Unsupervised Anomaly Detection on Cloud-Hosted Credit Decisioning Engines

by

Armando Ordorica

A thesis submitted in conformity with the requirements
for the degree of Master of Engineering
Graduate Department of Computer Engineering
University of Toronto

Abstract

Unsupervised Anomaly Detection on Cloud-Hosted Credit Decisioning Engines

Armando Ordorica

Master of Engineering

Graduate Department of Computer Engineering

University of Toronto

2020

Gaussian Mixture Models (GMMs) are known for being adequate at detecting anomalies in unbalanced datasets - when the number of negative examples heavily outweighs the number of positive examples. This paper demonstrates the efficiency of Multivariate GMMs in unsupervised anomaly detection on the nodes of a Credit Decisioning Engine hosted on Amazon Web Services (AWS). It is shown that this model reports high F1 scores for both training and testing sets.

A further improvement to account for impurities in the dataset is also described and it is shown that it produces a significant boost in F1 score. The modified version of the GMM adapted to this dataset proves to be powerful yet simple to implement and relatively easy to debug.

Acknowledgements

The completion of this project could have not been accomplished without the support of my boss and friend Sami Hajir. I also thank my manager and coach Daniel Gunn, who has always pushed me to take my personal development as a priority and to not take no for an answer.

Finally, I thank my supervisor and mentor Yuri Lawryshyn for his unconditional support and his constant push towards simplification and efficiency.

Contents

Chapter 1

Introduction

The incorporation of Cloud Computing in the banking industry has unleashed many possibilities due to the ability to store and process enormous amounts of data in real-time thanks to on-demand availability of computer system resources. This technology allows for real-time decision making, such as transaction-level underwriting. Another advantage is the ability to detect anomalies in real-time as opposed to batch, and therefore to build engines that self-correct in the presence of an anomaly.

Anomalies can be defined as observations that deviate so much from other observations as to arouse suspicion that they were caused by a different mechanism (Ben-Gal, 2005). Anomaly detection in time-series is an important area with respect to business.

Financial institutions experience system anomalies that result not only in monetary losses but also in poor customer experience. By leveraging Cloud Computing technologies and algorithms powered by Machine Learning, it is possible to build self-correcting mechanisms within the decisioning engines that deal with anomalies in real time. However, the application of advanced machine learning algorithms, i.e. black box algorithms, is restricted in the banking industry since explainability is of paramount importance in such a regulated industry (*Fair Credit Reporting Act*, n.d.).

A clear challenge in accurately detecting anomalies is the highly non-linear nature of real credit card authorizations data, which is affected by seasonality, as well as political and global economic effects (Coronado-Ramirez, n.d.). This report demonstrates that Gaussian Mixture Models can be powerful tools in detecting anomalies despite impurities in the dataset.

1.1 How this Report is Organized

- In Chapter 2, traditional methods are compared and contrasted against more novel approaches such as Deep Learning in anomaly detection. Pros and Cons are discussed.
- Chapter 3 describes the source, structure, and nature of the dataset under study. The Anomaly Detection Problem is defined and then the choice of algorithm performance metrics is justified. Finally, the assumptions made by Gaussian Mixture Models (GMMs) are discussed and the methods that will be used to validate whether these assumptions hold true.
- Chapter 4 delves into the details of data pre-processing, testing for normality, and finaly shows the results for both the Simple and the Modified versions of the GMM.
- All the figures pertaining to the detailed analysis of each node are included in the Appendix in Chapter 7.

1.2 Objective

The dataset under study contains traffic on the nodes of a Credit-Decisioning Engine hosted on Amazon Web Services (AWS). Each of these nodes approve or decline millions of transactions per day.

The goal is to explore Gaussian Mixture Models as well as a modified version of it and examine their performance in detecting anomalies. Their efficiency will be explored as well as their explainability and the ease to debug to finalize with a recommendation.

Two algorithms will be discussed:

- Multivariate Gaussian Mixture Model
- Modified Multivariate Gaussian Mixture Model

1.3 Contributions

Although the theory behind Gaussian Mixture Models is well understood, some alterations were necessary to account for impurities in the dataset. These impurities include but are not limited to credit

rules changing over time, unbalanced traffic across nodes, and cyclic non-anomalous data impurities.

The main contributions can be summarized as follows:

- Simple Gaussian Mixture Models are effective at detecting anomalies in the traffic of nodes of Cloud-Hosted Decisioning Engines. The report shows how GMMs can achieve an F1 score of 0.933 as unsupervised anomaly detection models.
- This report shows how accounting for discontinuities in time series (where time series are the features) improves performance in Gaussian Mixture Models. By applying this modification, F1 score was improved to 0.947.

Chapter 2

Literature Review

This chapter compares and contrasts traditional and novel approaches for Anomaly Detection, discussing the advantages and disadvantages of each. In Section 2.3, it is explained why the decision to use Gaussian Mixture Models over other approaches was made.

2.1 Traditional Methods for Anomaly Detection

Business and economic time series are often complex and exhibit difficult periodic patterns. A large number of model based approaches have been proposed which assume time series to be a multiplicative or additive combination of components such as trend, seasonality and residue (Hillmer & Tiao, 1982). Such temporal based decomposition methods go all the way back to Holt-Winters methodology (Chatfield, 1978). These approaches try to model each of these components separately. However, these decompositions are tuned and trained on a specific problem and therefore exhibit poor generalization (Godfrey & Gashler, 2017).

ARIMA based models (Schlittgen, 1997) are widely popular in outlier detection. The problem with this family of models is that they assume linearity and mean ergodicity (wide-sense stationary processes), when in reality the observed behaviour of business data is highly non-linear and exhibits heteroskedasticity (Kapoor, 2020).

In order to use GMMs in anomaly detection, one must assess the level of hetereoskedasticity, since normally distributed data is the main assumption behind GMMs (Le, n.d.). In statistics, heteroskedasticity (or heteroscedasticity) happens when the standard errors of a variable, monitored over a specific

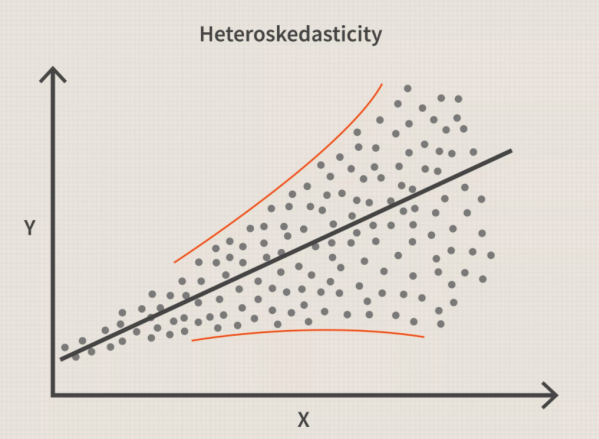


Figure 2.1: Heteroskedasticity is the term to characterize a process where the standard deviation is not constant over time and therefore may not be mean-ergodic.

amount of time, are non-constant (Knaub, 2007). With heteroskedasticity, the tell-tale sign upon visual inspection of the residual errors is that they will tend to fan out over time, as depicted in on Figure 2.1. Heteroskedasticity is a violation of the assumptions for linear regression modeling, and so it can impact the validity of econometric analysis or financial models (Hayes, 2020).

Principal Component Analysis (PCA) is another promising approach as used in anomaly detection by well-known technology companies such as Netflix (Blog, 2017). A key constraint introduced by PCA as an anomaly detection algorithm is that it requires high cardinality in the data, i.e. strong dependence between points in the dataset. Since the data used for this report belongs to real traffic of a decisioning engine that processes credit card transactions, the data points are indeed expected to be highly correlated. Although PCA is expected to perform well in this dataset (Hyndman, Wang, & Laptev, 2015), the level of explainability provided by Gaussian Mixtures is preferred. It is recommended that PCA anomaly detection algorithms are explored as future work for this dataset.

2.2 Deep Neural Networks for Anomaly Detection

Neural networks can be powerful tools in the field of anomaly detection that come at a cost of complexity (Singh, 2017). Recurrent Neural Networks (RNNs) are a family of neural networks for processing sequential data where the architecture dictates connections between hidden units (Goodfellow, Bengio, Courville, & Bengio, 2016).

LSTMs were developed to enable RNNs to learn long-term dependencies(Goodfellow et al., 2016). It

was discovered that vanilla RNNs do not work well for tasks where the relevant events (often referred to as teacher signals) were more than a few time steps old (Grosse, n.d.).

Vanilla RNNs are afflicted by the problem of exploding /vanishing gradients which prevents the model learning to converge to optimal parameters. LSTMs remove this problem by introducing a memory cell which ensures constant error flow and gating units to control information flow in and out of the cell (Gers, Eck, & Schmidhuber, 2002).

The original LSTM architecture worked well when the sequences in the input data had well defined boundaries but failed otherwise (Singh, 2017). The remedy was forget gates, which automatically refreshed LSTM memory as needed. Since then a number of variations have been proposed, with the most prominent being the use of peephole connections, which enable LSTMs to learn precise intervals. However, all this sophistication comes at the price of added complexity. It is widely acknowledged that LSTMs are difficult to train and optimize, as there are numerous parameters that need to be tuned. This is another indication that the standard LSTM can be quite overwhelming and we will most likely see simpler variants in future (Singh, 2017).

2.3 Model Selection Rationale

Even though LSTMs seem to be promising methods due to their power to learn non-linearities in the data, they come at a cost of being overwhelmingly complex, hard to debug, and are costly to run from a computational standpoint. LSTM's superiority does not carry over to certain simpler time series prediction tasks. This suggests to use LSTM only when simpler traditional approaches fail (Gers et al., 2002).

On another hand, Gaussian Mixture Models are very easy to understand and debut but are not as powerful at learning nonlinearities in the data, and only work under the assumption that random processes behave as Gaussian noise and exhibit mean ergodicity (Veracini, Matteoli, Diani, & Corsini, 2009).

Because of the arguments listed above, this paper will focus on analyzing the performance of two models:

- Gaussian Mixture Model (GMM)
- A Modified GMM explicitly designed to deal with data impurities such as nulls and time series

discontinuities

Chapter 3

Methodology

3.1 About the Dataset

The dataset under study contains traffic on the nodes of a Cloud-Hosted (AWS) Credit-Decisioning Engine that belongs to a major bank. Based on their nature, credit card authorizations get classified and end in one of 64 nodes, each of which is mapped to a final decision to approve or decline a transaction. Even though there are millions of transactions that flow through this decision tree every month, the 64 nodes have varying amounts of traffic. Some of these nodes barely get any traffic, while others register thousands or millions of transactions per day.

The dataset consists of 419 training examples and 64 features, each of these features are referred to as Nodes. Each training example corresponds to a specific calendar day.

The anomaly detection problem will be presented and we will discuss how this can be applied to our dataset. After that, we will present how we can test for normality, which is the assumption that must be true for us to use GMMs. Then we will introduce the theory behind Gaussian Distributions and Multivariate Gaussian Distributions and see how we can use the specific problem under study.

Finally, a section on Algorithm Evaluation will be presented, which explains how we will measure success across different algorithms given the unbalanced nature of our dataset - disproportionately more negative than positive examples.

3.2 The Anomaly Detection Problem

Consider a system with a set of n features $x_1, x_2, ...x_n$ that belong to a dataset represented by m training examples $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ where each of the elements in the dataset represents a time unit, such as a day (Ng, 2017a). We want to know whether a new data point x_{test} belongs to the dataset or not(Veracini et al., 2009). For this, we build a model to find the probability that a new dataset element belongs to the dataset and we denote it by $p(x)$ (Ng, 2017a).

Hence,

If:

$$p(x_{test}) < \epsilon \longrightarrow \text{Flag Anomaly} \tag{3.1}$$

Else if:

$$p(x_{test}) \geq \epsilon \longrightarrow \text{Not an Anomaly} \tag{3.2}$$

where:

- x_{test} is the new data point to be tested
- ϵ is a threshold that we use to decide when to flag something as an anomaly
  - Try different values of ϵ in order to maximize the F1-score.
  - The best ϵ will be the one that best generalizes on the cross-validation and testing sets.

3.3 Algorithm Evaluation

To evaluate the performance of this algorithm, we have obtained labels from an expert in the dataset who has categorized the detected anomalies into True Positives, True Negatives, False Positives, and False Negatives. We classify each data point as follows:

- $y_{true} = 1$ is a true anomaly
- $y_{true} = 0$ is not an anomaly
- $y_{pred} = 1$ is a predicted anomaly by the model
- $y_{pred} = 0$ is a predicted non-anomalous point by the model

Note that the labels were ONLY used to assess the performance and NOT used at any point to train the model.

Because the number of data points that are anomalous ($y_{true} = 1$) are a lot less than the points that will not be anomalous ($y_{true} = 0$), classification accuracy would not be a good metric to evaluate the performance of the classifier. Instead, we should use metrics that allow us to compute the fraction of true positives to false positives while computing the precision and recall of the algorithm, which can be achieved by metrics like F1-score. To calculate the F1-score, we need the following equations:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.3}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.4}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

The number of negative data points (non-anomalous) heavily outweighs the number of positive examples (anomalous). This will allow us to fit a normal distribution to the data and calculate $p(x)$.

3.4 Testing for Normality

A key assumption for one to be able to use the Gaussian Mixture Model is that the data must be sampled from a normal distribution. To assess this, we used the Shapiro-Wilk test for normality (Sarkadi, 1975).

Q-Q or quantile-quantile plots have the actual data on the y-axis with the values ordered from smallest to largest. In the case of our data, our values have been scaled to be limited between 0 and 1 for the y-axis. The x-axis shows what one would expect for the same mean and standard deviation as the data.

If the data are normally distributed, the points should follow along the straight line, especially in the middle of the distribution (Shapiro & Wilk, 1965). The slope of the line is the standard deviation of the normal distribution. If the points do not follow the straight line, then the distribution is not normal.

The Shapiro-Wilk Test is a formal test of the null hypothesis that one sample is drawn from the a normal distribution (Shapiro & Wilk, 1965). This test calculates a statistic called W (MIT, n.d.).

$$W = \frac{\left(\sum a_i x_i\right)^2}{\sum (x_i - \bar{x})^2} \tag{3.6}$$

The numerator is the slope of the Q-Q plot squared. If the data follows a normal distribution, the numerator should be an estimate of the population variance. The denominator is also an estimate of the population variance. Therefore, if the null hypothesis is true and the data are actually normal, then W should equal 1.

- If $W < 1$, the distribution may be significantly different from normal (p value determined from empirical distribution)

Note that this test has little power to reject the null hypothesis when the sample size is small and it is very likely to reject it when the sample size is very large and there are small deviations from normality.

3.5 Gaussian Distributions

If the data is normally distributed, one can approximate its distribution with a Gaussian curve, which is centered at μ an with a variance σ^2.

In mathematical notation, we say

$$x \sim \mathcal{N}(\mu, \sigma^2) \tag{3.7}$$

To calculate the probability that a point belongs to that distribution:

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2 * \pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.8}$$

To determine whether we can treat our data as a normal distribution, we need to evaluate its skewness, which is 0 in a perfectly normal distribution. Therefore, the further away from 0 it is, the more non-normal the distribution is. As an additional test for normality, we will use Shapiro-Wilk tests.

3.6 Multivariate Gaussian Distribution

The idea behind a Multivariate Gaussian distribution is to model $p(x)$ taking into account the contribution all individual features x_1, x_2, ... x_n. In our case, each feature corresponds to a different node.

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{(n/2)}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{3.9}$$

where:

- $\mu \in \mathbb{R}^n$ is a vector
- $\Sigma \in \mathbb{R}^{n\times n}$ is the Covariance matrix
- n is the number of features
- m is the number of examples

3.6.1 Parameter estimation problem

Given training set $\left\{x^{(1)}, x^{(2)}, ..., x^{(m)}\right\}$ we can calculate μ and Σ in the following way:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{3.10}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T \tag{3.11}$$

These two parameters μ and Σ will be computed for each of the 64 features of our dataset (nodes). We will use the μ and Σ from equations 3.10 and 3.11 respectively to plug them into equation 3.9.

We will flag an anomaly if $p(x_{test}) < \epsilon$. Note that the Multivariate model automatically captures correlation between features, but it is computationally more expensive because it is required to compute the inverse of the covariance matrix Σ. In addition, we must have $m > n$ for the covariance matrix to be invertible (number of examples greater than the number of available features). A good rule of thumb is to use multivariate Gaussian models when we have $m \geq 10n$ (Ng, 2017b).

The covariance matrix can be singular in two cases: if $m < n$ or if you have redundant features - a feature being a linear combination of other features.

Chapter 4

Analysis and Results

4.1 Time Series Analysis of Empirical Data

Figures 4.1, 4.2, 4.3, and 4.4 below shows the time series behaviour of each of the nodes in the decisioning engine. We can see that the date axes are different because every node corresponds to a rule in the decisioning algorithm and as credit policies change, rules may have different beginning and ending dates.

The data has been scaled using a MinMax scaler to be have comparable axes across the different nodes ranging from 0 to 1. The formula used for the MinMaxScaler is the following:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

It is also important to notice the aspect on seasonality, which is evident on time series such as the one on Nodes 2, 3, 15, and 17 (cyclo-stationary random processes). Seasonality becomes less evident on Nodes 4, 5, 22, 29 as an example. Sharp spikes in Node traffic such as Node 33 and Node 36 are due to anomalous behaviour, which can be driven by system malfunction, change in policy, or data issue (nulls, duplicates, or corrupted data files).

As we can see, not every node can be modeled as Gaussian noise. To evaluate which nodes we expect to be easily modelled as such, we will use statistical methods such as measuring the skew and the Shapiro-Wilk Test, which has been found to be the most powerful test in most situations (Shapiro & Wilk, 1965).

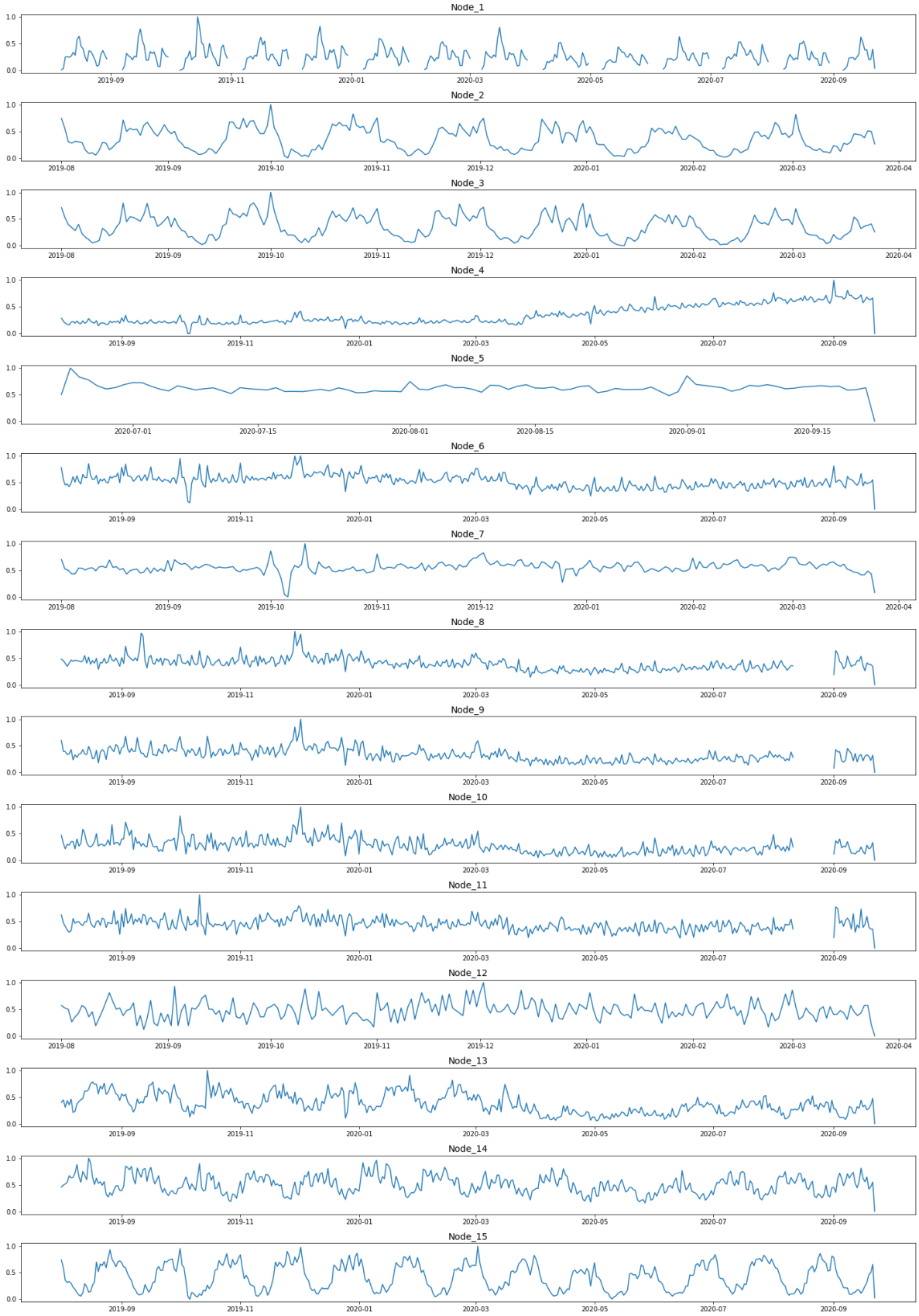


Figure 4.1: Behaviour of nodes 1-15 in decisioning engine

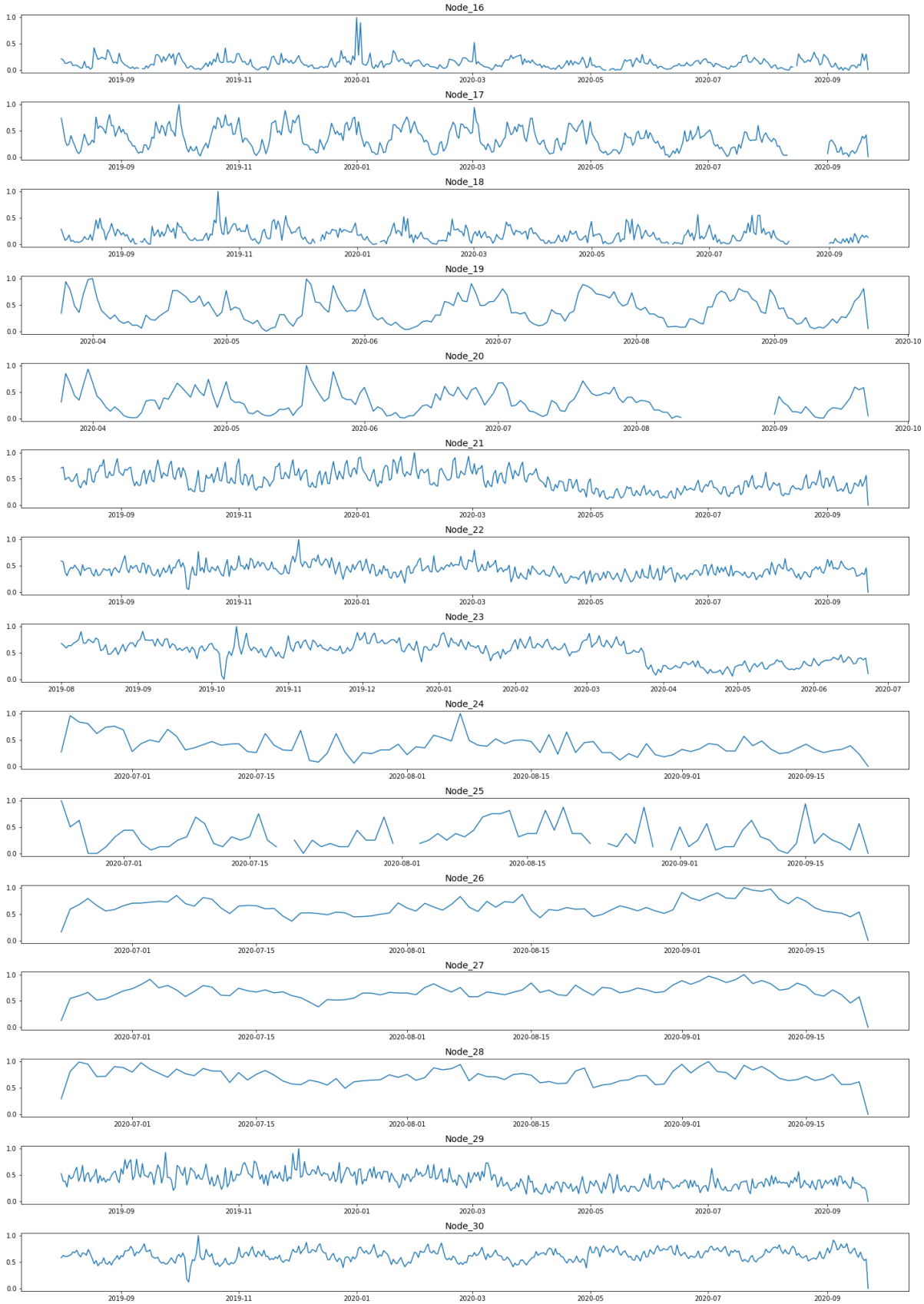


Figure 4.2: Behaviour of nodes 16-30 in decisioning engine

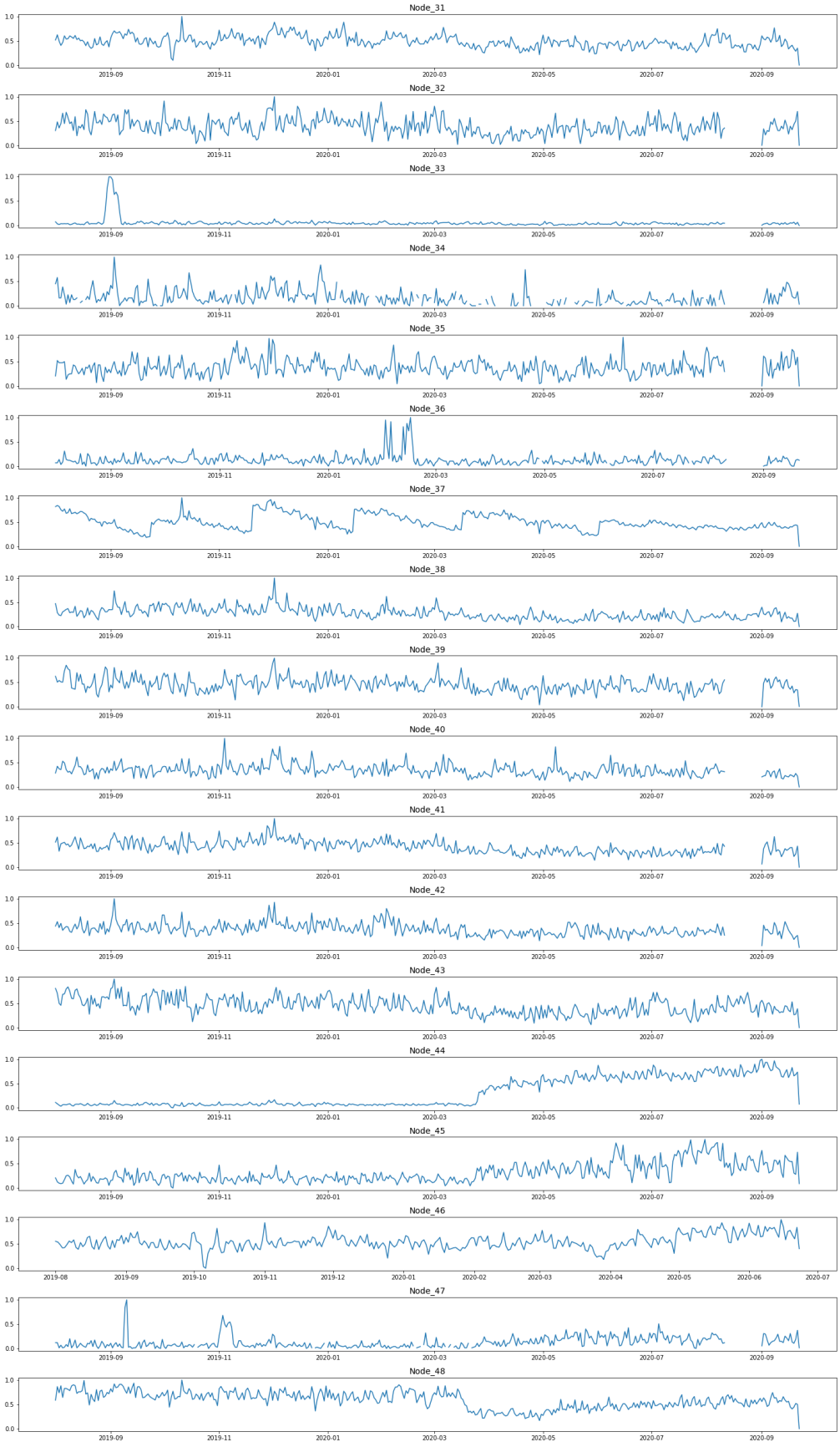


Figure 4.3: Behaviour of nodes 31-48 in decisioning engine

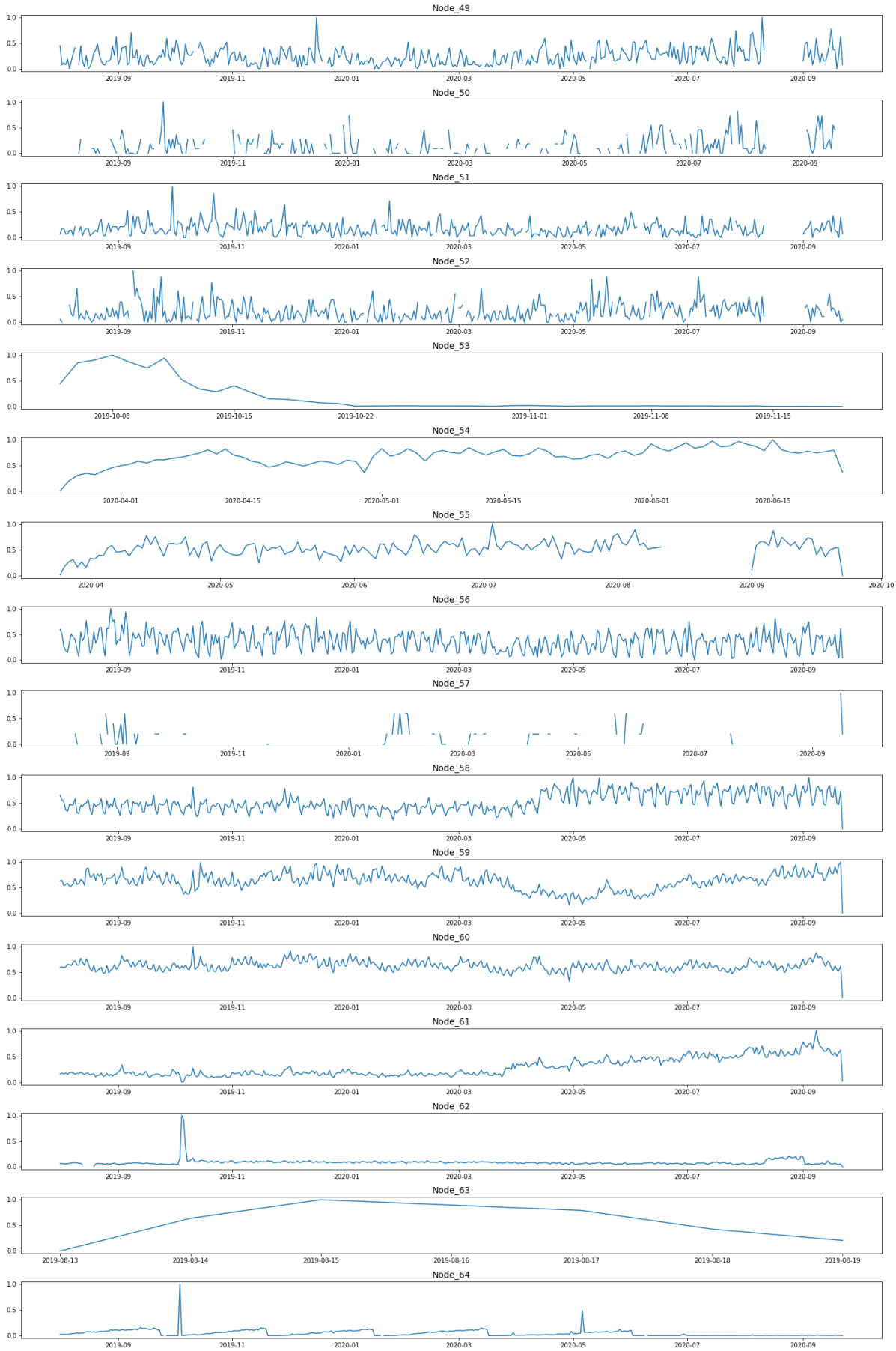


Figure 4.4: Behaviour of nodes 49-64 in decisioning engine

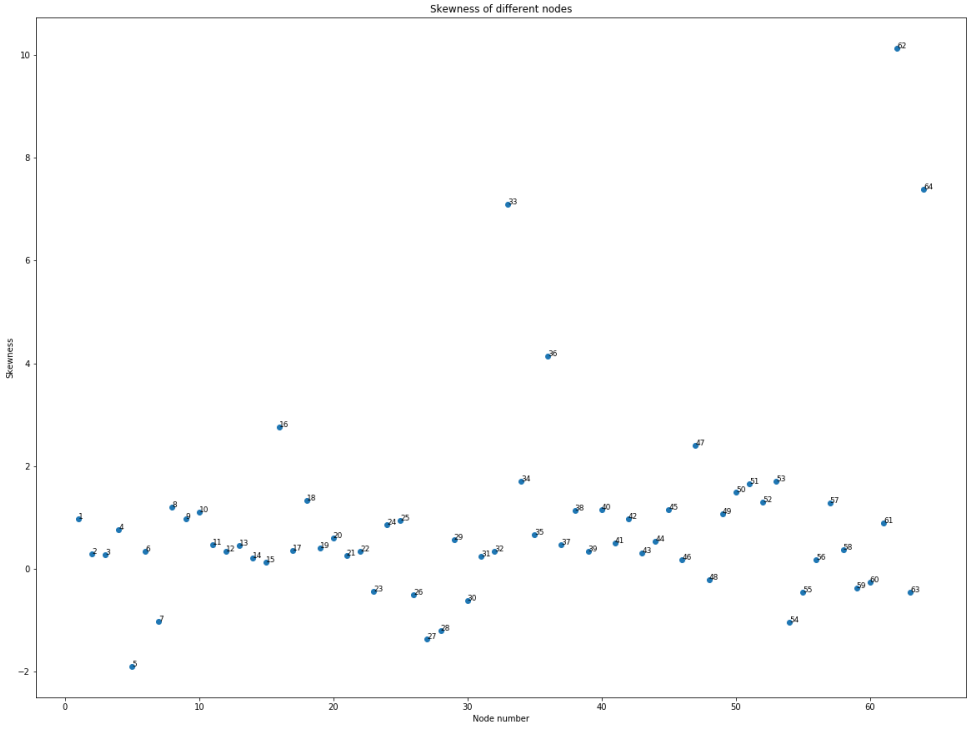


Figure 4.5: Comparing the distribution of the skewness of each node in the decisioning engine. Annotations show the node number.

The Figure 4.5 shows that while the traffic in most nodes has a skewness centered around zero and can therefore be modelled as Gaussian noise, other nodes were far from following a Gaussian distribution with skewness levels close to 10. More precisely, 39/64 nodes have a skewness lower than 1, 15/64 have a skewness between 1 and 2, and 10 have a skewness greater than 2. One can expect a model like Gaussian Mixtures to work well on nodes that follow closely a Gaussian distribution.

A natural suspicion is to assume that skewness heavily deviated from zero can be due to low sample size. However, as we can see on Figure 4.6, there was no evidence that the deviation of skewness was determined by the sample size, i.e. volume of requests on a given node.

4.1.1 Data Preparation

The dataset was normalized using Scikit learn's `StandardScaler`, which subtracts the mean and divides over the variance for each of the features. In other words, the standard score of a sample x is calculated

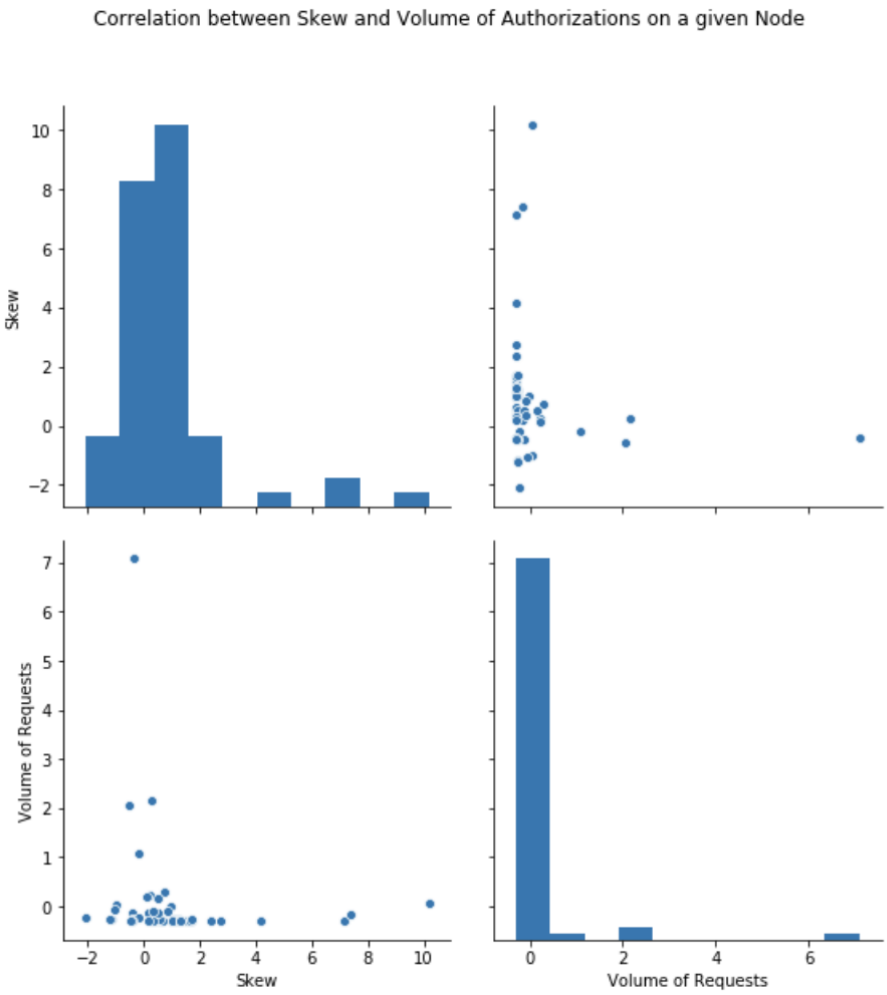


Figure 4.6: Comparing the node skewness with sample size. We can see that the volume of requests does not dictate skewness and viceversa.

as:

$$z = \frac{x - \mu}{\sigma} \tag{4.2}$$

The dataset under study is composed of 419 examples that correspond to days in the time series and 64 features, which correspond to nodes in the decisioning engine.

The dataset was split in 80% training and 20% testing, with `X_train` holding 335 examples and `X_test` holding 84 examples. Since it is a time series, the training set was used for the first 335 days and the testing set was the latest 84 days.

Note that while the strict requirement to use Mixture of Gaussians is to have more training examples (which we do) than features, it is recommended that this ratio is $10:1$ for optimal performance. However, that ratio is only $335/64 \approx 5:1$, which will likely limit the performance of this model on a dataset with a larger sample size.

4.1.2 Testing for Normality

We want to know whether data follows a normal distribution or not and we will use the Shapiro-Wilk test for this. We can see the results below. Note that the QQ plots are in agreement with Figure 4.8.

The procedure of going from a time series to fitting a Gaussian to its histogram and then assessing normality can be observed on Figure 4.7.

Q-Q or quantile-quantile plots have the actual data on the y-axis with the values ordered from smallest to largest. In the case of our data, it has been scaled to be between 0 and 1 so those are the limits for the y-axis. The x-axis shows the expected or theoretical normal quantiles, which are just a fraction of points smaller than a given value. The x-axis is what you would expect for the same mean and standard deviation as the data.

The nodes that had the lowest Shapiro tests are 33, 62, 64, 53, 36, and 5 starting from low to high Shapiro-Wilk statistics. We can see that these nodes that correspond to low Shapiro-Wilk statistics also deviate from the straight line on the QQ plots, as expected. However, out of the aforementioned nodes,

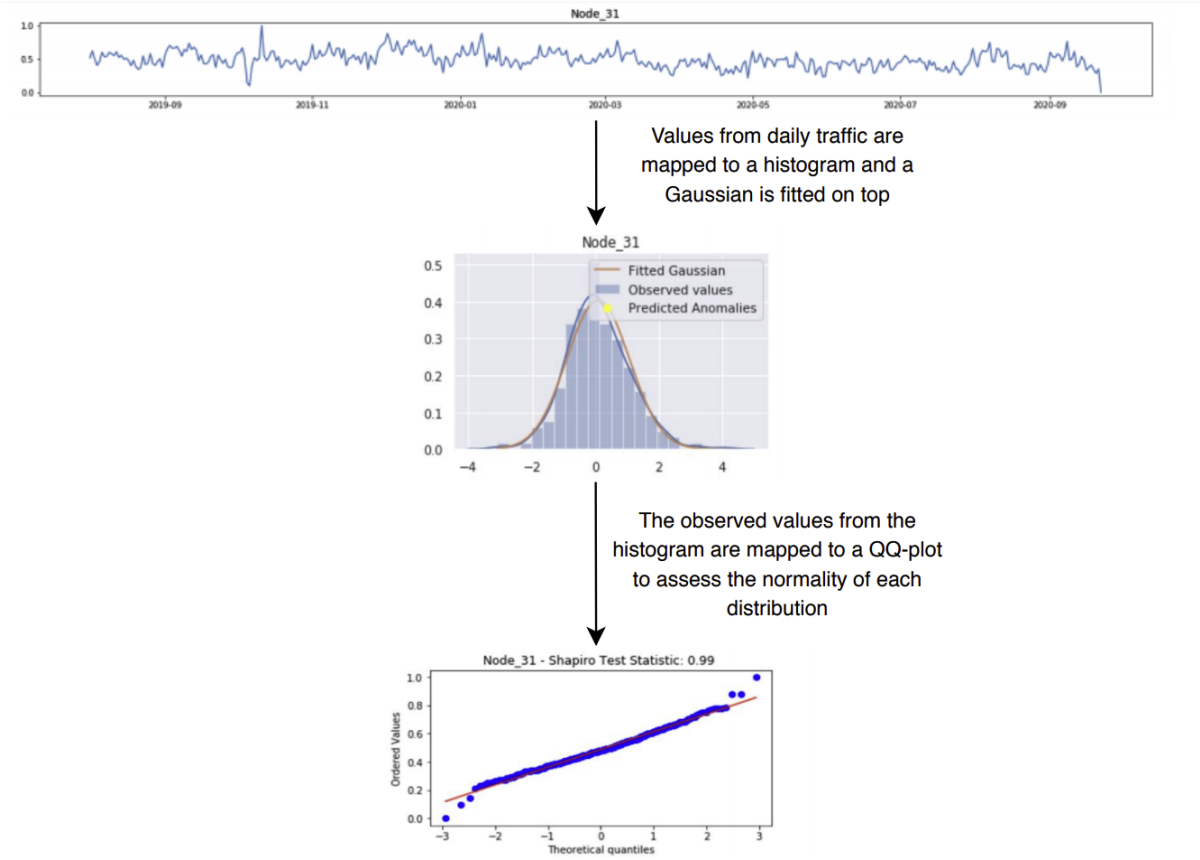
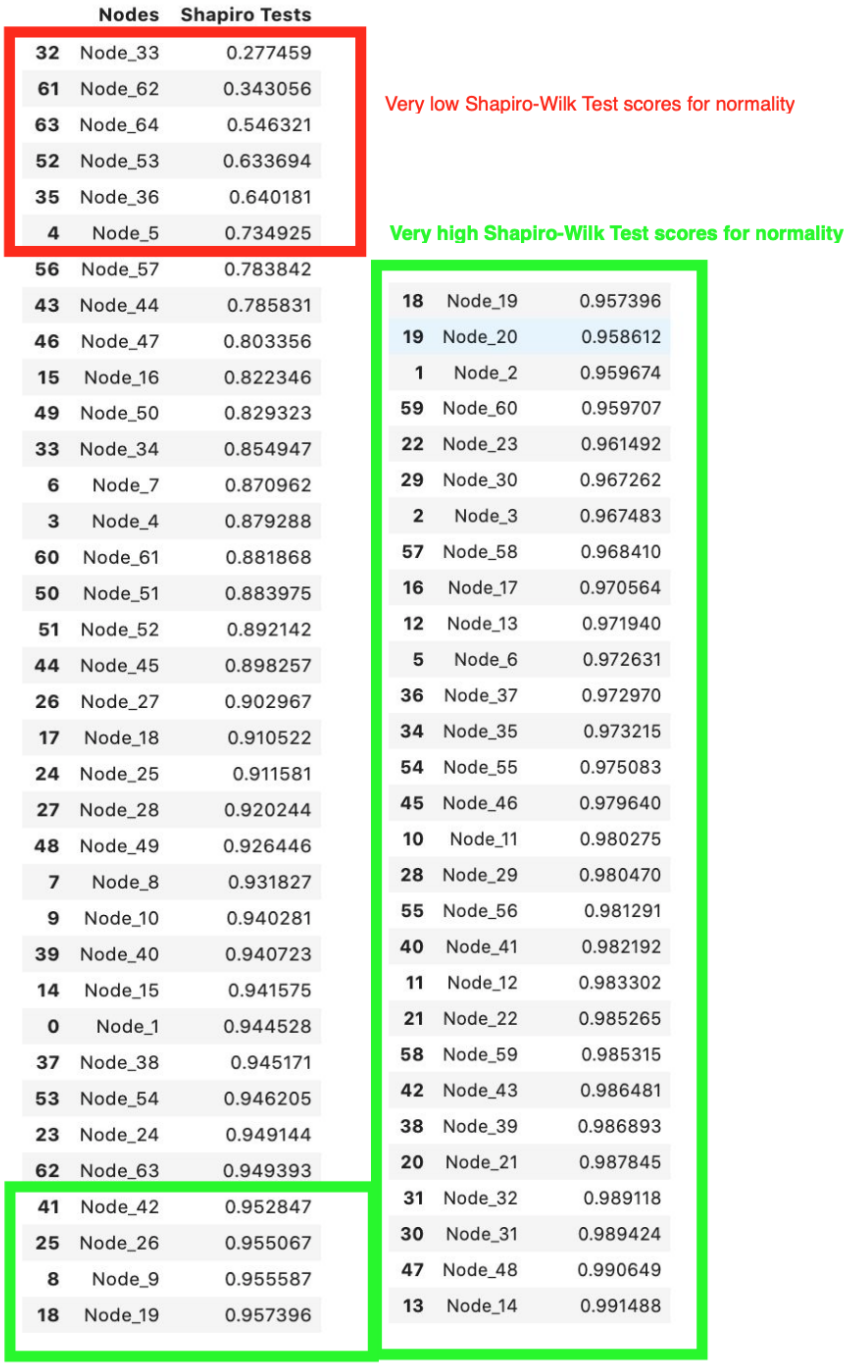


Figure 4.7: This Figure shows the flow of how traffic data on a single node maps to a normally distributed number of incoming requests per day and then a test of normality is performed.



	Nodes	Shapiro Tests
32	Node_33	0.277459
61	Node_62	0.343056
63	Node_64	0.546321
52	Node_53	0.633694
35	Node_36	0.640181
4	Node_5	0.734925
56	Node_57	0.783842
43	Node_44	0.785831
46	Node_47	0.803356
15	Node_16	0.822346
49	Node_50	0.829323
33	Node_34	0.854947
6	Node_7	0.870962
3	Node_4	0.879288
60	Node_61	0.881868
50	Node_51	0.883975
51	Node_52	0.892142
44	Node_45	0.898257
26	Node_27	0.902967
17	Node_18	0.910522
24	Node_25	0.911581
27	Node_28	0.920244
48	Node_49	0.926446
7	Node_8	0.931827
9	Node_10	0.940281
39	Node_40	0.940723
14	Node_15	0.941575
0	Node_1	0.944528
37	Node_38	0.945171
53	Node_54	0.946205
23	Node_24	0.949144
62	Node_63	0.949393
41	Node_42	0.952847
25	Node_26	0.955067
8	Node_9	0.955587
18	Node_19	0.957396

18	Node_19	0.957396
19	Node_20	0.958612
1	Node_2	0.959674
59	Node_60	0.959707
22	Node_23	0.961492
29	Node_30	0.967262
2	Node_3	0.967483
57	Node_58	0.968410
16	Node_17	0.970564
12	Node_13	0.971940
5	Node_6	0.972631
36	Node_37	0.972970
34	Node_35	0.973215
54	Node_55	0.975083
45	Node_46	0.979640
10	Node_11	0.980275
28	Node_29	0.980470
55	Node_56	0.981291
40	Node_41	0.982192
11	Node_12	0.983302
21	Node_22	0.985265
58	Node_59	0.985315
42	Node_43	0.986481
38	Node_39	0.986893
20	Node_21	0.987845
31	Node_32	0.989118
30	Node_31	0.989424
47	Node_48	0.990649
13	Node_14	0.991488

Figure 4.8: Ranking of the nodes from lowest to highest Shapiro-Wilk Score. The highest the score, the most likely it is to be sampled from a normal distribution.

only 33, 53 and 36 seem to have a strongly different distribution, since the QQ plots show significantly different distance from the midpoint of the straight line. It is important to note that deviations from this straight line near the edges is less relevant.

More than half of the nodes exhibit 90% or higher probability that the data was sampled from a normal distribution, which is very promising and compatible with anomaly detection algorithms such as Gaussian Mixtures. For this reason, we will carry the assumption that the traffic on every node behaves as Gaussian noise.

4.2 Simple Multivariate Gaussian Mixture Model

4.2.1 Fitting the Gaussians on Training Set

A Gaussian was fitted to each of the features in the training set by computing their mean and variance using equations and yielding results as shown on Figure 4.9. Note that the number of data points for each of the features is variable since time intervals where the node was not active were excluded. This was made in an effort to remove bias from an algorithm that would over-learn zeros, which would pass as "average" behaviour since the normalized time series are centered at zero.

Each node votes to determine whether there is an anomaly on a given day since the product of all the probabilities is calculated as shown on equation 4.3. Note that 64 is the number of features (nodes). Each of the individual probabilities are calculated using equation 3.9. The product of probabilities is referred to as global probability later in this article.

4.2.2 Predicting Anomalies on Training Set

The list of True Anomalies was manually created by an expert user in the credit card decisioning engine who used their judgement to determine the days when system anomalies had been present. Note that the definition of an anomaly remains a rather subjective concept and this is what the performance of the algorithm is judged upon.

After finding a global probability for each day, equations 3.1 and 3.2 are used to determine whether a given day should be flagged as anomalous.

As part of the grid search, the value of ϵ is manually tuned until the highest value of F1-score is obtained. Note that F1-score was used to evaluate the performance of the algorithms on this paper because of the unbalanced nature of positive and negative examples in an anomaly detection problem.

$$p(x; \mu, \Sigma) = p(x_1) \times p(x_2) \times ...(x_{64}) \tag{4.3}$$

where the subindex of x represents the node (feature) number.

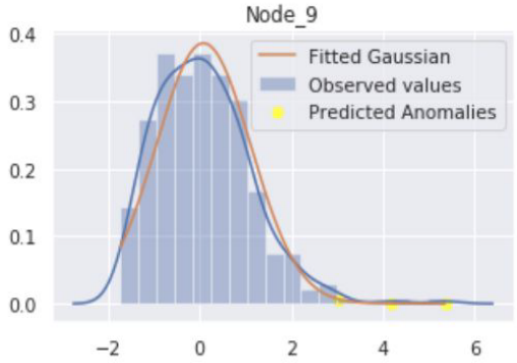


Figure 4.9: Example showing how a Gaussian was fitted to the histogram of observed data for a given node using the training data to find the mean μ and standard deviation σ.

4.2.3 Tuning the Model with the Training Set

A grid search was perform to search for the highest F1-score. For each iteration of the grid search, a different value for ϵ was explored. The **threshold** value shown on Figure 4.10 is really a multiplier on the standard deviation σ of the global probabilities.

Note that there were 31 total discrepancies from the True vs Predicted probabilities from a total of 335. Even though this represents 99% accuracy, this is not a good metric to measure the performance because the set is heavily unbalanced and therefore the majority of the examples on the training set are non-anomalous. There are 9 false positives vs 23 false negatives, implying that the algorithm is more likely to miss an anomaly than to report false alarms.

Upon manually sampling the missed discrepancies, it was noticed that most of the anomalies that were missed were due to null data files that end up in zero traffic for nodes on specific days. As an example, one can take a look at nodes 33, 34, 35, 39, 40, 41, and 42 on Figure 4.3.

Issues stemming from null data files are real concerns in practice since this may cause millions of transactions to be declined per day due to a data issue. An improvement to this algorithm to explicitly detect unusual pauses on node traffic is explored in section 4.3.

4.2.4 Validating Results on Testing Set

The model was fitted and tuned using data from the training set and its performance was later verified on the testing set. As mentioned above, because it is a time series, the training set was comprised of the

	Set	f1	precision	recall	threshold	method	num_dicrepancies	tn	fp	fn	tp
0	Training Set	0.812281	0.131148	0.275862	0.01	Simple Gaussian Mixture	74.0	253.0	53.0	21.0	8.0
1	Training Set	0.849320	0.179487	0.241379	0.02	Simple Gaussian Mixture	54.0	274.0	32.0	22.0	7.0
2	Training Set	0.868657	0.241379	0.241379	0.03	Simple Gaussian Mixture	44.0	284.0	22.0	22.0	7.0
3	Training Set	0.884634	0.333333	0.241379	0.04	Simple Gaussian Mixture	36.0	292.0	14.0	22.0	7.0
4	Training Set	0.892937	0.411765	0.241379	0.05	Simple Gaussian Mixture	32.0	296.0	10.0	22.0	7.0
5	Training Set	0.892937	0.411765	0.241379	0.06	Simple Gaussian Mixture	32.0	296.0	10.0	22.0	7.0
6	Training Set	0.895059	0.437500	0.241379	0.07	Simple Gaussian Mixture	31.0	297.0	9.0	22.0	7.0
7	Training Set	0.890349	0.400000	0.206897	0.08	Simple Gaussian Mixture	32.0	297.0	9.0	23.0	6.0

Figure 4.10: This table shows how the highest F1-score achieved was 0.895 for a threshold multiplier of 0.07

first 80% of the days and the testing set was comprised of the latest 20% of the days.

The probabilities of each node and each day being anomalous were calculated using equation 3.9 where μ and Σ were estimated using equations 3.10 and 3.11. To determine whether each point was anomalous, equations 3.1 and 3.2 were used - same as how it was done with Training set. The value for ϵ used was 0.07 as this was the value that achieved the highest F1-score for the training set as shown on Figure 4.10.

Figure 7.2.2 shows how the Gaussians fitted to the training test were adopted by the testing set.

Comparing predicted vs actual anomalies for the testing set generated the results on Figure 4.11. Note that the F1-score achieved for the testing set was 0.933, slightly higher than that of the training set on Appendix section 4.10. There were 9 total discrepancies, which were all false positives. There was a tendency to over-flag false alarms on the testing set (contrary to training set) but 9/9 true positives were detected. Similarly, 74/74 of true negatives were detected and no false negatives were flagged.

4.2.5 Interpretation of Results

While the results were not perfect, the algorithm performs fairly well at detecting anomalies on this dataset despite the non-linearities and the subjective nature of the definition of an anomaly. Further improvements would involve actively detecting no traffic on nodes caused by null data files, which will be explored in the next section.

Set	f1	precision	recall	threshold	method	num_dicrepancies	tn	fp	fn	tp
testing Set	0.933617	0.1	1.0	0.07	Simple Gaussian Mixture	9.0	74.0	9.0	0.0	9.0

Figure 4.11: This table shows how the highest F1-score for the testing set achieved was 0.933 for a threshold multiplier of 0.07

4.3 Modified Multivariate Gaussian Mixture Model

In this section we explore a modified version the Multivariate Gaussian Mixture Model used in the previous section to actively flag nulls in data files as anomalies. This modification is proposed because upon manual sampling of missed occurrences of anomalies (false negatives), it was found that they were mostly due to empty data files. These appear as zero traffic on the decisioning engine nodes and end up undetected because the time series are explicitly normalized to have a zero mean.

Note that this modification layer is added after all the steps from the previous section on the Simple Modified Multivariate Gaussian Mixture Model are performed. Therefore, there is no change to the probability density functions of each node.

4.3.1 Detection of Traffic Breaks on Nodes of Decisioning Engine

The goal of this modification of the Multivariate Gaussian Mixture Model is to detect time intervals with zero traffic on the nodes of the decisioning engine such as the ones shown on Nodes 39, 40, 41, and 42 between 2020-08 and 2020-09 as shown on Figure 4.3.

4.3.2 Results on Training Set

After all the anomalies were detected using the Simple version of the GMM, the dates where nulls in data files occurred were added to a list and then the `Predicted Anomaly` flag was set to 1 for those dates in the list as shown on Figure 4.12.

As shown on 4.13 the modified Multivariate Gaussian Mixture Model achieved a slightly higher F1-score of 0.919 against the 0.895 in the original version. The number of discrepancies dropped to 24/338 from 31/338.

4.3.3 Interpretation of Confusion Matrix on Training Set

There were 18 false negatives in this version of the model - in other words, true anomalies that the modified GMM model did not detect. This is still an improvement from the 23 false negatives in the Simple version of the model. The reason why these false negatives were flagged is that this type of anomaly is usually caused by a spike on single node, and if the rest of the nodes are behaving normally, the power of a

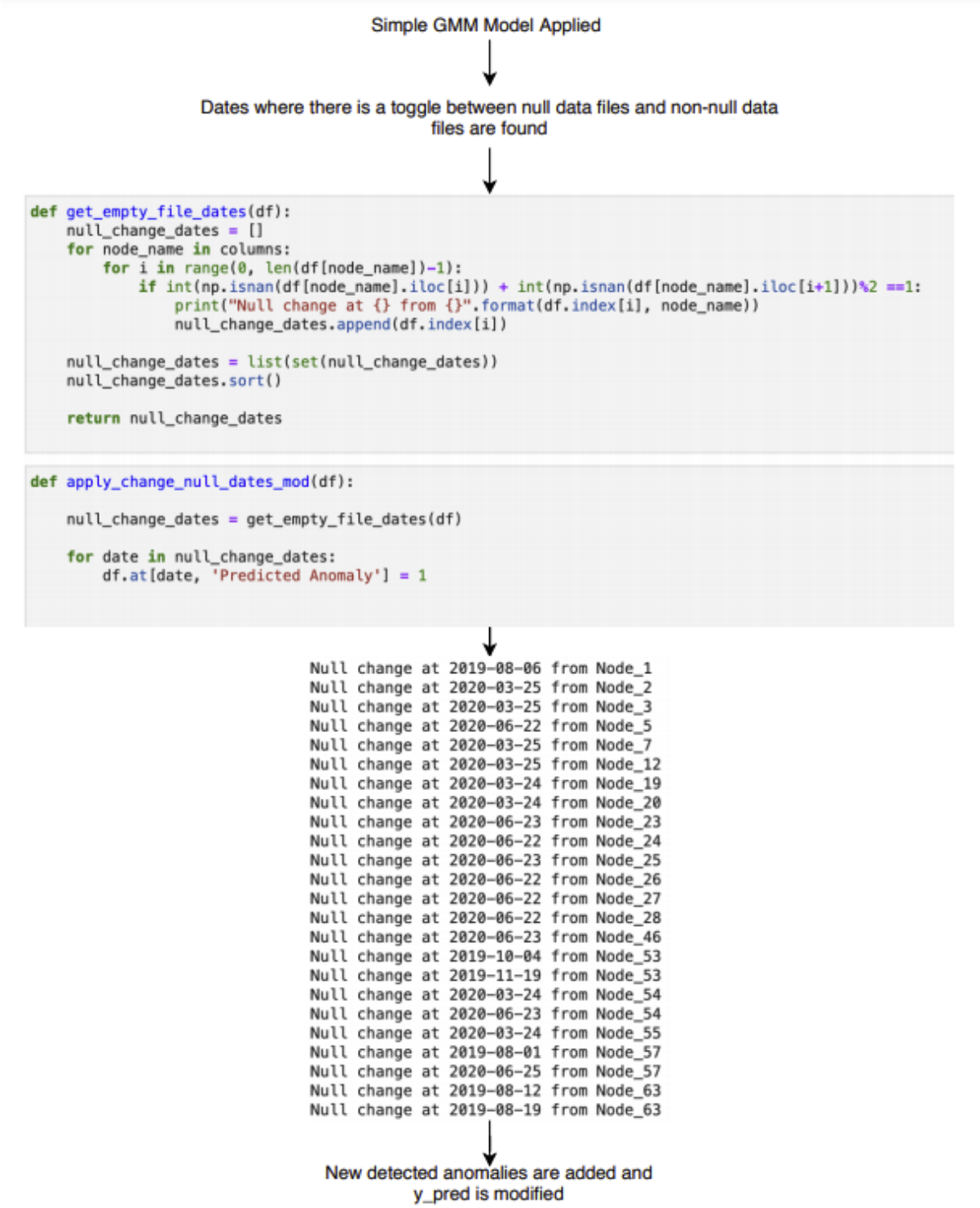


Figure 4.12: This table shows the flow of the proposed modification to the GMM to actively flag nulls in data files.

	Set	f1	precision	recall	threshold	method	num_dicrepancies	tn	fp	fn	tp
2	Training Set	0.821422	0.188406	0.448276	0.01	Modified GMM with empty data file detector	72.0	250.0	56.0	16.0	13.0
3	Training Set	0.862055	0.265306	0.448276	0.02	Modified GMM with empty data file detector	52.0	270.0	36.0	16.0	13.0
4	Training Set	0.882804	0.333333	0.448276	0.03	Modified GMM with empty data file detector	42.0	280.0	26.0	16.0	13.0
5	Training Set	0.897834	0.406250	0.448276	0.04	Modified GMM with empty data file detector	35.0	287.0	19.0	16.0	13.0
6	Training Set	0.906726	0.464286	0.448276	0.05	Modified GMM with empty data file detector	31.0	291.0	15.0	16.0	13.0
7	Training Set	0.906726	0.464286	0.448276	0.06	Modified GMM with empty data file detector	31.0	291.0	15.0	16.0	13.0
8	Training Set	0.908994	0.481481	0.448276	0.07	Modified GMM with empty data file detector	30.0	292.0	14.0	16.0	13.0
9	Training Set	0.905165	0.461538	0.413793	0.08	Modified GMM with empty data file detector	31.0	292.0	14.0	17.0	12.0
10	Training Set	0.907422	0.480000	0.413793	0.09	Modified GMM with empty data file detector	30.0	293.0	13.0	17.0	12.0
11	Training Set	0.909700	0.500000	0.413793	0.10	Modified GMM with empty data file detector	29.0	294.0	12.0	17.0	12.0
12	Training Set	0.907981	0.500000	0.379310	0.11	Modified GMM with empty data file detector	29.0	295.0	11.0	18.0	11.0
13	Training Set	0.912585	0.550000	0.379310	0.12	Modified GMM with empty data file detector	27.0	297.0	9.0	18.0	11.0
14	Training Set	0.914927	0.578947	0.379310	0.13	Modified GMM with empty data file detector	26.0	298.0	8.0	18.0	11.0
15	Training Set	0.917299	0.611111	0.379310	0.14	Modified GMM with empty data file detector	25.0	299.0	7.0	18.0	11.0
16	Training Set	0.917299	0.611111	0.379310	0.15	Modified GMM with empty data file detector	25.0	299.0	7.0	18.0	11.0
17	Training Set	0.917299	0.611111	0.379310	0.16	Modified GMM with empty data file detector	25.0	299.0	7.0	18.0	11.0
18	Training Set	0.917299	0.611111	0.379310	0.17	Modified GMM with empty data file detector	25.0	299.0	7.0	18.0	11.0
19	Training Set	0.919702	0.647059	0.379310	0.18	Modified GMM with empty data file detector	24.0	300.0	6.0	18.0	11.0
20	Training Set	0.919702	0.647059	0.379310	0.19	Modified GMM with empty data file detector	24.0	300.0	6.0	18.0	11.0
21	Training Set	0.919702	0.647059	0.379310	0.20	Modified GMM with empty data file detector	24.0	300.0	6.0	18.0	11.0
22	Training Set	0.919702	0.647059	0.379310	0.25	Modified GMM with empty data file detector	24.0	300.0	6.0	18.0	11.0
23	Training Set	0.915370	0.625000	0.344828	0.30	Modified GMM with empty data file detector	25.0	300.0	6.0	19.0	10.0

Highest F1 score achieved

Figure 4.13: This table shows the improved performance on the modified Gaussian Mixture Model, including detection for null data files.

single node to flag an anomaly becomes diluted and the threshold is not sensitive enough to detect them.

There were 6 false positives or *false alarms* in this version of the model. The bulk of them were introduced by data nulls that are actually non-anomalous due to the cyclo-stationary nature of some nodes, such as Node 1 as can be seen on 4.1. One can observe that there are days where this node gets no traffic at all and this gets caught by the data null detector layer added in this version of the model.

Low traffic nodes are also expected to flag false alarms on this version of the model such as on Node 19 on 4.2 (although breaks are so short that cannot be observed from the referenced Figure). This is because the traffic is so low that a value of zero incoming authorizations is not necessarily an anomaly.

4.3.4 Validating Results on Testing Set

The modified model performed better than the original version as shown on 4.14, reaching a maximum F1 score of 0.947, which is higher than the 0.933 achieved by the original version of the model as can be seen on 4.11.

	Set	f1	precision	recall	threshold	method	num_dicrepancies	tn	fp	fn	tp
31	Testing Set	0.919804	0.083333	1.0	0.07	Modified GMM with empty data file detector	11.0	72.0	11.0	0.0	11.0
32	Testing Set	0.926740	0.090909	1.0	0.08	Modified GMM with empty data file detector	10.0	73.0	10.0	0.0	10.0
29	Testing Set	0.947240	0.125000	1.0	0.18	Modified GMM with empty data file detector	7.0	76.0	7.0	0.0	7.0
30	Testing Set	0.947240	0.125000	1.0	0.19	Modified GMM with empty data file detector	7.0	76.0	7.0	0.0	7.0
28	Testing Set	0.947240	0.125000	1.0	0.20	Modified GMM with empty data file detector	7.0	76.0	7.0	0.0	7.0

Figure 4.14: This table shows the improved performance for the Testing Set as well on the modified Gaussian Mixture Model, which is able to detect nulls in data files.

4.3.5 Interpretation of Results

The Modified Multivariate Gaussian Mixture Model outperforms the Simple version with the added layer of detecting nulls in data files as well as time intervals with zero traffic. Nulls on data files may be a type of anomaly not present in every type of dataset but certainly applies to real systems in production. It was shown that this added layer increases the efficiency of the model because of how often data null anomalies take place.

Further improvements to decrease the number of false positives would include considering cyclic data nulls that are not necessarily anomalies as well as zeroes on nodes with low traffic.

Chapter 5

Future Work

Further improvements to decrease the number of false positives on the modified GMM would include removing cyclic data nulls as anomalies as well as zeroes on nodes with low traffic.

We also recommend to further explore architectures for LSTMs, which may be more promising at understanding non-linearities and cyclic behaviour in time series. The flagged an anomalies by LSTMs can be combined with the Modified GMM to produce an ensemble model which should at least perform as well as the Modified GMM.

Given the proven success of Principal Component Analysis (PCA) in multivariate anomaly detection, (Hyndman et al., 2015), we recommend to explore PCA-based anomaly detection algorithms for this dataset as wel.

Chapter 6

Conclusion

Gaussian Mixture Models work very well to detect anomalies on unlabeled data if the process is wide-sense stationary. GMMs are relatively easy to understand, trivial to implement and very quick to run. Although GMMs require that data is normally distributed with a reasonably large number of examples, we proved that these assumptions generally hold true on real production data and are therefore a good fit. Despite all the sample size limitations and data impurities, we were able to achieve an F1 score of 0.933 for the Simple GMM and 0.947 for the Modified GMM. While LSTMs may be more powerful in theory, they are prohibitively costly to implement and complex to debug for a well behaved wide-sense stationary process.

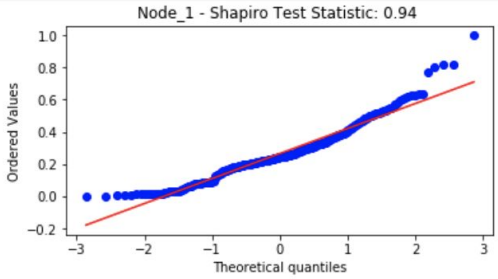
We recommend the use of Gaussian Mixture Models for unsupervised anomaly detection in wide-sense stationary processes with unbalanced datasets, such as to monitor the traffic of nodes on credit decisioning engines.

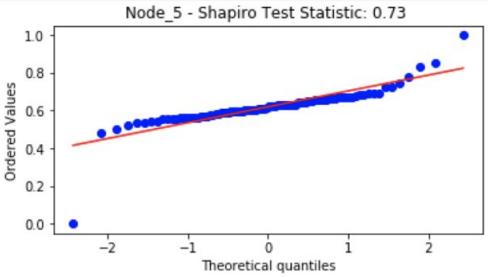
Chapter 7

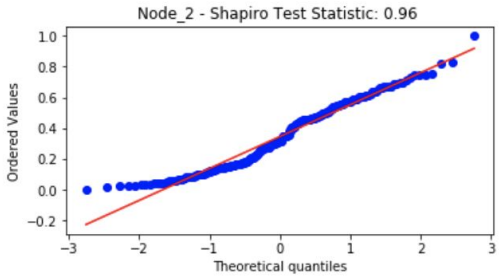
Appendix

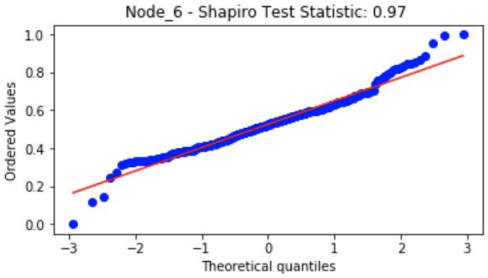
7.1 Testing for Normality

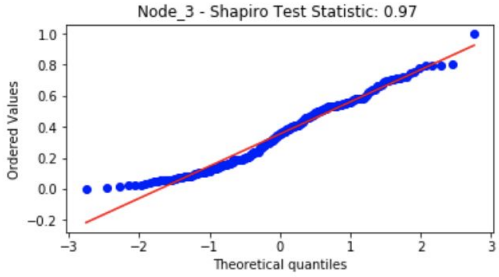
7.1.1 Shapiro-Wilk Plots

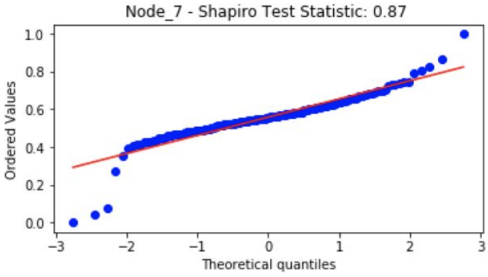

Node_1 - Shapiro Test Statistic: 0.94
Ordered Values
Theoretical quantiles

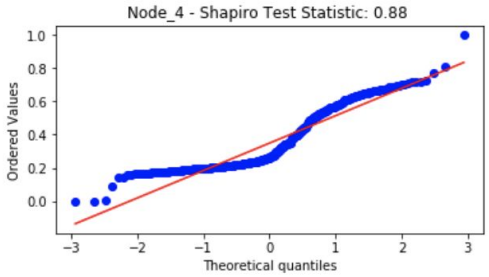

Node_5 - Shapiro Test Statistic: 0.73
Ordered Values
Theoretical quantiles

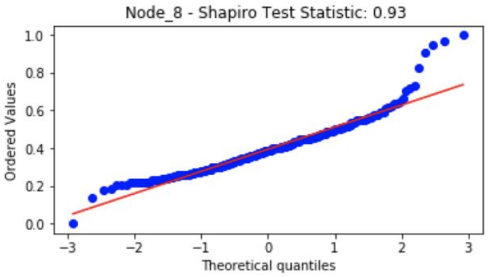

Node_2 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

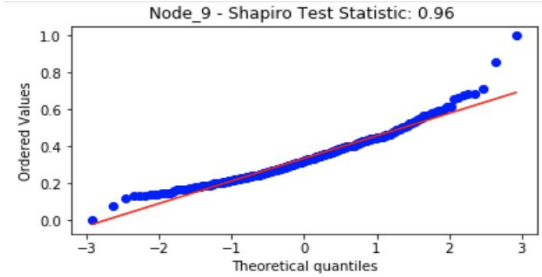

Node_6 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

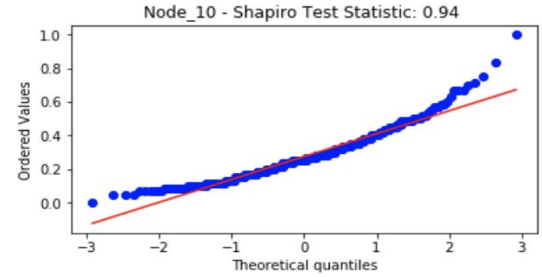

Node_3 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

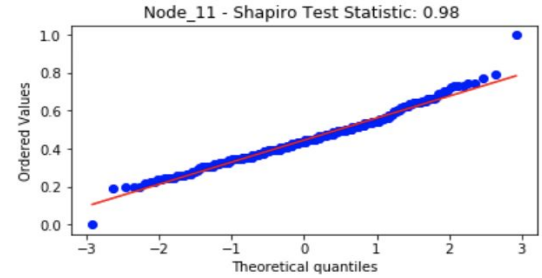

Node_7 - Shapiro Test Statistic: 0.87
Ordered Values
Theoretical quantiles

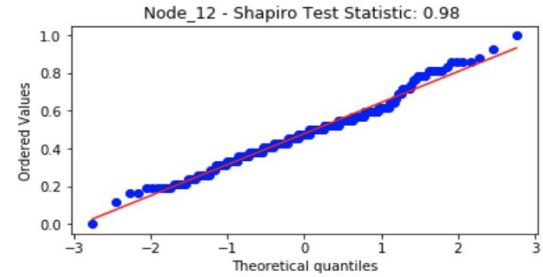

Node_4 - Shapiro Test Statistic: 0.88
Ordered Values
Theoretical quantiles

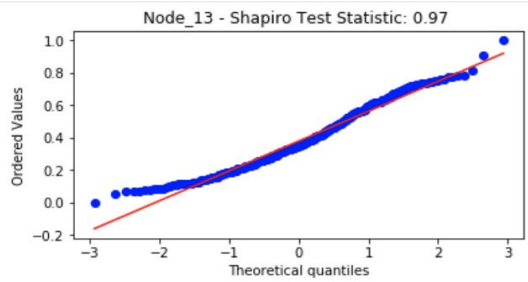

Node_8 - Shapiro Test Statistic: 0.93
Ordered Values
Theoretical quantiles

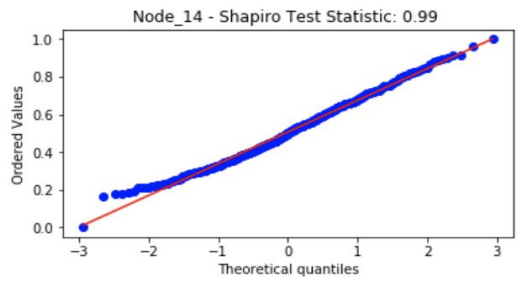

Node_9 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

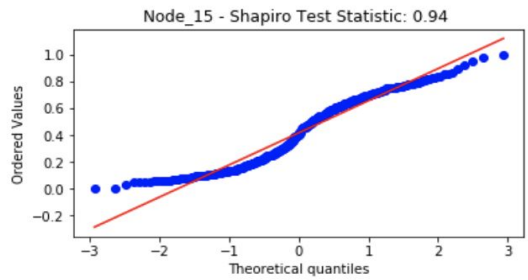

Node_10 - Shapiro Test Statistic: 0.94
Ordered Values
Theoretical quantiles

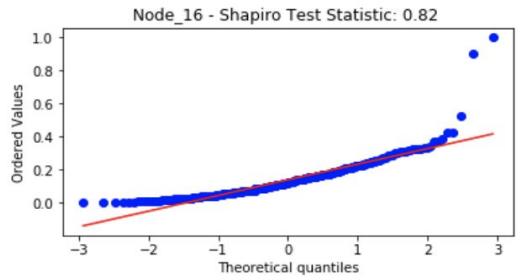

Node_11 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

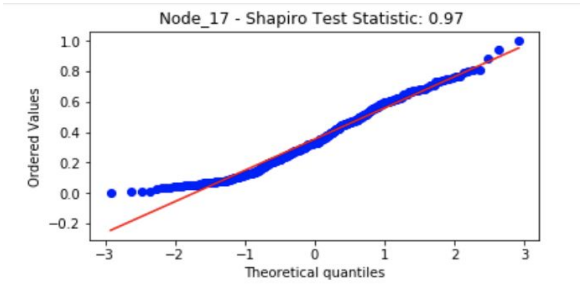

Node_12 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

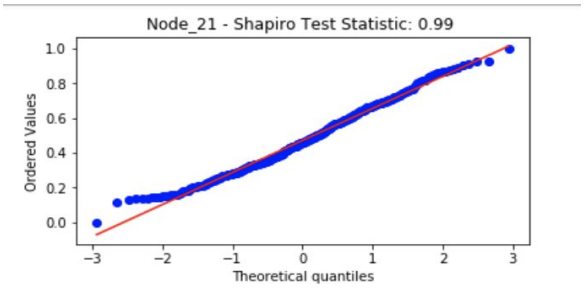

Node_13 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

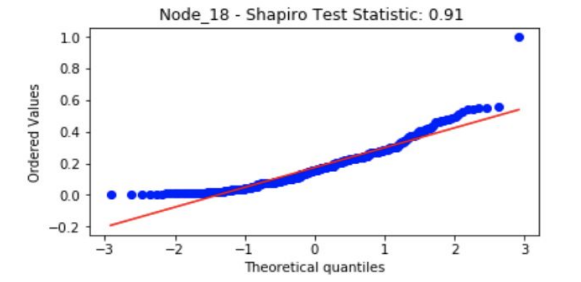

Node_14 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

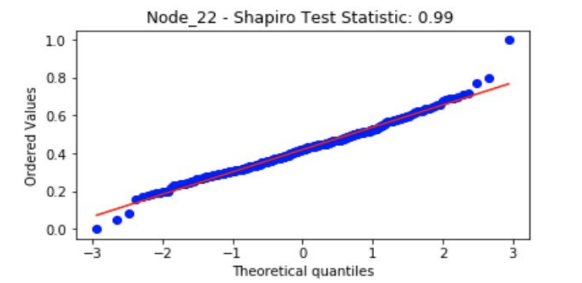

Node_15 - Shapiro Test Statistic: 0.94
Ordered Values
Theoretical quantiles

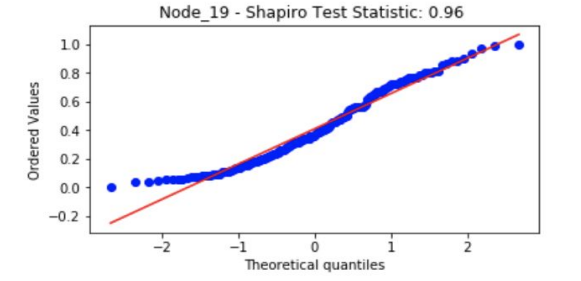

Node_16 - Shapiro Test Statistic: 0.82
Ordered Values
Theoretical quantiles

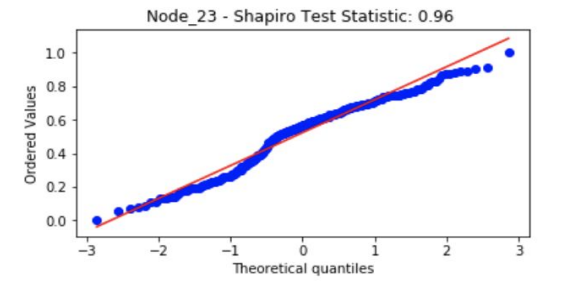

Node_17 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

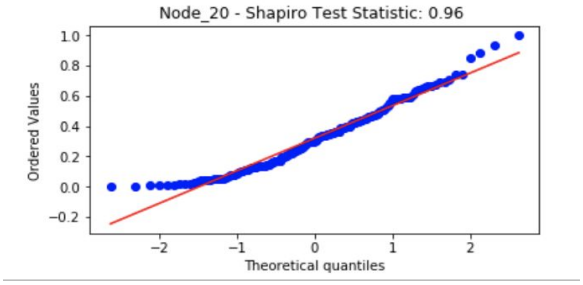

Node_21 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

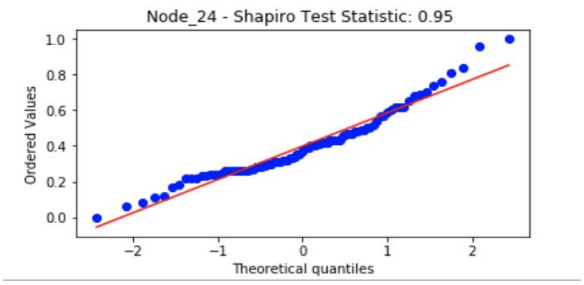

Node_18 - Shapiro Test Statistic: 0.91
Ordered Values
Theoretical quantiles

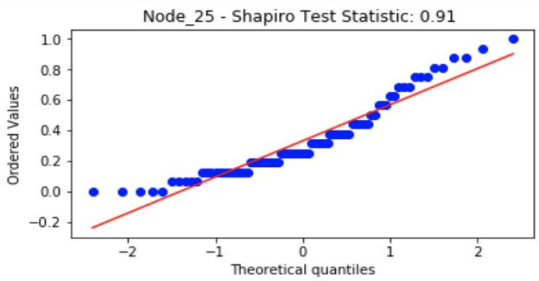

Node_22 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

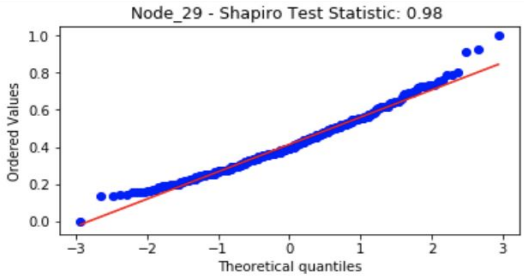

Node_19 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

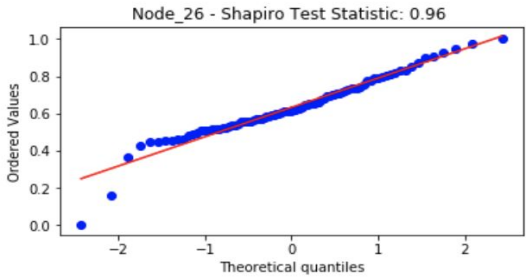

Node_23 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

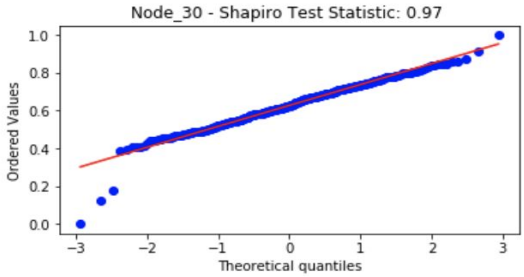

Node_20 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

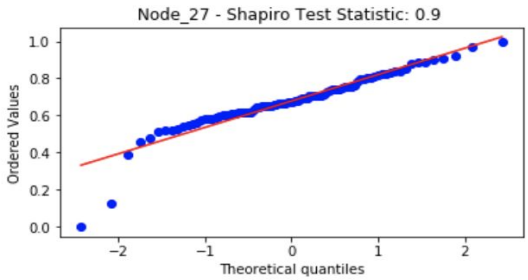

Node_24 - Shapiro Test Statistic: 0.95
Ordered Values
Theoretical quantiles

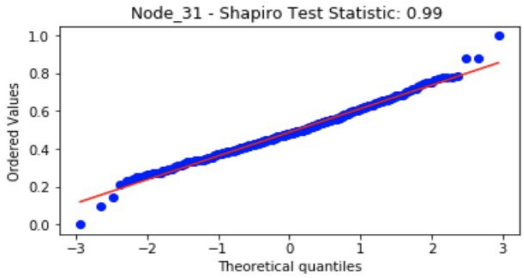

Node_25 - Shapiro Test Statistic: 0.91
Ordered Values
Theoretical quantiles

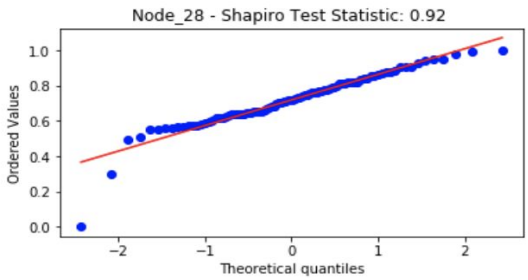

Node_29 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

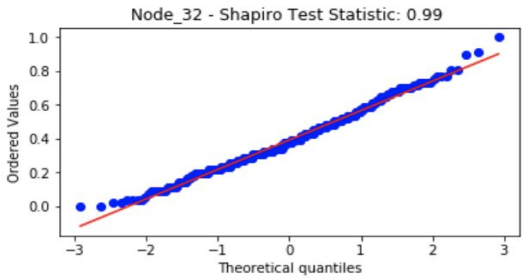

Node_26 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles

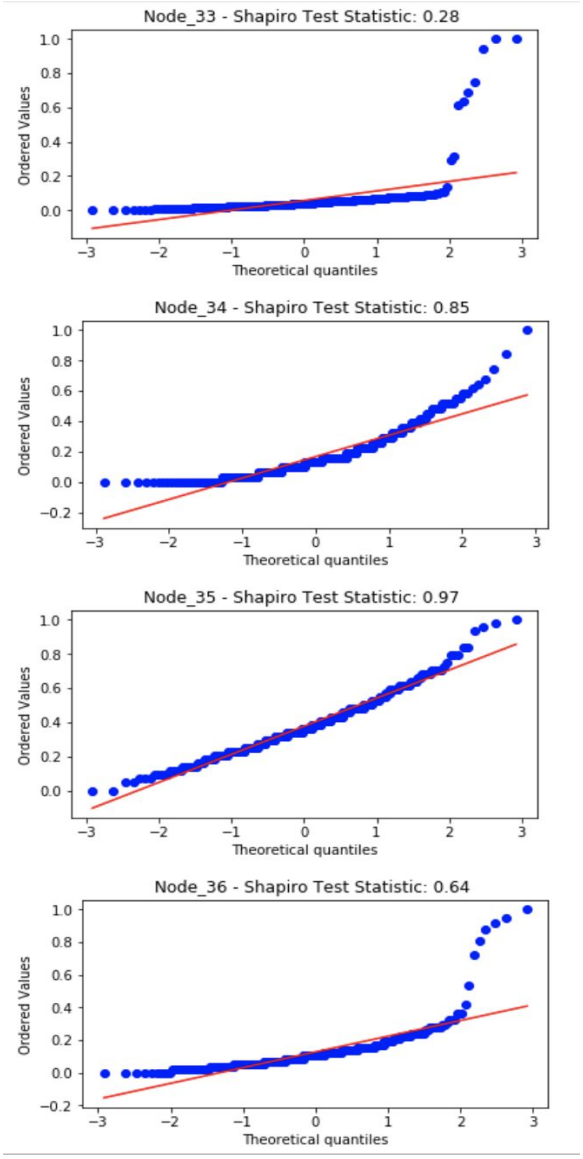

Node_30 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

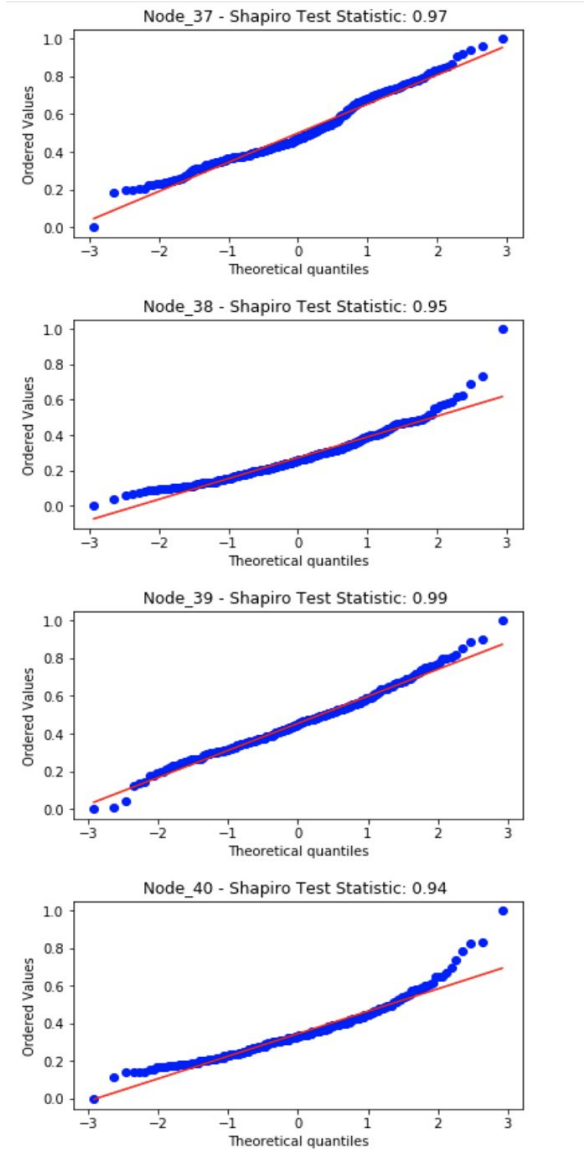

Node_27 - Shapiro Test Statistic: 0.9
Ordered Values
Theoretical quantiles

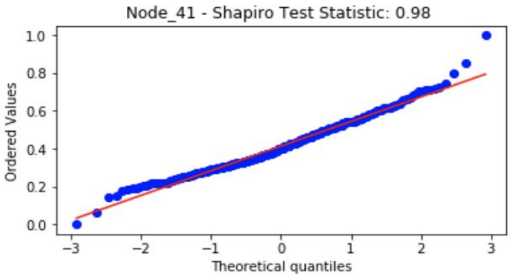

Node_31 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

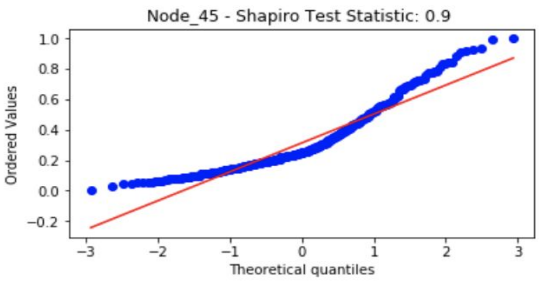

Node_28 - Shapiro Test Statistic: 0.92
Ordered Values
Theoretical quantiles

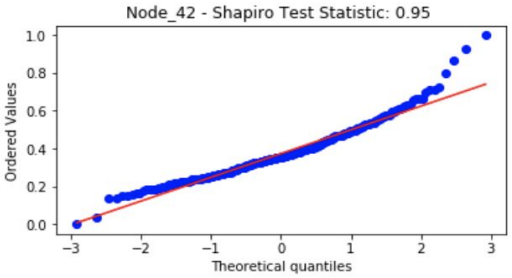

Node_32 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

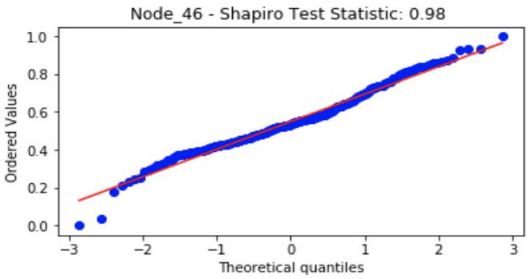

Node_33 - Shapiro Test Statistic: 0.28
Node_34 - Shapiro Test Statistic: 0.85
Node_35 - Shapiro Test Statistic: 0.97
Node_36 - Shapiro Test Statistic: 0.64
Ordered Values
Theoretical quantiles

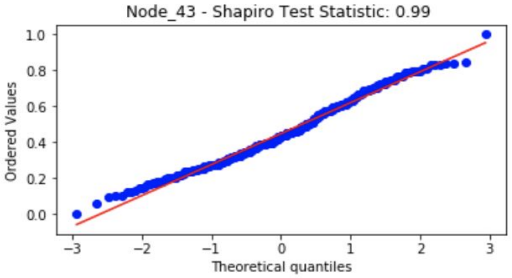

Node_37 - Shapiro Test Statistic: 0.97
Node_38 - Shapiro Test Statistic: 0.95
Node_39 - Shapiro Test Statistic: 0.99
Node_40 - Shapiro Test Statistic: 0.94
Ordered Values
Theoretical quantiles

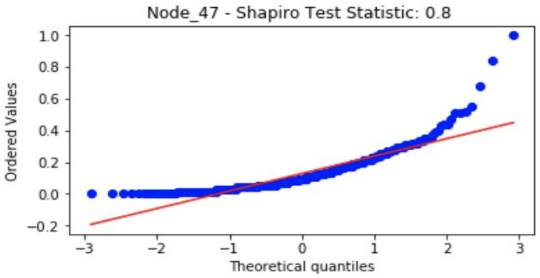

Node_41 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

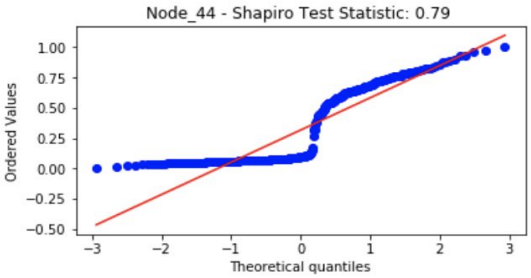

Node_45 - Shapiro Test Statistic: 0.9
Ordered Values
Theoretical quantiles

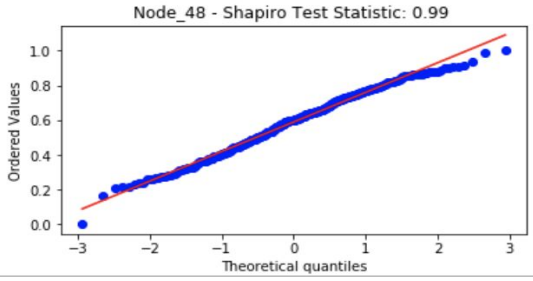

Node_42 - Shapiro Test Statistic: 0.95
Ordered Values
Theoretical quantiles

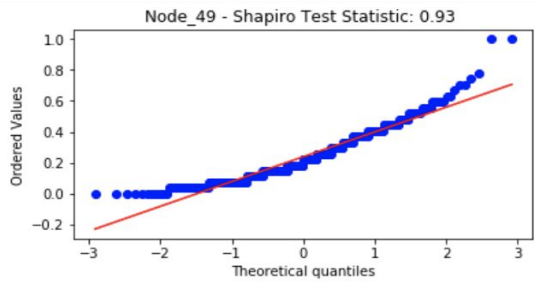

Node_46 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

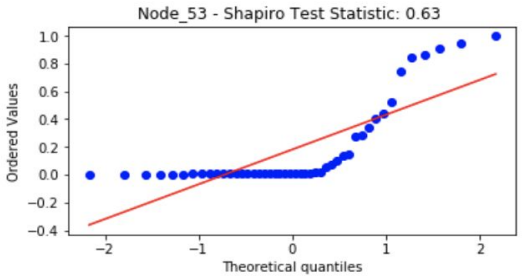

Node_43 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

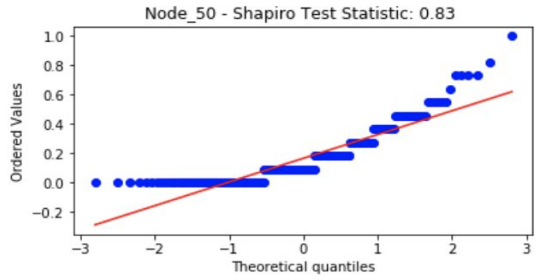

Node_47 - Shapiro Test Statistic: 0.8
Ordered Values
Theoretical quantiles

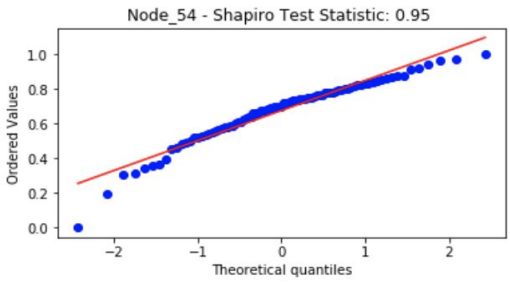

Node_44 - Shapiro Test Statistic: 0.79
Ordered Values
Theoretical quantiles

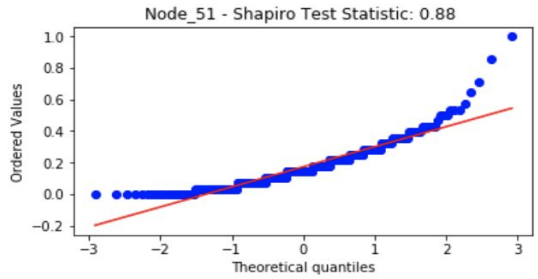

Node_48 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles

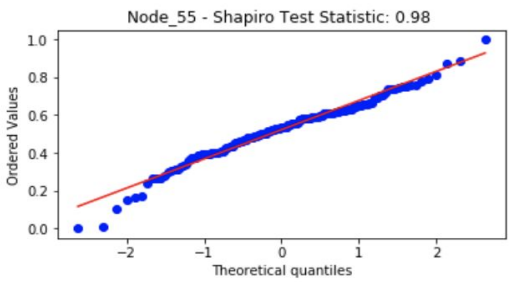

Node_49 - Shapiro Test Statistic: 0.93
Ordered Values
Theoretical quantiles

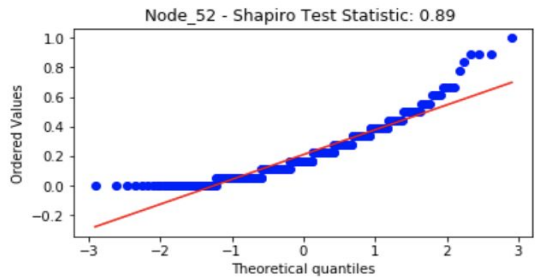

Node_53 - Shapiro Test Statistic: 0.63
Ordered Values
Theoretical quantiles

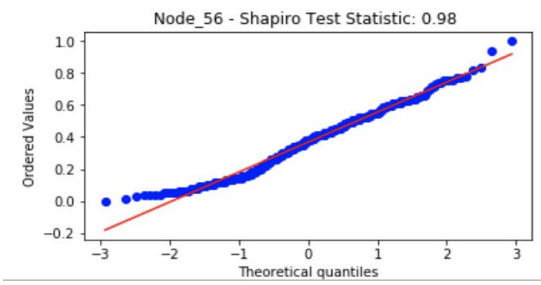

Node_50 - Shapiro Test Statistic: 0.83
Ordered Values
Theoretical quantiles

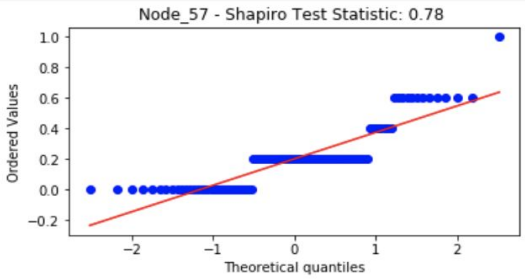

Node_54 - Shapiro Test Statistic: 0.95
Ordered Values
Theoretical quantiles

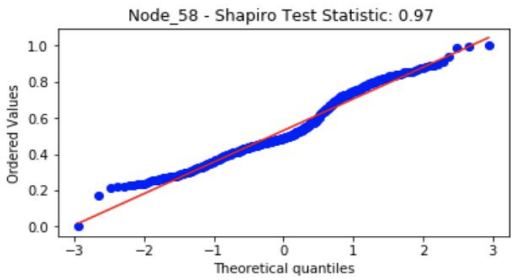

Node_51 - Shapiro Test Statistic: 0.88
Ordered Values
Theoretical quantiles

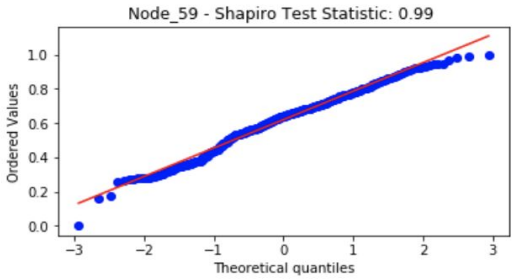

Node_55 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

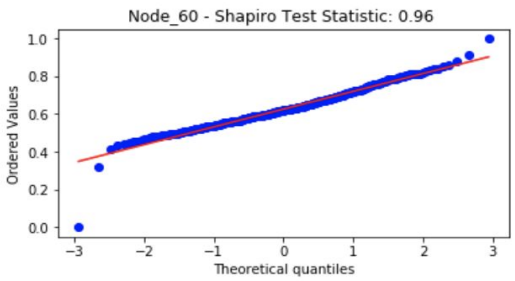

Node_52 - Shapiro Test Statistic: 0.89
Ordered Values
Theoretical quantiles

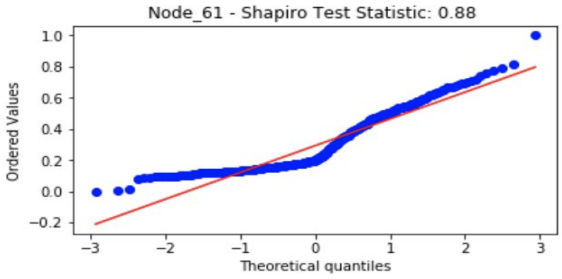

Node_56 - Shapiro Test Statistic: 0.98
Ordered Values
Theoretical quantiles

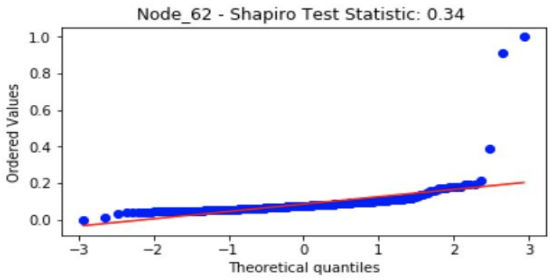

Node_57 - Shapiro Test Statistic: 0.78
Ordered Values
Theoretical quantiles

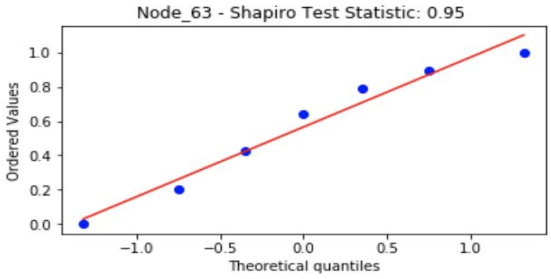

Node_58 - Shapiro Test Statistic: 0.97
Ordered Values
Theoretical quantiles

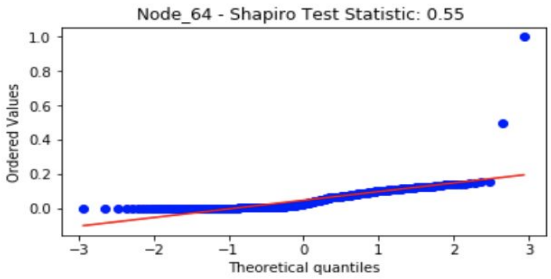

Node_59 - Shapiro Test Statistic: 0.99
Ordered Values
Theoretical quantiles


Node_60 - Shapiro Test Statistic: 0.96
Ordered Values
Theoretical quantiles


Node_61 - Shapiro Test Statistic: 0.88
Ordered Values
Theoretical quantiles

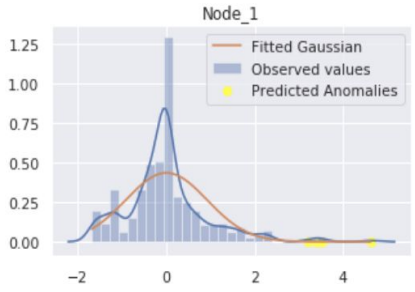

Node_62 - Shapiro Test Statistic: 0.34
Ordered Values
Theoretical quantiles

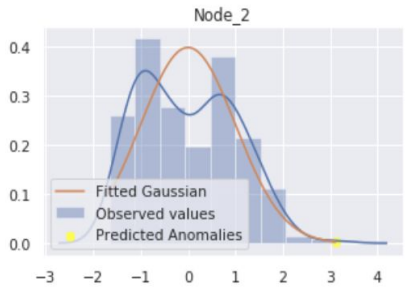

Node_63 - Shapiro Test Statistic: 0.95
Ordered Values
Theoretical quantiles

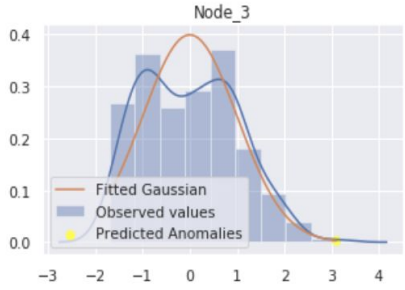

Node_64 - Shapiro Test Statistic: 0.55
Ordered Values
Theoretical quantiles

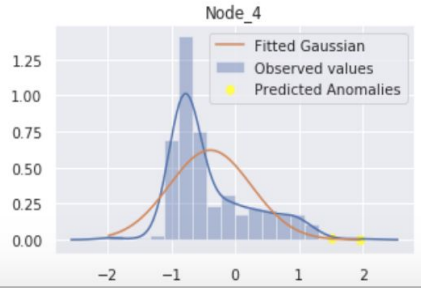
7.2 Simple Multivariate Gaussian Mixture Model

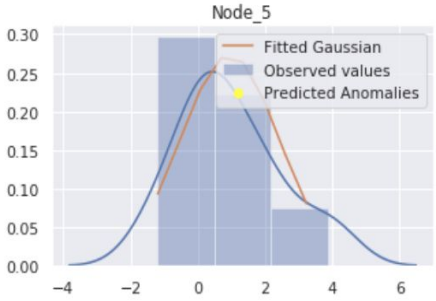
7.2.1 Fitting on Training Set

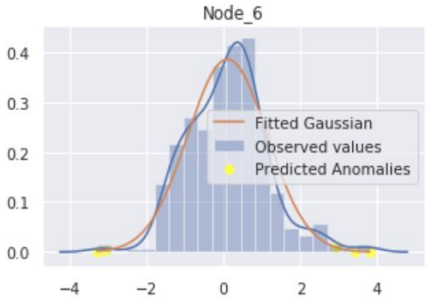

Node_1
Fitted Gaussian
Observed values
Predicted Anomalies

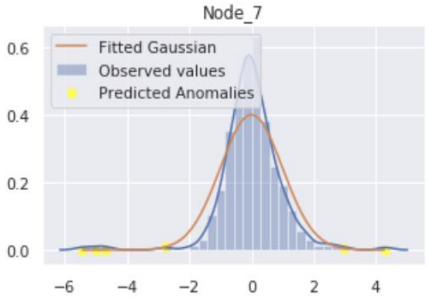

Node_2
Fitted Gaussian
Observed values
Predicted Anomalies

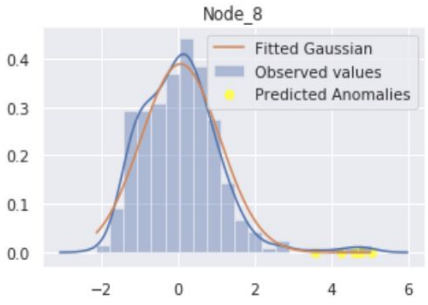

Node_3
Fitted Gaussian
Observed values
Predicted Anomalies

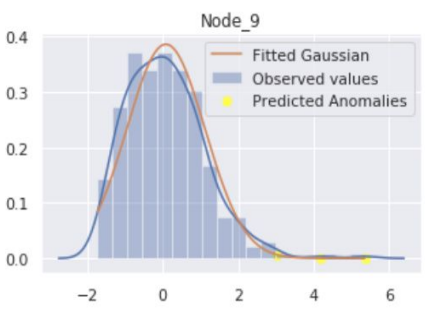

Node_4
Fitted Gaussian
Observed values
Predicted Anomalies

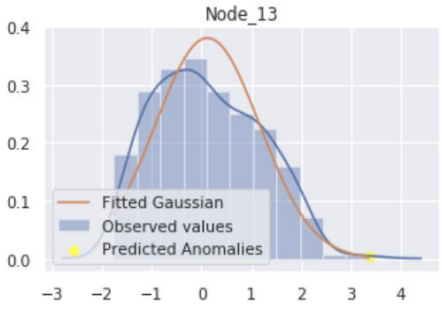

Node_5
Fitted Gaussian
Observed values
Predicted Anomalies

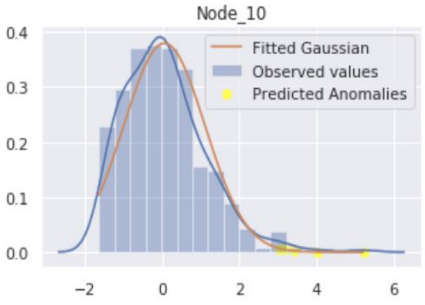

Node_6
Fitted Gaussian
Observed values
Predicted Anomalies

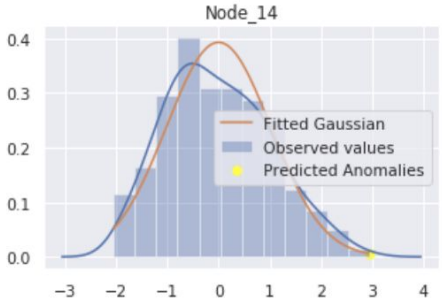

Node_7
Fitted Gaussian
Observed values
Predicted Anomalies

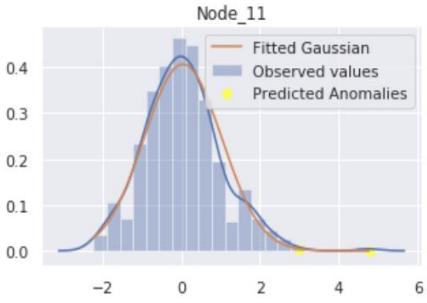

Node_8
Fitted Gaussian
Observed values
Predicted Anomalies

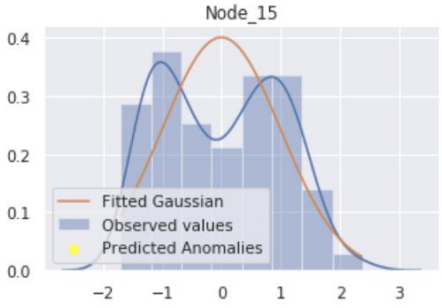

Node_9
Fitted Gaussian
Observed values
Predicted Anomalies

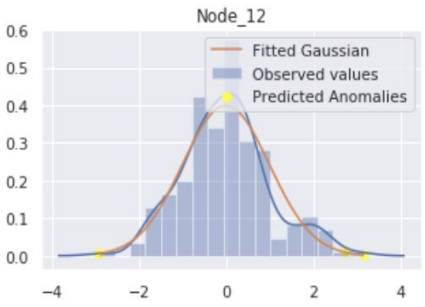

Node_13
Fitted Gaussian
Observed values
Predicted Anomalies

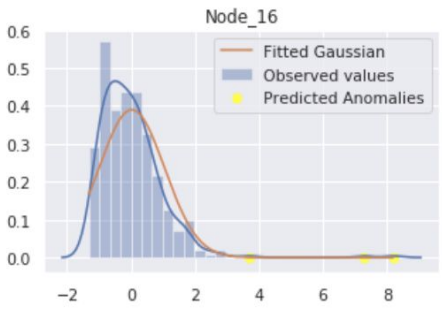

Node_10
Fitted Gaussian
Observed values
Predicted Anomalies

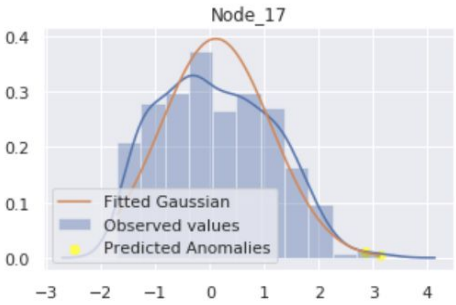

Node_14
Fitted Gaussian
Observed values
Predicted Anomalies

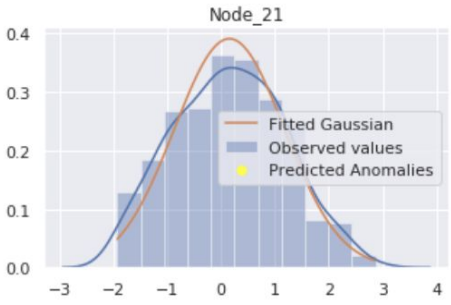

Node_11
Fitted Gaussian
Observed values
Predicted Anomalies

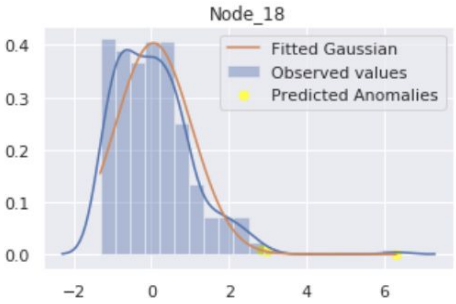

Node_15
Fitted Gaussian
Observed values
Predicted Anomalies

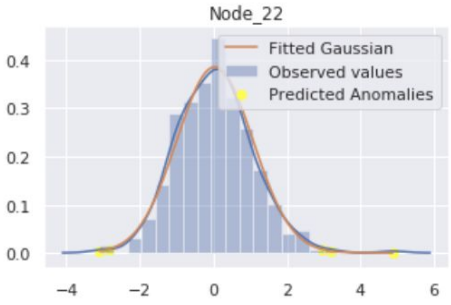

Node_12
Fitted Gaussian
Observed values
Predicted Anomalies

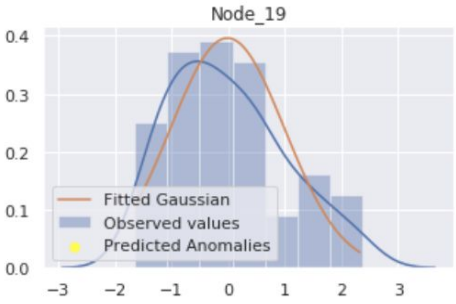

Node_16
Fitted Gaussian
Observed values
Predicted Anomalies

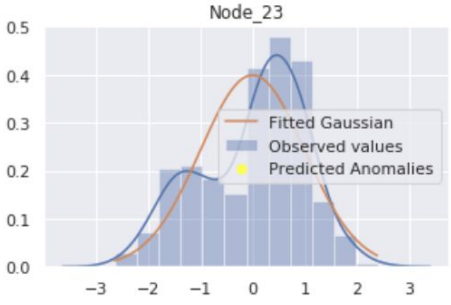

Node_17
Fitted Gaussian
Observed values
Predicted Anomalies

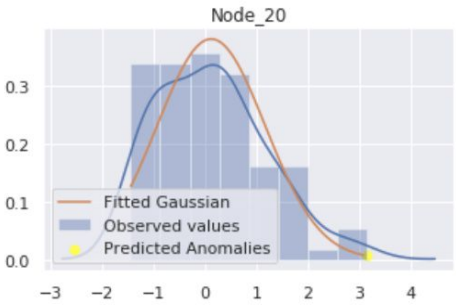

Node_21
Fitted Gaussian
Observed values
Predicted Anomalies

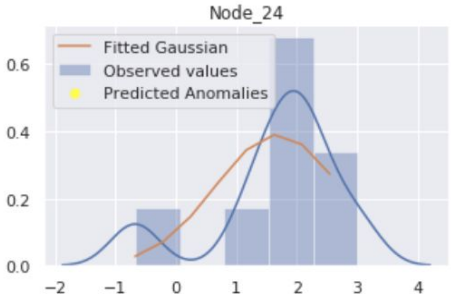

Node_18
Fitted Gaussian
Observed values
Predicted Anomalies

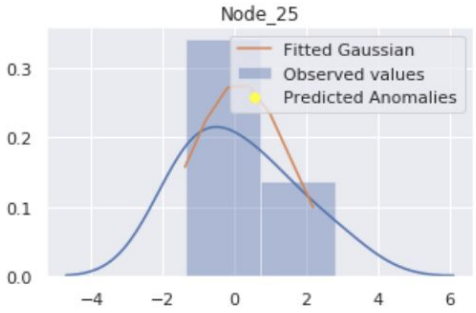

Node_22
Fitted Gaussian
Observed values
Predicted Anomalies

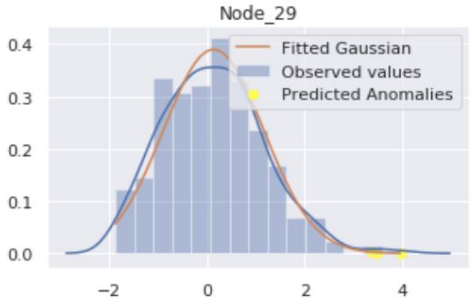

Node_19
Fitted Gaussian
Observed values
Predicted Anomalies

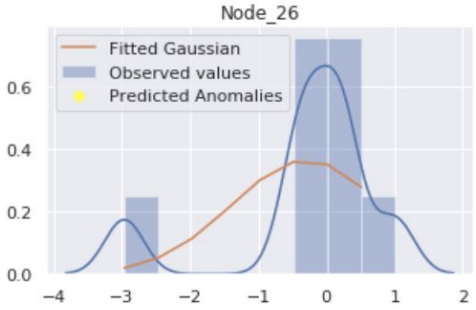

Node_23
Fitted Gaussian
Observed values
Predicted Anomalies

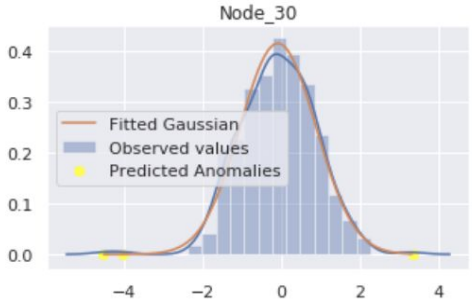

Node_20
Fitted Gaussian
Observed values
Predicted Anomalies

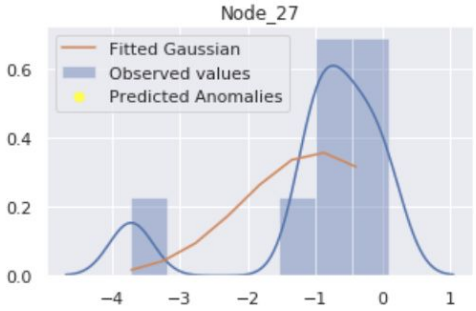

Node_24
Fitted Gaussian
Observed values
Predicted Anomalies

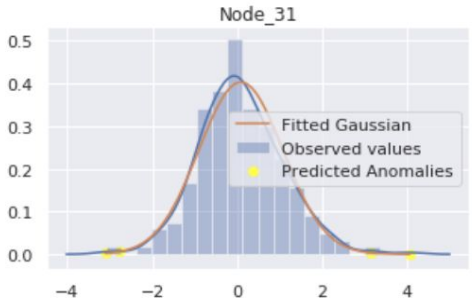

Node_25
Fitted Gaussian
Observed values
Predicted Anomalies

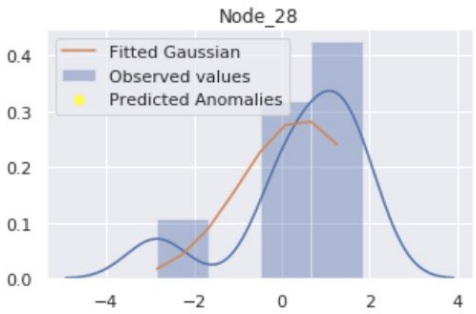

Node_29
Fitted Gaussian
Observed values
Predicted Anomalies

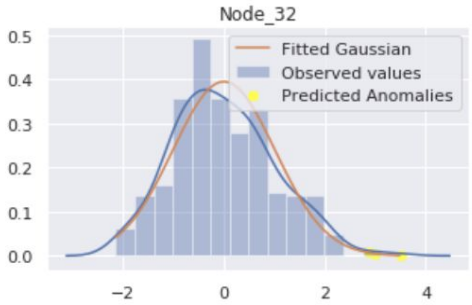

Node_26
Fitted Gaussian
Observed values
Predicted Anomalies

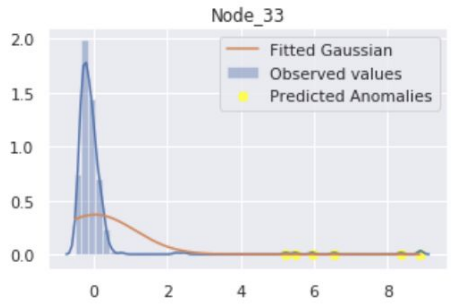

Node_30
Fitted Gaussian
Observed values
Predicted Anomalies

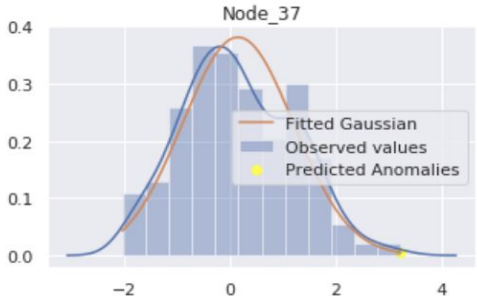

Node_27
Fitted Gaussian
Observed values
Predicted Anomalies

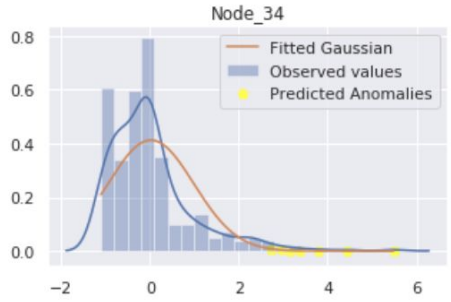

Node_31
Fitted Gaussian
Observed values
Predicted Anomalies

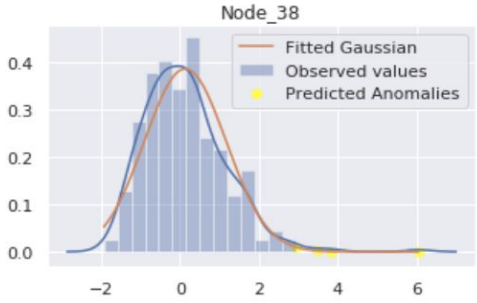

Node_28
Fitted Gaussian
Observed values
Predicted Anomalies

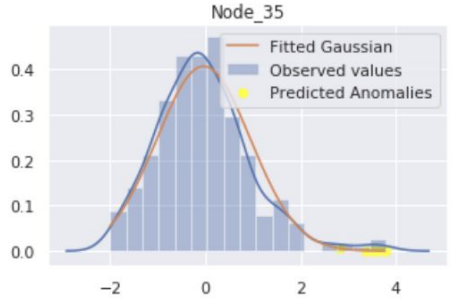

Node_32
Fitted Gaussian
Observed values
Predicted Anomalies

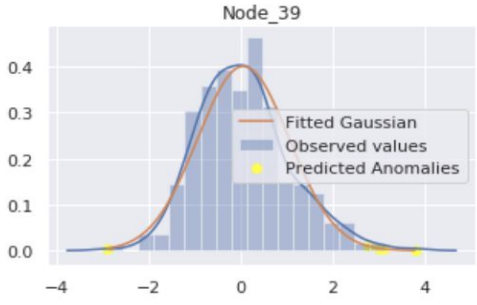

Node_33
Fitted Gaussian
Observed values
Predicted Anomalies

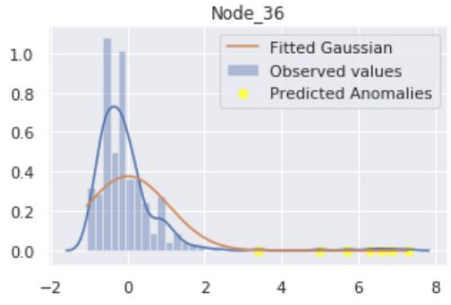

Node_37
Fitted Gaussian
Observed values
Predicted Anomalies

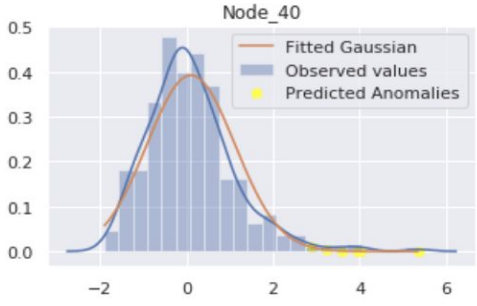

Node_34
Fitted Gaussian
Observed values
Predicted Anomalies

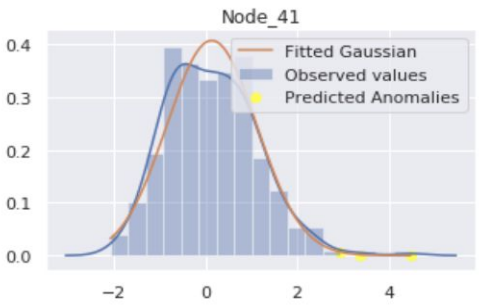

Node_38
Fitted Gaussian
Observed values
Predicted Anomalies

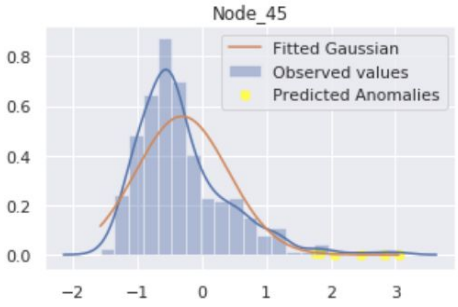

Node_35
Fitted Gaussian
Observed values
Predicted Anomalies

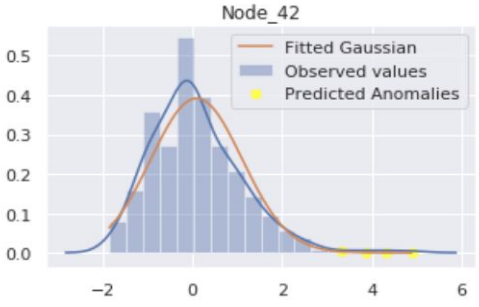

Node_39
Fitted Gaussian
Observed values
Predicted Anomalies

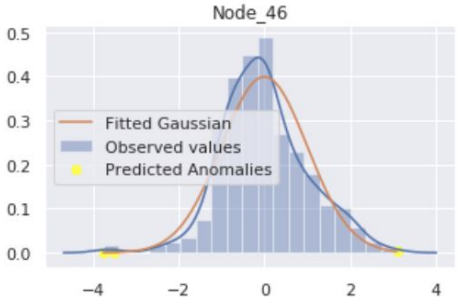

Node_36
Fitted Gaussian
Observed values
Predicted Anomalies

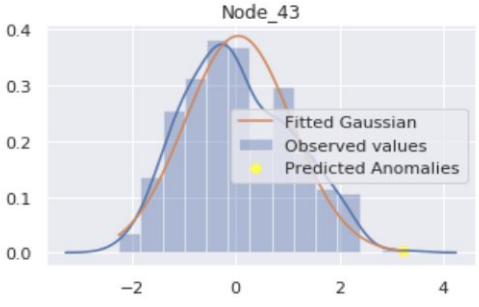

Node_40
Fitted Gaussian
Observed values
Predicted Anomalies

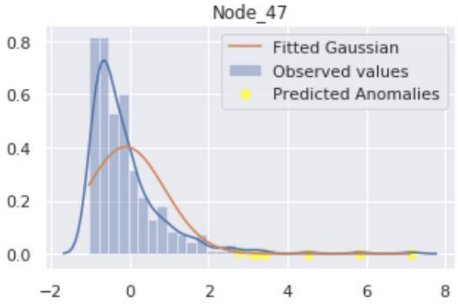

Node_41
Fitted Gaussian
Observed values
Predicted Anomalies

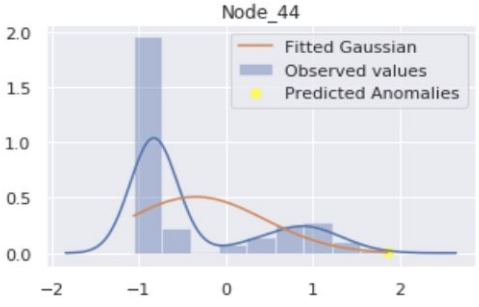

Node_45
Fitted Gaussian
Observed values
Predicted Anomalies

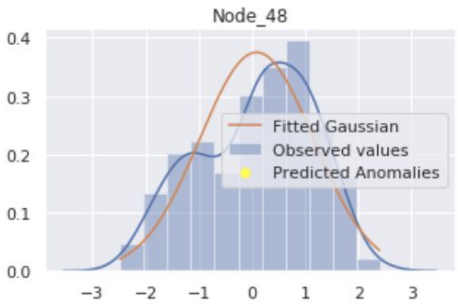

Node_42
Fitted Gaussian
Observed values
Predicted Anomalies

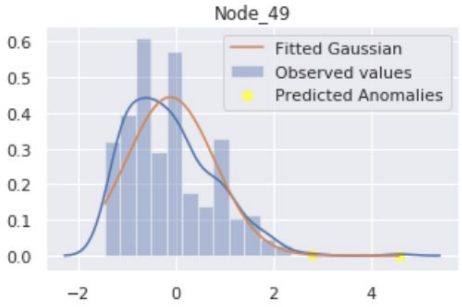

Node_46
Fitted Gaussian
Observed values
Predicted Anomalies

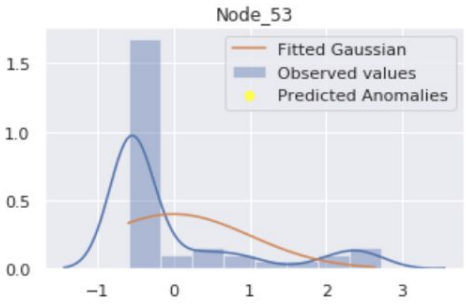

Node_43
Fitted Gaussian
Observed values
Predicted Anomalies

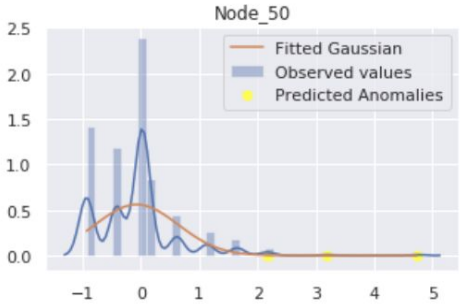

Node_47
Fitted Gaussian
Observed values
Predicted Anomalies

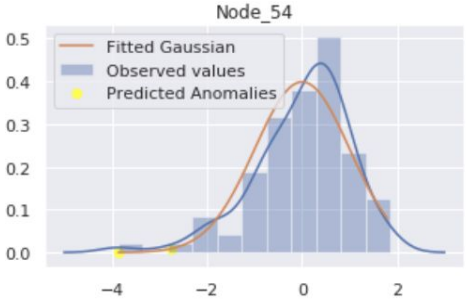

Node_44
Fitted Gaussian
Observed values
Predicted Anomalies

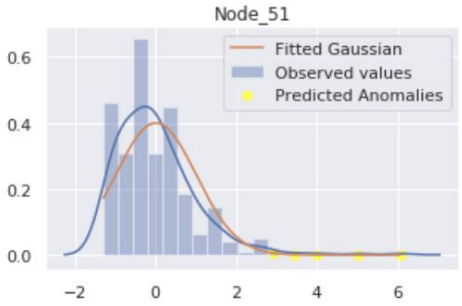

Node_48
Fitted Gaussian
Observed values
Predicted Anomalies

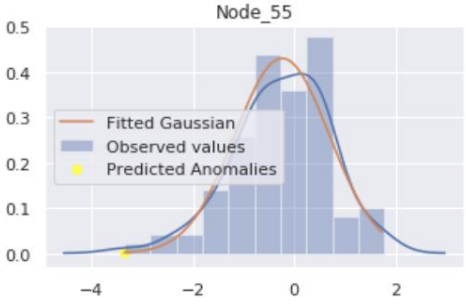

Node_49
Fitted Gaussian
Observed values
Predicted Anomalies

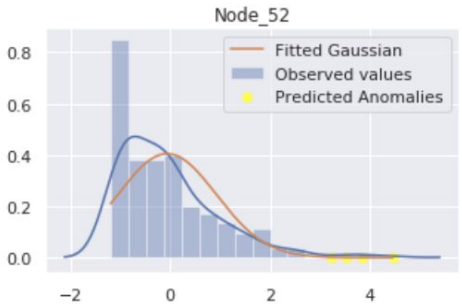

Node_53
Fitted Gaussian
Observed values
Predicted Anomalies

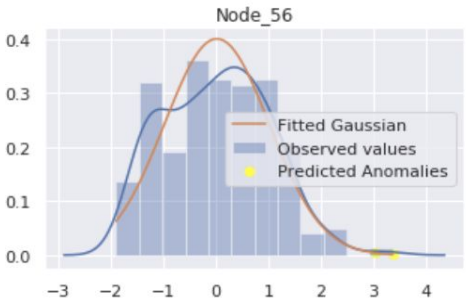

Node_50
Fitted Gaussian
Observed values
Predicted Anomalies

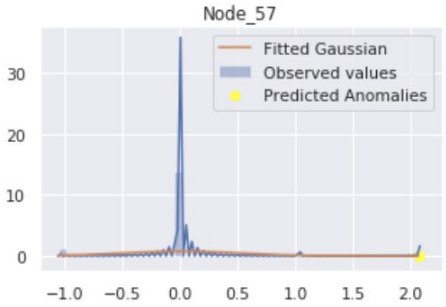

Node_54
Fitted Gaussian
Observed values
Predicted Anomalies

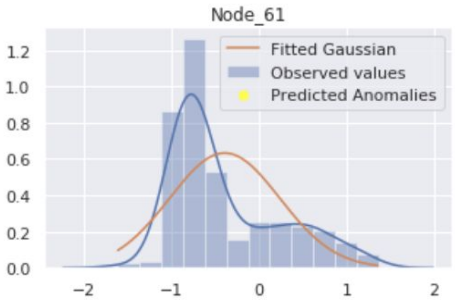

Node_51
Fitted Gaussian
Observed values
Predicted Anomalies

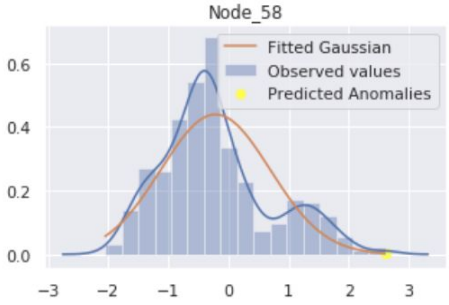

Node_55
Fitted Gaussian
Observed values
Predicted Anomalies

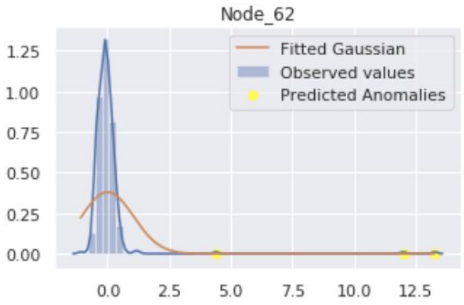

Node_52
Fitted Gaussian
Observed values
Predicted Anomalies

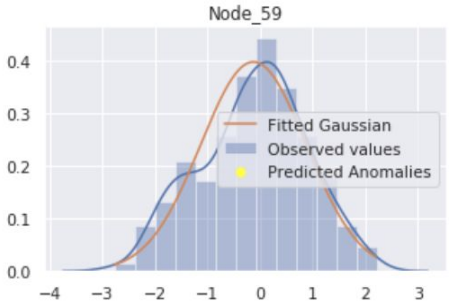

Node_56
Fitted Gaussian
Observed values
Predicted Anomalies

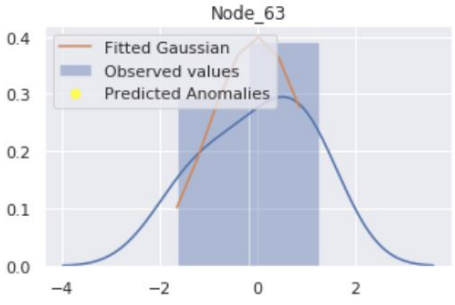

Node_57
Fitted Gaussian
Observed values
Predicted Anomalies

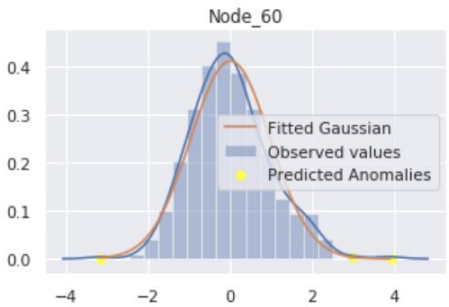

Node_61
Fitted Gaussian
Observed values
Predicted Anomalies

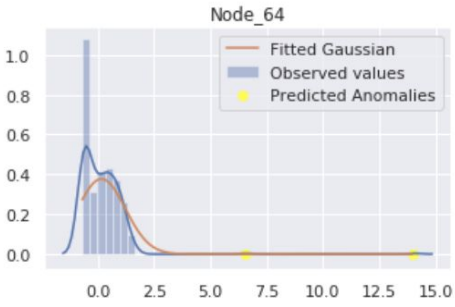

Node_58
Fitted Gaussian
Observed values
Predicted Anomalies


Node_62
Fitted Gaussian
Observed values
Predicted Anomalies

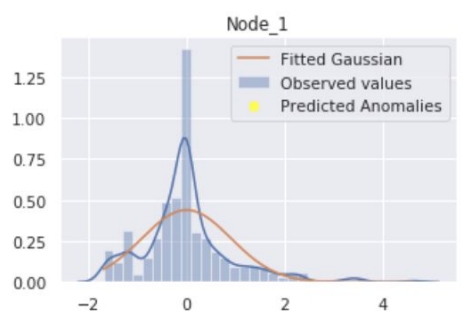

Node_59
Fitted Gaussian
Observed values
Predicted Anomalies

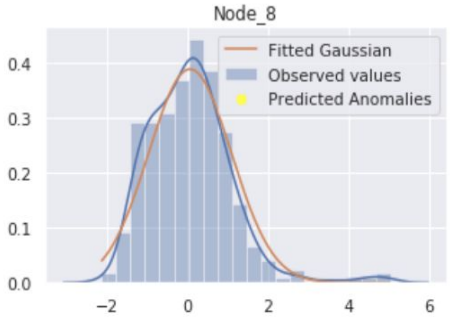

Node_63
Fitted Gaussian
Observed values
Predicted Anomalies

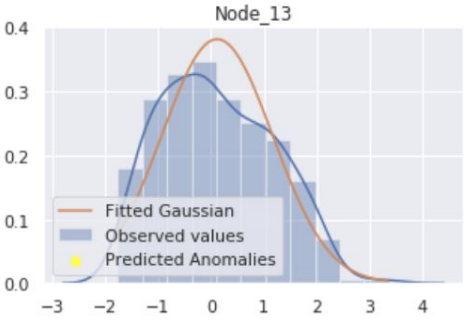

Node_60
Fitted Gaussian
Observed values
Predicted Anomalies


Node_64
Fitted Gaussian
Observed values
Predicted Anomalies

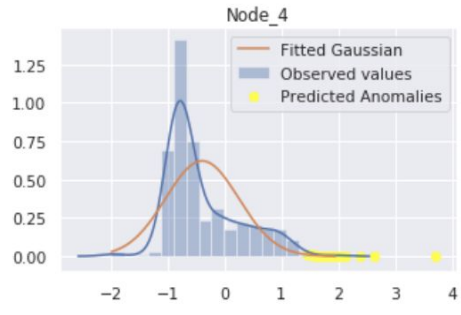
7.2.2 Testing Set Results

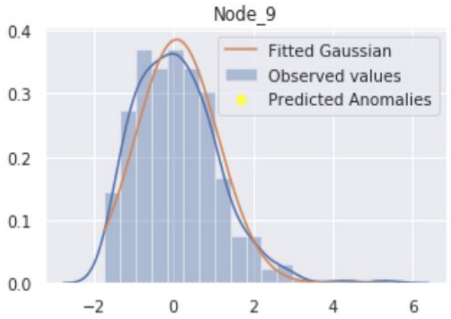


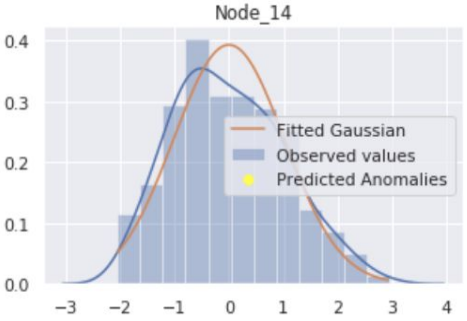


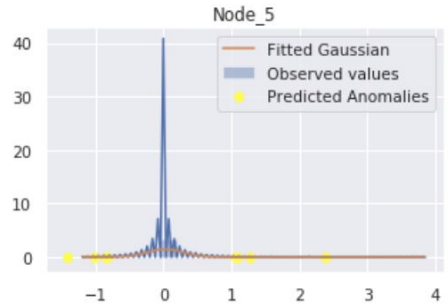


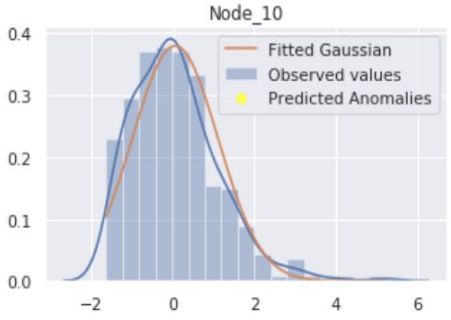
<Figure size 360x216 with 0 Axes>
<Figure size 360x216 with 0 Axes>

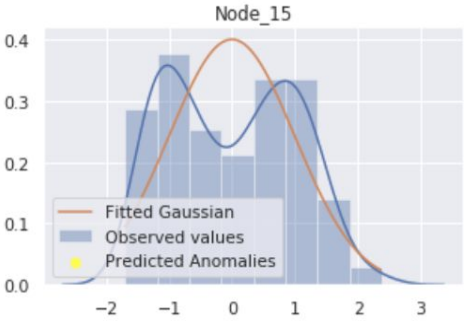


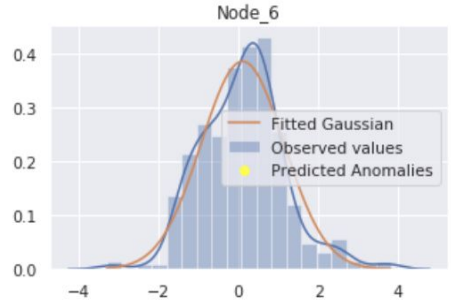


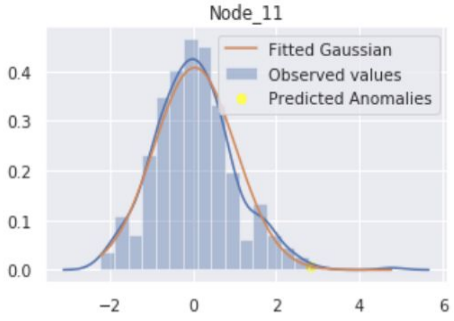


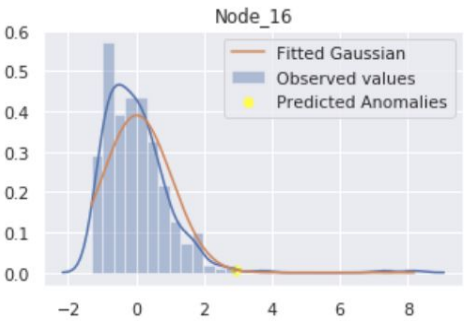


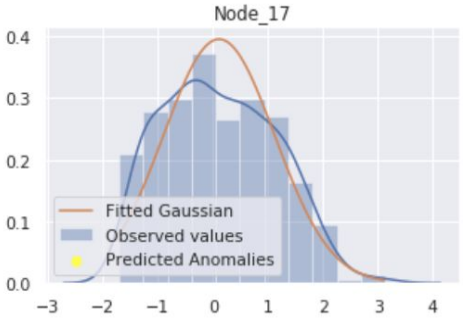


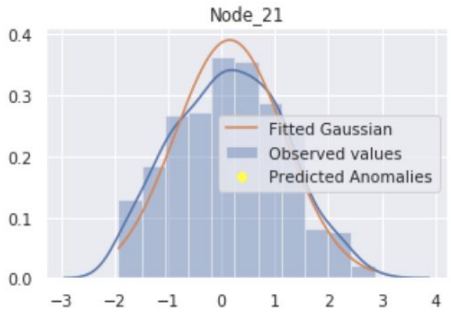


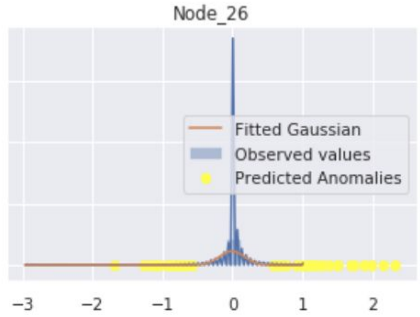


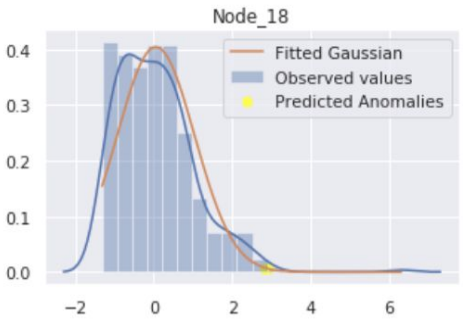


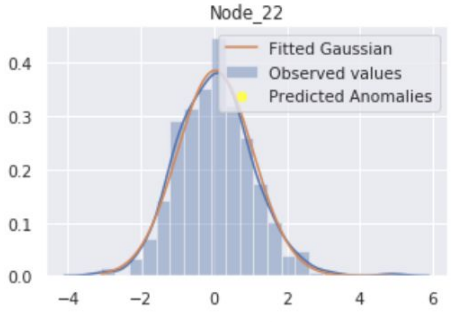


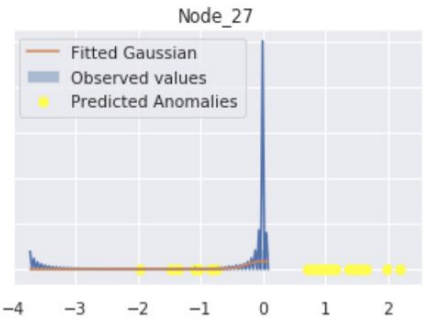




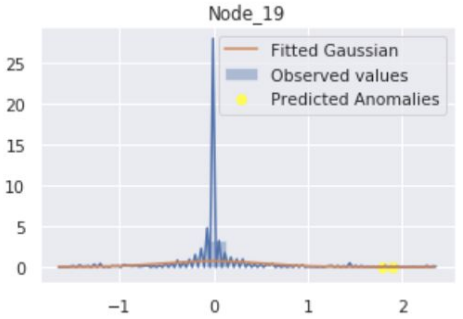


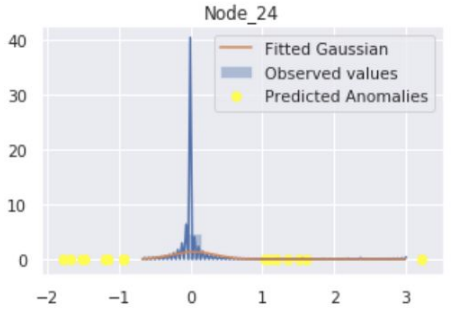


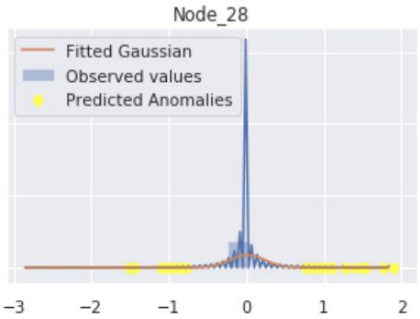


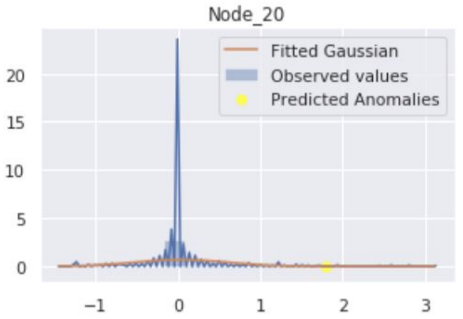


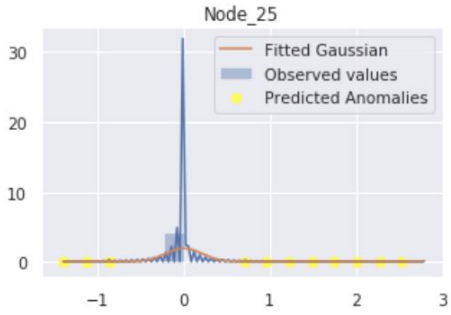
<Figure size 360x216 with 0 Axes>

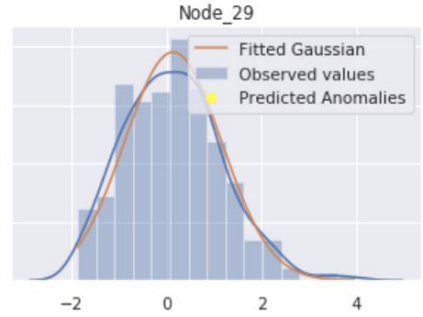


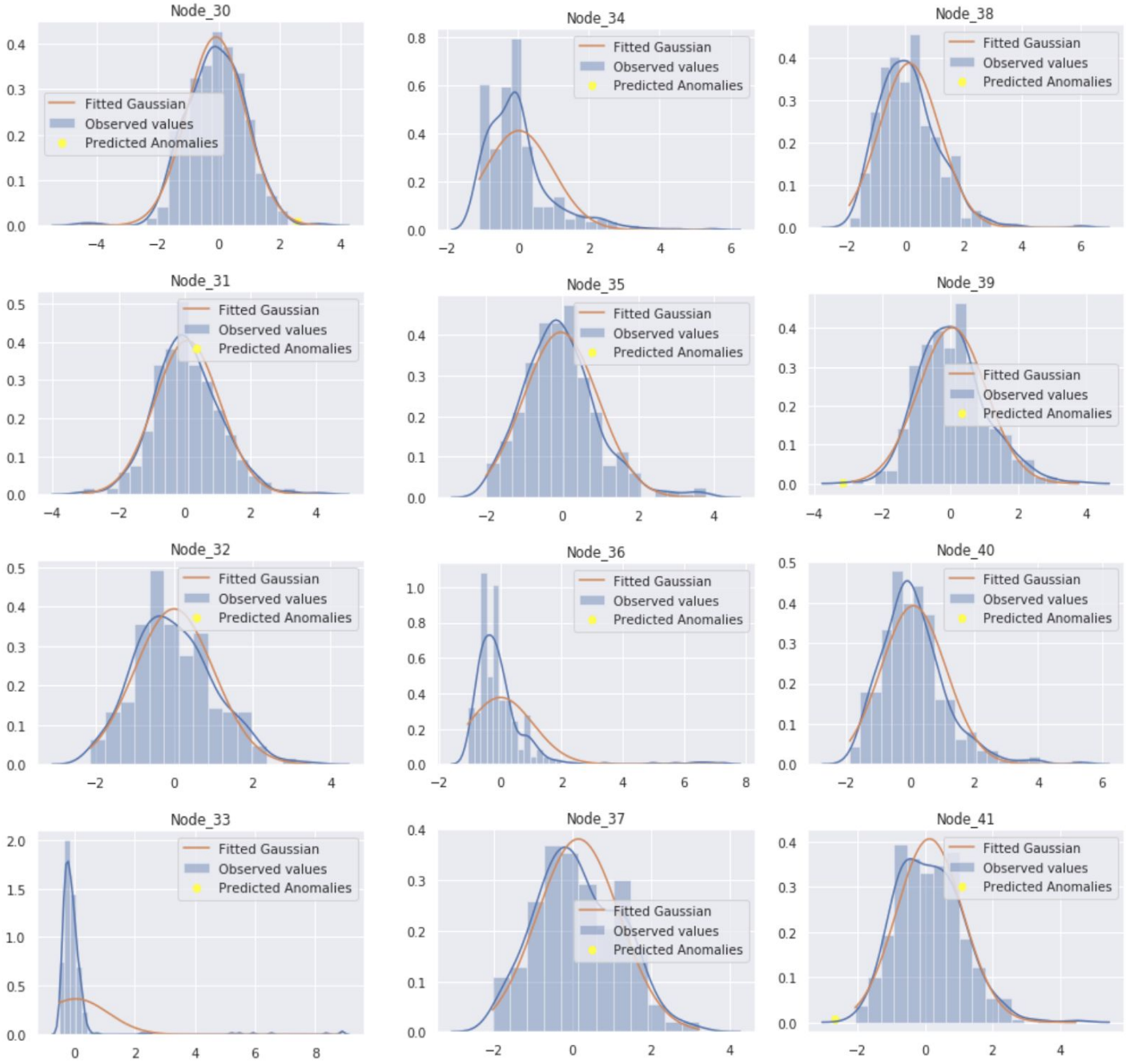


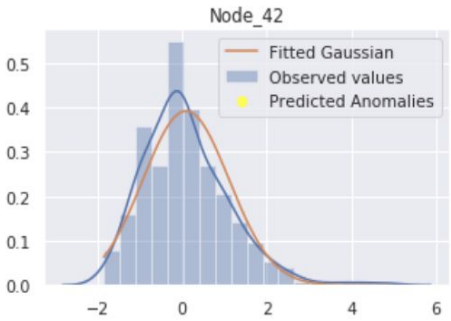


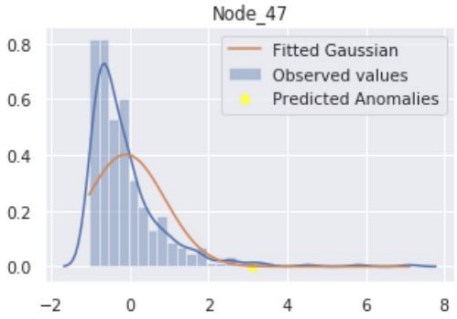


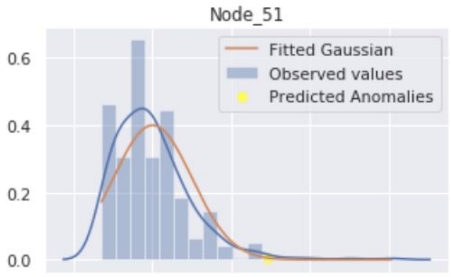


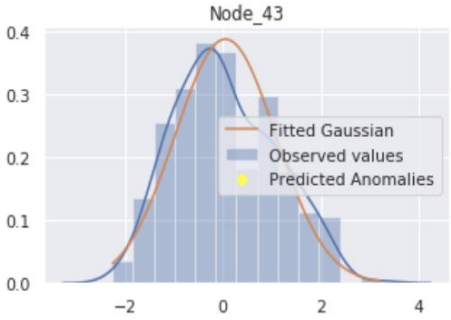


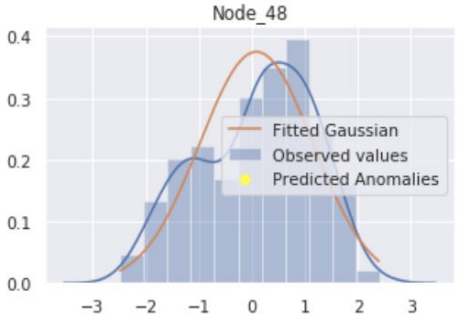

Node_30
Node_34
Node_38
Node_31
Node_35
Node_39
Node_32
Node_36
Node_40
Node_33
Node_37
Node_41
Fitted Gaussian
Observed values
Predicted Anomalies

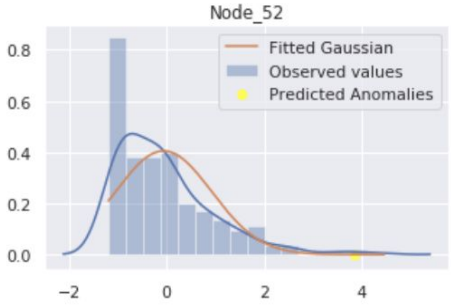




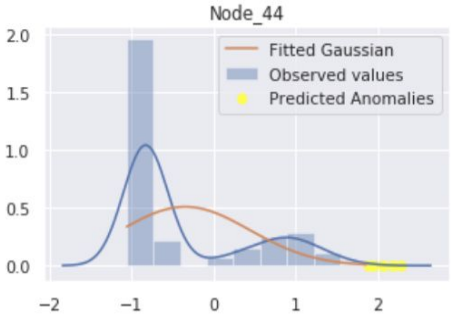


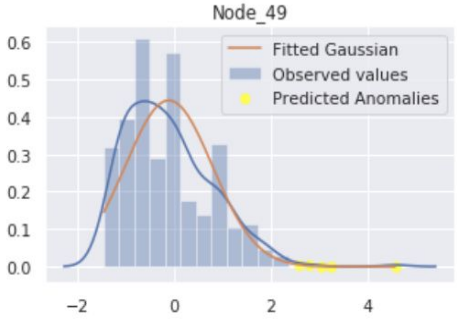


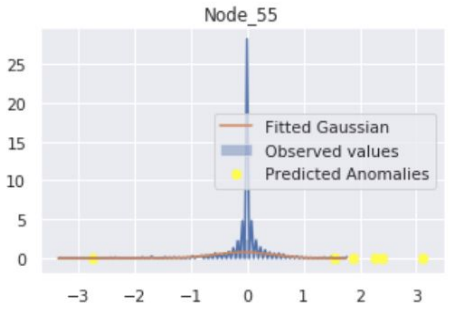


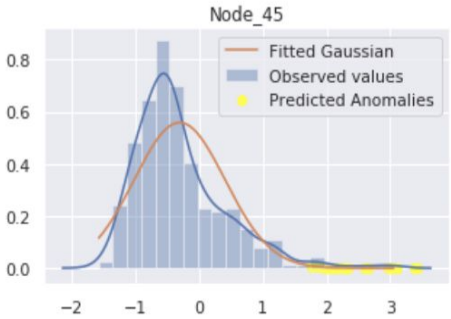


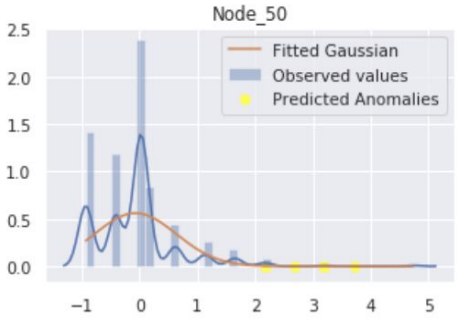
<Figure size 360x216 with 0 Axes>
<Figure size 360x216 with 0 Axes>

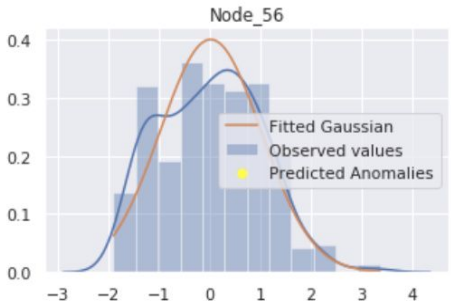


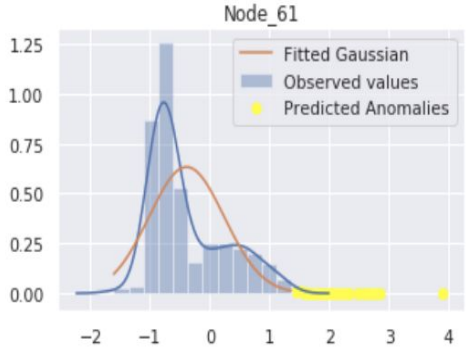


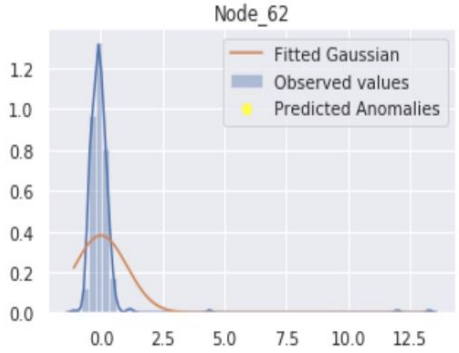




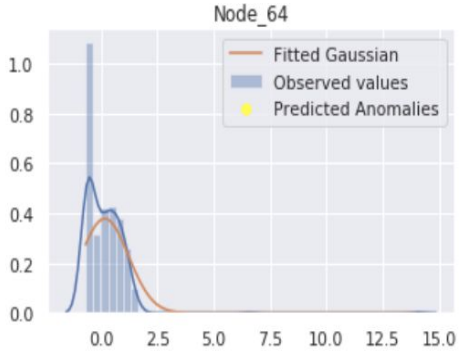








<Figure size 360x216 with 0 Axes>



References

Ben-Gal, I. (2005). Outlier detection. In *Data mining and knowledge discovery handbook* (pp. 131–146). Springer.

Blog, N. T. (2017, Apr). *Rad-outlier detection on big data.* Netflix TechBlog. Retrieved from https://netflixtechblog.com/rad-outlier-detection-on-big-data-d6b0494371cc

Chatfield, C. (1978). The holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *27*(3), 264–279.

Coronado-Ramirez, T.-P. (n.d.). *Non linear time series.indd.* http://www.cucea.udg.mx/include/publicaciones/coorinv/pdf/Nonlinear-time-series-and-finance.pdf ((Accessed on 09/29/2020))

Fair credit reporting act. (n.d.). https://www.consumer.ftc.gov/articles/pdf-0111-fair-credit-reporting-act.p ((Accessed on 09/29/2020))

Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying lstm to time series predictable through time-window approaches. In *Neural nets wirn vietri-01* (pp. 193–200). Springer.

Godfrey, L. B., & Gashler, M. S. (2017). Neural decomposition of time-series data for effective generalization. *IEEE transactions on neural networks and learning systems*, *29*(7), 2973–2985.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT press Cambridge.

Grosse, R. (n.d.). *L14 recurrent neural nets.pdf.*

Hayes, A. (2020, Aug). *Heteroskedasticity.* Investopedia. Retrieved from https://www.investopedia.com/terms/h/heteroskedasticity.asp

Hillmer, S. C., & Tiao, G. C. (1982). An arima-model-based approach to seasonal adjustment. *Journal of the American Statistical Association*, *77*(377), 63–70.

Hyndman, R. J., Wang, E., & Laptev, N. (2015). Large-scale unusual time series detection. In *2015 ieee international conference on data mining workshop (icdmw)* (pp. 1616–1619).

Kapoor, K. (2020). A novel algorithm for optimized real time anomaly detection in timeseries. *arXiv*

preprint arXiv:2006.04071.

Knaub, J. (2007). Heteroscedasticity and homoscedasticity. In *Encyclopedia of measurement and statistics* (pp. 431–432). SAGE.

Le, Q. V. (n.d.). *Lesmocan05.pdf.* `https://cs.stanford.edu/ quocle/LeSmoCan05.pdf`. ((Accessed on 09/30/2020))

MIT. (n.d.). *shapiro.dvi.* `https://math.mit.edu/ rmd/465/shapiro.pdf`. ((Accessed on 09/30/2020))

Ng, A. (2017a). *cs229-notes7b.dvi.* `http://cs229.stanford.edu/notes/cs229-notes7b.pdf`. ((Accessed on 09/29/2020))

Ng, A. (2017b). *Lecture 15.7 — anomaly detection — multivariate gaussian distribution — [andrew ng] - youtube.* `https://www.youtube.com/watch?v=JjB58InuTqM`. ((Accessed on 09/30/2020))

Sarkadi, K. (1975). The consistency of the shapiro—francia test. *Biometrika*, *62*(2), 445–450.

Schlittgen, R. (1997). Chris chatfield: The analysis of time series. an introduction. vii+ 283 pp. chapman & hall 1996,£ 18.99 (paperback). *Biometrical Journal*, *39*(4), 508–508.

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3/4), 591–611.

Singh, A. (2017). *Anomaly detection for temporal data using long short-term memory (lstm).*

Veracini, T., Matteoli, S., Diani, M., & Corsini, G. (2009). Fully unsupervised learning of gaussian mixtures for anomaly detection in hyperspectral imagery. In *2009 ninth international conference on intelligent systems design and applications* (pp. 596–601).