STUDENT SUMMER INTERNSHIP TECHNICAL REPORT

Subtle Process Anomalies Detection using Machine Learning Methods

DOE-FIU SCIENCE & TECHNOLOGY
WORKFORCE DEVELOPMENT PROGRAM

Date submitted:

December 20, 2019

Principal Investigators:

Roger Boza (DOE Fellow Student)
Florida International University

Mike Griffel, Mentor
Idaho National Lab (INL)

Ahmad Al Rashdan, Project Investigator
Idaho National Lab (INL)

Tammie Border, Department Manager
Idaho National Lab (INL)

Florida International University:

Ravi Gudavalli, Ph.D. Program Manager
Leonel Lagos Ph.D., PMP® Program Director

Submitted to:

U.S. Department of Energy
Office of Environmental Management
Under Cooperative Agreement # DE-EM0000598

FIU | Applied Research Center
FLORIDA INTERNATIONAL UNIVERSITY

ABSTRACT

Nuclear power plant equipment can fail catastrophically and unexpectedly before scheduled maintenance periods. Such events lead to unscheduled plant shutdown for many days in order to facilitate the repairs. The shutdown is necessary to protect the integrity of the power plant environment and stay in compliance with the regulatory standards for the industry. This course of action causes a significant loss of revenue for the plant. An approach to mitigate the occurrence of equipment failure is to apply machine learning methods to detect subtle process anomalies before the events happen. Machine learning models can be used as a preventative maintenance solution or as an early warning system while the nuclear power plant is operational. The approach taken was to use supervised and unsupervised learning techniques to detect and infer anomalies. K-Means and Isolation forest were used for the unsupervised classification task but did not show any meaningful results. Long Short-Term Memory (LSTM) was used for the supervised task due to the cyclical and temporal nature of the problem and showed promising results.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Nuclear power plant equipment can fail catastrophically and unexpectedly before scheduled maintenance periods. Such events lead to unscheduled plant shutdown for many days in order to facilitate the repairs. A power plant shutdown is necessary to protect the integrity of the power plant environment and stay in compliance with the regulatory standards for the industry. All reparations are safely carried out within the time window provided during this period. The problem with this course of action is that it causes a significant loss of revenue for the plant and drastically reduces the production of electricity.

Ideally a nuclear power plant would have a system that could monitor the equipment status and determine when a failure is about to happen. With a system like this the equipment maintenance plan would be dynamic and scheduled when needed as opposed to being in a fixed time basis. Equipment longevity and performance would be optimal since preventative maintenance is keeping them in peak operating conditions. Plant reliability and stability would also increase allowing for a safer environment.

Knowing when equipment failure is about to happen would allow the nuclear power plant management personnel to act accordingly and proactively. Random and unexpected catastrophic events could be prevented if enough warning and lead time is given by a system at the onset of an anomaly detection. A plan of action could be devised ahead of time to minimize the disruption of the nuclear power plant as well.

Currently the ideal system doesn't exist but there is a large magnitude of sensor data being collected from equipment that needs analysis and inferencing. This data can be used to mitigate the unexpected occurrences of equipment failure by applying machine learning methods to detect subtle process anomalies before the events occur. The machine learning models can be used as a preventative maintenance solution or as an early warning system while the nuclear power plant is operational.

2. EXECUTIVE SUMMARY

This research work has been supported by the DOE-FIU Science & Technology Workforce Initiative, an innovative program developed by the US Department of Energy's Environmental Management (DOE-EM) and Florida International University's Applied Research Center (FIU-ARC). During the summer of 2019, a DOE Fellow intern Roger Boza spent 11 weeks doing a summer internship at Idaho National Lab (INL) under the supervision and guidance of Mike Griffel. The intern's project was initiated on June 1, 2019, and continued through August 16, 2019 with the objective of performing data analysis and anomaly detection using machine learning methods.

3. RESEARCH DESCRIPTION

The research was conducted in three phases. Each phase was dedicated to one type of Machine learning algorithm. At the beginning of each phase the data was prepared, curated, and formatted properly for each algorithms. Considerations for space and time complexity were taken since K-Means, Isolation Forest, and LSTM models all behave differently. Each model has specific requirements and excels at their intended purpose.

K-Means

K-Means is an unsupervised machine learning method for classical quantization in signal processing and cluster analysis in data mining. It is most often used with datasets that don't have predefined categories or groups. The goal of the algorithm is to make *k* distinct clusters from *n* total observations. It works by assigning each data point in the dataset to one of the *k* groups based on all the features available. The approach is computationally difficult and falls under the category of NP-hard (non-deterministic polynomial-time) problems. Although K-Means falls under such category, an efficient heuristic algorithm can converge on a solution rather quickly and find a local optimum.

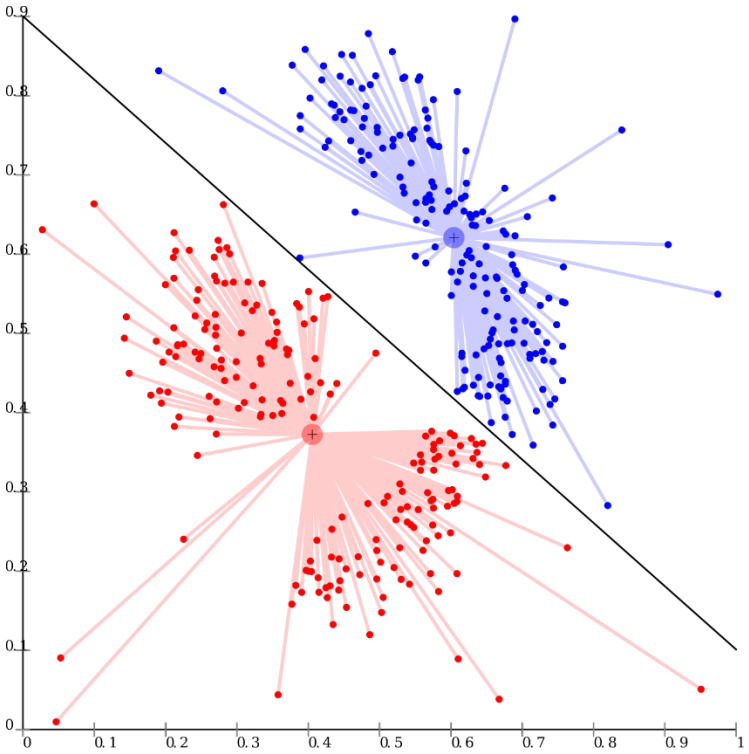

Figure 1. Sample two-dimensional dataset to visualize the distance from points to centroid. Farthest points can be considered anomalous. (https://commons.wikimedia.org/w/index.php?curid=17085333)

Each one of the *k* clusters is defined by its centroid. The centroids are the mathematically computed arithmetic mean of the points within the individual clusters (geometric center). The K-means algorithm operates by iterating between two main steps. At the initiation of the process each cluster gets a random computed centroid or a randomly selected one from the original

dataset. The first step is to assign data points to the nearest centroid. The second step is to recalculate the centroid with all the data points assigned to the cluster. These two steps are done until a stopping condition or criteria is met. Common stopping conditions include reaching a set limit for the amount iterations, the sum of the distances has been minimized, or no data points have changed cluster assignment.

K-Means Pseudocode:

1. Choose the number of *k* clusters
2. Obtain the data points from the dataset
3. Place centroids c_1, c_2, …, c_k randomly
4. Repeat steps 5 and 6 iteratively until stopping condition is met
5. For each data point x_i
    a. Find the nearest centroid (c_1, c_2, …, c_k)
    b. Assign the data point to that cluster
6. For each cluster c_1, c_2, …, c_k
    a. New centroid = mean of all points currently assigned to that cluster.
7. End.

In most cases the number of clusters, *k*, is not known a priori and must be found through empirical observations. For this task we use a technique called the Elbow method and its interpretation for validation. The Elbow method is a heuristic method designed to help find the appropriate number of clusters in a dataset. The analysis begins by determining the minimum and maximum number of clusters for the algorithm. The minimum number of clusters is trivially chosen and set to one ($k = 1$). The maximum number of clusters is the total number of features present in the dataset (k = # of features). After the domain of clusters has been chosen, the analysis proceeds by calculating a score value for each number of *k* clusters. The score is implemented in the Scikit-learn python library as the “opposite of the value of X on the K-Means objective”, where the objective function is to minimize the sum of squares of the distances of all data points in their respective clusters. All the scores are then graphed in a line plot and analyzed for the elbow criterion. The optimal number of k clusters is where the graph shows minimal gain by increasing the number of clusters, the “elbow” in the graph.

K-Means Cluster Analysis Pseudocode

1. Determine the total number of features *n* in dataset
2. For each cluster $j = 1 \ldots n$
    a. Calculate score for K-Means cluster using $k=j$
3. Graph number of clusters and its respective score
4. Look for elbow criteria on the line graph, optimum value
5. Set *k* to optimum value
6. End

The K-Means algorithm can be used as an anomaly detector by analyzing the distance each data point has to its centroid. Closely related points are concentrated around the centroid and are considered normal. Data points that fall far away from the centroid are still considered to be related to its neighboring points but with a lower affinity and thus can be classified as anomalous. The anomaly detector could work by using the distance as a notion of relationship and apply it as a discriminator between normal and abnormal. A threshold on the distance can then be set to determine if a data point is normal or an anomaly.

To execute the K-Means algorithm an environment was properly created within the Anaconda distribution for the Python programming language. Anaconda is entirely open-source and aims to simplify package management and deployment through the conda system. The environment was setup with Python 3.6.8 and Jupyter Notebook version 4.4.0. Multiple libraries were installed using the conda system to facilitate data manipulation. Numpy and Pandas were mainly used to store the time-series datasets. Scikit-learn was used to leverage the machine learning algorithm and the internal preprocessing ability. Matplotlib was used extensively to visualize the analysis and show the results of the anomaly detector.

Isolation Forest

Isolation Forest is another unsupervised machine learning algorithm that is built based on decisions trees. The main idea, which is very different in representation from many other popular outlier and anomaly detection algorithms, is that an Isolation Forest explicitly focuses on identifying anomalies instead of characterizing the normal data points in the data set. The Isolation Forest algorithm ‘isolates’ observations by randomly selecting a feature from the feature space and then randomly selecting a split point from the maximum and minimum values of the selected feature. In principle, outliers have much less frequent occurrences than regular observations in the data and have subtle differences in terms of values.

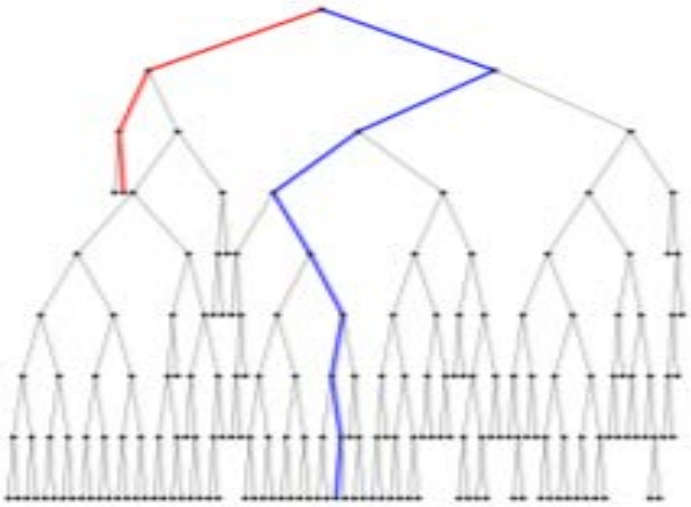

Figure 2. Representation of a single tree in the Isolation Forest. The red path is short and close to the root of the tree meaning anomalous. The blue path is long and qualifies as a normal observation (Hariri, Kind, & Brunner, 2018).

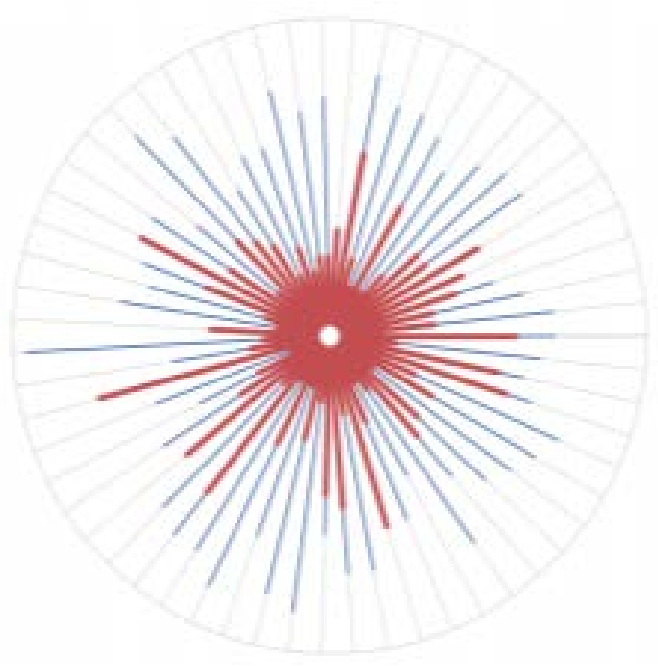

Figure 3. Visual representation of a full forest were each radial line represents the path length of a single tree. Red represent anomalous while blue represents normal. (Hariri et al., 2018).

Isolating anomalies is easier as fewer conditions are needed to separate them from the rest of the dataset set. In contrast, normal observations require more conditions to isolate them. Since recursive partitioning can be natively represented by a tree structure, the number of splits required to isolate an observation is equivalent to the length of the path from the root node of the tree to the leaf the observation is in. The path length, when averaged over all random trees in the forest, is a measure for anomaly in the observation.

Isolation Forest Pseudocode

1. Select a point from the dataset to isolate
2. For each feature f_i
    a. Set the minimum in the range
    b. Set the maximum in the range
3. Choose a feature at random
4. Pick a value within the range at random
5. Repeat steps 3 and 4 until the point is isolated
    a. The point is the only one within the range for all features
6. Count how many times steps 3 and 4 were done
    a. This is the path length
        i. p_{length} = count of steps 3 and 4

7. Repeat steps 2 to 6 for each tree t_i in the forest
8. Sum p_{length} for all t_i and get the average a_{pl}
9. End

The process of random partitioning generally produces shorter path length for anomalous data. It is very likely for an observation to be an actual anomaly when a random generated forest collectively produces shorter paths for such observation. Randomizing the process helps correct the issue decision trees naturally have of overfitting based on the training set.

The approach taken and implemented was to use the path length metric in the Random Forest algorithm and apply it over a search space for multiple time frames and contamination rates. The contamination rate is a parameter passed to the Scikit-learn machine learning module in Python that describes the proportion of outliers in the data set. It is used when fitting the data to the model to define the threshold on the decision function.

Datasets where prepared with different time scale granularities. This decision was made to find different abstract patterns in the data as well as decrease model computation time during the learning process. The time-series data resampling performed was based on the arithmetic mean and each dataset was saved in an individual file for faster processing. The files were serialized using Python's "pickling" algorithm for object serialization. The contamination rate search space was composed of the values r_1, r_2, and r_3. These values are selected specifically to fine tune the model and to calibrate the sensitivity to anomalous observations. PCA was also applied to the data and an additional analysis was also conducted to evaluate the effects of dimensionality reduction on the algorithm.

Implementation Pseudocode

1. Create time granularity t = (minutes, hours, days, weeks)
    a. Resample original dataset using arithmetic mean
2. Set contamination rate
        i. $r = (r_1, r_2, r_3)$
3. For each t_i
    a. Create Isolation Forest
    b. For each contamination rate r_i
        i. Select data points average path length < r_i
        ii. Classify them as anomalous
4. End

The Anaconda environment used for the K-Means algorithm was reused to execute the Isolation Forest implementation. One additional library within the Scikit-learn package, IsolationForest, was imported to handle the creation and logic for the algorithm. This library handled the creation of all random trees in the forest and returned the binary (-1 or 1) classification for the dataset. A one-to-one mapping was done from -1 to 1 and 1 to 0 were the mapped 0 meant "normal" and the mapped 1 meant "anomalous". Matplotlib was used to visualize the mapped results as a color-coded scatter plot.

LSTM

A recurrent neural network (RNN) is a class of artificial neural networks (ANN) where the connections between nodes form a directed graph (digraph). The digraph is made up of a set of vertices which are connected by edges. All the edges have a start and end vertex as well as a direction associated with them. In the case of the RNN, the digraph is formed along a temporal sequence which allows the network to exhibit a dynamic temporal behavior. RNNs are exceptionally well suited to process sequence of inputs due to their inherent internal state (memory). This makes them applicate to tasks dealing with time-series dataset such as the one for this project.

Long short-term memory (LSTM) is an artificial RNN architecture that extends the memory of a regular RNN. The 'short term memory' in the LSTM is exhibited through persistent previous information that is used in the current neural network. It also mitigates the vanishing gradient problem, which is the networks inability to learn because the updates to the weights within the nodes become too small (insignificant) to alter the output. The LSTM does the mitigation by using a series of gates contained in memory blocks that regulate the flow of information.

A typical LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The input gate scales input to the cell and behaves as a write operation. The output gate scales output to cells and behaves as a read operation. The forget gate scales old cell values and behaves as a reset operation. Each of the gates in the LSTM unit behaves as a switch to control read/write operations and thus gives the effect of long-term memory to the model.

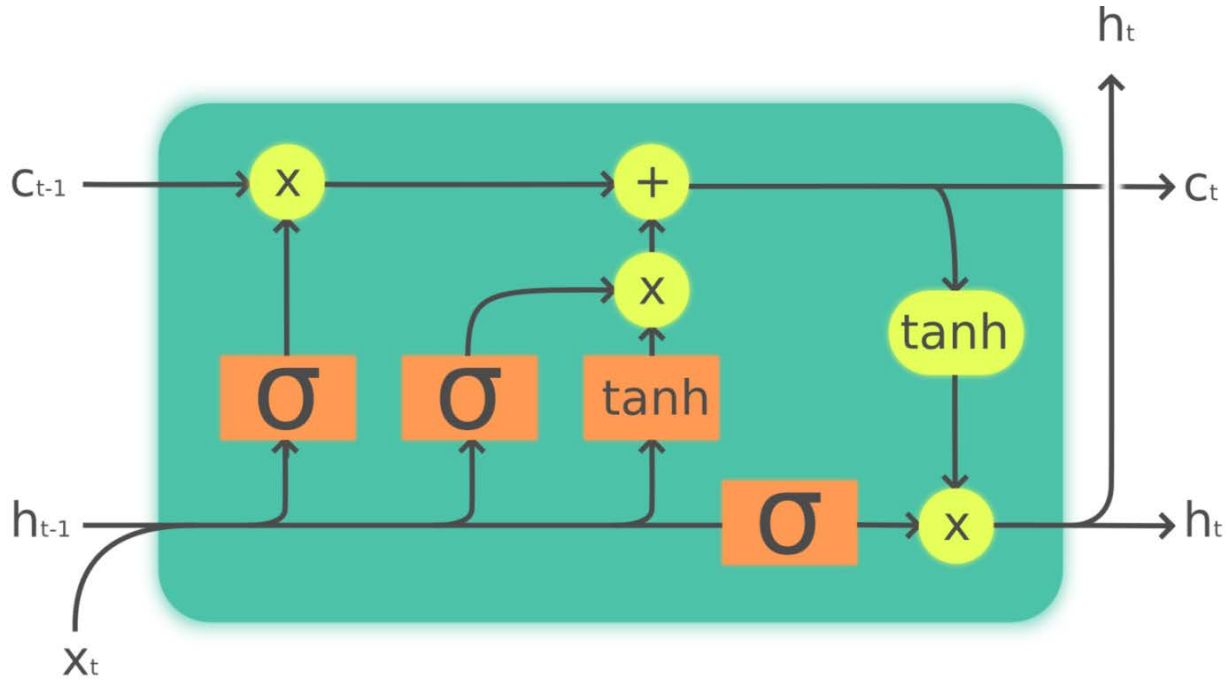

Figure 4. Single LSTM unit showing the input, output, and forget gates which give the network memory and the ability to process data sequentially. (https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png)

Memory allows LSTM networks to excel at finding complex relationships within a multivariate feature space. LSTMs ability to process sequential data and have internal memory cell units made it perfect to model the time-series dataset for the project. The memory attribute of the architecture would be able to remember seasonal patterns as well as cyclic patterns that constitute normal operating conditions. Given enough data, LSTMs have the potential to learn patterns from one year to the next as well.

To train the LSTM the dataset had to be properly curated and processed. A standalone Jupyter Notebook was created to handle the curation process. A custom defined function was used to remove inconsistent data from the original dataset file. After the original dataset was curated it had to be formatted accordingly to be used as input to the LSTM network. It had to be structured in time-windows of size *l*. The variable *l* is a hyperparameter and it is not trivial to choose the optimum value. A value of ten ($l = 10$) was used in the initial assessment and used as a starting guide.

To construct each time-window a moving frame of size *l* is applied to the dataset iteratively with a step of one unit. Each time-window must be verified for usability as all the datapoints inside have to be temporally sequential in reference to the time unit. There are cases where a time-window contain a set of data points which are not temporally sequential as a result of the data curation process and must be disregarded as an input to the LSTM.

Dataset Time-window Pseudocode

1. Use curated dataset
2. Set l =10
3. For each record r_i in the dataset
    a. Create an array (time-window) with row values from r_i to r_{i+l}
    b. Verify that the array is temporally sequential
        i. If true
            1. Add array to training set
        ii. If false
            1. Disregard time-window
4. End

The time-window formatted dataset is given to the LSTM as training with the purpose to output the desired feature in the following time step. The internal architecture of the network is composed of multiple stacked LSTM layers followed by a Dense layer which is responsible for the output of the model. The loss function which needs to be minimized in the learning process was set to mean squared error (MSE). The optimizer used in the network was root mean square propagation (rmsprop) as it has the benefit of automatically adjusting the learning rate.

To execute the LSTM algorithm an additional environment was created within the Anaconda distribution. The environment was setup with Python 3.6.8 and Jupyter Notebook version 4.4.0. The Keras machine learning library was installed and used to handle the architecture of the LSTM. Keras is a wrapper library for the TensorFlow library that provides machine learning capabilities. Since this approach for anomaly detection is very computationally expensive, the graphical processing unit (GPU) version of Keras and TensorFlow were activated.

GPU processing speeds up the computation time exponentially when compared to the central processing unit (CPU). Numpy and Pandas were used to store the time-series datasets and create time-windows for input to the network. Matplotlib was used to visualize how the model was learning during the training and to check the loss function as time was advancing. The graphing abilities were also used to show the actual values of the training set compared to the predicted values of the final model after training.

4. RESULTS AND ANALYSIS

The goal for the K-Means algorithm was to cluster the dataset and select the outliers as anomalies. A search space for the threshold was created that contained the top 1%, 2%, and 5% points farthest from the centroid. Anomalies were detected with the 5% threshold but included a high quantity of false positives. A dimensionality reduction technique called Principle Component Analysis (PCA) was applied to the dataset as an effort to lower the misclassification error. The number of components used from the PCA was selected after finding the elbow in the component plot which graphed the cumulative explained variance. With the PCA dataset the algorithm was able to detect anomalies at 2% threshold but still contained false positive.

For the Isolation Forest approach, the models were trained with the regular dataset and the PCA dataset. The goal for the models was to catch the anomalies and label them appropriately while reducing the number of false positive predictions. A grid search for contamination rates within the forest was created that included values 0.01, 0.05, and 0.1. Unfortunately, none of the models were able to the anomaly classification task while having the misclassification rate below the desired threshold. An important observation made was that the PCA dataset allowed the Isolation Forest to make better predictions.

The LSTM models showed promising results during the training phase as it was able to learn what the normal behavior was. The model's loss function consistently decreased for both the training set and validation set. More importantly, the loss function for both sets didn't deviate from each other and converged at a low value. Convergence of the loss function for the datasets mean that the predictive model abstracted enough information to learn and not memorize the relations. The LSTM models also had very low root mean squared error (RMSE) for both the training data set and the internal validation set. The low values in the RMSE are indications that the model also learned some of the natural operating behaviors and cycles.

In order to predict that data is anomalous with the LSTM a postprocessing technique was used. The technique used statistical analysis such as standard deviation and algebraic relations such as line intercept functions to look for a characteristic "phenomenon" in the predictions.

5. CONCLUSION

The K-Means and Isolation Forest machine learning models were not able to catch the subtle anomalies in the equipment sensor dataset with enough accuracy for the approach to be viable. The LSTM model was able to do much better but required a new post processing technique. Using this approach, the model was able to foresee an anomaly days ahead of a catastrophic event. The subtle anomaly detected by the LSTM model could be used as the early warning sign or as a preventative maintenance measure. Further testing and modifications are still required for the post processing approach since the model has not eradicated the issue with false positives.

6. REFERENCES

Alex. (2018, November 4). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. Retrieved from https://arxiv.org/abs/1808.03314.

Anomaly detection. (2019, December 9). Retrieved from https://en.wikipedia.org/wiki/Anomaly_detection.

Chunming, X., Haibo, J., & Jianjiang, Y. (2008). Robust two-dimensional principle component analysis. *2008 27th Chinese Control Conference*. doi: 10.1109/chicc.2008.4605066

Ji, L., Lin, H., Hong, T., & Zhongtao, H. (2018). Clustering Analysis Based Modeling Method of Bridge State Anomaly Distribution. *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. doi: 10.1109/iicspi.2018.8690391

K-means clustering. (2019, December 3). Retrieved from https://en.wikipedia.org/wiki/K-means_clustering.

Kaur, P. (2016). Outlier Detection Using Kmeans and Fuzzy Min Max Neural Network in Network Data. *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*. doi: 10.1109/cicn.2016.142

Long short-term memory. (2019, December 1). Retrieved from https://en.wikipedia.org/wiki/Long_short-term_memory.

Luo, W., Liu, W., & Gao, S. (2017). Remembering history with convolutional LSTM for anomaly detection. *2017 IEEE International Conference on Multimedia and Expo (ICME)*. doi: 10.1109/icme.2017.8019325

Malhotra, Pankaj, Ramakrishnan, Anusha, Anand, Vig, … Gautam. (2016, July 11). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. Retrieved from https://arxiv.org/abs/1607.00148.

Principal component analysis. (2019, December 11). Retrieved from https://en.wikipedia.org/wiki/Principal_component_analysis.

Ramos, R. G. D. S., L., P. R., & Cardoso, J. V. D. M. (2016). Anomalies Detection in Wireless Sensor Networks Using Bayesian Changepoints. *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. doi: 10.1109/mass.2016.064

Xu, D., Wang, Y., Meng, Y., & Zhang, Z. (2017). An Improved Data Anomaly Detection Method Based on Isolation Forest. *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*. doi: 10.1109/iscid.2017.202