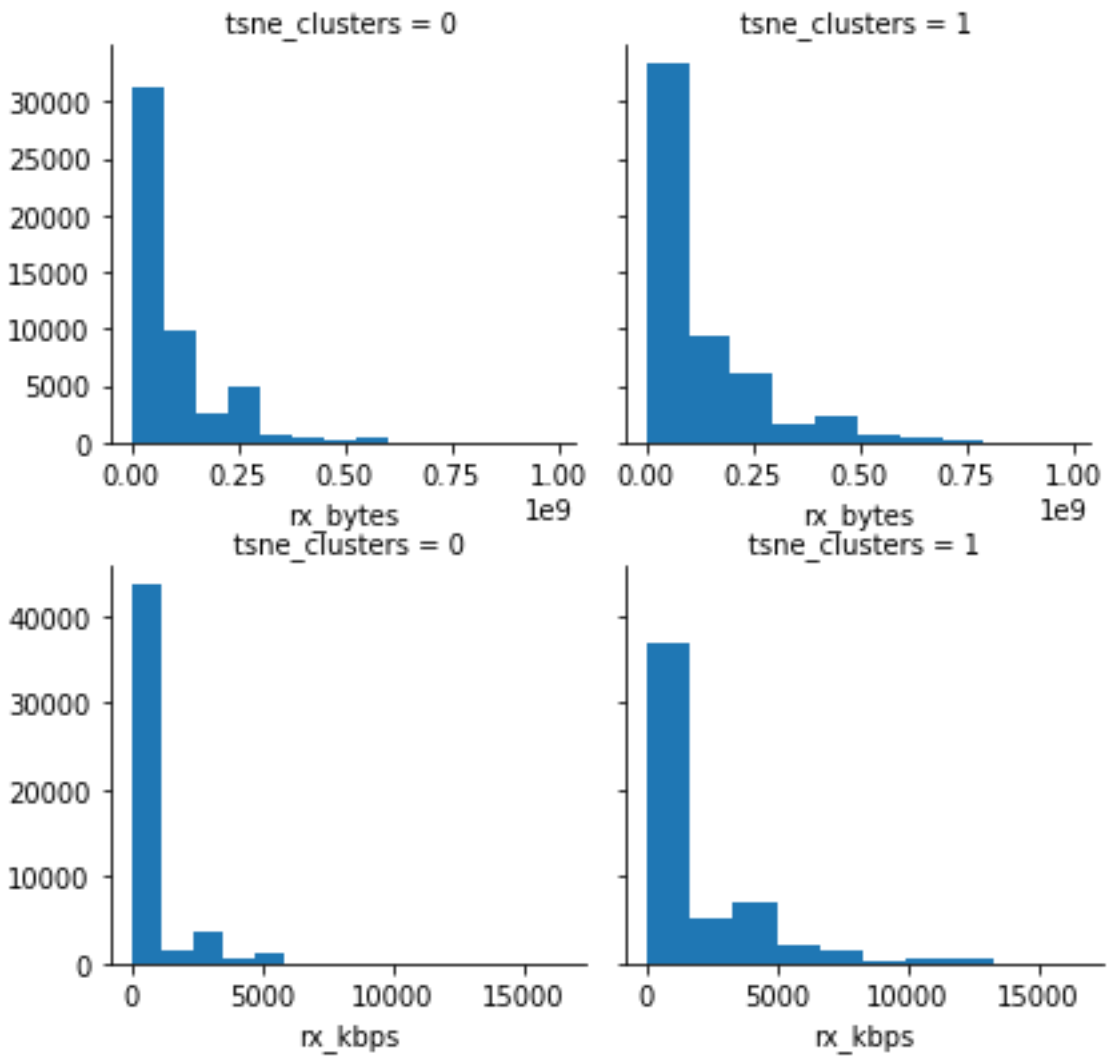

pca_clusters = 0
pca_clusters = 1
15000
12500
10000
7500
5000
2500
0
2
4
6
8
10
switch

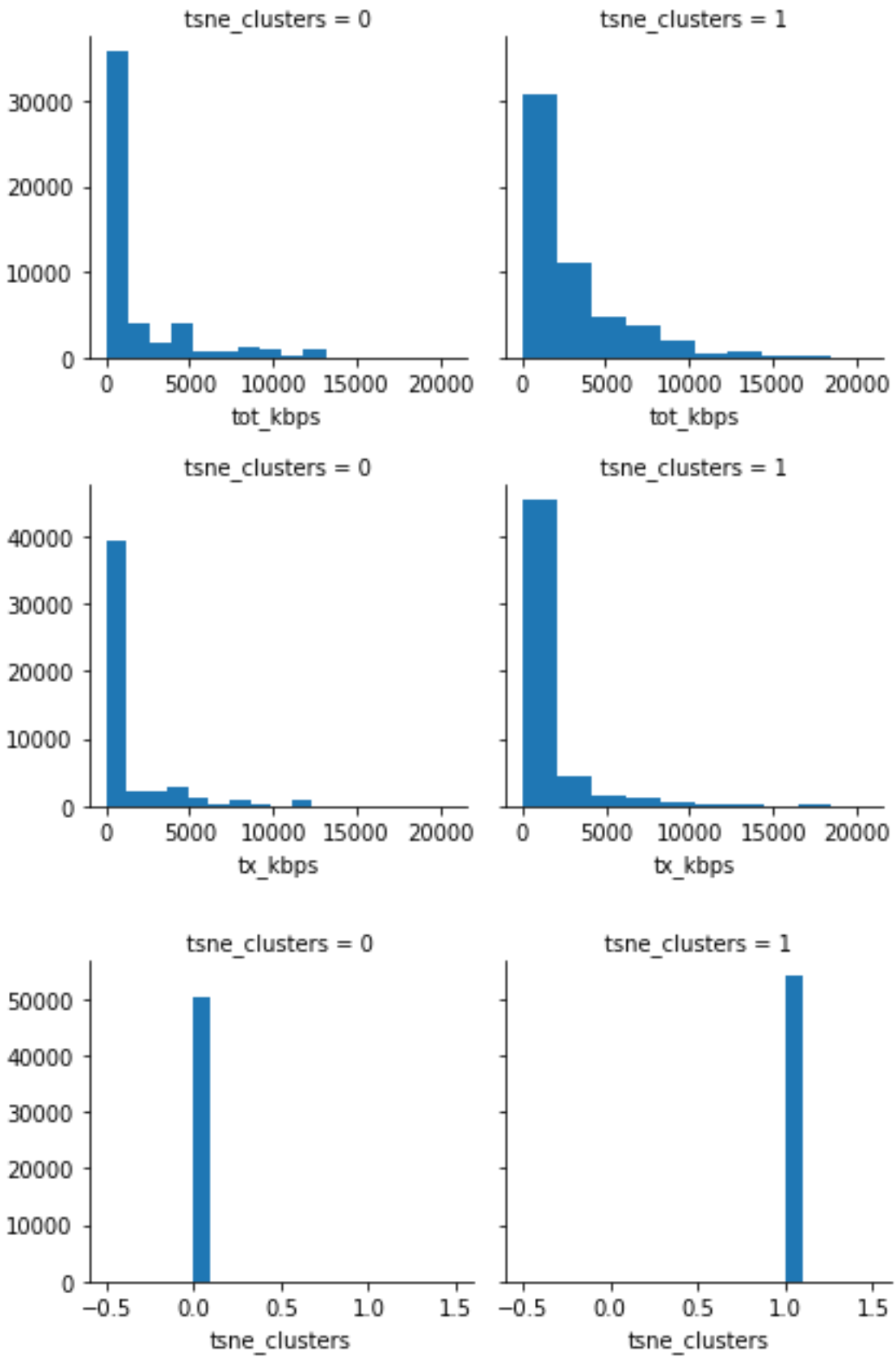

pca_clusters = 0
pca_clusters = 1
30000
20000
10000
0
−1.5
−1.0
−0.5
0.0
byteperflow
1e8
pca_clusters = 0
pca_clusters = 1
8000
6000
4000
2000
0
0
500
1000
1500
dur
pca_clusters = 0
pca_clusters = 1
30000
20000
10000
0
−100000
−50000
0
pktperflow

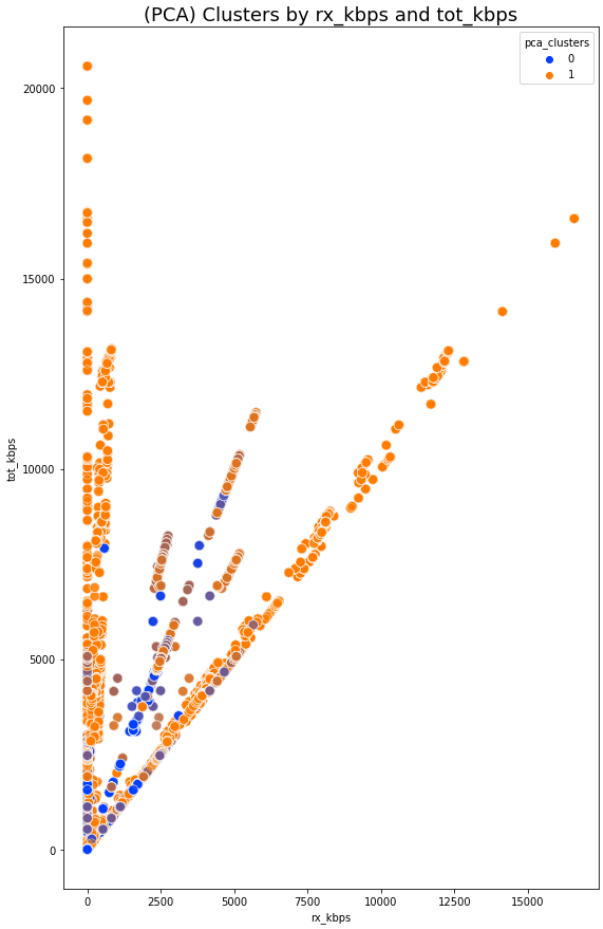

pca_clusters = 0
pca_clusters = 1
30000
20000
10000
0
0.00
0.25
0.50
0.75
1.00
Pairflow
−4000
−2000
0
pktrate
1
0
2
Protocol

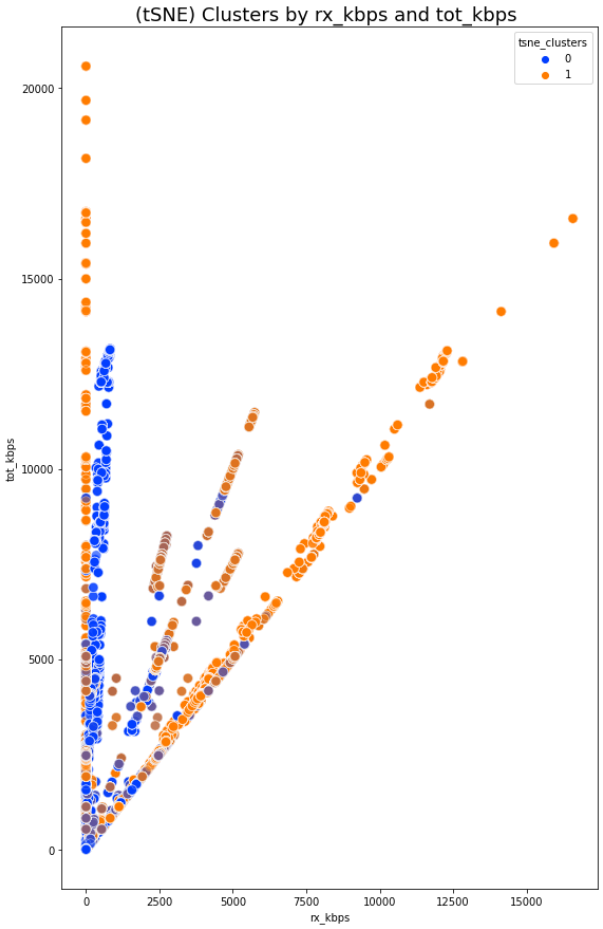

pca_clusters = 0
pca_clusters = 1
50000
40000
30000
20000
10000
0
0.00
0.25
0.50
0.75
1.00
rx_bytes
1e9
8000
6000
4000
2000
0.0
0.5
1.0
1.5
tot_dur
1e12
40000
30000
20000
10000
0
5000
10000
15000
20000
tot_kbps


pca_clusters = 0
pca_clusters = 1
50000
40000
30000
20000
10000
0
0.0
0.5
1.0
tx_bytes
1e9
pca_clusters = 0
pca_clusters = 1
50000
40000
30000
20000
10000
0
0
5000
10000
15000
20000
tx_kbps
pca_clusters = 0
pca_clusters = 1
50000
40000
30000
20000
10000
0
0
5000
10000
15000
rx_kbps
pca_clusters = 0
pca_clusters = 1
60000
40000
20000
0
−0.5
0.0
0.5
1.0
1.5
pca_clusters


tsne_clusters = 0
tsne_clusters = 1
30000
25000
20000
15000
10000
5000
0
0.00
0.25
0.50
0.75
1.00
1e9
rx_bytes
tsne_clusters = 0
tsne_clusters = 1
40000
30000
20000
10000
0
0
5000
10000
15000
rx_kbps

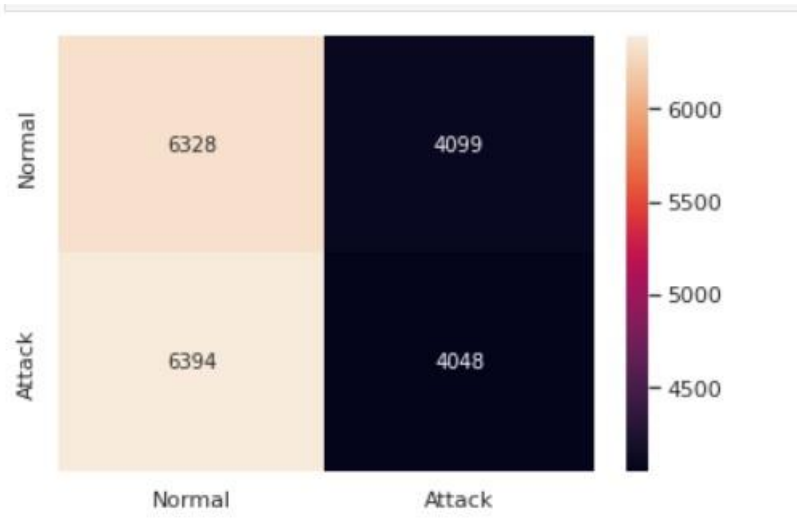

tsne_clusters = 0
tsne_clusters = 1
30000
20000
10000
0
0
5000
10000
15000
20000
tot_kbps
tsne_clusters = 0
tsne_clusters = 1
40000
30000
20000
10000
0
0
5000
10000
15000
20000
tx_kbps
tsne_clusters = 0
tsne_clusters = 1
50000
40000
30000
20000
10000
0
−0.5
0.0
0.5
1.0
1.5
tsne_clusters

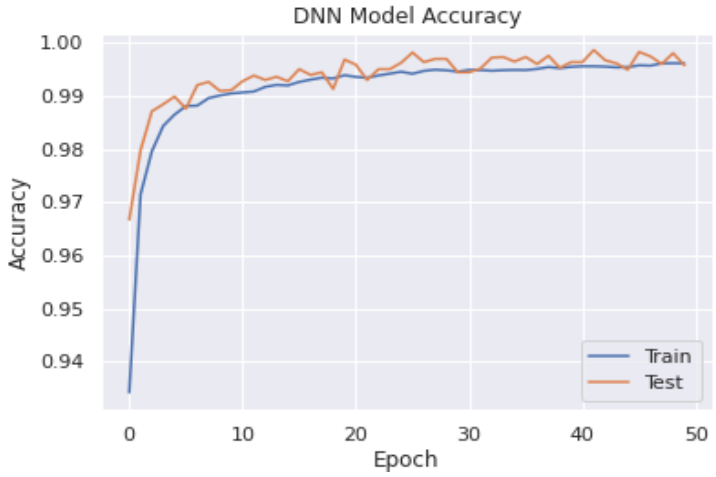

(PCA) Clusters by rx_kbps and tot_kbps
pca_clusters
0
1
tot_kbps
rx_kbps
0
5000
10000
15000
20000
0
2500
5000
7500
10000
12500
15000

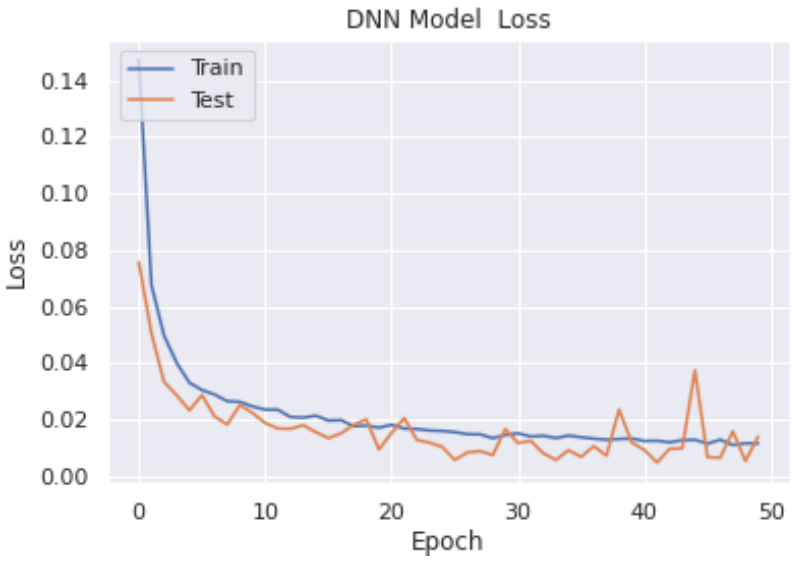

(tSNE) Clusters by rx_kbps and tot_kbps
tsne_clusters
0
1
tot_kbps
rx_kbps
0
5000
10000
15000
20000
0
2500
5000
7500
10000
12500
15000

- **Deep Neural Network (DNN) for DDoS Detection**

  - Load the dataset
  - Splitting the dataset into features and label
  - Splitting the dataset in to training and testing (20% for testing and 80% for training)
  - Feature Scaling or Standardization using Standard Scaler
  - Create DNN with sequential model having input layer, 3 hidden layer and output layer
  - The input layer consist of 22 neurons corresponds to 22 features selected. The relu activation function used
  - The hidden layers consist of equal number of 30 neurons and relu activation function is used
  - There are also two dropout layers with dropout ratio 0.2
  - The output layer consists of 2 neurons with sigmoid activation function because it is a binary classification problem
  - The output label '0' indicates the network traffic is normal and '1' indicates the traffic is DDoS attack
  - Compile the model using 'Sparse categorical cross entropy' loss function, 'accuracy'` metrics and 'adam' optimizer
  - Fit the model with 50 epochs and batch size selected is 16
  - Plot the training and validation accuracy values
  - Plot the training and validation loss values
  - Make predictions on the test set
  - Obtain confusion matrix
  - Evaluate the model and obtain the accuracy
    - Result:- The accuracy obtained is **99.39 %**


Normal
Attack
6328
4099
6394
4048
Normal
Attack
6000
5500
5000
4500


DNN Model Accuracy
Accuracy
Epoch
Train
Test
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0
10
20
30
40
50


DNN Model Loss
Loss
Epoch
Train
Test
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
10
20
30
40
50

- **Auto Encoder (AE) for DDoS Detection**

The Auto Encoder accepts high dimensional input data, compresses down to the latent space representation in the bottleneck hidden layer. The Decoder takes the latent representation of the data as an input to reconstruct the original input data. AE tries to minimize the reconstruction error part of its training. Anomalies are detected by checking the magnitude of the reconstruction loss.

- ❖ Import the required libraries and load the dataset
- ❖ The dataset consists of label '0' and '1', 0 denotes normal and 1 denotes the attack (Anomaly)
- ❖ Split the data for training and testing (20% for testing)
- ❖ Scale the data using Minmax scaler
- ❖ Use normal data only for training

  AE are trained to minimize the reconstruction error. The reconstruction errors are used as the anomaly score. When we train the AE on normal data, we can hypothesize that the anomalies will have higher reconstruction errors than normal data.
- ❖ Create an Auto Encoder class with output units equal to number of input data and the number of units in the bottleneck (code size) equal to 16
- ❖ The encoder of the model consists of 4 layers that encode the data into lower dimensions
- ❖ The decoder of the model consists of 4 layers that reconstruct the input data
- ❖ The model is compiled with metrics equal to mse and adam optimizer
- ❖ The model is trained with 200 epochs with a batch size of 16
- ❖ The reconstruction errors are considered as anomaly score. The Threshold is then calculated by summing the mean and standard deviation of the reconstruction errors
- ❖ The reconstruction error above this threshold is DDoS attack (anomalies)
  - ❑ Result : Accuracy obtained is **68.27 %**
- ❖ Optimize the model with keras tuner

- Accuracy improved to **71.39 %**

- **Auto Encoder and XGBOOST for DDoS Detection**

  Auto Encoder is a type of neural network that can be used to learn a compressed representation of raw data. An autoencoder is composed of an encoder and a decoder sub-model. The encoder compresses the input, and the decoder attempts to recreate the input from the compressed version provided by the encoder. After training, the encoder model is saved, and the decoder is discarded. The encoder can then be used as a data preparation technique to perform feature extraction on raw data that can be used to train a different machine learning model.

  - Importing the required libraries
  - Load the dataset
  - Split the dataset in to train and test
  - Scale the dataset using Min Max scaler
  - Define the Auto Encoder model with encoder model, bottleneck, and decoder model
  - Fit the autoencoder model to reconstruct input
  - Define and save the encoder model without decoder
  - Compress the input data using encoder model
  - Import the XGBoost classifier
  - Encode the train data
  - Encode the test data
  - Fit the XGBoost classifier model on the training set
  - Make predictions on the test data
  - Calculate classification accuracy
    - Result :

Evaluation Metric	Obtained Value
Accuracy	97.58 %
Precision	97.41 %
Recall	96.25 %
F1 score	97.43 %

- **CNN for DDoS Detection :**

  Deep learning is a form of machine learning and CNN is a type of deep neural network. CNN has proven to be effective in many various studies and applications specifically in image classification field. Any CNN consists of multiples layers: input layer, convolutional layers, pooling layers, fully connected layer and output layer, the deepness of the CNN dependence on the number of layers used.
  1D CNN is used for the DDoS detection in SDN environment.

  - ❖ Import the required libraries
  - ❖ Load the dataset
  - ❖ Split the data for training and testing (20% for testing)
  - ❖ Standardize the data using Standard Scaler
  - ❖ Create the CNN with sequential model
  - ❖ The CNN model consist of two Conv1D layer, Batch Normalization layer, Max pooling layer, Dropout layer, Flatten layer and dense layer. The relu activation function used
  - ❖ The output layer having softmax activation and 2 neurons corresponds to 2 output classes
  - ❖ The model is compiled with sparse categorical cross entropy and adam optimizer
  - ❖ Fit the CNN model with 20 epochs and batch size=16
  - ❖ Plot the CNN model loss and CNN model accuracy
    Make predictions using the trained CNN model
  - ❖ Plot the Confusion matrix
    - ❑ Result : The accuracy obtained is **99.50%**

- **LSTM for DDoS Detection**

  Long-term and short-term memory model is a special RNN model, which is proposed to solve the problem of gradient dispersion of RNN model. Among them, the internal state of RNN network can show the dynamic sequential behavior. Unlike feedforward neural networks, RNN can use its internal memory to process input sequences of arbitrary time series, which makes it easier to process such as non-segmented handwriting recognition, speech recognition and so on. However, RNN has two problems, namely,

gradient disappearance and gradient explosion. The original intention of LSTM design is to solve the problem of long-term dependence in RNN, so that remembering long-term information becomes the default behavior of neural network, rather than a lot of effort to learn. LSTM model replaces RNN cells in the hidden layer with LSTM cells to make them have long-term memory ability.

- ❖ Import the required libraries
- ❖ Load the dataset
- ❖ Split the dataset in to training and testing
- ❖ Standardize the data using standard scalar
- ❖ Reshape the inputs for LSTM (samples, time steps, features)
- ❖ Create the LSTM with sequential model
- ❖ The LSTM architecture consist of one LSTM layer with 22 neurons and tanh activation function, the kernel_regularizer is l2
- ❖ The Dense layer having relu activation function and l2 regularizer
- ❖ The output layer having sigmoid activation function
- ❖ Compile the model with binary cross entropy loss function and adam optimizer
- ❖ Fit the model with 50 epochs
- ❖ Plot the LSTM model loss and LSTM model accuracy
- ❖ Plot the Confusion matrix
- ❑ Result :The accuracy obtained is **94.37%**

- **Temporal Convolutional Network (TCN) for DDoS Detection**

Temporal Convolutional Networks, or simply TCN, is a variation of Convolutional Neural Networks for sequence modelling tasks, by combining aspects of RNN and CNN architectures. Preliminary empirical evaluations of TCNs have shown that a simple convolutional architecture outperforms canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets while demonstrating longer effective memory. The distinguishing characteristics of TCNs are

- The convolutions in the architecture are causal, meaning that there is no information “leakage” from future to past.

- The architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN. TCNs possess very long effective history sizes (i.e., the ability for the networks to look very far into the past to make a prediction) using a combination of very deep networks (augmented with residual layers) and dilated convolutions.

The TCN is based upon two principles: the fact that the network produces an output of the same length as the input, and the fact that there can be no leakage from the future into the past. To accomplish the first point, the TCN uses a 1D fully-convolutional network (FCN) architecture, where each hidden layer is the same length as the input layer, and zero padding of length (kernel size − 1) is added to keep subsequent layers the same length as previous ones. To achieve the second point, the TCN uses *causal convolutions*, convolutions where output at time *t* is convolved only with elements from time *t* and earlier in the previous layer.

- ❖ Import the required libraries
- ❖ Load the dataset
- ❖ Split the dataset for training and testing
- ❖ Standardize the dataset using Standard scalar
- ❖ Create the TCN model from keras tcn
- ❖ The relu activation is used, no of filters=22, kernel_initializer is he_normal, dilations used [2,4,8],number of stacks is 1, kernel size used is 6 and use skip connections
- ❖ Fit the model with 20 epochs
- ❖ Plot the TCN model loss and tcn model accuracy
- ❖ Make prediction using trained TCN model
- ❖ Plot the confusion matrix
- ❑ Result : The accuracy obtained is **99.65%**

References:-

[1]. Elsayed, M.S., Le-Khac, N.A., Dev, S. and Jurcut, A.D., 2020, August. Ddosnet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)* (pp. 391-396). IEEE.

[2]. Cil, A.E., Yildiz, K. and Buldu, A., 2021. Detection of DDoS attacks with feed forward based deep neural network model. *Expert Systems with Applications*, *169*, p.114520.

[3]. Haider, S., Akhunzada, A., Mustafa, I., Patel, T.B., Fernandez, A., Choo, K.K.R. and Iqbal, J., 2020. A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *Ieee Access*, *8*, pp.53972-53983.

[4]. Tonkal, Ö., Polat, H., Başaran, E., Cömert, Z. and Kocaoğlu, R., 2021. Machine Learning Approach Equipped with Neighbourhood Component Analysis for DDoS Attack Detection in Software-Defined Networking. *Electronics*, *10*(11), p.1227.

[5]. Catak, F.O. and Mustacoglu, A.F., 2019. Distributed denial of service attack detection using autoencoder and deep neural networks. *Journal of Intelligent & Fuzzy Systems*, *37*(3), pp.3969-3979.

Guide Details:

Name : Prof. Sumod Sundar
Email: sumodsundar@tkmce.ac.in