Anomaly Detection in Time Series

Probabilistic and Forecasting models on Numenta Anomaly Benchmark (NAB) dataset

Udayveer Singh Andotra

RU-ID: 227001036

1.Introduction

Anomaly is an observation which suspiciously deviates from other observations and is suggestive that it was generated by a different mechanism. These strange and interesting events can be categorized into point anomaly and subsequence anomaly. A pictorial representation can be seen below in Figure 1. A robust anomaly detection technique would be efficient in correctly identifying such events (local/global outliers and anomalous sequences) irrespective of their type, and doing so with a reasonable amount of data. This report is a comparative study of a trivial outlier detection technique and two different unsupervised learning techniques, namely hidden markov model and autoencoder. Methods, results and conclusions pertaining to the techniques are discussed further.

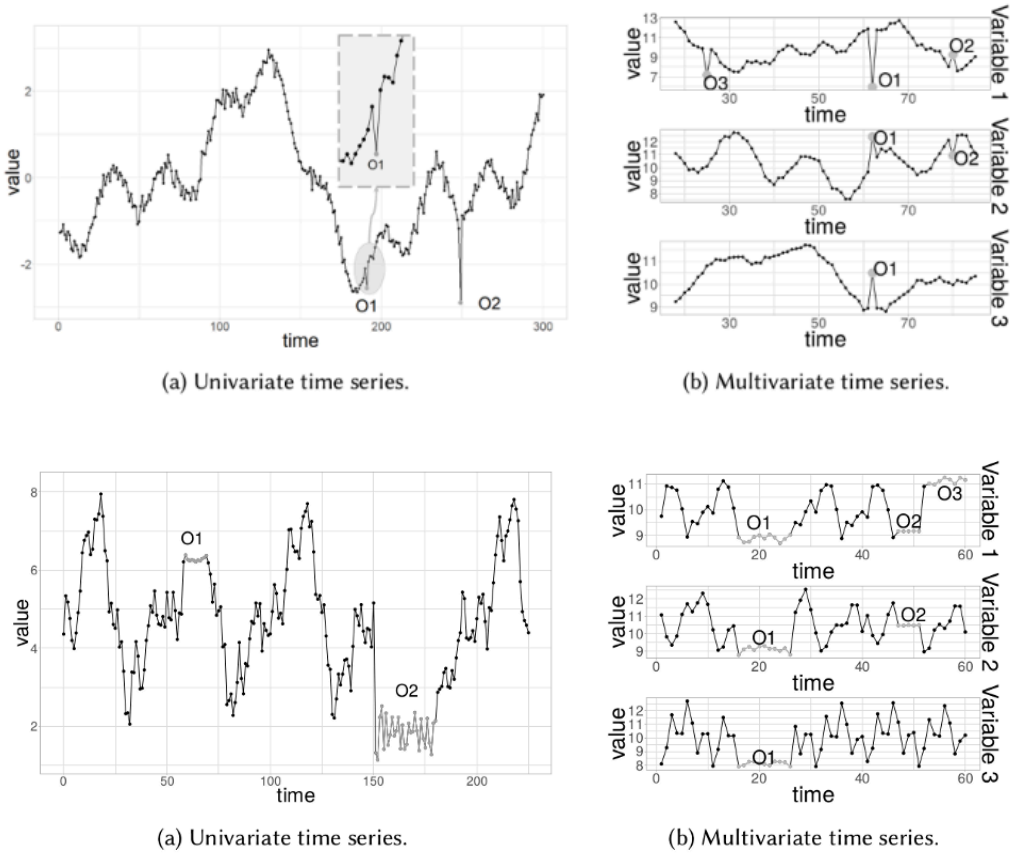

Figure 1. Point Anomaly (top) and Subsequence Anomaly (bottom)

Source

2.Data and Methods

The dataset used here is the Numenta's Anomaly Benchmark (NAB) dataset- Twitter_volume_IBM. NAB is a novel benchmark for evaluating algorithms for anomaly detection in streaming, real-time applications. The dataset is ordered, timestamped and has single-valued metrics. The true labels for an anomaly are not not available but there are anomalous events present in the data. Due to a lack of actual anomaly identifier variable we evaluate our results based on an anomaly detector - ARTime. Figure 2 below shows the normalized benchmark scores for different detectors which is based on NAB's algorithm which penalizes for both false positives and false negatives. ARTime is presently performing the best amongst other detectors for NAB datasets.

Detector	Standard Profile	Reward Low FP	Reward Low FN
Perfect	100.0	100.0	100.0
ARTime	74.9	65.1	80.4
Numenta HTM*	70.5-69.7	62.6-61.7	75.2-74.2
CAD OSE†	69.9	67.0	73.2
earthgecko Skyline	58.2	46.2	63.9
KNN CAD†	58.0	43.4	64.8
Relative Entropy	54.6	47.6	58.8
Random Cut Forest ****	51.7	38.4	59.7
Twitter ADVec v1.0.0	47.1	33.6	53.5
Windowed Gaussian	39.6	20.9	47.4
Etsy Skyline	35.7	27.1	44.5
Bayesian Changepoint**	17.7	3.2	32.2
EXPoSE	16.4	3.2	26.9
Random***	11.0	1.2	19.5
Null	0.0	0.0	0.0

Figure 2. Numenta Anomaly Detector Benchmark Score

Source

Twitter_volume_IBM is a collection of Twitter mentions of IBM. The metric value represents the number of mentions for a given ticker symbol every 5 minutes. The data has 15,893

observations from early March, 2015 to late April, 2015. Figure 3 below shows the time stamped observations with the anomalous events as per ARTime. It estimates 1590 such events.

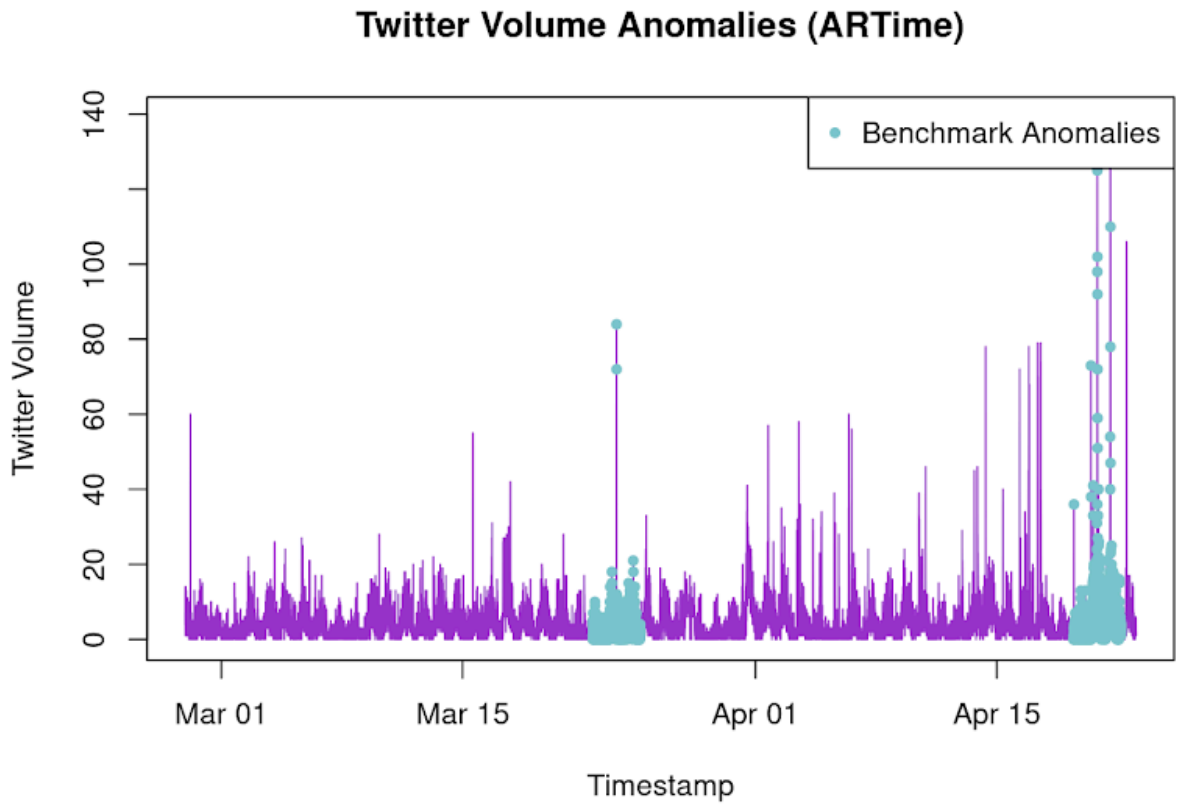

Figure 3. Twitter Volume IBM series with ARTime anomalies marked. Anomaly count: 1590

The essence of all discussed methods is having a threshold on the deviations. These deviation metrics are unique to each method.

2.1.Time Series Decomposition

Decomposing a time series to trend, seasonality and residuals components. If y_t is a time series then m_t (trend), s_t (seasonality) and r_t (residue) are components of it. The decomposition can be additive or multiplicative.

$$y_t = m_t * s_t * r_t \quad \text{or} \quad y_t = m_t + s_t + r_t$$

Additive decomposition is used here. Figure 4 shows this decomposition (done in R). The residual medians are used for threshold. The big and small residuals are there marked as

anomalies. This method is more suited to catch local and global outliers. However, I have used Twiter's AnomalyDetection library (in R). It employs time series decomposition and using robust statistical metrics, viz., median together with ESD (Extreme Studentized Deviate) test which tries to capture the outliers more efficiently. Besides time series, the package can also be used to detect anomalies in a vector of numerical values.

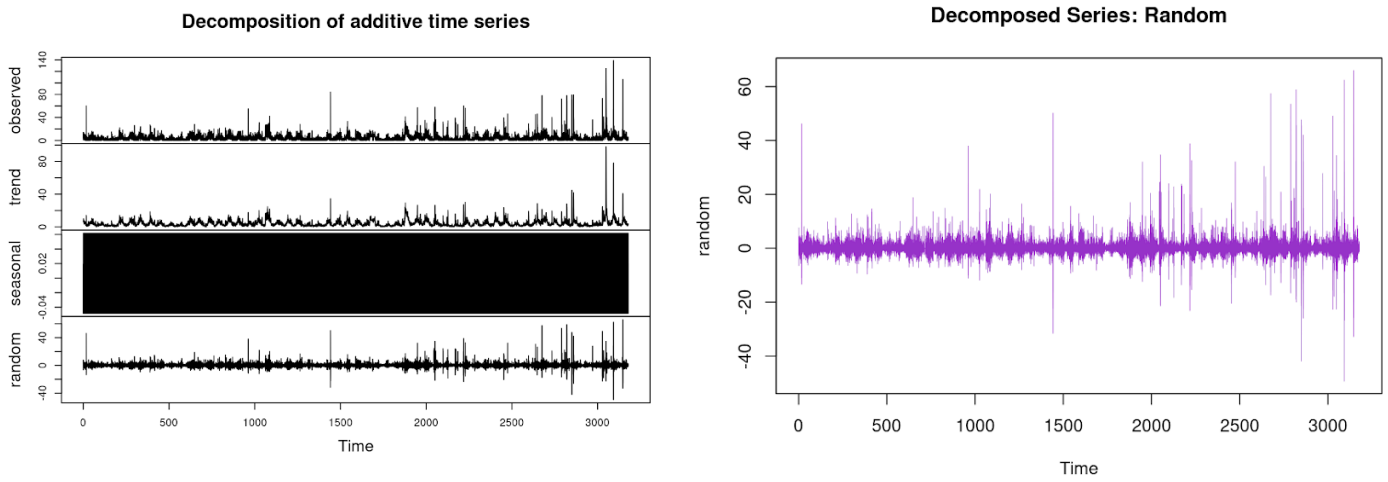

Figure 4. Twitter Volume IBM series, Left: Classical Decomposition & Top-Right: Residual series

2.2.Hidden Markov Model

Hidden Markov Model (HHM) is a probabilistic unsupervised learning model used for sequential data. An HMM consists of a hidden state sequence and an observable sequence. The hidden state sequence represents the underlying process that generates the observations, while the observable sequence is the sequence of observations that are directly visible. In essence, the goal of HMM is to model the probability distribution of the hidden state sequence given the observed sequence. It has three main algorithms:

- **Baum-Welch Algorithm**: An iterative method that estimates the parameters of the HMM given a training set of observed sequences. The Baum-Welch algorithm is based on the principle of expectation-maximization (EM).
- **Forward-Backward Algorithm:** Provides an efficient way to calculate the likelihood of a sequence of observations given the model parameters and to estimate the hidden states that generated these observations.

- **Viterbi Algorithm:** Dynamic programming algorithm used to solve the maximum likelihood decoding problem for hidden Markov models (HMMs) find the most likely sequence of hidden states that generated a given observation sequence (called the “decoding task”)

To detect anomalies I used the forward-backward algorithm. Viterbi gives the most likely sequence, whereas Forward-Backward gives the most likely state at each position. If the number of sequences correctly predicted is priority, then the Viterbi algorithm is preferable, but if the number of individual state errors is, then, Forward-Backward is better.

The hidden state in HMM is a markov process which is hidden and the markov assumption is applicable : If we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. The figure below shows a simplified HMM architecture with hidden sequence, observation sequence and the state transition probabilities. The emission probabilities are simply conditional probabilities, P(D | H) and have not been shown in the figure. Based on the markov assumption the posterior probability can be calculated as P(D). Placing a lower threshold on this probability will help us identify the anomalies.

Hidden Seq, H: $\{H_1,H_2….H_n\}$
Observed Seq, D:$\{D_1,D_2….D_n\}$
$P(D)=\sum_{\{Hidden\ States\}} P(H) P(D | H)$

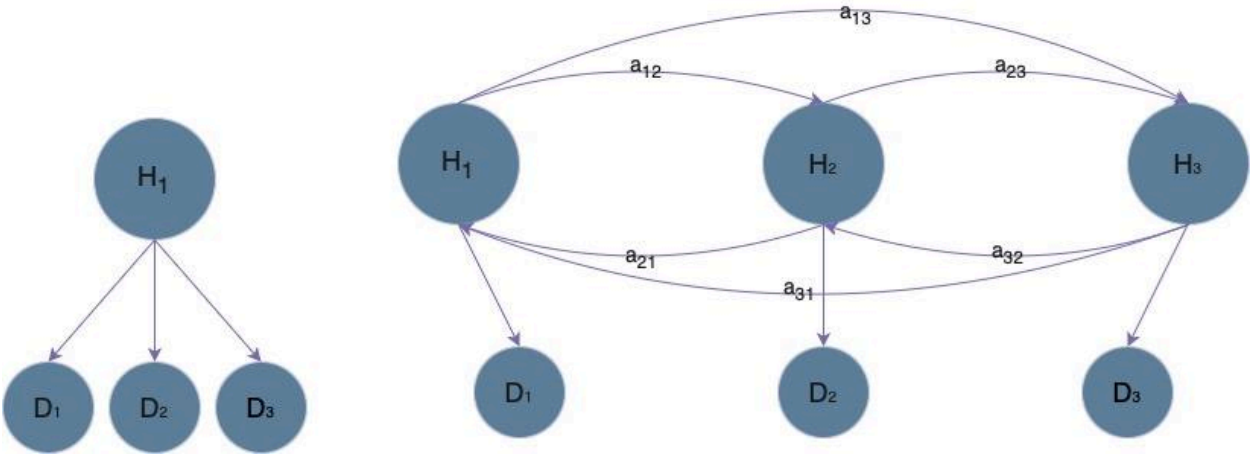

Figure 5. Hidden Markov Model Architecture

2.3.Autoencoder

Autoencoder is a neural network designed to learn an identity function in an unsupervised way to reconstruct the original input while compressing the data in the process so as to discover a more efficient and compressed representation.

Autoencoders are trained via backpropagation and consist of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code (latent-space representation), the decoder then reconstructs the input only using this code. The output of the autoencoder will not be exactly the same as the input, it will be a close but degraded representation.

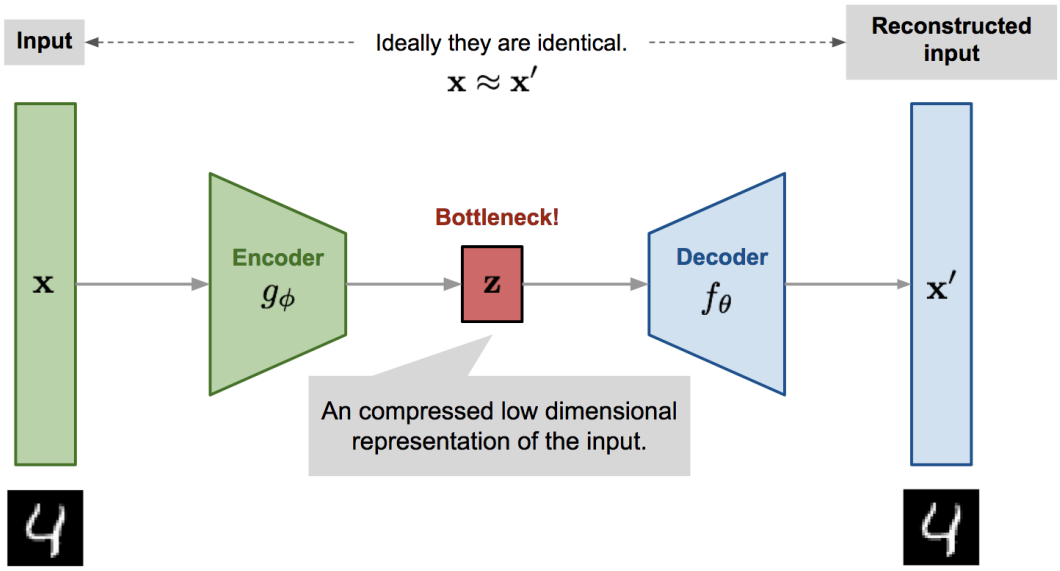

Figure 6. Autoencoder Architecture

See references for further details on the working of the encoder/decoder and the robust dimensionality reduction process of autoencoder. The loss function for the reconstruction error used is mean-squared-error (MSE) and placing a threshold will help to identify the anomalies.

3.Results and Conclusion

3.1.Time Series Decomposition : AnomlyDetection

As discussed in the methods section and quite evident from figure 7 below, the use of residual magnitude in correctly identifying anomalies is highly efficient. The method identified 539 anomalies as compared to our reference count of 1590 from ARTime.

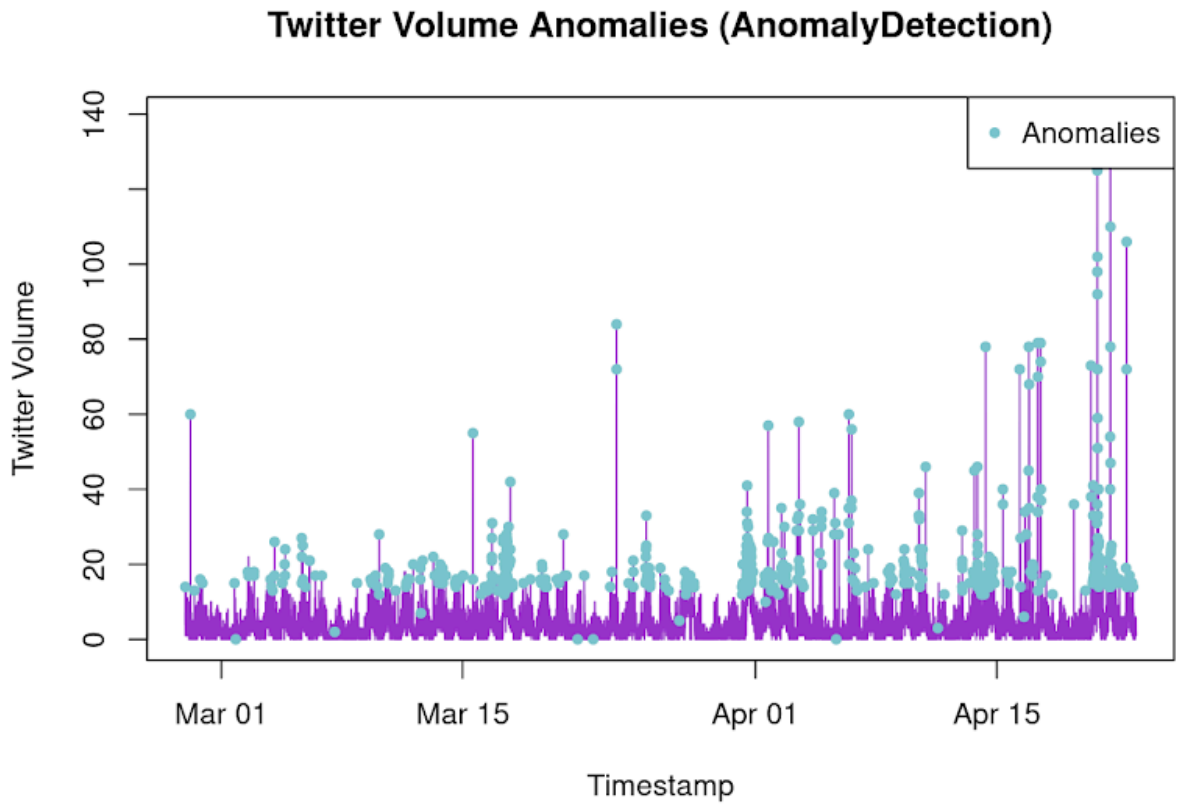

Figure 7. Twitter Volume IBM series, Anomalies using AnomalyDetection. Anomaly count: 539

The anomaly events here are scattered quite evenly while capturing some of the point anomalies but failing completely to get the subsequence anomalies.

3.2.Hidden Markov Model: Forward-Backward Algorithm

The forward-backward algorithm for posterior observation probability and a threshold yielded 776 anomalies as shown in figure 8. The number of hidden states used by the model is 8. The

number of hidden states and the model parameters are a challenging aspect of optimizing HMM. The model parameters estimated by the model fit produced alongside setting the number of hidden states to 8 gave the best result (based on the limited set of hidden states I tested). The results however seem to be biased towards the extremities. This is an optimisation problem of the model as the given prior distribution and the number of hidden states can't accurately depict the process that generated the observation sequence.

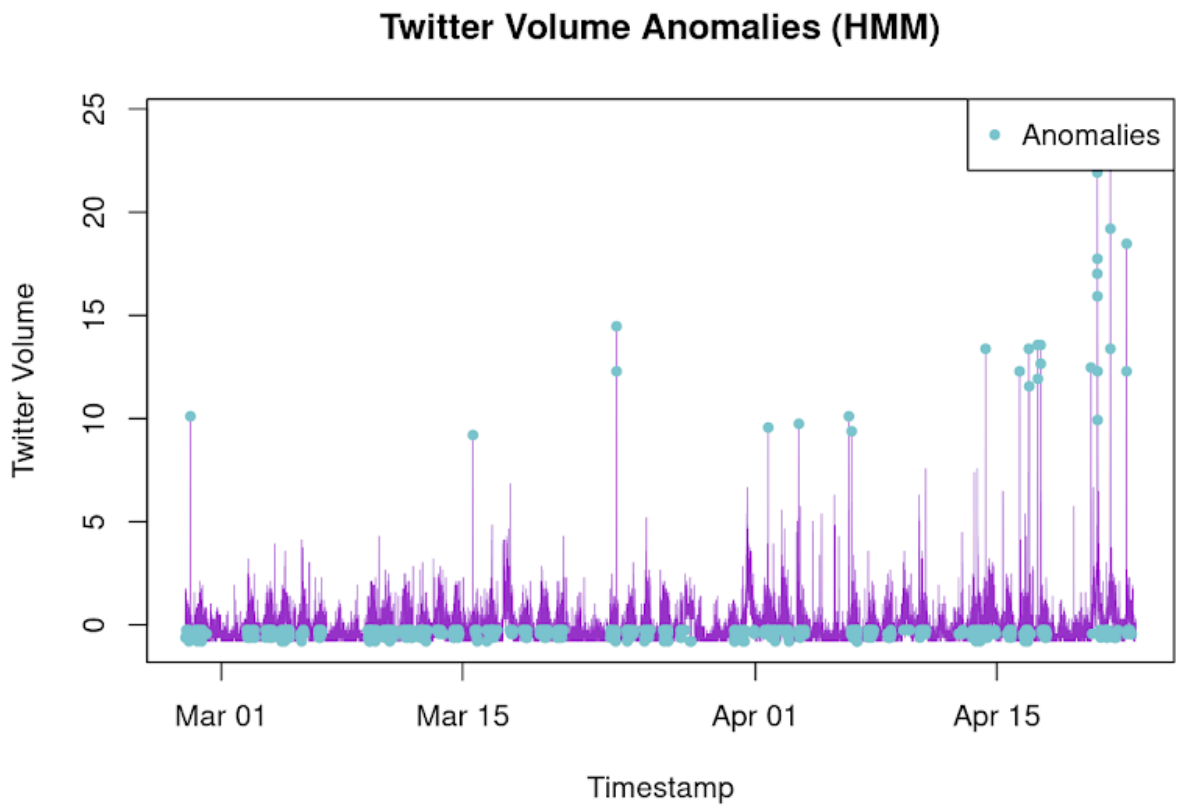

Figure 8. Twitter Volume IBM series, Anomalies using Hidden Markov Model (Gaussian, No. of Hidden States = 8). Anomaly count: 776

3.3.Autoencoder: MSE as loss function

The autoencoder was defined with an encoding layer with 64 neurons. Rectified linear unit activation function is used here, which will output the input directly if it is positive, otherwise, it will output zero. The decoding layer has the same number of neurons as the input layer i.e. 288 and uses a linear activation function, meaning it will output the weighted sum of the inputs plus the bias.

Figure 9 below shows the anomaly flagged results from the autoencoder. A count of 781 such events is reported. This count is only a bit more than HMM but the anomalous events are much better classified by the autoencoder. It does not consider any prior distribution of input data and is not limited due to the unoptimized number of hidden states, as is the case with HMM. The model runs for 50 epochs and has identified the point as well as subsequence anomalies for the anomalous cluster event towards April end (as per ARTime).

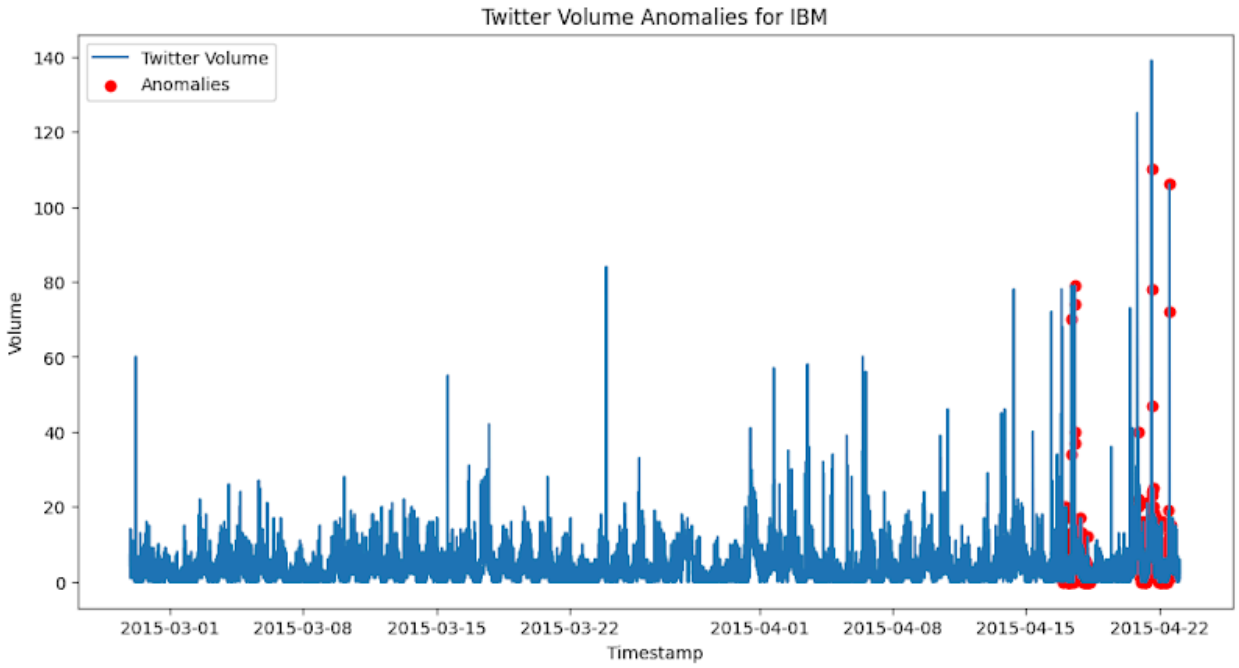

Figure 9. Twitter Volume IBM series, Anomalies using Autoencoder (No. of Epochs = 50). Anomaly count: 781

3.4.Conclusion

The three methods discussed hold their applicability as per the task and observation set at hand. No singular present day technique is robust and universal in dealing with real world anomalous events. Autoencoder performed the best amongst the three but it does not perform well with longer sequences or large datasets. HMM can easily deal with multivariate sequences but its utility is limited when used alone. Ensemble models address this problem with a framework that minimizes the incapabilities of the individual components. ARTime is also an ensemble model and was expected to work better than what we discussed. The normalized benchmark scores can be obtained for these techniques. A guide is available in NAB's github repository.

4.References

1. [2002.04236] A review on outlier/anomaly detection in time series data
   https://arxiv.org/abs/2002.04236
2. GitHub - numenta/NAB: The Numenta Anomaly Benchmark
   https://github.com/numenta/NAB
3. GitHub - twitter/AnomalyDetection: Anomaly Detection with R
   https://github.com/twitter/AnomalyDetection
4. Multivariate time series anomaly detection: A framework of Hidden Markov Models
   https://doi.org/10.1016/j.asoc.2017.06.035
5. Intro to Autoencoders | TensorFlow Core
   https://www.tensorflow.org/tutorials/generative/autoencoder
6. Autoencoders for Anomaly Detection in an Industrial Multivariate Time Series Dataset
   https://doi.org/10.3390/engproc2022018023

5.Appendix

The dataset can be accessed directly from the server and this has been incorporated in the code. A copy of the dataset is in the zipped file too. To replicate the results the code files Autoencoder.ipynb and Decomp_HMM.Rmd can be found in the zipped file also. I have performed autoencoder implementation in Google Colab (python 3) and used Rstudio (version 2023.12.1+402) for time series decomposition and hidden markov model.