REPORT

Project Dataset: Tweets by Amazon.

Dataset Source:
https://raw.githubusercontent.com/numenta/NAB/master/data/realTweets/Twitter_volume_AMZN.csv

Logic:

Train the data on the model. Get the loss between the actual and the predicted value. Get a normally distributed curve of the errors. Get the mean and standard deviation of the distribution. Fix on a threshold value of loss. If the loss for any data point is more than the threshold value consider it an anomaly. We have considered the threshold value to be one standard deviation, that is about 65% of the error distribution around the mean.

PART 1:

Setup:
Consider previous 288 values and predict the next 5 values.
Model: Dense layers
Building models with dense layers and LSTM. The output is shown for dense layers. We have only implemented this project using LSTM whose layers are also shown below (commented). For LSTM the input should be 3 dimension and for dense it should be 2 dimensions.

For the different setups we have used different window sizes (72, 144, 288).

Each setup is submitted in a different notebook with name "Part**#_windowsize".**

The three error metrics that we used are:

1. MAE - Mean Absolute Error
2. MSE – Mean Squared Error
3. R2 score

Anomalies are plotted using each of these error metrics. Similar results were seen in all the cases.

PART 2:

We have implemented autoencoder with LSTM, Conv1d and Dense layers setup.

Observations:

1. LSTM takes longer to run compared to the other two setups.
2. Lower the batch size and timesteps, lower/faster is the run time. We have run the setup with the following batch sizes: 128, 64, 32.
3. Loss was higher for higher timesteps value. We have run the setup with the following time steps: 288, 144, 72, 36.
4. Adding more layers gave better predictions.

5. Increasing the number of neurons in the autoencoder layers resulted in more loss. We have run the setups with (500,250), (250,125), (200,100) and (100, 50). Reducing the number of neurons gave less loss and good anomaly detection.
6. In few cases ReLU gave a huge loss or nan, changing the activation function to Linear gave good results.

PLOTS:

Part 1:

1. Plot for original data.
2. Part 1 has plots for train vs validation loss.
3. Error distribution plot.
4. Plots detecting anomalies using different error metrics**.** The red spots are the anomalies in the data and the blue plot is the actual data)

Part 2:

1. Plot for original data.
2. Part 2 has plots for train vs validation loss.
3. Error distribution plot.
4. Plots detecting anomalies using Mean Absolute Error metric**.** The red spots are the anomalies in the data and the blue plot is the actual data)

REFERENCES:

1. https://blog.keras.io/building-autoencoders-in-keras.html
2. https://colab.research.google.com/github/jdwittenauer/ipython-notebooks/blob/master/notebooks/keras/AnomalyDetection.ipynb#scrollTo=-t2rLTCeBK-5
3. https://towardsdatascience.com/autoencoder-on-dimension-reduction-100f2c98608c