# *Data Analytics – Final Project Report*

***Name: Ajith Muthu (U91561222)***

***Title: Anomaly detection in Time Series Data***

## *INTRODUCTION*

Anomalies are the data points that stand out amongst other data points in the dataset. These data points do not confirm the normal behavior in the data, which says, they are deviating from the dataset's normal behavior patterns,

Outliers are the merely extreme datapoint of the dataset, they may or may not be the true prediction. They are the unlikely event given the improbability in our assumptions, also known as base distributions

The first step in data analysis that identifies the datapoint deviating from dataset's normal behavior is the Anomaly detection which is the cyber-attacks in my project.

As the prediction is made coordinative to time, the anomaly detected links its capability of precisely examining real Time series data. The time series data consists sequential values for every timestamp where each data point has 2 items, one for timestamp of the metric being measured and one for its value.

We detect which of these datapoints or attacks will be predicted over time, and this is Anomaly detection in time series data.

There are 3 types of time series Anomalies:

- Global outliers or point outliers:
  These outliers are detected at the border of dataset.

- Contextual outliers or conditional outliers:
  The values of these anomalies deviate from its data points in the same region of dataset. These are usual in time series data as they hold records of specific measure at known time. They are detected within global expectation but appears to be anomalous after some seasonal patterns.

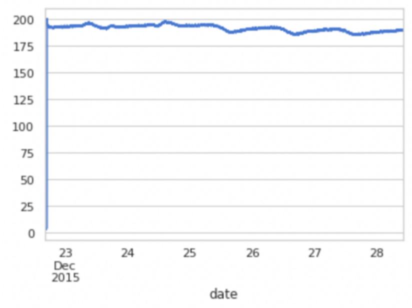
- Collective outlier:
  when a group of datapoints appears to be anomalous in the dataset with same pattern, these outliers are called collective outliers. The individual behavior varies from one dataset to other being trained or predicted, so more substantial anomalies get better.

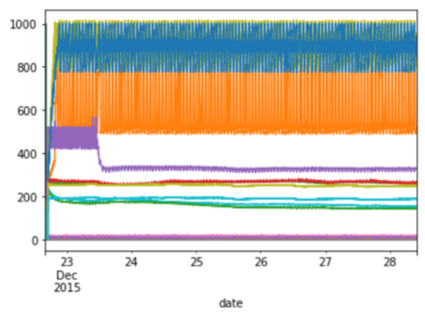
The objective of my project is to detect anomalies in time series data corresponding to Cyber-attacks in our dataset.

The project uses two data files:

- Training-Nominal.csv:

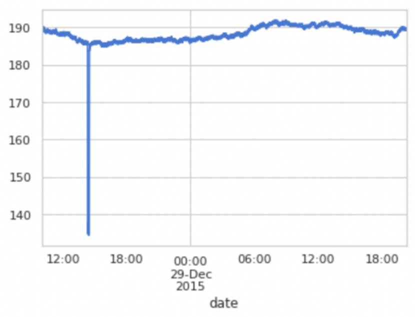
  The training dataset only consists of nominal instances, so my goal in training will be to develop an algorithm which can learn the nominal patterns and detect deviations from the nominal patterns, when predicted with other time series dataset, namely my test data set which are anomalous instances.
  The training dataset consists of 20 features or columns X1 to X20 and indexed with date column. Instances are recorded for ever one second from 22nd dec, 2015 4pm to 28th dec, 2015 10 AM, with 496799 timesteps or records for every feature.

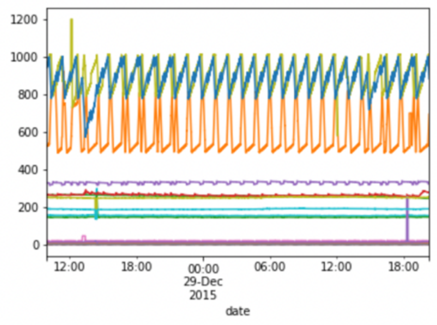




- Test.csv:

  Testing dataset will contain both nominal and anomalous instances which will detected by our model.
  To measure the performance of my model, I evaluated the performance metrics to find, False positives and False negatives i.e., the measure of false alarm and missed anomalies. The performance is visualized with ROC curve with AUC score and F1 score. In addition to this, I have measured the accuracy, precession and recall value of my prediction and confusion matrix representing the True positives, False Negatives, True Negatives and False positives of the prediction.

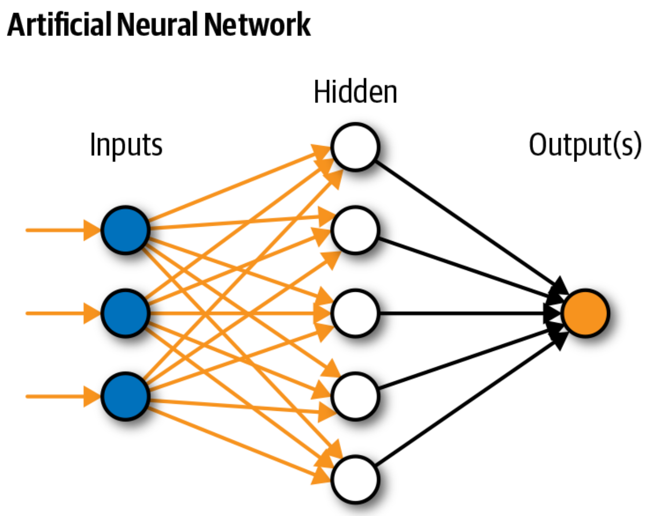




On context to Analysis, I have built a recurrent neural network (RNN) model for sequence prediction. Training data set is used for training the model with non-anomalous data instances and the same model is used to predict on test data. The model reconstructs the data instances to its initial structure using LSTM Auto Encoders. The anomalous data that show are the anomalies in our time series data.

## *PERFORMED ANALYSES*

For prediction I have used single feature of the dataset, X20. The model algorithm is built using Recurrent Neural network.
Key performance of a neural network is to reflect the behavior of the human brain, allowing the computer programs or machines to recognize and resolve the problems in the field of Intelligence. It can be Machine Learning, Deep Learning, or AI. Neural networks are also called Artificial neural networks (ANN). ANN is comprised of an input layer and an output layer with one or more hidden layers in between.

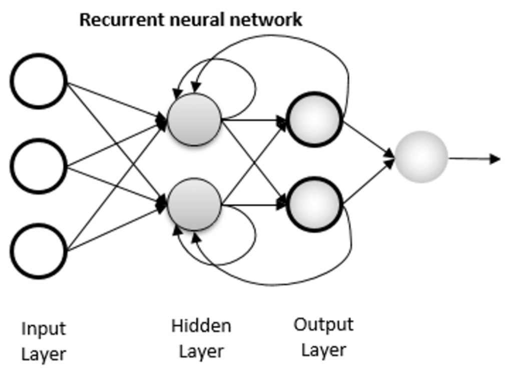


**Neural networks c**onduct its tasks of predictions using its AI system via algorithms. These algorithms predict outputs depending on the inputs provided. The Algorithms can be Regression or Classification. In our model we use classification to detect the anomalous data.

**RNN**: *Recurrent neural network* is an ANN which uses sequential data or time series data. These algorithms are used to detect and solve temporal problems, in Natural language processing NLP, Anomaly detection, Image captioning and label predictions, etc.

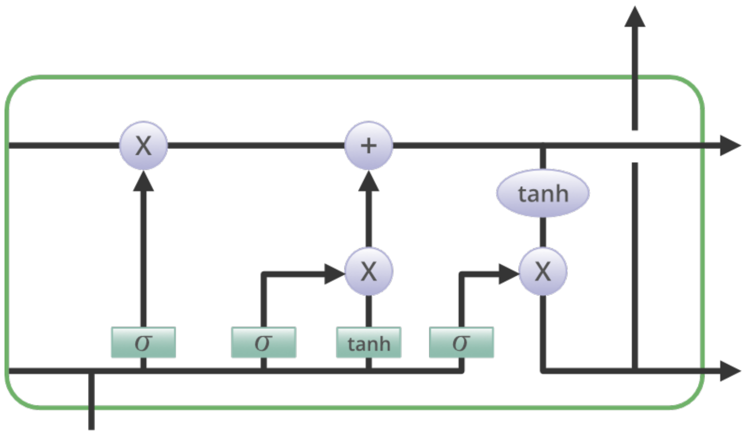
Like Convolutional neural network, these algorithms use training data to learn. They take information from the prior inputs to populate the present input and output.

We use Recurrent Neural Networks for data processing as it works well with recognizing sequential data and predict patterns.



The drawback of RNN is that they cannot remember long term dependencies due to vanishing gradient, and to overcome this issue, LSTMs are designed explicitly.

**LSTM:**
*Long Short-Term Memory (LSTM)* networks are a type of RNN and they are capable of learning order dependencies in sequence prediction problems.
Here the RNN uses LSTM algorithm to receive inputs and create outputs. LSTM block is a complex unit, it consists of weighted inputs, activation functions, sequential inputs from previous blocks and sequential outputs. The block diagram is shown below.



The LSTM unit is referred to as Long short-term memory as its architecture is based on short-term memory process to predict or create longer-term memory. Else ways, LSTM predicts values after referring to its previous instances. Hence LSTM is an accepted and common concept in mastering Recurrent Neural Networks.

## LSTM autoencoders:

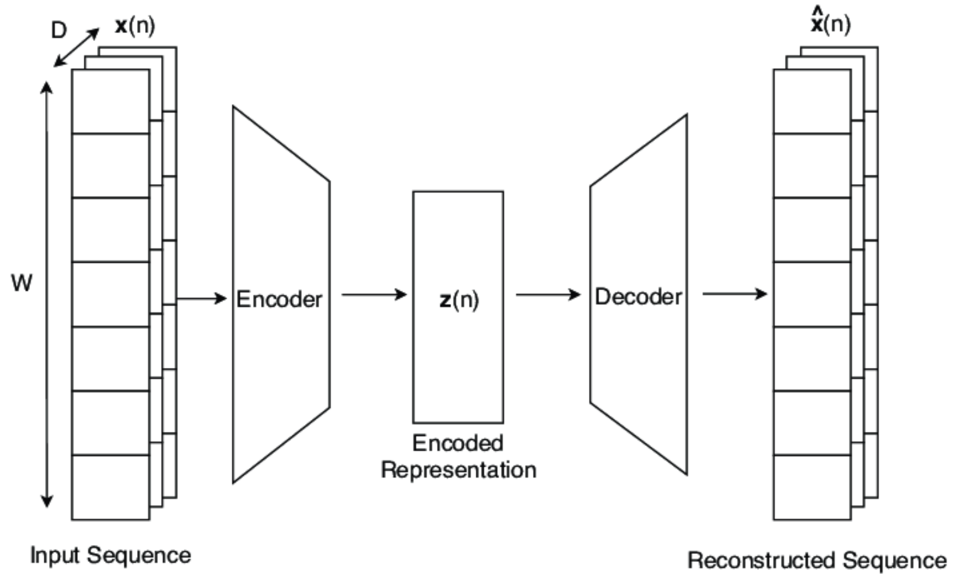
They focus on encoder-decoder architecture. They compress data using encoder and decode it to its original structure using a decoder.

The subsequent predicted values are within 1 standard deviation in this algorithm. Autoencoders are neural network models that represent a compressed form of an input (encoded). They are trained as a portion of larger model that results in recreating the input, hence they are unsupervised learning even though they are trained using supervised learning methods.

The main objective of the autoencoders is to reconstruct the input at the best possible way by minimizing the loss function. They can be Mean Absolute Error (MAE), Mean Squared Error (MSE) and cross entropy loss. We monitor the MAE loss in this model.

One use of LSTM autoencoder is feature extraction model. Once the model is fit, the reconstruction aspect of the model is considered until a certain bottle neck in the dataset from which the reconstruction of the input data is carried out. That is, time series data is provided as input to the model and the output of the model at certain point is the vector representing compressed input pair which is used as feature vector of our model to visualize the data instances or to undergo dimensionality reduction.

They are more precise than Recurrent Neural networks.



**LSTM Model:**

The LSTM model is built using TensorFlow keras library. It is a sequential model, and it contains 2 LSTM layers with 128 neurons, dropout layers, repeat vectors (Recurrent neural network model) and time distributed.

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It comes with built in model to process time series model.

Model uses sequential structure as we predict the data instances of a time series data over predefined timestamps.  The number of units to the LSTM layers are the number of number of neurons or LSTM memory units connected to the inputs of the time series data and each transfers the filtered information to the next memory unit.

A dropout of 20% is used to randomly drop the instances out of the batch size when the LSTM model produces sequences.

Repeat vectors are used in intervals of timesteps (interval between two consecutive data points) to repeat the input fed for defined number of times. It is timesteps in our case.

Return sequences of the LSTM model is set to True which implies the sequences related to current hidden state is returned for all instances individually.

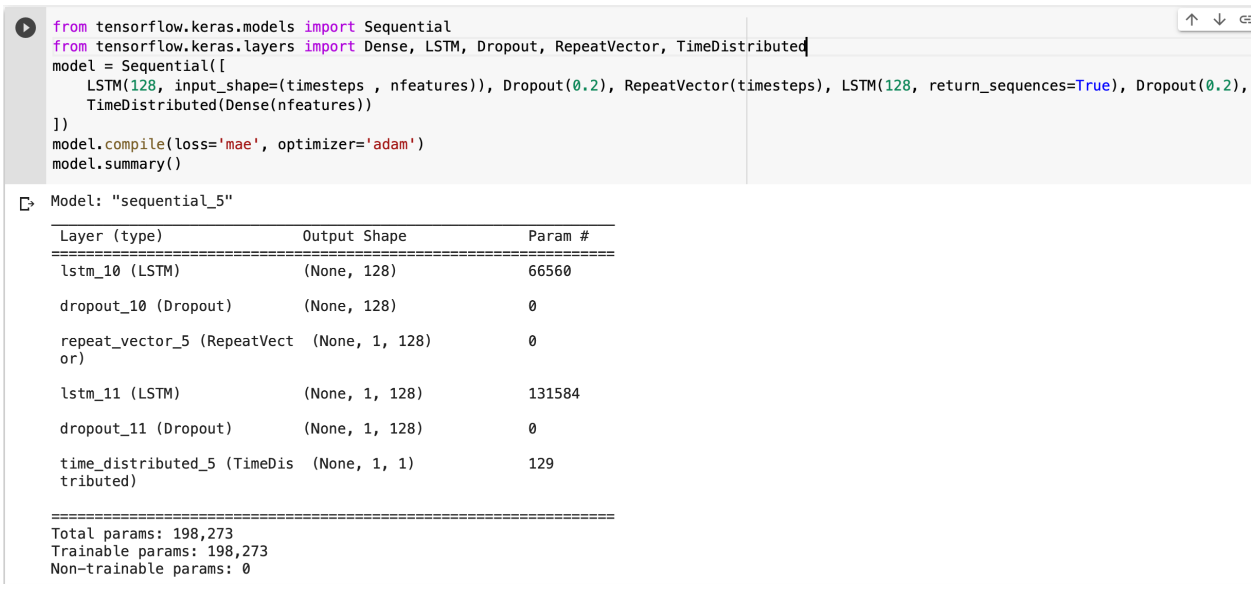
We use a time distributed layer in our LSTM model to apply a layer to every temporal portion of an input. IN LSTM every input fed is of 3 dimension and the dimension index of the first input will be the temporal dimension added by timesteps.

The model is optimized using adaptive moment estimation (Adam). Adam is an algorithm for optimization techniques for gradient descent.

The mean absolute error or MAE is calculated for this model. MAE denotes the loss function more robust to outliers. The math behind MAE is the average of absolute difference between the actual and predicted values.

MAE is used as a decision factor for detecting anomalies in our sequential model.

The picture below shows the LSTM sequential model



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, RepeatVector, TimeDistributed
model = Sequential([
    LSTM(128, input_shape=(timesteps , nfeatures)), Dropout(0.2), RepeatVector(timesteps), LSTM(128, return_sequences=True), Dropout(0.2),
    TimeDistributed(Dense(nfeatures))
])
model.compile(loss='mae', optimizer='adam')
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_10 (LSTM)              (None, 128)               66560     
                                                                 
 dropout_10 (Dropout)        (None, 128)               0         
                                                                 
 repeat_vector_5 (RepeatVect  (None, 1, 128)           0         
 or)                                                             
                                                                 
 lstm_11 (LSTM)              (None, 1, 128)            131584    
                                                                 
 dropout_11 (Dropout)        (None, 1, 128)            0         
                                                                 
 time_distributed_5 (TimeDis  (None, 1, 1)             129       
 tributed)                                                       
                                                                 
=================================================================
Total params: 198,273
Trainable params: 198,273
Non-trainable params: 0
```

The sequential model built using LSTM layers are subjected to fit the training data. The model undergoes learning process as it is fit with non-anomalous data or training data.

In Recurrent model, the model is fit with input as the training sequences (training_X) and the output is targeted at training_Y. I have used a batch size of 32, epochs as 10, 10% of the dataset as the validation split.

*To fit the model:*

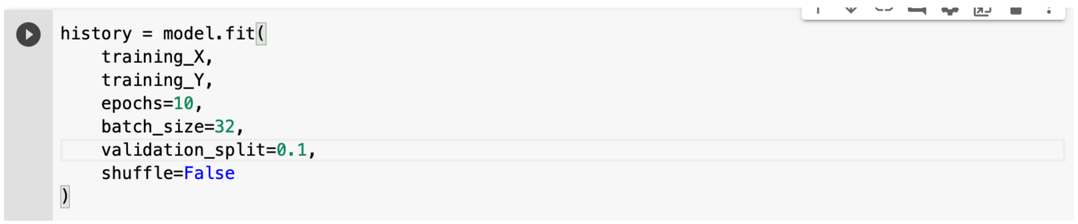
Training_X : The input array fed to the model to fit.

Training_Y: The target where output is stored

Epochs: It is defined as one pass over complete dataset. This separated the training data into distinct phases useful for periodic evaluation. In my analysis, I have used 10 epochs which means evaluation is run at end of each epoch.

Batch_size: It refers to the number of training instances utilized in one iteration. I have used 32 in my analysis.

Validation Split: denoted the portion of training data to be used as validation data.

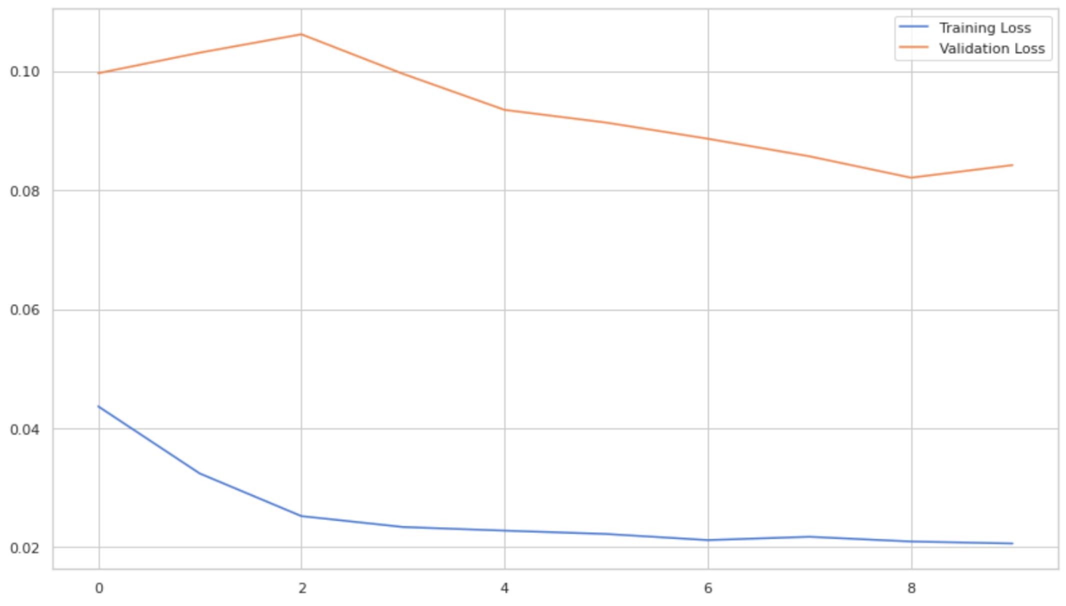


```
history = model.fit(
    training_X,
    training_Y,
    epochs=10,
    batch_size=32,
    validation_split=0.1,
    shuffle=False
)
```

```
Epoch 1/10
13973/13973 [==============================] - 106s 7ms/step - loss: 0.0437 - val_loss: 0.0996
Epoch 2/10
13973/13973 [==============================] - 94s 7ms/step - loss: 0.0324 - val_loss: 0.1031
Epoch 3/10
13973/13973 [==============================] - 96s 7ms/step - loss: 0.0252 - val_loss: 0.1062
Epoch 4/10
13973/13973 [==============================] - 96s 7ms/step - loss: 0.0234 - val_loss: 0.0995
Epoch 5/10
13973/13973 [==============================] - 95s 7ms/step - loss: 0.0228 - val_loss: 0.0935
Epoch 6/10
13973/13973 [==============================] - 96s 7ms/step - loss: 0.0222 - val_loss: 0.0913
Epoch 7/10
13973/13973 [==============================] - 94s 7ms/step - loss: 0.0212 - val_loss: 0.0886
Epoch 8/10
13973/13973 [==============================] - 96s 7ms/step - loss: 0.0218 - val_loss: 0.0857
Epoch 9/10
13973/13973 [==============================] - 99s 7ms/step - loss: 0.0210 - val_loss: 0.0821
Epoch 10/10
13973/13973 [==============================] - 97s 7ms/step - loss: 0.0206 - val_loss: 0.0842
```

As the model is fit, the training loss and validation loss is monitored over the complete training process or learning process.

```
<matplotlib.legend.Legend at 0x7f22c614efd0>
```



Loss is the value of the cost function for the training data, and the training loss tells us how well the model is fitting the training data. It is evident that the model is working good as the loss value is decreasing from the start of epoch to the end.

The Val_loss is the value of cost function for validation data. They evaluate the performance of proposed model.

As the model is fit and the loss is monitored, we use the model to predict non-anomalous data or training data to determine the MAE loss which will be our threshold, and the same model is used to predict the anomalous or test data and the anomalies are detected from these test predictions are detected when the MAE loss of the test data satisfies the anomalous condition.

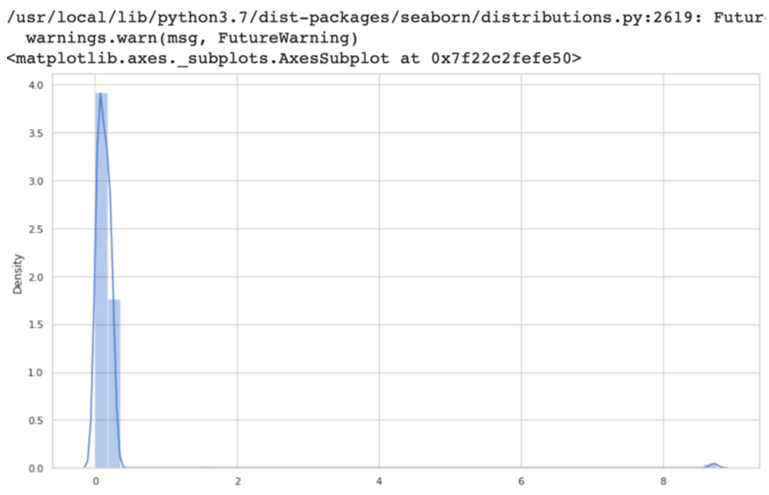
## *RESULTS AND DISCUSSION*

The prediction of training dataset and the testing dataset are carried out successfully and the training MAE loss is calculated as the average of the absolute difference between the actual and the predicted values. The maximum of this loss monitored over non-anomalous data prediction serves as the threshold function for detecting anomalies in the test data being predicted.
The test MAE loss is calculated over predicted test data samples and actual test data samples. The predicted data instance is an anomalous data if the MAE loss of the instance predicted exceeds the thresholding condition.

*Condition*:
Anomaly = predicted_test_data>threshold.
Threshold = maximum of training MAE loss.

We can also detect anomalies by monitoring Mean Squared Error of both training and test predictions and compare it with threshold.

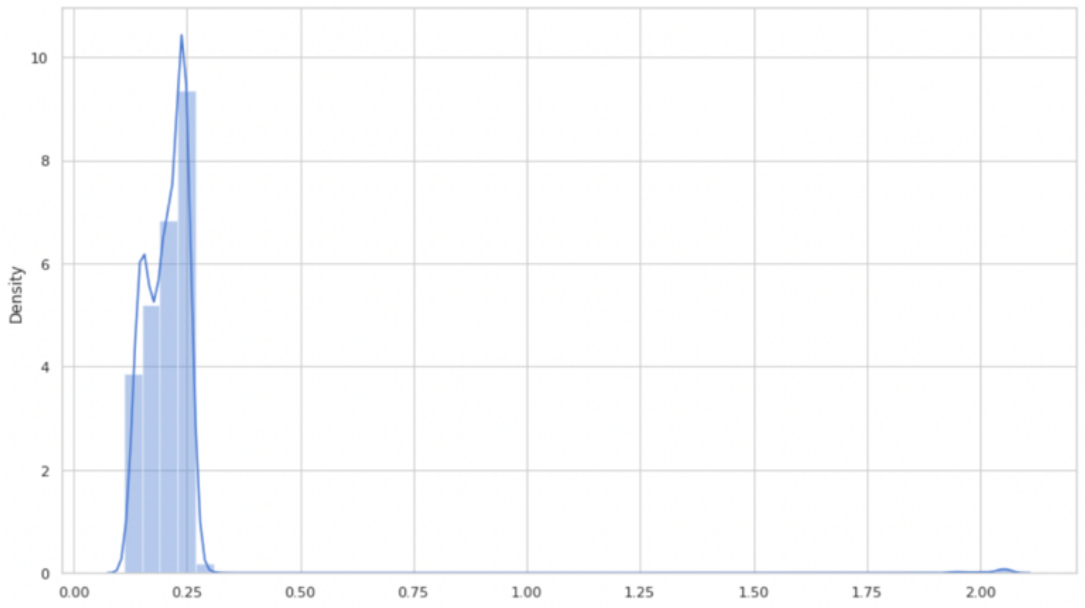
The MAE loss determined over training data predictions are:



```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futur
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f22c2fefe50>
```

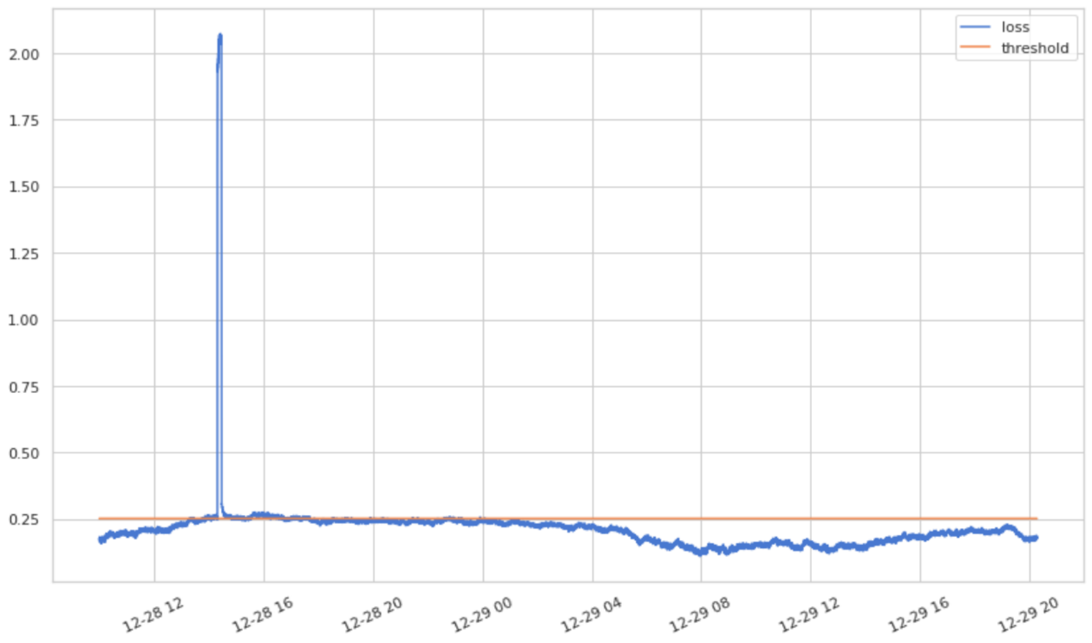
The MAE loss determined over test data predictions are:

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futur
  warnings.warn(msg, FutureWarning)
```

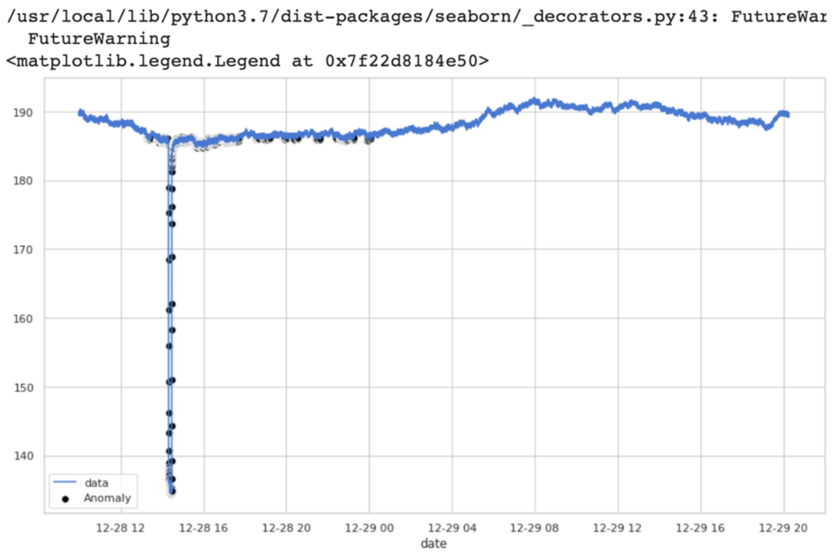


Threshold is determined from the training predictions and the thin line in orange represents the threshold value and the data points having MAE more than this value is anomalous datapoint.

```
<matplotlib.legend.Legend at 0x7f22d81b6350>
```



The predicted test data and the actual test data are plot over each other using scatter plot. The data in blue is the actual test data and the data in black points are the anomalous test data points that are detected.

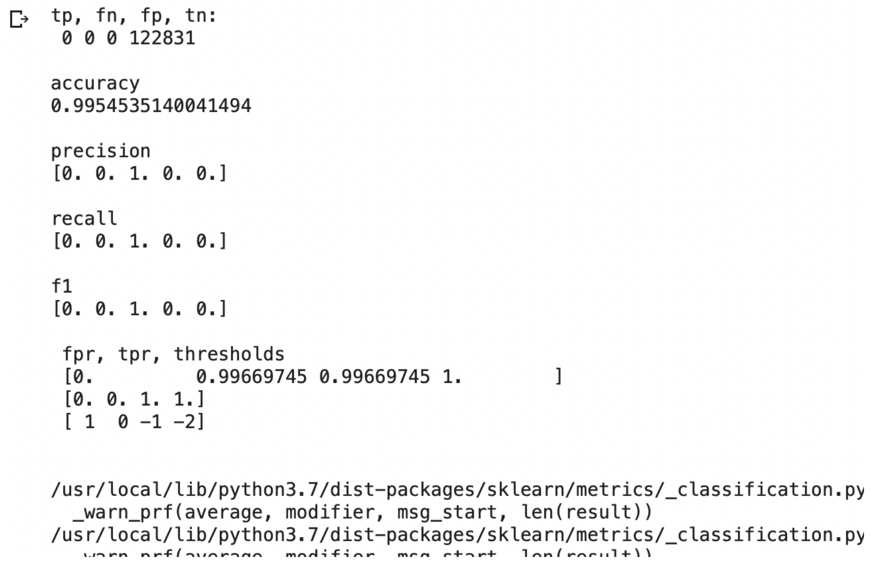


```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
  FutureWarning
<matplotlib.legend.Legend at 0x7f22d8184e50>
```

As the anomalous data points are identified, we must evaluate the performance of the model using performance metrices.

- The performance metrices are:
- In anomaly detection the predicted values can be:
- True Positive (TP): Model predicts the positive class correctly.
    - True Negative (TN): Model predicts the negative class correctly
    - False Positive (FP): Model predicts the positive class incorrectly.
    - False Negative (FN): Model predicts the negative class incorrectly.
- Accuracy: It is the extent to which data is predicted correctly, i.e., the number of correct predictions over total number of predictions.
    - Accuracy = (TP + FN)/(TP+TN+FP+FN)
- Precision: Depicts the portion of anomalous data being predicted correctly.
    - Precision = TP/(TP+FP)

- Recall: Denotes the true anomalies being identified.
    - Recall = True Positive/ (False Positive + False Negative)
- F1 score: It is the harmonic mean of precision and recall. It denotes the overall performance of the anomaly detection of our model.
    - F1 score = 2*Recall*Precision / (Precision + recall)
- True Positive Rate (TPR): It gives the correct positive results predicted over all positive instances.]
    - TPR = True Positive/ (True Positive + False Negative)
- False Positive Rate (FPR): It represents the number of incorrect positive results over all negative instances of the dataset.
    - FPR = False Positive/ (False Positive + True Negative)
- Confusion matrix: It is the technique used to summarize the performance of the algorithm. It consists of TP, FP, TN, FN in the form of a matrix.
- ROC AUC: Receiver Operating Characteristic (ROC) is the graphical representation of performance evaluation of the model at all thresholds. It is a plot between the TPR and FPR. AUC is the area under the ROC curve. The AUC score provides overall measure of performance across all possible thresholds.
- The performance metrics are evaluated to be as:



```
tp, fn, fp, tn:
 0 0 0 122831

accuracy
0.9954535140041494

precision
[0. 0. 1. 0. 0.]

recall
[0. 0. 1. 0. 0.]

f1
[0. 0. 1. 0. 0.]

 fpr, tpr, thresholds
 [0.         0.99669745 0.99669745 1.        ]
 [0. 0. 1. 1.]
 [ 1  0 -1 -2]


/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py
  _warn_prf(average, modifier, msg_start, len(result))
```

## *CONCLUSION*

It is clear from the Analysis through LSTM sequential model that 10711 instances were recorded as anomaly. The file output.csv contains the datafile with 2 columns, one for timestamp and other of Anomalous results as 0 to no cyber-attack and 1 for cyber-attack in our dataset.
The anomalous datapoints detected are contextual outliers.
The metrics were calculated to find the measure of performance of our model.
Other than RNN LSTM model, the best approaches are One class SVM, clustering and Density based techniques like KNN, LOF and Isolation Forest.
The model is performing well with reducing loss and it predicts the outcome values having actual values as input and it can be improvised by increasing the epoch size, batch size, number of LSTM layers, etc., to get mere satisfactory results.

## *REFERENCES*

[1] Pandas library: https://pandas.pydata.org/docs/user_guide/index.html#user-guide
[2] Scikit-learn: https://scikit-learn.org/stable/user_guide.html
[3] NumPy: https://numpy.org/doc/stable/user/index.html#user
[4] Detecting Mobile Traffic Anomalies through physical control channel Fingerprinting: a Deep Semi-supervised Approach. Francoise Beaufays
[5] Hawkins, Douglass M. (1980). Identification of Outliers. Chapman and Hall London; New York
[6] Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. Hasim Sak, Andrew Senior.