FauDigPro: A Machine Learning based Fault Diagnosis and Prognosis System for Electrocardiogram Sensors

Aakash Agarwal
Dept. of Electronics and Communication
IIIT Naya Raipur
Chhattisgarh, India
aakash19101@iiitnr.edu.in

Aparna Sinha
Dept. of Electronics and Communication
IIIT Naya Raipur
Chhattisgarh, India
aparna@iiitnr.edu.in

Debanjan Das
Dept. of Electronics and Communication
IIIT Naya Raipur
Chhattisgarh, India
debanajn@iiitnr.edu.in

***Abstract*—Nowadays, sensors play a vital role in monitoring many appliances in various sectors, including the medical field. The electrocardiogram (ECG) is a test used to check the heart's electric activity, using the ECG sensors. These sensors can get faulty and cause serious havoc. In this paper, a real-time classification and prognostics system is proposed that could classify between normal and faulty ECG signals and could even predict whether the sensor will get faulty based on previous data. Drift and bias fault are considered in this study. Four different machine learning algorithms, the Support Vector Machine, K-Nearest Neighbour, Decision Tree and Naive Bayes, are used and compared for classification purposes. Three statistical time-domain features, mean of the signal, energy of the signal and the peak-to-peak value of the signal, are used as the feature set in this study. The autoregression algorithm is used for fault prognosis. To replicate a practical scenario in an industrial system, an unbalanced dataset is generated having a larger number of normal signals than faulty signals. The model is implemented on raspberry pi for low resource hardware implementation. The performance of the classifiers is evaluated using different metrics. From the results, we can see that the KNN algorithm achieves the best accuracy of 95% which is suitable for real-time deployment. We also see that a very good result in the case of fault prognosis is achieved owing to the fact that the faults introduced are linear in nature. Different Root Mean Squared Error-values are presented in the paper for both faults.**

***Index Terms*—Fault Diagnosis, Prognostics, Electrocardiogram, Machine Learning.**

I. Introduction

In today's era, hundreds of sensors are deployed every day in sectors like smart cities and industries. These sensors have to face complex and challenging environments. The performance of these sensors is affected by natural factors such as electromagnetic interference and many other factors. A faulty sensor can either produce meaningless signals or stop making signals altogether. Such a sensor acts unstably. Sensor fault detection is a field researchers have worked in the past few years. Such systems are required to improve the quality and reliability of a sensor system. Sensor systems in a healthcare facility require a lot of attention because money and life are at stake. A fault is expressed as an unusual behaviour of a system or a machine. [1].

Since the 1980s, sensor fault detection has been a hot topic owing to its importance in society. Studies have been carried out for fault detection in industrial facilities using mathematical or physical-based models. Since the computational power was not very advanced back then, these approaches had to be limited within a specific environment to reduce the number of model variables as the industrial systems are highly complex. Recently, data-driven methods have been proposed using machine learning techniques that can identify the patterns in the data and give better results. Currently, we have very complex systems in the industries thus making mathematical and physical-based approaches obsolete and difficult to use. Whereas data-driven methods work fantastically on real systems using the collected or historic data. [2], [3], [4].

Although several papers exist regarding sensor fault identification, none of them have talked about working explicitly about ECG sensor. Not much work have been done in deploying a real time fault identification system. We propose a real time fault detection system and test it on Raspberry Pi, which is a low computational power hardware.

In a mechanical system, the fault can occur in actuators or in sensors. In the past, fault detection in the rolling elements of a mechanical machine has been studied extensively which produced good results. In a medical setting, faults in a sensor can cause serious life-threatening events leading to consequences that affect both life and money. Studies have also been done on what kind of faults a sensor can have. Out of the many faults that can occur in a sensor, in this work, we focus on the two most occurred faults, the drift and the bias fault. In a drift fault, the sensor readings increase or decrease from the normal reading linearly. In a bias fault, the sensor readings increase or decrease by a constant. More on these faults have been explained in later sections. The authors in [4] presented a comparative study of the performance of the different machine learning techniques, like KNN, SVM, etc., for real-time drift fault detection in Raspberry Pi. In this work, the authors have only considered drift fault and have worked on binary classification. Also, they have not considered any specific use case sensor but have considered a general sensor. Since

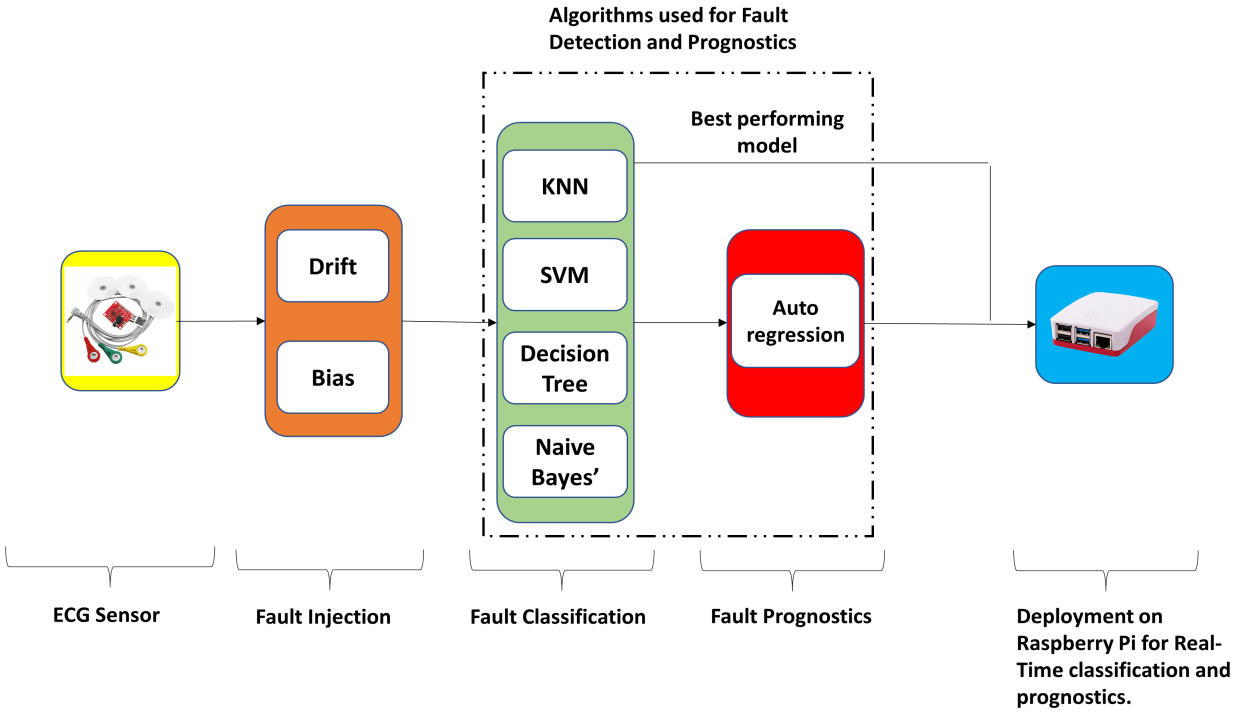

Fig. 1. Framework of the proposed methodology

it is only a two-class classification, the accuracies reported are good. In [5], the authors worked on fault produced in temperature to voltage converter data. They used the Support Vector Machine algorithm to classify the five different kinds of fault (erratic, drift, hard-over, spike, and stuck). They used 5 to 10 different features and got an accuracy of about 90%. They used the cross-validation technique to tackle the problem of overfitting.In this work [6], the authors have again used the Support Vector Machines algorithm to classify faults in a Wireless sensor network (WSN). They have considered 6 fault types (Offset, Gain, Stuck-at, Out-of-bounds, Data-loss and Random). They used an already available dataset of the wireless sensor network data.

The contributions of this paper are outlined as follows:

- A Lightweight System is proposed which is computationally very cheap and low cost in terms of power consumption.
- A Real-Time Fault Detection and Prognostics methodology is proposed which uses a machine learning approach. A real-time system is necessary since we are working in health-related domain.
- An imbalanced dataset is used for training and testing to mimic a real-world scenario as chances of a sensor being faulty in real-world are generally very low. This makes our system robust and adaptive to even small changes.

II. Proposed Fault Classification and Prognostics Methodology

A. Data Acquisition

The dataset used in this paper is the WESAD dataset [7]. Since the collected data is noisy, pre-processing techniques such as such as taking the moving average was done to remove the noise. There is no data available publicly containing the faulty ECG sensor values therefore we have created our own synthetic data set by injecting faults manually using the MATLAB software. There are mainly two types of fault that we induce in this study, the bias fault and the drift fault.

First is the drift fault. In a drift fault, the output of the sensor increases or decreases in a linear fashion from the normal state. The fault can be injected using the equation (1) which shows the value of the output value S_{drift} at time $t+1$ for an input S at time t. [4].

$$S_{drift}(t+1) = S(t) + x, \tag{1}$$

where $S_{drift}(0) = S(0)$ and x is the drift constant value that can be positive or negative but for our case it ranges between 0.01-0.5. Fig. 2(a) shows an example of a drift fault injected in a normal ECG sensor data.

The second type of fault is the bias fault. In this type of fault, the output value of the sensor increases or decreases by a constant value. WE have induced such type of fault at random parts of the signal. The output value S_{bias} at time t for original S at time t is given in Eq. (2) [4].

$$S_{bias}(t) = S(t) + b, \tag{2}$$

where b is the bias constant value that can be positive or negative but for out case it ranges between -1 to 1. Fig. 2(b) shows an example sample of a normal and bias fault injected signal.

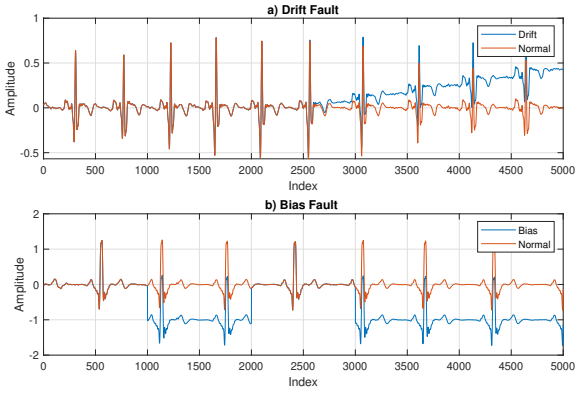

Fig. 2. a) Drift added to the normal ECG signal. b) Bias added to random parts of the normal ECG signal

The data is divided into 1200 samples of normal ECG data and 400 each of both bias and drift fault injected data. Each sample consists of 5000 data elements. The resultant dataset consisted of $(1200 \times 5000 + 2 \times 400 \times 5000)$ data elements. The difference in the number of files is kept to simulate a real life example as the probability of occurring of faults in real life will be comparatively low. This will make our system robust.

B. Feature Extraction

Feature selection is a necessary step in not just fault diagnosis problems but also in every machine learning related problems [8]. Feature selection techniques have been explored in many studies [9]– [10]. There are three types of features that can be extracted to give information about the faulty signal. These are the time-domain features, frequency domain features, or time–frequency-domain features. The simplest of the three are the Time-domain features which are very easy to extract, compared to features extracted from the frequency-domain and time–frequency-domain. Time domain features contains enough information for us to make classification for the faults that we have considered in this research. These features are the Mean, Energy and the Peak to Peak value of the signal. Table I shows the features used and how to calculate these features.

TABLE I
FEATURES USED IN THE PROPOSED METHOD

Feature	Value
Mean	$\frac{1}{N}\sum_{i=1}^{N} S(i)$
Energy	$\frac{1}{N}\sum_{i=1}^{N} S(i)^2$
Peak to Peak	$S_{max} - Smin$

C. Machine learning for Classification and Prognostics

In supervised machine learning the program learns itself from the data points. The labels to which class they belong is also provided to the program in the training phase. The algorithm learns from this data and makes predictions on brand new unseen data. Classification is one such that can be tackled with supervised machine learning. In our work, we have a multi-class classification model that classifies the incoming data into one of the three classes, namely, bias, drift and normal.

In this work, we test 04 machine learning classification and 1 prognostics algorithms which are mentioned below.

1) Support Vector Machine (SVM): Support Vector Machine is a very robust supervised machine learning algorithm that is used to solve both classification and regression problems. Although, commonly it is used for classification purposes only. In this algorithm, each data point is assumed to be a point in a space with n dimensions, where n represents the total features that are extracted from the data assuming each feature is a particular co-ordinate. Then a hyperplane is calculated which optimally draws a line between any two classes. The optimized hyperplane is mathematically represented as below, [4]

$$w^t x + b = 0 \tag{3}$$

where w represents the weight vector, x represents the input vector and b is the bias vector. We have used multi-class SVm with Radial Basis Function (RBF0 as the kernel function.

2) Naïve Bayes Classifier (NB):: The Naïve Bayes classification algorithm uses a probabilistic approach, as the name suggests, based on Bayes's theorem, given in equation 4. It works on the assumption that every pair of features is independent. It performs well in multiclass classification in many real world scenarios, inspite of the simple assumption.

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{4}$$

There are several NB classifiers which are specifically used for different tasks, i.e., Bernoulli Naive Bayes, Multinomial Naive Bayes and Gaussian Naive Bayes. Earlier these classifiers were majorly used in sentiment analysis and recommendation systems. We have used the Naive Bayes classifier to compare its performance with the rest of the well-known classifiers in the problem of fault detection.

3) K-Nearest Neighbors (KNN):: The K-nearest neighbour algorithm is a non-parametric supervised learning algorithm. It is one of the best-known and high-performing algorithm majorly used for classification problems. It works on the principle of the nearest neighbour's algorithm. It predicts the class of the new data based on the similarities with the existing data points. This similarity is the Euclidean Distance between the new and the existing data points. This algorithm assumes similar kind of data points lies close to each other. In this algorithm, a point is classified based on its K nearest neighbours. In our work, we have set the number of neighbours to 5 with the default leaf as 30 and uniform weight.

4) Decision Tree (DT): : The decision tree is a very versatile and robust machine learning algorithm. This algorithm is also used to solve both classification and regression problems. Each node of a decision tree represents a feature, and each branch represents the decision that could be made. The last leaf node represents the output of the decision tree. The dataset is broken into smaller subsets, while a decision tree is simultaneously generated. It is seen that the performance of this algorithm is good even on multioutput problems. Like SVMs, DTs can also fit highly complex datasets.

5) Autoregression: Autoregression is an algorithm that is used to forecast future values using a linear combination of past values of the variable in consideration. An auto-regression model can be mathematically [11] represented as,

$$y_t = b + A_1 y_{t-1} + A_2 y_{t-2} + A_3 y_{t-3} \cdots + A_m y_{t-m} + \epsilon_t \quad (5)$$

where ϵ_t is white noise. We can think of this as a usual multiple regression algorithm but it uses lagged values of y_t as it's predictors.

III. Hardware Implementation

The 4 different classification algorithms are trained on a system with 16GB RAM and RTX 3070 GPU. After the training phase, the best performing model, i.e., K-Nearest Neighbour model, is then deployed on a Raspberry PI. The python Pickle module is used to serialize the trained model by converting the python object into a character stream. After the trained model is loaded on the Raspberry Pi, data is collected using the ADA8232 ECG sensor from 7 different subjects with their consent. Faults of both kinds are injected in these data files. These files are then used to test the accuracy of the model loaded on the Raspberry Pi.

The inferences are made in just 0.0291 seconds and the Fig. 3 shows Receiver Operating Characteristic (ROC) with the Area Under Curve (AUC) mentioned. Since we have a multi class classification, we have used the one-vs-rest technique to calculate the ROC curve. We see that an average AUC of 0.92 is achieved which is close to 1. Hence our classifier performances well in real-time environment too. We achieve a testing accuracy of 85.71% with only 3 misclassifications. Fig. 4 shows the hardware implementation of our solution in RaspberryPi

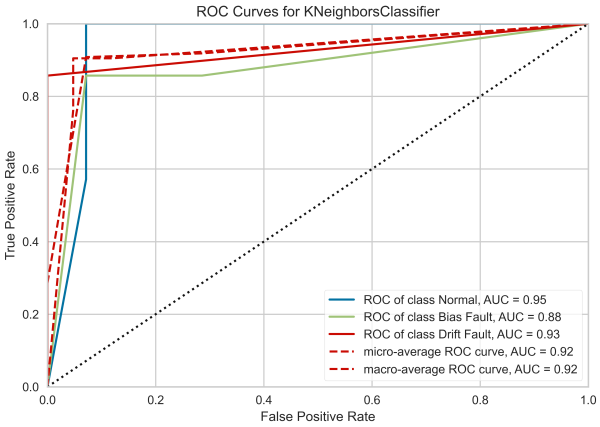

Fig. 3. ROC curve of the KNN model.

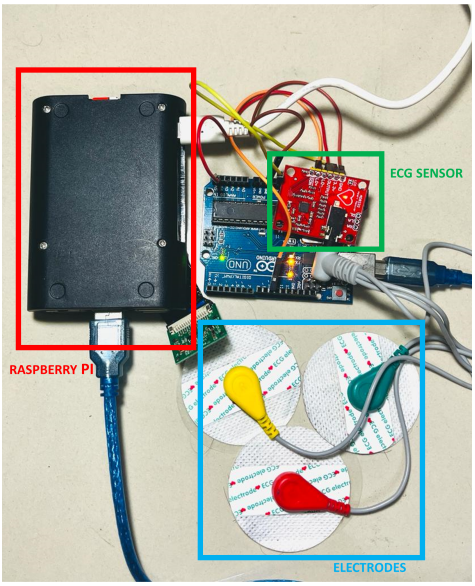

Fig. 4. Hardware implementation using Raspberry Pi

IV. Results

A. Classification

1) Training: In a real-life situation, generally, the probability of a sensor producing faulty outputs is much less than the sensor working absolutely fine. This creates the issue of class imbalance which means that the accuracy might not be the best metric for the evaluation of any algorithm. The Confusion matrix, on the other hand, provides us with deep information from where much more detailed metrics can be calculated. The Confusion Matrices for the four different algorithms are calculated and are presented in Fig. 5

From the Table II, we can observe that KNN, SVM and Decision Tree give high and nearly similar values for Precision and Recall. Hence, all the three models are suitable for deployment in the sensor nodes for fault detection. But since the accuracy of the KNN is highest, so it is considered for Fault Identification Module in this paper. The KNN outperformed other classifiers because the dataset is small, low-dimensional and has no outliers. The Naïve Bayes gave the lowest values for accuracy and recall. Fig. 5 shows the confusion matrix for the 4 different algorithm in the testing phase.

2) Testing: For the testing of our algorithm, we have deployed our machine learning model on a raspberry pi single-board computer (SBC). Raspberry pi is a low-powered, low-cost and modular computer. We have taken data from 7 different people using the ECG AD8232 sensor. The AD8232 extracts and amplifies the electrical signals from the heart. After recording the data from 7 different people, both drift and bias fault are added to each of the seven different ECG signals. Now we have a total of 21 data signals. Since the KNN algorithm offers the best accuracy in the training phase,

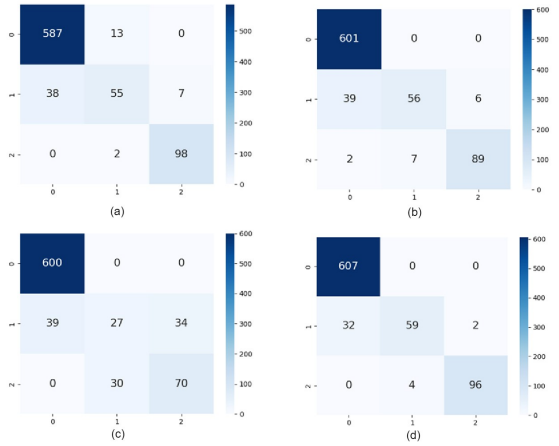

Fig. 5. Confusion Matrix for (a) Decision Tree Classifier, (b) Support Vector Machine (c) Naïve Bayes Classifier (d) K-Nearest Neighbours with k=5. (0, and 2 represents Normal, Bias and Drift class respectively)

TABLE II
TRAINING SIMULATION RESULTS COMPARISON FOR DIFFERENT ML ALGORITHMS

ML Used	**Fault Condition**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
SVM	Drift	0.93	0.88	0.90	92.75%
	bias	0.86	0.54	0.66	
	Normal	0.93	1.00	0.97	
Decision Tree	Drift	0.93	0.98	0.96	92.5%
	Bias	0.79	0.55	0.65	
	Normal	0.94	0.98	0.96	
Naïve Bayes	Drift	0.67	0.70	0.69	87.125%
	Bias	0.47	0.27	0.34	
	Normal	0.94	1.00	0.97	
KNN	Drift	0.97	0.95	0.96	95.25%
	Bias	0.92	0.58	0.71	
	Normal	0.94	1.00	0.97	

we choose KNN as our algorithm, and the predictions are made using the KNN model. We use the pickle library in python to serialize our machine learning model so that it can be deployed on the raspberry pi. Even with a low number of test subjects, the classification accuracy of 90.4% is achieved with only 2 wrong classifications.

B. Prognostics

For the fault prognostics, we have used the Autoregression algorithm. Since the faults that we have considered are linear faults, choosing the autoregression algorithm gives us very good results. We have trained and tested the algorithm assuming different percentages of the signal values known to the model. In case of bias error, we achieve an Root Mean Square Error (RMSE) of just 0.2608% when only 10% of the signal is known for training. While in the case of Drift error, we achieve a RMSE of just 6.359×10^{-4}. The RMSE value of drift error is so low because of the fact that the error is linear in nature and can be easily captured by Autoregression algorithm. More results are shown in Table

TABLE III
ROOT MEAN SQUARED ERROR OF THE PREDICTED FAULTS USING THE AUTOREGRESSION ALGORITHM

Percentage used for training	Bias RMSE	Drift RMSE
20%	0.2766	6.359×10^{-4}
40%	0.0201	6.359×10^{-4}
60%	0.0080	6.359×10^{-4}

Fig 6 shows the real and the predicted bias fault part of the signal using the Autoregression algorithm.

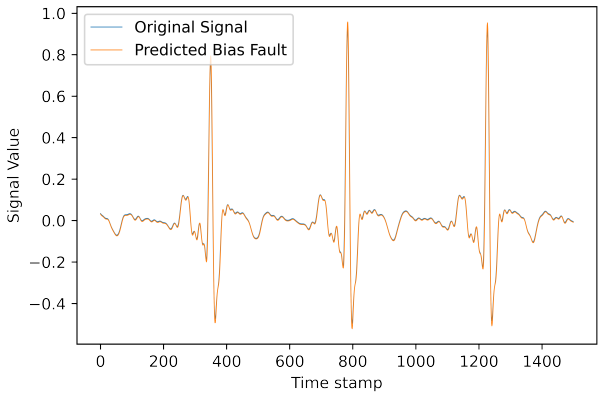

Fig. 6. Predicted vs original value of the bias fault signal

V. CONCLUSION

In this paper, the authors identify two types of faults in sensor fault detection problems, namely, drift and bias faults in the Electrocardiogram (ECG) sensor. For use in smart systems, a low computational power system (based on raspberry-pi) was proposed so that real-time fault detection could be made possible. Popular classification machine learning algorithms were used to classify data into one of the three classes (two faults and one normal class). From the experimental results, we can see that SVM and KNN performed tremendously well. A fault prognostic methodology was also proposed using the autoregression algorithm in which fault could be predicted beforehand. For future work, a more complex system can be developed, which could be deployed on a powerful motherboard. This system can have multiple sensors instead of just having one. We can also have different types of sensors, such as accelerometers or pressure sensors rather than having only ECG sensors. One more way to explore is to include more faults and make this model a robust system.

REFERENCES

[1] D. Li, Y. Wang, J. Wang, C. Wang, and Y. Duan, "Recent Advances in Sensor Fault Diagnosis: A Review," *Sensors and Actuators A: Physical*, vol. 309, p. 111990, 2020.

[2] X. Dai, F. Qin, Z. Gao, K. Pan, and K. Busawon, "Model-based On-line Sensor Fault Detection in Wireless Sensor Actuator Networks," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 2015, pp. 556–561.

[3] B. de Bruijn, T. A. Nguyen, D. Bucur, and K. Tei, "Benchmark Datasets for Fault Detection and Classification in Sensor Data," in *SENSORNETS*, 2016, pp. 185–195.

[4] U. Saeed, S. U. Jan, Y.-D. Lee, and I. Koo, "Machine Learning-based Real-Time Sensor Drift Fault Detection using Raspberry Pi," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2020, pp. 1–7.

[5] J. S. Sana Ullah Jan, Young-Doo Lee, "Sensor Fault Classification Based on Support Vector Machine and Statistical Time-Domain Features," in *IEEE Access*. IEEE, 2017, pp. 1–7.

[6] T. M. Salah Zidi and B. Alaya, "Fault Detection in Wireless Sensor Networks Through SVM Classifier," in *IEEE Sensors Journal*. IEEE, Jan 2018, pp. 340 – 347.

[7] A. R. P. Schmidt and K. V. L. R. Duerichen, ""Introducing WeSAD, a multimodal dataset for wearable stress and affect detection,," in *n ICMI 2018 - Proceedings of the 2018 International Conference on Multimodal Interaction*. ICML, 2018, pp. 400–408.

[8] F. D. A. B. T. W. Rauber and F. M. Varejáo, "Heterogeneous feature models and feature selection applied to bearing fault diagnosis,," *IEEE Trans. Ind. Electron*, vol. 62, no. 1, pp. 637–646, 2015.

[9] W. J. S. Ding, H. Zhu and C. Su, "'A survey on feature extraction for pattern recognition," *' Artif. Intell. Rev*, vol. 37, no. 3, pp. 169–180, 2012.

[10] N. J. Pithadia and V. D. Nimavat, "A review on feature extraction techniques," *n Proc. WiSPNET*, vol. 1, no. 3, p. 1263–1268.

[11] Y. Zhao and L. Shen, "Application of time series auto regressive model in price forecast ," *International Conference on Business Management and Electronic Information (BMEI)*, vol. 5.