System fault detection for multivariate time series

Fubao Wu

Research Intern, Bently Systems, WaterTown, CT

1. Overview

Nowdays, there are a large number of infrastructures and facilities constructed such as buildings, bridges, railways and factory facilities, etc. Many of them are becoming aging with time evolved. To monitor the health of them, a large amount of sensors are deployed around them. Each sensor is a time series data. These leads to the generation of larger multivariate time series data. To analyze these time series data, detecting the system fault and giving proper and timely alerting is a popular and important task. We propose a system fault detection technique with reduced top-k important features for high dimensional multivariate time series based on our previous univariate time series outlier detection methods and on windows techniques.

2. Preliminary

A time series is defined as an ordered sequence of real-valued measurement taken at timestamps $X = x_1, x_2, , , x_T$ with a times step interval st. st could be several seconds, minutes, hours or days. A multivariate time series $D = \{X_i\}_{i=1}^{p}$ is a collection of time series that correspond to the measurements of p real-valued variables spanning the same time interval. Each measurement of is also called one dimension/ feature. There are p dimensions/features for the multivariate time series data. -

Outlier: For a univariate time series, if the data object at a timestamp deviates significantly from the expected value or from the rest of the objects, as if it were generated by a different mechanism, we call the data object has a outlier at that timestamp.

System fault: If there is a high probability that most of the measurements have outliers detected, the system is more likely to break down, we call it the system fault.

Problem statement Given a multivariate time series dataset D with the number of existing fault/breaking down event dates G_d, the problem is trying to detect the probability of the system fault on each time step t with new datasets.

3. Dataset

In our problem, The multivariate time series data are from sensors deployed for a facility of a diamond-mining related company-De Beers, which are deployed around the Southern African ocean. They measure the status of the diamond mining facility. The data are composed of $p = 41$ features, which correspond to 41 measurements of sensors. Each time series span two and half month periods from 02/01/2018 to 04/25/2018. The time step are 5 minutes. Therefore, there are $T = 24000$ records. Each feature corresponds to a real-valued measurement, such as temperature, pressure, voltage or current and so on. There are 4 known system break-down event dates-01/31/2018, 03/10/2018, 03/19/2018, 03/21/2018 when the system has broken down, which are considered as the ground truth of fault dates.

4. Methodology

For univariate time series, the outlier detection method is relatively mature and reliable. In our problem, we have very high dimensional data. One intuitive way is to get a model that can learn the potentially important features from the high dimensions. Considering there are only 4 ground truths for this long periods of time series, if we consider the traditional supervised learning classification method, the unbalanced classes of ground truths would lead us great bias and unexpected test accuracy for a learned model. Therefore we propose to use a semi-supervised learning approach to reduce the high-dimensional features and use the top-k features to do system fault prediction based on univariate outlier detection methods.

The approach is organized as the following steps, which are also shown in Figure 1

1. Divide the multivariate time series data into each univariate time series and use an existing outlier detection approach for each univariate time series detection, which includes preprocessing, decomposition and outlier detection.

2. Based on each univariate time series outlier detection result and the ground truth system fault date, we use a partitioning (fixed window) technique to calculate the feature outlier score for each feature based on different fixed window (m_w) size, a ranking of top features can be obtained.
3. Based on a top-k features, we propose to use moving windows to calculate the probability of system fault for each specified timestamp, e.g. 5 minutes. In this way, each specified timestamp would have a probability of system fault detection with different top-k feature and different moving window size m_w. The optimized f_w, top-k and m_w values are picked based on the best performances that has the best system fault detection score that we designed to evaluate the fault model.
4. Use the the best fixed window f_w, top-k feature and moving window m_w as the suggested default values to do new data prediction, or users can set top-k and m_w values to do system fault detection with new dataset. Each specified timestamp would have a system fault probability, a threshold could be set to decide the probable system fault warning.

4.1. Outlier Detection for Univariate Time Series

For each univariate time series, there are four existing outlier detection methods: Xbar, EWMA, Cusum and SH-ESD. The methods are effective in detecting outlier for univariate time series. It can detect the low outlier and high outlier. There are basically three steps for each outlier detection method. (1) Data preprocess (2) Decomposition (3) Outlier detection. The detailed description could be referred in the technical report [1] For the outlier detection steps, there are some differences we have done here, compared with the operations in the report [1].

4.1.1. Preprocessing

1. To decrease the influence of the noise, we consider a moving window for preprocessing. For example, we set the windows size $ws = 12$, for each continuously 12 time step ts, we get the accumulated average as the new values for each time series. Note the time step is still not changed, but the value in each time step is average of accumulated hourly values of original data.
2. Remove the negative values. For all the dataset, the negative values are removed.

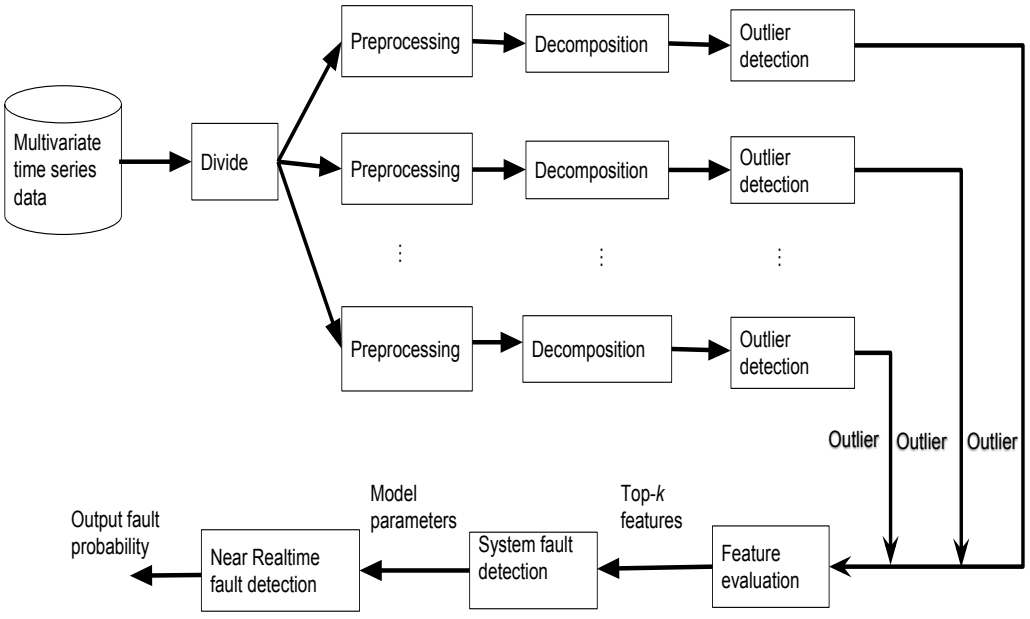


Figure 1: System fault detection overflow

3. Fill the missing time steps, irregular time step and remove duplicated time steps same as in the report [1].

4.1.2. Decomposition

Considering our data do not have obvious periodicals, we use the following decomposing method for each univariate time series value

$$New\ remainder = Original\ value - Trend \tag{1}$$

4.1.3. Outlier Detection

For each time series, we use the new remainder to do the outlier detection with the four outlier detection methods.

4.1.4. Feature Evaluation

With previous steps, for each time series, we can get the result of outlier detections with low or high outliers for each timestamp. To avoid the curse of dimensionality, how to use the important features or use feature reduction to effective and efficient detection of system fault is critical. We propose the

following way to do feature selection. We calculate the feature outlier score for each feature and then get the ranking of the features based on the score.

Feature outlier score Based on the ground truth date and all the univariate-time series outlier detection results, we propose a partitioning way (that is to use fixed windows) to calculate the feature outlier score for each feature. Let's consider one outlier detection method here as an example, such as Xbar. a feature i's detection score is then calculated in this following way with Xbar. For other features and other outlier detection methods, it would have the similar processes.

Figure **??** shows an example of feature i time series.

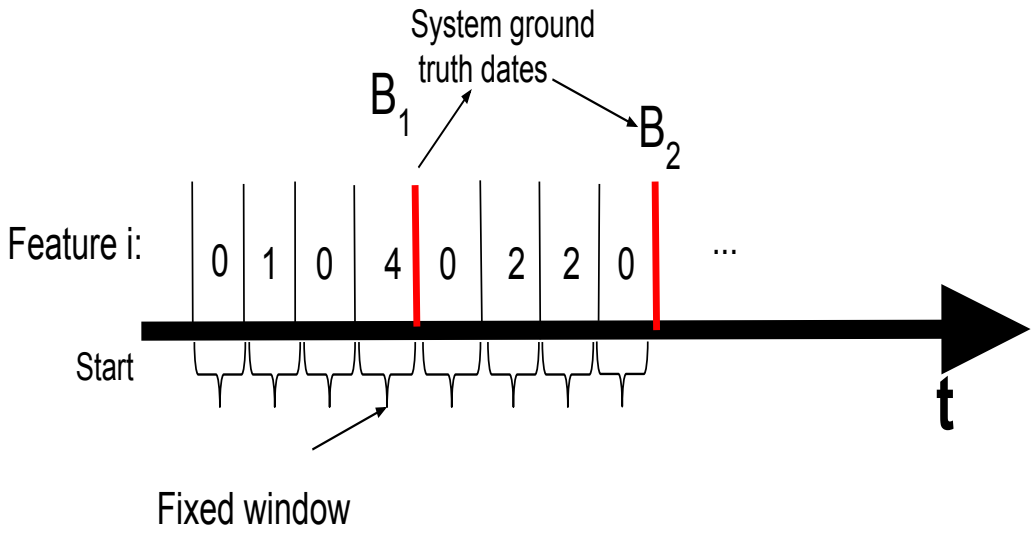


Figure 2: Example of feature outlier score calculation

1. We divide the timestamps of this whole data into several value intervals based on ground truth fault dates. In Figure 2, the red line indicates the start date or a ground truth fault date. The number of interval N equals to the number of ground truth fault dates.
2. Each ground truth fault date and previous ground truth fault date (or the starting date) in each time series comprise of the value interval X_{st} starting indexing from s and ending at t. For each value interval X_{st}, we split the timestamps in this interval based on a fixed windows

parameter f_w. f_w could set $1, 2, 3, ..., 7$ days. Therefore, we get a number of partitions based on a fixed f_w for each interval i. The partition date closest to ground truth date is called the ground truth partition date, other partition dates are called normal dates. In Figure 2, there are interval j and $j+1$, the Partition 1 between red and blue line is the ground truth partition date, the partition 2, 3,... in interval j are normal dates.

3. Based on a f_w value, we calculate the detected value of outlier in each partition for each feature in each interval. If in the interval, there is at least one high or low outlier detected, the value would be "1", otherwise, it is "0". These binary value are shown in Figure 2 in each partition.
4. We calculate the F_1 (or precision) score FS_i of each feature i as the feature's outlier score as follows:

$$FS_i = \frac{1}{N} \sum_{j=1}^{N} f_j \tag{2}$$

Where f_j is the F_1 score of each interval j. We calculate the precision p_j and recall r_j to get each interval's F_1 score f_j, For each interval j, the precision p_j is calculated in this way below based on the true positive and false positive. The recall r_j is calculated based on the true positive and false negative.
True positive (tp_j): number of "1" values in ground truth partition date in an interval j. This is the value in partition 1 in Figure 2
False positive (fp_j): number of "1" values in normal dates in an interval j. This is the number of "1" values in partition 2, 3 and 4 in Figure 2.
False negative (fn_j): number of "0" values in ground truth partition date in an interval j. This is the number of "0" values in partition 1 in Figure 2
Then each interval j's precision is:

$$p_j = \frac{tp_j}{tp_j + fp_j} \tag{3}$$

Each interval j's recall is:

$$r_j = \frac{tp_j}{tp_j + fn_j} \tag{4}$$

Each interval j's F_1 score is:

$$f_j = 2 \cdot \frac{p_j \cdot r_j}{p_j + r_j}. \tag{5}$$

In the interval 1 for ground truth B_1 of Figure 2, $p_1 = \frac{1}{1+4} = 0.2$. $r_1 = \frac{1}{1+0} = 1$. So $f_1 = 2 \cdot \frac{1 \cdot 0.2}{1+0.2} = 0.3333$.

After getting each feature i's outlier score FS_i, we can rank the features by their outlier scores in non-ascending order.

4.1.5. System Fault Model Construction

In this section, we describe how to use our previous proposed technique to do the system fault detection for multivariate time series to construct system fault detection model. After we preset the top-k values, we then use the top-k features to do the system fault detection for this model. To monitor the system status timely and detect the system fault effectively, we propose a probability-based method to detect how likely the system will have faults based on moving windows. Given multivariate time series data, we calculate each specified timestamp t's probability of system fault P_t using a moving windows size m_w. The procedure is described as follows:

1. Set a time unit of observation, such as "each time-step" or "each day".
2. Set a moving window size m_w based the set time unit.
3. **System fault probability:** Calculate the system fault probability P_t at timestamp t based on the m_w and top-k features. p_t is the percentage of number of features detecting outliers in each window time over the value of k.

   $$p_t = n_{tk}k \tag{6}$$

   where n_k is the number of features detecting outlier in the window staring at timestamp t

4.2. System Fault Model Evaluation

To evaluate the "goodness" of the model, we propose system fault detection score (FDS) to evaluate each model's performance. We can try different combinations of parameters of fixed windows size, top-k and moving window size to get different models. For each model, we get a FDS. The FDS is constructed according to the fault probability result and ground truth events. Intuitively, if a probability is closer to a ground truth event, it can have

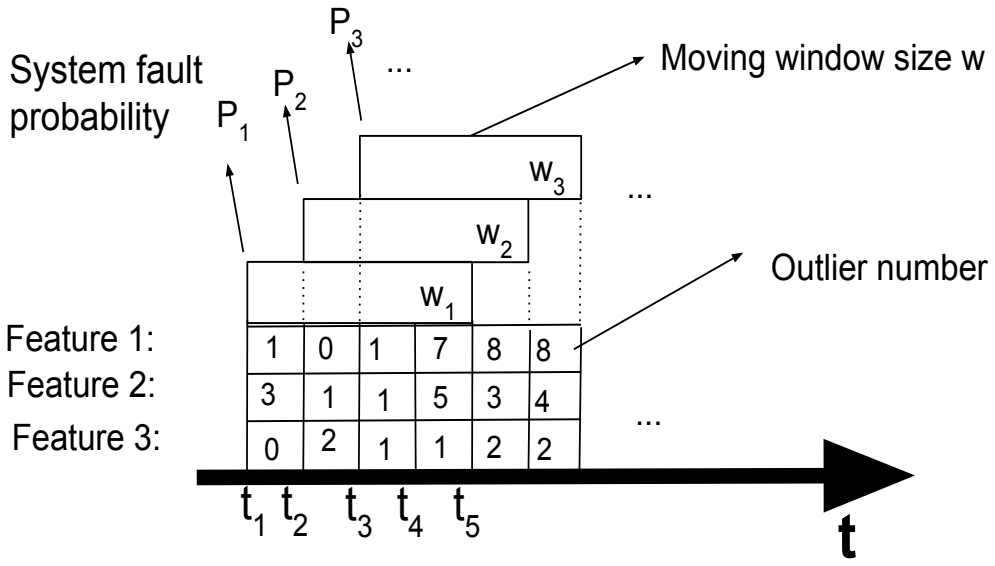


Figure 3: Example of system fault probability

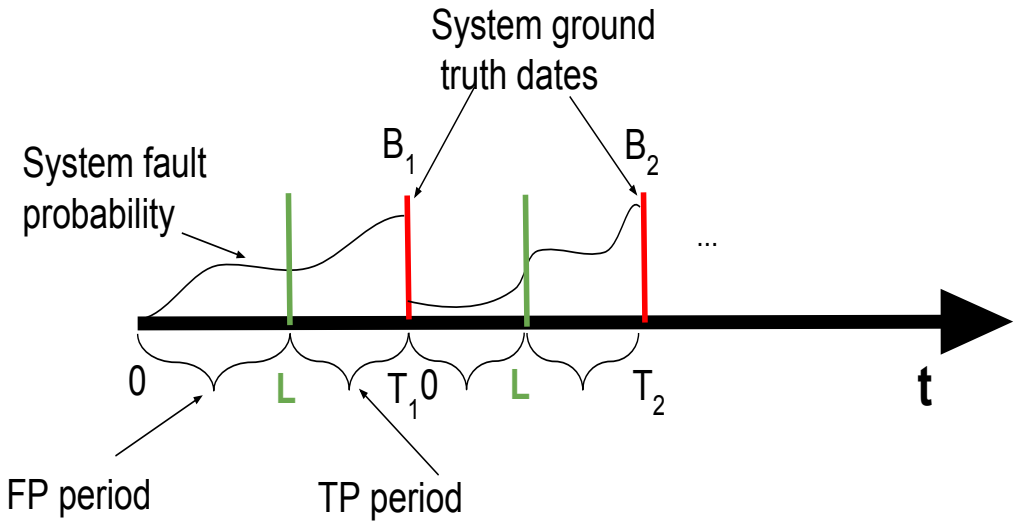


Figure 4: Example of model evaluation with system fault detection score

higher weight. As shown in the Figure **??** We define a ground truth period as a period between a starting date time and a ground truth date time, or between two ground truth date times. Therefore, we divide a ground truth period B_i that has T_i days as a true positive period, and a false positive period L days, which is called a penalty-day parameter. We define a true positive score for a true positive period as $TPS(B_i)$

$$TPS(B_i) = \sum_{t=L}^{T_i} P_t * (1 - \frac{T_i - t}{T_i}) \tag{7}$$

Where $1 - \frac{T_i - t}{T_i}$ is the weight for the fault probability P_t in true positive period. If the t index is closer to the final index T_i, the P_t will have higher weight.

Also, the false positive score as penalty score for a false positive period B_i is defined as $FPS(B_i)$

$$FPS(B_i) = \sum_{t=L}^{T_i} P_t * (\frac{T_i - t}{T_i}) \tag{8}$$

$\frac{T_i - t}{T_i}$ is the weight for the fault probability in false positive period

Therefore, the total system fault detection score DS for a dataset D containing N ground truth dates is the total summation of the differences between true positive score and false positive score

$$DS(D) = \sum_{i=1}^{N} TPS(B_i) - FPS(B_i) \tag{9}$$

Generally, if a fault detection score is higher, the constructed model will have better performance. Therefore, we select different configurations of parameters to find the best model.

4.2.1. Select Different Parameter Configurations

The parameters for a model configuration include fixed window size f_w in feature evaluation stage, the number of top feature k and moving window size m_w.

We could also set the parameters manually, or we could use an automatic way to get the optimized parameter configurations. To find the best configuration of parameters,f_w, k and m_w for each different outlier detection method.

$$f_w, k, m_w = \underset{f_w,k,m_w}{\operatorname{argmax}} S_{wk} \tag{10}$$

4.3. Near Realtime System Fault Detection

When a new dataset comes, we could calculate each specified time step's probability p_t of system fault with the best model configuration found in the previous step. Then we could set a threshold p_low to decide the appropriate warning of system fault when the p_t is above the threshold p_low.

5. Experimental Evaluation

We developed an integrated tool with user interface in C# language. We experimentally verify the approach and test the tool in a Windows platform. Here we show the experimental evaluation results with feature evaluation and system fault detection.

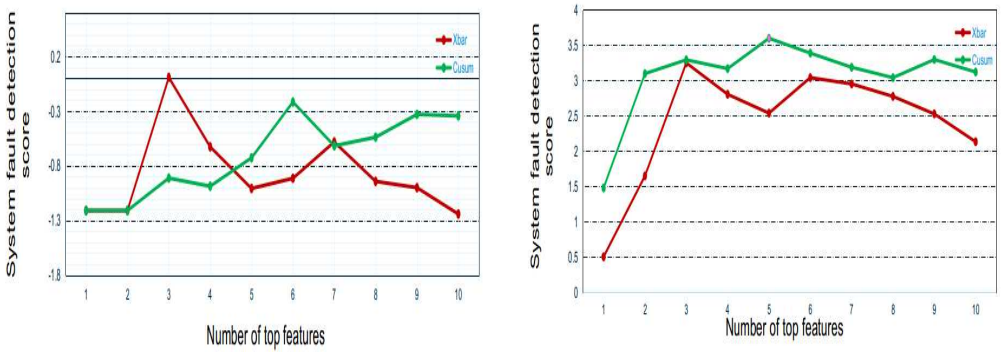
5.1. Dataset Usage

We divide the multivariate time series data from De Beers company into two parts. The first part is used for model construction to find best model, called model construction data D_1, the second part is used for validating the best model, called model validation data D_2. D_1 has 41 features that spans from 02/01/2018 to 03/21/2018, which has 3 ground truth events on 03/10/2018, 03/19/2018, 03/21/2018. Each feature corresponds to a univariate time series that has 5 minute time step. D_2 has 41 features that spans from 03/22/2018 to 04/25/2018, which has 1 ground truth event on 04/10/2018. Each feature also corresponds to a univariate time series that has 5 minute time step.

5.2. Top Feature Evaluation

Top-k features play an important role in deciding the performance of a system fault. To evaluate the effectiveness of top features, we try different numbers of feature to observe the models' performance. Here we show a model's fault detection score to different number of top feature from 1 to 10 for penalty-day $L = 7$ and $L = 14$ with Xbar and Cusum method, respectively. Figure 5a shows different models' performance results for $L = 7$. It shows with the increasing of number of top features, the model fault detection score for both Xbar and Cusum methods increases and then decrease. For Xbar method, the maximum score is reached when the number of top features is 3. For Cusum method, top-6 features correspond to the maximum fault detection score. However, the fault detection score values are all negative, which is not good indication for a model performance.

Figure 5b shows different models' performance results for $L = 14$. It shows the similar result as $L = 7$ for Xbar and Cusum method too. The peak score value is at the top-3 features for Xbar method. The maximum score value for Cusum method is reached at top-5 features.



(a) L = 7 days (b) L = 14 days

Figure 5: System fault detection score vs Number of top features

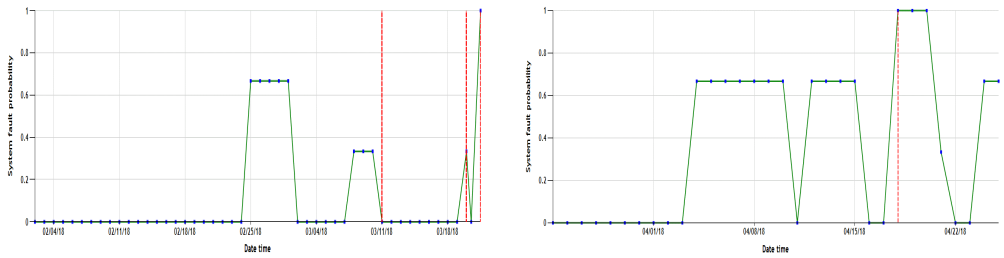
5.3. Examples of Model Construction and Validation

Here we show the model construction and validation result for Xbar and Cusum method based on the best model performance. Figure 6a shows the model construction for Xbar result based on the best top-3 features on data D_1. The top-3 features are. From the figure, the three ground truth events are indicated in the red dotted line. The probabilities of time steps around the ground truth events have basically higher probabilities than other time steps even though there exist a few time steps with high probabilities that are possible to be false positive.

Figure 6b shows the model validation for Xbar result on D_2 with the model constructed in Figure 6a. It shows the validation has very good result around the ground truth event date, which demonstrate the effectiveness of our system fault detection model.

Figure 7a shows the model construction for Cusum result based on the best top-5 features on data D_1. The top-5 features are. Similarly, we can see that the probabilities of time steps around the ground truth events have basically higher probabilities than other time steps even though there exist a few time steps with high probabilities that are possible to be false positive.

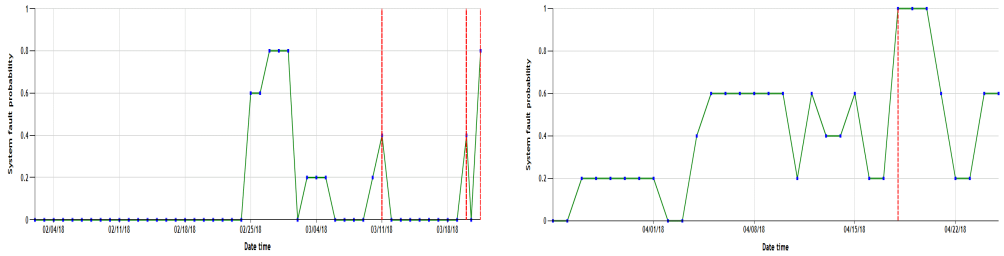
Figure 7b shows the model validation for Xbar result on D_2 with the model constructed in Figure 7a. It shows the validation has very good result



(a) Model construction result for Xbar (b) Model validation result for Xbar

Figure 6: Model construction fault probability and validation fault probability result for Xbar method

around the ground truth event date, which demonstrate the effectiveness of our system fault detection model.



(a) Model construction result for Cusum (b) Model validation result for Cusum

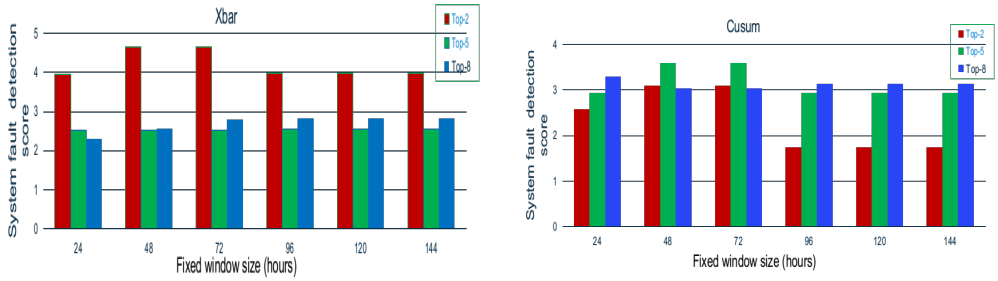
Figure 7: Model construction fault probability and validation fault probability result for Cusum method

5.4. Fixed Window Evaluation

Figure 8a shows the fault detection model performance to different fixed windows f_w (24, 48 to 144 hours) with top-2, top-5 and top-8 for Xbar method. It shows similar trends for these top features that detection score begin to increase and then decrease. The larger f_w does not help much to the performance of the model. Figure 8b shows the fault detection score to different fixed windows for Cusum method. It also shows when the f_w increases to a large number, the fault detection score does not increase. The maximum detection score are basically achieved at f_w at 72 or 96 hours.

5.5. Moving Window Evaluation

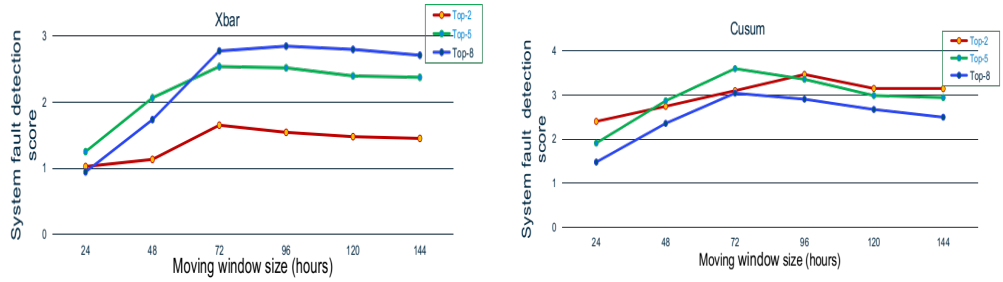
Figure 9a shows the fault detection model performance to different moving windows f_w (24, 48 to 144 hours) with top-2, top-5 and top-8 for Xbar. It



(a) Fixed window evaluation for Xbar (b) Fixed window evaluation for Cusum

Figure 8: Model fault detection score to different fixed window evaluation for Xbar and Cusum methods

shows similar trends for these top features that detection score begin to increase and then decrease. The larger f_w also does not help much to the performance of the model. Figure 9b shows the fault detection score to different moving windows for Cusum method. It also shows when the f_w increases to a large number, the fault detection score does not increase. The maximum detection score are basically achieved at f_w at 72 or 96 hours.



(a) Fixed window evaluation for Xbar (b) Fixed window evaluation for Cusum

Figure 9: Model fault detection score to different moving window evaluation for Xbar and Cusum methods

5.6. Conclusion

We have proposed and developed a system fault detection approach for multivariate time series data and developed the integrated tool to experimentally verified the approach and the tool. From the experimental results, we can see that the different outlier methods show different top feature results. The best model performance

References

[1] Technique report on event detection for smart water management (2018).