2.2 Data Quality Test Approaches for Sequence Data

A time series, also known as sequence data [60], is a set of time-ordered records [61]. Large volumes of real-world time series data are increasingly collected from various sources, such as Internet of Things (IoT) sensors, network servers, and patient medical flow reports [61–63]. Existing approaches for finding anomalies by discovering the constraints in individual data records cannot be used for testing time-series data as there may be constraints over multiple attributes and records in a time series. The records in a sequence have strong correlations and dependencies with each other, and their violations cannot be discovered by analyzing records in isolation [64]. In this section, we propose a classification framework for the techniques that detect anomalies in time-series data.

2.2.1 Time Series

A time series *T* is a sequence of *d*-dimensional records [61] described using the vector $T =< R_1, ..., R_n >$, where $R_i = (a_i^1, ..., a_i^d)$ is a record at time i, for $1 \leq i \leq n$ and a_i^j is the j^{th} attribute of the i^{th} record. Existing data analysis approaches [61] assume that the time gaps between any pair of consecutive records differ by less than or equal to an epsilon value, i.e., the differences between the time stamps of any two consecutive records are nearly the same.

A time series can be *univariate* (d=1) or *multivariate* (d>1) [62]. A univariate time series has one time-dependent attribute. For example, a univariate time series can consist of daily temperatures recorded sequentially over 24-hour increments. A multivariate time series is used to simultaneously capture the dynamic nature of multiple attributes. For example, a multivariate time series can consist of precipitation, wind speed, snow depth, and temperature.

The research literature [1, 66] uses various features that describe the relationships among the time-series records and attributes. Trend and seasonality [67] are the most commonly used features. Trend is defined as the general tendency of a time series to increment, decrement, or stabilize over time [67]. For example, there may be an upward trend for the number of patients with cancer diagnosis. Seasonality is defined as the existence of repeating cycles in a time series [67]. For example,

Table 2.4: Time Series Features [1]

Feature	Description
F_1: Mean	Mean value
F_2: Variance	Variance value
F_3: Lumpiness	Variance of the variances across multiple blocks
F_4: Lshift	Maximum difference in mean between consecutive blocks
F_5: Vchange	Maximum difference in variance between consecutive blocks
F_6: Linearity	Strength of linearity
F_7: Curvature	Strength of curvature
F_8: Spikiness	Strength of spikiness based on the size and location of the peaks and troughs
F_9: Season	Strength of seasonality based on a robust STL [65] decomposition
F_{10}: Peak	Strength of peaks
F_{11}: Trough	Strength of trough
F_{12}: BurstinessFF	Ratio between the variance and the mean (Fano Factor)
F_{13}: Minimum	Minimum value
F_{14}: Maximum	Maximum value
F_{15}: Rmeaniqmean	Ratio between interquartile mean and the arithmetic mean
F_{16}: Moment3	Third moment
F_{17}: Highlowmu	Ratio between the means of data that is below and upper the global mean
F_{18}: Trend	Strength of trend based on robust STL decomposition

the sales of swimwear is higher during summers. A time series is *stationary* (non-seasonal) if all its statistical features, such as mean and variance are constant over time. Table 2.4 shows a set of features defined by Talagala et al. [1] to describe a time series.

A constraint is defined as a rule over the time-series features. For example, the mean (F_1) value of the daily electricity power delivered by a household must be in the [0.1 KWH, 0.5 KWH] range. We categorize the faults that can violate the constraints over time-series features as *anomalous records* and *anomalous sequences*.

Anomalous records. Given an input time series *T*, an anomalous record R_t is one whose observed value is significantly different from the expected value of *T* at *t*. An anomalous record may violate constraints over the features F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, and F_{17}. For example, if there is a constraint that imposes a range of values (F_{13}, F_{14}) for the infant patients' weights during their first three months, a record in the first three months with a weight value outside this range must be reported as faulty.

Anomalous sequences. Given a set of subsequences $T = \{T_1, ..., T_m\}$ in a time series T, a faulty sequence $T_j \in T$ is one whose behavior is significantly different from the majority of subsequences in *T*. An anomalous sequence may violate constraints over any of the features F_1 through F_{18} . For example, consider the constraint that imposes an upward trend (F_{18}) for the

number of cars passing every second at an intersection from 6 to 7 am on weekdays. A decrease in this trend is anomalous.

Figure 3 shows the classification framework we propose for the techniques that detect anomalous records and sequences in time-series data. The framework presents *what* is detected in terms of anomaly types and *how* they are detected.

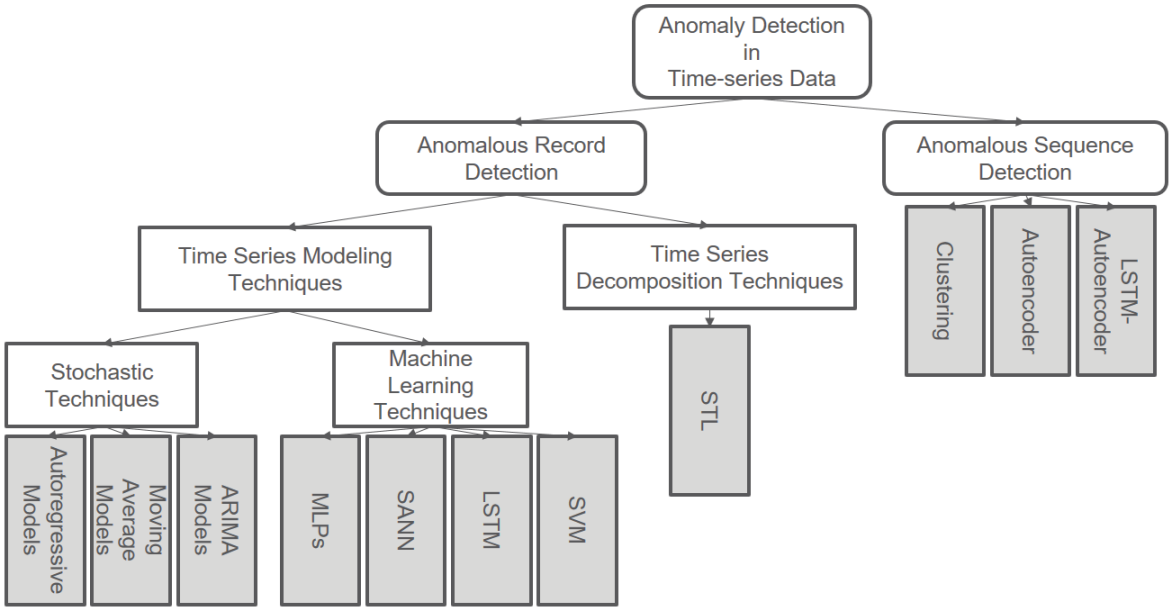

Figure 2.3: Classification Framework for Anomaly Detection Approaches for Sequence Data

2.2.2 Approaches to Detect Anomalous Records

We categorize these approaches as *time series modeling techniques* and *time series decomposition techniques*.

Time Series Modeling Techniques

Given a time series $T = \{R_t\}$, these techniques model the time series as a linear/non-linear function f that associates current value of a time series to its past values. Next, the techniques use f to provide the predicted value of R_t at time t, denoted by R'_t, and calculate a prediction error $PE_t = |R_t - R'_t|$. The techniques report R_t as outlier if the prediction error falls outside a fixed

threshold value. Every model f has a set of parameters, which are estimated using *stochastic* or *machine learning* techniques.

In the stochastic modeling techniques, a time series is considered as a set of random variables $T = \{R_t,\ t = 0, ..., n\}$, where R_t is from a certain probability model [67]. Examples of these techniques are *Autoregressive*, *Moving Average*, and *Autoregressive Integrated Moving Average* models.

Autoregressive (AR) models. In an Autoregressive model, the current value of a record in a time series is a linear combination of the past record values plus a random error. An autoregressive model makes an assumption that the data records at previous time steps (called as lag variables) can be used to predict the record at the next time step. The relationship between data records is called correlation. Statistical measures are typically used to calculate the correlation between the current record and the records at previous time steps. The stronger the correlation between the current record and a specific lagged variable, the more weight that autoregressive model puts on that variable. If all previous records show low or no correlation with the current one, then the time series problem may not be predictable [68]. Equation 2.3 shows the mathematical expression for an AR model.

$$R_t = \sum_{i=1}^{p} A_i R_{t-i} + E_t \tag{2.3}$$

where R_t is the record at time t and p is the order of the model. For example, an autoregressinve model of order two indicates that the current value of a time series is a linear combination of the two immediately preceding records plus a random error. The coefficients $A = (A_1, ..., A_p)$ are weights applied to each of the past records. The random errors (noises) E_t are assumed to be independent and following a Normal $N(0, \sigma^2)$ distribution. Given the time series T, the objective of AR modeling is to estimate the model parameters (A, σ^2). The linear regression estimators [69], likelihood estimators [70], and Yule-Walker equations [67] are typical stochastic techniques used to estimate this model parameters.

The AR model is only appropriate for modeling univariate stationary time-series data [67]. Moreover, it does not consider the non-linear associations between the data records in a time series.

Moving Average (MA) models. In these models, a data record at time t is a linear combination of the random errors that occurred in past time periods (i.e. E_{t-1}, E_{t-2},...,E_{t-p}). Equation 2.4 shows the mathematical expression for an MA model.

$$R_t = \mu \sum_{i=1}^{p} B_i E_{t-i} + E_t \tag{2.4}$$

Where μ is the series mean, p is the order of the model, and $B = (B_1, ..., B_p)$ are weights applied to each of the past errors. The random errors E_t are assumed to be independent and following a Normal $N(0, \sigma^2)$ distribution.

The MA model is appropriate for univariate stationary time series modeling [67]. Moreover, it is more complicated to fit an MA model to a time series than fitting an AR model. Because in an MA model, the random error terms are not foreseeable [67].

Autoregressive Integrated Moving Average (ARIMA) models. ARIMA is a mixed model, which incorporates: (1) Autoregression (AR) model, (2) an Integrated component, and (3) Moving Average (MA) model. The integrated component stationarized the time series by using transformations like differencing [71], logging [72], and deflating [73]. ARIMA can model time series with non-stationary behaviour. However, this model assumes that the time series is linnear and follows a known statistical distribution, which makes it inapplicable to many practical problems [67].

In machine Learning-based modeling techniques, a time series is considered to follow a specific pattern. Examples of these techniques are *Multi Layer Perceptron (MLP)*, *Seasonal Artificial Neural Networs (SANN)*, *Long Short Term Network (LSTM)*, and *Support Vector Machine (SVM)* models.

Multi Layer Perceptron (MLP). This technique is a type of Artificial Neural Network (ANN) [74], which supports non-linear modeling, with no assumption about the statistical distribution of the data [67]. An MLP model is a fully connected network of information processing units that are organized as input, hidden, and output layers. Equation 2.5 shows the mathematical expression of an MLP for time series modeling.

$$R_t = b + \sum_{j=1}^{q} \alpha_j g \left(b_j + \sum_{i=1}^{p} \beta_{ij} R_{t-i} \right) + E_t \tag{2.5}$$

where R_{t-i} $(i = 1, .., p)$ are p network inputs, R_t is the network output, α_j and β_{ij} are the network connection weights, E_t is a random error, and g is a non-linear activation function, such as logistic sigmoid and hyperbolic tangent.

The objective is to train the network and learn the parameters of the non-linear functional mapping f from the p past data records to the current data record R_t (i.e., $R_t = f(R_{t-1}, ..., R_{t-p}, w) + E_t$). Approaches based on minimization of an error function (equation 2.6) are typically used to estimate the network parameters. Examples of these approaches are Backpropagation and Generalized Delta Rule [74].

$$Error = \sum_{t} e_t^2 = \sum_{t} (R_t - R'_t)^2 \tag{2.6}$$

where R'_t is the actual network output at time t.

An MLP can model non-linear associations between data records. However, it is appropriate for univariate time series modeling. Moreover, because of the limited number of network inputs, it can only discover the short-term dependencies among the data records.

A Seasonal Artificial Neural Networ (SANN) model is an extension of MLPs for modeling seasonal time-series data. The number of input and output neurons are determined based on a seasonal parameter *s*. The records in the i^{th} and (i+1)th seasonal period are used as the values of network input and output respectively. Equation 2.7 shows the mathematical expression for this model [67].

$$R_{t+l} = \alpha_l + \sum_{j=1}^{m} w_{1jl} g \left(\theta_j + \sum_{i=0}^{s-1} w_{0ij} R_{t-i} \right) \tag{2.7}$$

where $R_{t+l}(l = 1, .., s)$ are *s* future predictions based on the *s* previous data records ($R_{t-i}(i = 0, ..., s-1)$); w_{0ij} and w_{1jl} are connection weights from the input to hidden and from hidden to output neurons respectively; g is a non-linear activation function and α_l and θ_j are network bias terms.

This network can model non-linear associations in seasonal time-series data. However, it is appropriate for modeling univariate time series. Moreover, the values of records in a season are considered to be dependent only on the values of the previous season. As a result, the network can only learn short-term dependencies between data records.

Long Short Term Network (LSTM). An LSTM is a Recurrent Neural Network (RNN) [4] that contains loops in its structure to allow information to persist and make network learn sequential dependencies among data records [75]. An RNN can be represented as multiple copies of a neural network, each passing a value to its successor. Figure 2.4 shows the structure of an RNN [4]. In this Figure, *A* is a neural network, X_t is the network input, and h_t is the network output.

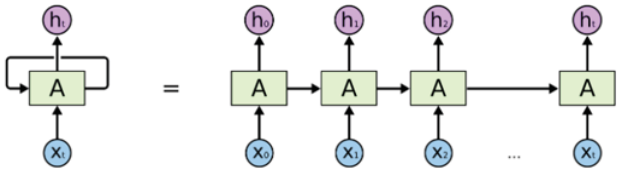

Figure 2.4: An Unrolled RNN [4]

The original RNNs can only learn short-term dependencies among data records by using the recurrent feedback connections [62]. LSTMs extend RNNs by using specialized gates and memory cells in their neuron structure to learn long-term dependencies.

Figure 2.5 shows the structure of an LSTM network. The computational units (neurons) of an LSTM are called *memory cells*. The horizontal line passing through the top of the neuron is called

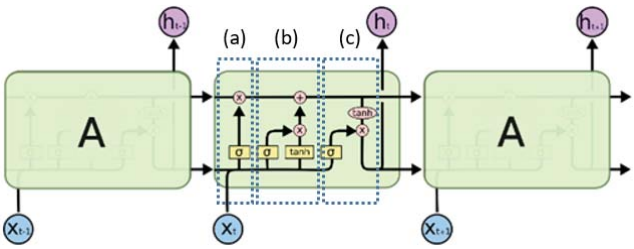

Figure 2.5: LSTM Structure [4]

the memory cell state. An LSTM has the ability to remove or add information to the memory cell state by using *gates*. The gates are defined as weighted functions that govern information flow in the memory cells. The gates are composed of a *sigmoid layer* and a *point-wise operation* to optionally let information through. The sigmoid layer outputs a number between zero (to let nothing through) and one (to let everything through).

There are three types of gates, namely, *forget*, *input*, and *output*.

- *Forget gate* (Figure 2.5 (a)): Decides what information to discard from the memory cell. Equation 2.8 shows the mathematical representation of the forget gate.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.8}$$

  where W_f is the connection weight between the inputs (h_{t-1} and x_t) and the sigmoid layer; b_f is the bias term and σ is the sigmoid activation function. In this gate, $f_t = 1$ means that completely keep the information and $f_t = 0$ means that completely get rid of the information.

- *Input gate* (Figure 2.5 (b)): Decides which values to be used from the network input to update the memory state. Equation 2.9 shows the mathematical representation of the input gate.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.9}$$

where C_t is the new memory cell state and C_{t-1} is the old cell state, which is multiplied by f_t to forget the information decided by the forget gate; $\tilde{C}_t$ is the new candidate value for the memory state, which is scaled by i_t as how much the gate decides to update the state value.

- *Output gate* (Figure 2.5 (c)): Decides what to output based on the input and the memory state. Equation 2.10 shows the mathematical representation of the output gate. This gate pushes the cell state values between -1 and 1 by using a hyperbolic tangent function and multiplies it by the output of its sigmoid layer to decide which parts of the input and the cell state to output.

$$\begin{aligned} o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{2.10}$$

An LSTM network for time series modeling takes the values of p past records ($R_{t-i},\ (i = 1, ..., p)$) as input and predicts the value of the current record (R_t) in its output.

The LSTM modeling techniques can model non-linear long-term sequential dependencies among the data records in univariate/multivariate time series, which makes them more practical to real-world applications. Moreover, LSTMs have ability to learn seasonality [76]. However, the trained network is a complex equation over the attributes of the data records, which is not human interpretable.

Support Vector Machine (SVM [67]). An SVM model maps the data from the input space into a higher-dimensional feature space using a non-linear mapping (referred to as a Kernel Function) and then performs a linear regression in the new space. The linear model in the new space represents a non-linear model in the original space.

An SVM for time series modeling uses the training data as pairs of input and output, where an input is a vector of *p* previous data records in the time series and the output is the value of the current data record. Equation 2.11 shows the mathematical representation of a non-linear SVM regression model.

$$R_t = b + \sum_{p} \alpha_i \varphi(R_{t-i}) \tag{2.11}$$

where R_t is the data record at time t, φ is a kernel function, such as Gaussian RBF [77], and R_{t-i} is the i^{th} previous record in the time series.

The SVM modeling techniques can model both linear and non-linear functions for predicting time series values. However, these techniques require an enormous amount of computation, which makes them inapplicable to large datasets [67]. Moreover, the trained model is not human interpretable.

Time Series Decomposition Techniques

These techniques decompose a time series into its components, namely level (the average value of data points in a time series), trend (the increasing or decreasing value in the time series), seasonality (the repeating cycle in the time series), and noise (the random variation in the time series) [78, 79]. Next, they monitor the noise component to capture the anomalies. These approaches report as anomalous the data record R_t whose absolute value of noise is greater than a threshold.

These techniques consider the time series as an additive or multiplicative decomposition of level, trend, seasonality, and noise. Equation 2.12 and 2.13 shows the mathematical representation of additive and multiplicative models respectively.

$$R_t = l_t + \tau_t + s_t + r_t \tag{2.12}$$

$$R_t = l_t * \tau_t * s_t * r_t \tag{2.13}$$

where R_t is the data record at time t, l_t is the level as the average value of data records in a time series, τ_t is the trend in time series, and s_t is the seasonal signal with a particular period, and r_t is the residuals of the original time series after the seasonal and trend are removed and is referred to as *noise*, *irregular*, and *remainder*. In this model, s_t can slowly change or stay constant over time.

In a linear additive model the changes over time are consistently made by the same amount. A linear trend is described as a straight line and a linear seasonality has the same frequency (i.e., width of cycles) and amplitude (i.e., height of cycles) [80].

In a non-linear multiplicative model the changes increase or decrease over time. A non-linear trend is described as a curved line and a non-linear seasonality has increasing/decreasing frequency/amplitude over time [80].

Different approaches are proposed in the literature to decompose a time series into its components. *Seasonal-Trend decomposition using LOESS* (*STL*) is one of the most commonly used approaches, which is described as follows.

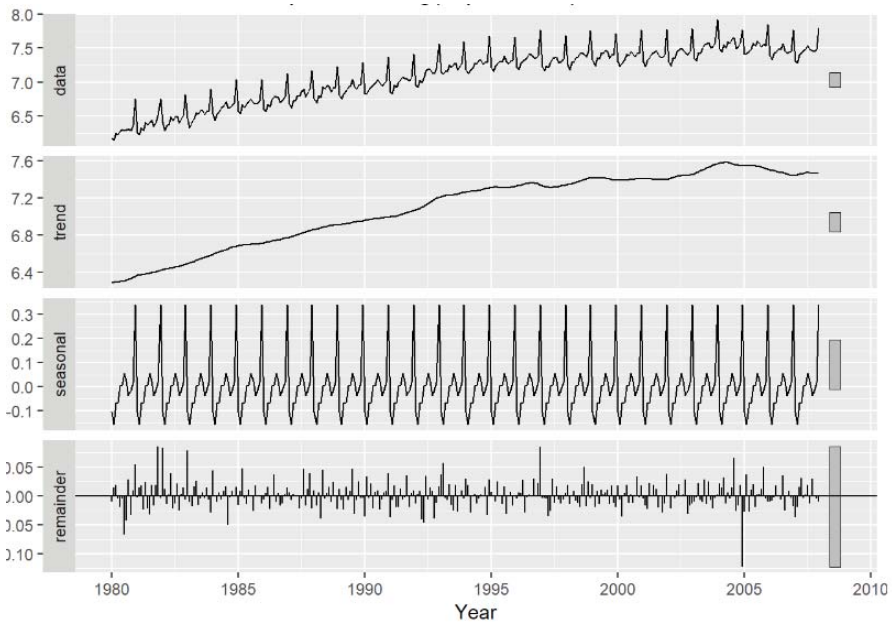

Figure 2.6: STL Decomposition of Liquor Sales Data [5]

Seasonal-Trend decomposition using LOESS (STL). This approach uses LOESS (LOcal regrESSion) smoothing technique to detect the time series components. LOESS is a non-parametric smoother that models a curve of best fit through a time series without assuming that the data must follow a specific distribution. This method is a local regression based on a least squares method; it is called local because fitting at point t is weighted towards the data nearest to t. The effect of a

neighboring value on the smoothed value at a certain point t decreases with its distance to t. Figure 2.6 shows an example of the STL decomposition for a liquor sales dataset. This Figure shows the trend, seasonality, and noise components extracted from an original time-series data.

The time series decomposition techniques provide non-comlex models that can be used to analyze the time-series data and detect anomalies in the data. However, in real-world applications, we may not be able to model a specific time series as an additive or multiplicative model, since the real-world datasets are messy and noisy [80]. Moreover, the decomposition techniques are only applicable to univariate time series data.

2.2.3 Approaches to Detect Anomalous Sequences

The approaches proposed in the literature to detect anomalous sequences are based on (1) splitting the time-series data into multiple subsequences, typically based on a fixed size overlapping window, and (2) detecting as anomalous those subsequences whose behavior is significantly different from the majority of subsequences in the time series. Examples of these approaches are *Clustering*, *Autoencoder*, and *LSTM-Autoencoder*.

Clustering [79]. These techniques extract subsequence features, such as trend and seasonality. Table 2.4 shows the time series features from the TSFeatures CRAN library [66]. Next, an unsupervised clustering technique, such as K-means [54] and Self-Organizing Map (SOM) [81] is used to group the subsequences based on the similarities between their features. Finally, *internal* and *external* anomalous sequences are detected. An internal anomalous sequence is a subsequence that is distantly positioned within a cluster. An external anomalous sequence is a subsequence that is positioned in the smallest cluster.

Distance-based clustering algorithms cannot derive relationships among multiple time series features in their clusters [57]. Moreover, these techniques only detect anomalous sequences without determining the records/attributes that are the major causes of invalidity in each sequence.

Autoencoder [61]. An autoencoder is a deep neural network that discovers constraints in the unlabeled input data. An autoencoder is composed of an *encoder* and a *decoder*. The encoder compresses the data from the input layer into a short representation, which is a non-linear combination of the input elements. The decoder decompresses this representation into a new representation that closely matches the original data. The network is trained to minimize the reconstruction error (RE), which is the average squared distance between the original data and its reconstruction [14].

The anomalous sequence detection techniques based on autoencoders (1) take a subsequence (i.e., a matrix of *m* records and *d* attributes) as input, (2) use an autoencoder to reconstruct the subsequence, (3) assign an invalidity score based on the reconstruction error to the subsequence, and (4) detect as anomalous those subsequences whose invalidity scores are greater than a threshold.

In an autoencoder network for anomalous sequence detection, the input (T_i) and output (T_i') are fixed-size subsequences. T_i is the i^{th} subsequence that contains w records, w is the window size, and $X_{i,j} = [x_{i,j}^0, ..., x_{i,j}^{d-1}]$ is the j^{th} record in T_i with d attributes. The network output has the same dimensionality as the network input. The encoder investigates the dependencies from the input subsequence and produces a complex hidden context (i.e., d' encoded features). The decoder reconstructs the subsequence from the hidden context and returns a subsequence with shape $(d*w)$. The reconstruction error for this network is defined as follows [14]:

$$RE = \frac{1}{m}\sum_{i=1}^{m}(T_i' - T_i)^2 \tag{2.14}$$

where T_i and T_i' are the i^{th} network input and output and m is the total number of subsequences.

These techniques can learn complex non-linear associations among data attributes in the time series as a result of using a deep architecture with several layers of non-linearity. However, these techniques are not able to model temporal dependencies among the data records in an input subsequence.

LSTM-Autoencoder [61]. An LSTM-Autoencoder is an extension of an autoencoder for time-series data using an encoder-decoder LSTM architecture. As described in Section 2.5, an LSTM

network uses internal memory cells to remember information across long input sequences. As a result, an LSTM-Autoencoder can capture the temporal dependencies among the input records by using LSTM networks as the layers of the autoencoder network.

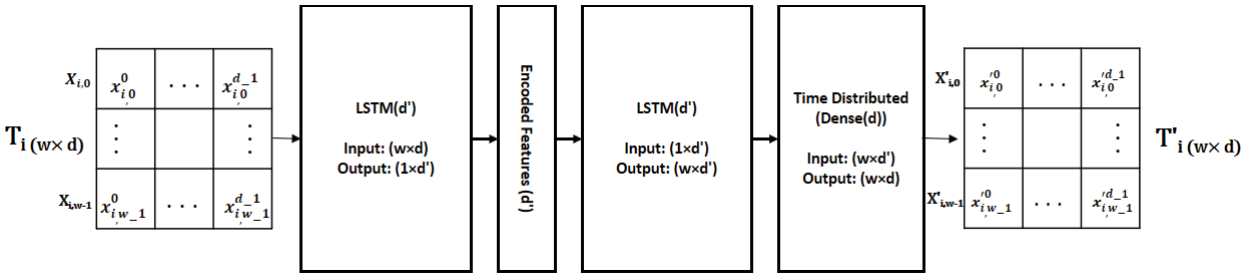

Figure 2.7: An LSTM-Autoencoder Network

Figure 5.3 shows the LSTM-Autoencoder architecture. The input and output are fixed-size time series matrices. $X_{i,j} = [x_{i,j}^0, ..., x_{i,j}^{d-1}]$ is the j^{th} record with d attributes, T_i is the i^{th} time series that contains w records, and w is the window size. The network output has the same dimensionality as the network input. The network is composed of two hidden layers that are LSTMs with d' units. The first LSTM layer functions as an encoder that investigates the dependencies from the input sequence and produces a complex hidden context (i.e., d' encoded time series features, where the value of d' depends on the underlying encoding used by the autoencoder). The second LSTM layer functions as a decoder that produces the output sequence, based on the learned complex context and the previous output state. The TimeDistributed layer is used to process the output from the LSTM hidden layer. This layer is a dense (fully-connected) wrapper layer that makes the network return a sequence with shape $(d * w)$. The reconstruction error for this network is defined as follows [14]:

$$RE = \frac{1}{m}\sum_{i=1}^{m}(T_i' - T_i)^2 \tag{2.15}$$

where T_i and T_i' are the i^{th} network input and output and m is the total number of subsequences.

These techniques can learn complex non-linear long-term associations among multiple data records and attributes as a result of using a deep network and the memory cells in their architecture. However, these associations are in the form of complex equations that are not human interpretable.

2.2.4 Summary

Table 2.5 summarizes different data quality test approaches for anomalous record and sequence detection. We have identified the following open problems in testing the time-series data.

Table 2.5: Data Quality Test Approaches for Sequence Data

Approach	Time Series Type	Fault Type	Modeling Non-linearity	Modeling Seasonality	Modeling Long-term Dependencies
AR	Univariate	Anomalous records			
MA	Univariate	Anomalous records			
ARIMA	Univariate	Anomalous records			
SARIMA	Univariate	Anomalous records		✓	
MLP	Univariate	Anomalous records	✓		
SANN	Univariate	Anomalous records	✓	✓	
LSTM	Mulivariate	Anomalous records	✓	✓	✓
SVM	Mulivariate	Anomalous records	✓		
STL	Univariate	Anomalous records	✓	✓	
Clustering	Univariate	Anomalous sequences	✓	✓	
Autoencoder	Multivariate	Anomalous sequences	✓		
LSTM-Autoencoder	Multivariate	Anomalous sequences	✓	✓	✓

Having potential to generate false alarms. The unsupervised approaches, such as Autoencoder and LSTM-Autoencoder have the potential to learn incorrect constraints pertaining to the invalid data records and sequences and generate false alarms. False alarms can make the anomaly inspection overwhelming for the domain experts [9]. We propose to use an interactive learning-based LSTM-Autoencoder to minimize the false alarms.

Lacking of a systematic approach to set input size. In the existing Anomalous sequence detection approaches, constraints are discovered within an input subsequence, the size of which is typically selected based on a fixed-sized window [82] or by using an exhaustive brute-force approach [83]. Since the window size can considerably affect the correctness of the discovered constraints, fixed-sized windows are not appropriate. Brute-force window-size tuning can be ex-

pensive. We propose a systematic autocorrelation-based windowing technique that automatically adjusts the input size based on how far the records are related to their past values.

Lacking of explanation. The existing data quality test approaches for sequence data do not explain which constraints are violated by the anomalous sequences. Moreover, they do not determine the records/attributes that are major causes of invalidity of the anomalous sequences. We generate visualization diagrams of two types to describe the detected faults: (1) suspiciousness scores per attribute and (2) decision tree.

Bibliography

[1] P. D. Talagala, R. J. Hyndman, K. Smith-Miles, S. Kandanaarachchi, and M. A. Munoz, "Anomaly Detection in Streaming Nonstationary Temporal Data," *Journal of Computational and Graphical Statistics*, pp. 1–21, 2019.

[2] "UCI ML Repository," https://archive.ics.uci.edu/ml/index.php (Accessed 2019-05-14).

[3] A. Singh, N. Thakur, and A. Sharma, "A Review of Supervised Machine Learning Algorithms," in *3rd International Conference on Computing for Sustainable Global Development*, 2016, pp. 1310–1315.

[4] "Recurrent Neural Networks," https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Accessed (21/10/2019).

[5] "Time series decomposition," http://course1.winona.edu/bdeppa/ Accessed (25/10/2019).

[6] "Patient Safety Errors are Common with Electronic Health Record Use," https://healthitanalytics.com/news/patient-safety-errors-are-common-with-electronic-health-record-use (Accessed 2019-08-17).

[7] "Informatica," https://www.informatica.com/ (Accessed 2019-02-12).

[8] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier Analysis*. Springer International Publishing, 2017, pp. 1–34.

[9] B. N. Saha, N. Ray, and H. Zhang, "Snake Validation: A PCA-based Outlier Detection Method," *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 549–552, 2009.

[10] B. Kaminski, M. Jakubczyk, and P. Szufel, "A Framework for Sensitivity Analysis of Decision Trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.

[11] R. M. Konijn and W. Kowalczyk, "An Interactive Approach to Outlier Detection," in *Rough Set and Knowledge Technology*, J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, Eds. Springer Berlin Heidelberg, 2010, vol. 6401, pp. 379–385.

[12] H. Homayouni, S. Ghosh, and I. Ray, "ADQuaTe: An Automated Data Quality Test Approach for Constraint Discovery and Fault Detection," in *IEEE 20th International Conference on Information Reuse and Integration for Data Science*, Los Angeles, CA, 2019, pp. 61–68.

[13] H. Homayouni, S. Ghosh, I. Ray, and M. Kahn, "An Interactive Data Quality Test Approach for Constraint Discovery and Fault Detection," in *IEEE Big Data*, Los Angeles, CA, 2019.

[14] C. Zhou and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders," in *23rd ACM International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.

[15] H. Homayouni, S. Ghosh, I. Ray, and S. Gondalia, "IDEAL: An Approach for Interactive Detection and Explanation of Anomalies using Autocorrelation-based LSTM-Autoencoder for Time-Series Data," in *Submitted to 15th International Conference on Availability, Reliability and Security*, 2020.

[16] P. Kromkowski, S. Li, W. Zhao, B. Abraham, A. Osborne, and D. E. Brown, "Evaluating Statistical Models for Network Traffic Anomaly Detection," in *Systems and Information Engineering Design Symposium*, 2019, pp. 1–6.

[17] "HDC," http://www.ucdenver.edu/about/departments/healthdatacompass/ (Accessed 2020-07-07).

[18] "CSU Plant Diagnostic Clinic," https://plantclinic.agsci.colostate.edu/ (Accessed 2020-05-07).

[19] Y. S. Traffic, "A Benchmark Dataset for Time Series Anomaly Detection," Mar. 2020. [Online]. Available: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly

[20] U. M. Repository, "NASA Shuttle," Mar. 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)

[21] M. Golfarelli and S. Rizzi, "Data Warehouse Testing: A Prototype-based Methodology," *Information and Software Technology*, vol. 53, no. 11, pp. 1183–1198, 2011.

[22] J. Gao, C. Xie, and C. Tao, "Big Data Validation and Quality Assurance – Issuses, Challenges, and Needs," in *IEEE Symposium on Service-Oriented System Engineering*, 2016, pp. 433–441.

[23] S. Dakrory, T. Mahmoud, and A. Ali, "Automated ETL Testing on the Data Quality of a Data Warehouse," *International Journal of Computer Applications*, vol. 131, no. 16, pp. 0975–8887, 2015.

[24] M. G. Kahn, T. J. Callahan, J. Barnard, A. E. Bauck, J. Brown, B. N. Davidson, H. Estiri, C. Goerg, E. Holve, S. G. Johnson, S.-T. Liaw, M. Hamilton-Lopez, D. Meeker, T. C. Ong, P. Ryan, N. Shang, N. G. Weiskopf, C. Weng, M. N. Zozus, and L. Schilling, "A Harmonized Data Quality Assessment Terminology and Framework for the Secondary Use of Electronic Health Record Data," *EGEMS (Washington, DC)*, vol. 4, no. 1, p. 1244, 2016.

[25] M. A. Weiss, *Data Structures & Algorithm Analysis in C++*, 4th ed. Pearson, 2013.

[26] H. Homayouni, S. Ghosh, and I. Ray, "Data Warehouse Testing," *Accepted for Publication in Advances in Computers*, 2018.

[27] "Quality Assurance (QA) by the National Weather Service (NWS)," http://www.nws.noaa.gov/directives/ (Accessed 2018-03-05).

[28] "i-tree Eco," https://www.itreetools.org/ (Accessed 2018-02-11).

[29] "Achilles," https://github.com/OHDSI/Achilles (Accessed 2019-02-12).

[30] G. Hripcsak, J. D. Duke, N. H. Shah, C. G. Reich, V. Huser, M. J. Schuemie, M. A. Suchard, R. W. Park, I. C. K. Wong, P. R. Rijnbeek, J. van der Lei, N. Pratt, G. N. Norén, Y.-C. Li, P. E. Stang, D. Madigan, and P. B. Ryan, "Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers," *Studies in Health Technology and Informatics*, vol. 216, pp. 574–578, 2015.

[31] "PEDSnet," https://pedsnet.org/ (Accessed 2017-05-11).

[32] "Pcori," https://www.pcori.org/ (Accessed 2019-05-15).

[33] D. Loshin, "Rule-based Data Quality," in *11th ACM International Conference on Information and Knowledge Management*, 2002, pp. 614–616.

[34] V. J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[35] Y. Chang, W. Li, and Z. Yang, "Network Intrusion Detection Based on Random Forest and Support Vector Machine," in *IEEE International Conference on Computational Science and Engineering (CSE)*, vol. 1, 2017, pp. 635–638.

[36] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[37] J. Zhang and M. Zulkernine, "A Hybrid Network Intrusion Detection Technique Using Random Forests," in *1st International Conference on Availability, Reliability and Security (ARES'06)*, 2006, pp. 8 pp.–269.

[38] P. B. de Laat, "Algorithmic Decision-Making based on Machine Learning from Big Data: Can Transparency Restore Accountability," *Philosophy & Technology*, vol. 31, no. 4, pp. 525–541, 2018.

[39] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013.

[40] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[41] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154.

[42] M. Swarnkar and N. Hubballi, "OCPAD: One Class Naive Bayes Classifier for Payload based Anomaly Detection," *Expert Systems with Applications*, vol. 64, pp. 330–339, 2016.

[43] P. Lam, L. Wang, H. Y. T. Ngan, N. H. C. Yung, and A. G. O. Yeh. (2017) Outlier Detection in Large-Scale Traffic Data by Naive Bayes Method and Gaussian Mixture Model Method.

[44] R. Zhen, Y. Jin, Q. Hu, Z. Shao, and N. Nikitakos, "Maritime anomaly detection within coastal waters based on vessel trajectory clustering and naïve bayes classifier," *The Journal of Navigation*, vol. 70, no. 3, pp. 648–670, 2017.

[45] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support vector machine for outlier detection in breast cancer survivability prediction," in *Advanced Web and Network Technologies, and Applications*, Y. Ishikawa, J. He, G. Xu, Y. Shi, G. Huang, C. Pang, Q. Zhang, and G. Wang, Eds. Springer Berlin Heidelberg, 2008, pp. 99–109.

[46] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, 2010.

[47] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion Detection by Machine Learning: A Review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[48] M. Markou and S. Singh, *Novelty detection: a review—part 1: statistical approaches*, 2003.

[49] A. Daneshpazhouh and A. Sami, "Semi-supervised Outlier Detection with Only Positive and Unlabeled Data based on Fuzzy Clustering," *Information and Knowledge Technology*, pp. 344–348, 2013.

[50] S. Agrawal and J. Agrawal, "Survey on Anomaly Detection using Data Mining Techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[51] H. Lukashevich, S. Nowak, and P. Dunker, "Using one-class SVM outliers detection for verification of collaboratively tagged image training sets," in *2009 IEEE International Conference on Multimedia and Expo*, 2009, pp. 682–685.

[52] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A Comparative Evaluation of Outlier Detection Algorithms: Experiments and Analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[53] C. C. Aggarwal, "Outlier Analysis," in *Data Mining: The Textbook*. Springer International Publishing, 2015, pp. 237–263.

[54] G. Gan and M. K.-P. Ng, "k-means clustering with outlier removal," *Pattern Recognition Letters*, vol. 90, pp. 8–14, 2017.

[55] M. Ahmed and A. Naser, "A Novel Approach for Outlier Detection and Clustering Improvement." IEEE, 2013-06, pp. 577–582.

[56] F. Cao, J. Liang, and L. Bai, "A New Initialization Method for Categorical Data Clustering," *Expert System Applications*, vol. 36, no. 7, pp. 10 223–10 228, 2009.

[57] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007.

[58] F. Doshi Velez and B. Kim, "Considerations for Evaluation and Generalization in Interpretable Machine Learning," in *Explainable and Interpretable Models in Computer Vision*

and Machine Learning, H. J. Escalante, S. Escalera, I. Guyon, X. Baro, Y. Gucluturk, U. Guclu, and M. van Gerven, Eds. Springer International Publishing, 2018, pp. 3–17.

[59] "Informatica," https://www.informatica.com/ (Accessed 2017-04-14).

[60] G. Dong and J. Pei, *Sequence Data Mining*. Springer Science & Business Media, 2007, vol. 33.

[61] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *19th IEEE International Conference on Mobile Data Management (MDM)*, pp. 125–134.

[62] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 816–825.

[63] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," vol. 33, pp. 1409–1416.

[64] H. Lu, Y. Liu, Z. Fei, and C. Guan, "An outlier detection algorithm based on cross-correlation analysis for time series dataset," vol. 6, pp. 53 593–53 610.

[65] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "Robuststl: A robust seasonal-trend decomposition algorithm for long time series," *CoRR*, vol. abs/1812.01767, 2018. [Online]. Available: http://arxiv.org/abs/1812.01767

[66] TSfeatures, "Cran," Mar. 2020. [Online]. Available: https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html

[67] R. Adhikari and R. K. Agrawal, *An Introductory Study on Time Series Modeling and Forecasting*. LAP LAMBERT Academic Publishing.

[68] "Autoregression Models for Time Series Forecasting with Python," https://machinelearningmastery.com/ Accessed (23/10/2019).

[69] G. A. F. Seber and A. J. Lee, *Linear Regression Analysis*, 2nd ed. Wiley.

[70] I. J. Myung, "Tutorial on maximum likelihood estimation," vol. 47, no. 1, pp. 90–100.

[71] A. Hatua, T. T. Nguyen, and A. H. Sung, "Information Diffusion on Twitter: Pattern Recognition and Prediction of Volume, Sentiment, and Influence," in *Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*. Association for Computing Machinery, 2017, p. 157–167.

[72] D. F. Findley, D. P. Lytras, and T. S. McElroy, "Detecting seasonality in seasonally adjusted monthly time series," *Statistics*, vol. 3, 2017.

[73] N. Tomar, D. Patel, and A. Jain, "Air Quality Index Forecasting using Auto-regression Models," in *IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5.

[74] C. M. Bishop and P. o. N. C. C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[75] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[76] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *Artificial Neural Networks — ICANN 2001*, G. Dorffner, H. Bischof, and K. Hornik, Eds. Springer, 2001, pp. 669–676.

[77] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, May 2009.

[78] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale Unusual Time Series Detection," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015-11, pp. 1616–1619.

[79] N. Laptev, S. Amizadeh, and I. Flint, "Generic and Scalable Framework for Automated Time-series Anomaly Detection," in *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1939–1947.

[80] "How to decompose time series data into trend and seasonality," https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/ Accessed (25/10/2019).

[81] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[82] D. Park, Y. Hoshi, and C. C. Kemp, "A MultimodalAnomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[83] B. Wang, Z. Wang, L. Liu, D. Liu, and X. Peng, "Data-driven Anomaly Detection for UAV Sensor Data Based on Deep Learning Prediction Model," in *2019 Prognostics and System Health Management Conference*, 2019, pp. 286–290.

[84] W. Zhang, T. Du, and J. Wang, "Deep Learning over Multi-field Categorical Data," in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science, N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, Eds. Springer International Publishing, 2016, pp. 45–57.

[85] "Scaling," https://scikit-learn.org/stable/modules/preprocessing.html (Accessed 2019-03-23).

[86] "One-hot Encoding," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html (Accessed 2019-06-24).

[87] L. A. Shalabi and Z. Shaaban, "Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix," in *International Conference on Dependability of Computer Systems*, 2006, pp. 207–214.

[88] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, and T. Huang, "Unsupervised Sequential Outlier Detection with Deep Architectures," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4321–4330, 2017.

[89] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, "A Comparative Study of RNN for Outlier Detection in Data Mining," in *IEEE International Conference on Data Mining*, 2002, pp. 709–712.

[90] W. Zhang and X. Tan, "Combining Outlier Detection and Reconstruction Error Minimization for Label Noise Reduction," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–4.

[91] C. Zhou and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders," in *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.

[92] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2011.

[93] Chen Jin, Luo De-lin, and Mu Fen-xiang, "An Improved ID3 Decision Tree Algorithm," in *4th International Conference on Computer Science Education*, 2009, pp. 127–130.

[94] "Outlier Detection Datasets," http://odds.cs.stonybrook.edu/ (Accessed 2019-08-17).

[95] J. Bergstra and Y. Bengio, "Random Search for Hyper-parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[96] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, USA, 2011, pp. 2546–2554.

[97] M. Kampffmeyer, S. Løkse, F. M. Bianchi, R. Jenssen, and L. Livi, "Deep kernelized autoencoders," in *Image Analysis*, P. Sharma and F. M. Bianchi, Eds. Springer International Publishing, 2017.

[98] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, *Neural Network Design*, 2nd ed. Martin Hagan, 2014.

[99] S. Narayan and G. Tagliarini, "An Analysis of Underfitting in MLP Networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, pp. 984–988.

[100] "H2O," https://www.h2o.ai/products/h2o/ (Accessed 2019-02-25).

[101] "Tensorflow," https://www.tensorflow.org/ (Accessed 2019-05-13).

[102] "Scikit Learn," https://scikit-learn.org/ (Accessed 2019-05-13).

[103] "Pandas," https://pandas.pydata.org/ (Accessed 2020-05-11).

[104] "Statistics," https://docs.python.org/3/library/statistics.html (Accessed 2020-05-11).

[105] "Graphviz," https://pypi.org/project/graphviz/ (Accessed 2020-05-11).

[106] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An Overview on Data Representation Learning: From Traditional Feature Learning to Recent Deep Learning," *Journal of Finance and Data Science*, vol. 2, no. 4, pp. 265–278, 2016.

[107] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[108] Z. Huang, "Clustering Large Data Sets with Mixed Numeric and Categorical Values," in *1st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997, pp. 21–34.

[109] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*, 2nd ed. Wiley-IEEE Press.

[110] "Murdock," https://murdock-study.com/ (Accessed 2019-06-24).

[111] G. Loganathan, J. Samarabandu, and X. Wang, "Sequence to Sequence Pattern Learning Algorithm for Real-Time Anomaly Detection in Network Traffic," in *IEEE Canadian Conference on Electrical Computer Engineering*, 2018, pp. 1–4.

[112] K. I. Park, *Fundamentals of Probability and Stochastic Processes with Applications to Communications*, 1st ed. Springer Publishing Company, Incorporated, 2017.

[113] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. L. Traon, and M. Harman, "Chapter six - Mutation Testing Advances: An Analysis and Survey," *Advances in Computers*, vol. 112, pp. 275–378, 2017.

[114] "Time Series Features," https://tsfresh.readthedocs.io/en/latest/ Accessed (28/10/2019).