Fire Detection With Erroneous Input Correction

ECGR 4105/5105 - Introduction to Machine Learning

1st Jaskin Kabir
ECGR Student
UNC Charlotte
Charlotte, NC
jkabir@charlotte.edu

2nd Axel Leon Vasquez
ECGR Student
UNC Charlotte
Charlotte, NC
aleonvas@charlotte.edu

3rd Matthew Anderson
ECGR Student
UNC Charlotte
Charlotte, NC
mande137@charlotte.edu

***Abstract*—The accuracy of typical fire alarm systems is often limited due to their reliance on single sensors with constrained detection capabilities. Moreover, a sensor failure renders the entire system ineffective, posing a critical risk. While AI-based fire detectors have massively outperformed traditional systems, little work has addressed handling sensor failures. To address this challenge, we propose AlarmNet, a robust fire detection system that incorporates erroneous sensor data during training. We investigate two implementations of AlarmNet: a centralized model trained on a single server and a simulation of a federated model where training is distributed across multiple clients and aggregated into a global model. In our experiments, sensor errors were introduced by randomly replacing 24.2% of the dataset's measurements with missing or faulty values. We applied a pre-processing step that replaced these values with the median of the corresponding training features. Our 3-layer artificial neural network The model achieved an accuracy of 98.38% and successfully detected 99.73% of fires, despite the high rate of sensor errors. This resulted in only a 0.25% reduction in detection performance compared to a model trained on complete data, and a significant 41.51% improvement over a simulated commercial-grade fire alarm system." The federated approach delivered comparable results while utilizing a less computationally intensive model. In future work, we aim to extend this approach to develop a real-world fire detection system capable of maintaining robustness against sensor failures.**

***Index Terms*—machine learning (ML), algorithm, model, Artificial Neural Network, Imputation, Sensor fusion, Fire Alarms, Fire Detection, Error Handling, Federated Learning**

I. Introduction and Motivation

Fire alarm systems play a critical role in safeguarding lives and property by providing early warnings of fire hazards. Traditional systems, however, do not perform as well as their critical role would suggest. For example, a 2013 study found the Honeywell FS90, a widely used commercial fire detection system, only achieves an accuracy of 87.5% [1]. This contributes to deaths, injuries, and damage due to fires in two ways. (1) When a fire occurs and the fire detection system fails to recognize it, the occupants of the building can only react to the fire once they are close enough to observe it, often once it is too late to evacuate. (2) A 1995 report stated that fewer than 25% of people interpreted the sound of the fire alarm as a potential indication of a real emergency [2]. Due to the high frequency of false or 'nuisance' alarms, occupants may not evacuate or contact emergency services, even if the alarm is triggered by a real fire.

As these systems typically rely on just one or two sensors and simple logic to detect fires, they are especially susceptible to faulty sensors rendering them completely ineffective. While artificial intelligence (AI)-based fire detectors have demonstrated significant improvements in accuracy over traditional systems, they generally rely on error-free sensor data [3] [4] [5]. This limits their practical applicability, especially in real-world scenarios wherein sensor errors and failures are common.

To address these challenges, we propose AlarmNet, a robust, AI-based fire detection system specifically engineered to remain accurate in the presence of sensor errors. AlarmNet is trained on a dataset from which randomly selected measurements were removed to simulate missing or faulty sensor readings. This approach enables the model to learn to detect fires accurately based on unreliable sensor data.

The system is evaluated in two configurations: a centralized model trained on a single server and a simulation of a federated model, where training is distributed across multiple clients and aggregated into a global model. This dual approach allows us to investigate two key questions: (1) whether it is possible to train a model that can handle sensor errors, and (2) whether a federated model can achieve comparable performance while utilizing fewer computational resources. If the federated model meets this goal, it would demonstrate AlarmNet's viability as a real-world fire detection system. This paper first details the analysis and preparation of the chosen dataset, followed by a disussion of the design, implementation, and evaluation of both versions of AlarmNet.

II. Approach

III. Data Preparation

A. Dataset

The dataset used in this project is the Smoke Detection Dataset provided by Stefan Blattmann [6]. The dataset consists of approximately 62,629 sensor readings encompassing both normal and fire-related scenarios, providing diverse training and evaluating fire detection models.

To generate this data, Blattman constructed an Arduiono-based system to collect data from an array of six sensors at a 1Hz sampling rate. These sensors were:

- Bosch BMP390: Pressure Sensor
- Bosch BME688: Humidity, Temperature, and Gas Sensor (Volatile Organic Compounds/VOC)
- Bosch BMP388: Pressure Sensor
- Sensirion SPS30: Particular Matter Sensor
- Sensirion SHT31: Humidity and Temperature Sensor
- Sensirion SPG30: Gas sensor (VOC)

Note that several sensors measure the same environmental factors, such as humidity and pressure. This redundancy is intentional and serves to enhance the robustness of the model. Before uploading this dataset to Kaggle, Blattman utilized sensor fusion to combine the data from these six sensors into a single dataset. The only sensor with no redundancy is the Sensirion SPS30, which measures particulate matter and is the most expensive sensor.

Blattman placed his sensor array into various fire scenarios and recorded the measurements. He provided the following short list of scenarios during which he collected data:

- Normal indoor
- Normal outdoor
- Indoor wood fire, firefighter training area
- Indoor gas fire, firefighter training area
- Outdoor wood, coal, and gas grill
- Outdoor high humidity
- etc.

[7]

B. Train/Test Split

Before the dataset was passed to the models, it was split using an 80/20 train/test split. This means that 80% of the data were randomly sampled and used to train the model, while the remaining 20% were used to evaluate the model's performance. This split ensures that the model is not overfitting to the training data, as it is evaluated on data that it has not seen before.

C. Feature Scaling

After the train/test split, the features were scaled using standard scaling. This method subtracts the mean of the feature and divides by the standard deviation, ensuring that all features have a mean of 0 and a standard deviation of 1. This scaling is important because it ensures that the model does not learn to prioritize features with larger values, which could lead to poor generalization. The mean and standard deviation are calculated based on the training data and then applied to the test data to ensure that the model is evaluated on the same scale that it was trained on.

D. Feature Elimination

The dataset provided by Blattman originally included 15 features, which are listed in Table I. Of these 15, the two sample count features and the timestamp were removed as they do not provide any useful information for the model. Of the remaining 12 features, a crucial task was to determine which features were the most relevant to the target variable. Eliminating irrelevant variables reduces the complexity and computational cost of the model, as its input space shrinks in dimensionality. This reduction can also help prevent overfitting, as the model is less likely to learn noise in the data.

Feature	**Description**
Timestamp	UTC Timestamp of the sample
Number	Unique identifier for each sample
Unnamed: 0	Unintended duplicate of Number
Temperature	Air temperature in degrees Celsius
Humidity	Air humidity in percentage
Pressure	Air pressure in hectoPascals
TVOC	Total Volatile Organic Compounds in parts per billion
eC02	Carbon Dioxide in parts per million
Raw H2	Raw, uncompensated molecular hydrogen
Raw Ethanol	Raw, uncompensated ethanol
PM1.0	Percentage of particles in the air less than $1.0\mu m$ in diameter
PM2.5	Percentage of particles in the air less than $2.5\mu m$ in diameter
NC0.5	Number concentration of particles in the air greater than $0.5\mu m$ in diameter
NC1.0	Number concentration of particles in the air greater than $1.0\mu m$ in diameter
NC2.5	Number concentration of particles in the air greater than $2.5\mu m$ in diameter

TABLE I
SMOKE DETECTION DATASET FEATURES

To achieve this, we calculated the absolute correlation of each feature with the target variable, 'Fire Alarm'. The features were then ranked based on this correlation, and the top four features were selected for the model. This number was chosen by incrementally removing the least correlated features from the dataset and training a model on the these data until a significant drop in accuracy was measured. These top four features were:

1) Humidity
2) Raw Ethanol
3) Pressure
4) TVOC

While a sufficiently accurate model could be trained on these data, this list of features does not include any of the particulate matter measurements. If this system were to ignore one of its sensors, it would be significantly less resilient against sensor errors. To address this, we added back the NC0.5 feature to the data, as it is the most correlated with the target variable among the particulate matter measurements.

E. Error Insertion

To simulate sensor errors, we developed a method to randomly remove certain measurements from the dataset accord-

ing to some assumed rate of error. This rate of error represents the chance that a given sensor reading will provide incorrect or missing data during any given sample. We chose to assume this error rate to be 40%. This is an unrealistically high number, but it is low enough to ensure that the model is not completely useless while still providing a significant challenge.

Most of the features–with the exception of the particulate matter measurements–were collected using two sensors. To include this redundancy in the simulation, we assumed that both sensors that generate any feature would have to fail simultaneously in order for that feature to be missing for a given a sample. In other words, while 40% of the particulate matter measurements were removed, only 20%[1] of the other measurements were removed. After applying this error simulation to the dataset, approximately 24.2% of the data was removed.

F. Imputation

Before this data could be used to train a model, the missing values had to be replaced. We chose to use median imputation, which replaces missing values with the median of the corresponding feature calculated from the training dataset. This method was chosen because it is robust to outliers and skewed distributions, which are common in sensor data.

IV. Model Training and Evaluation

A. Evaluation Methods

1) Multiple Models

To ensure that the techniques used in this project are effective, we trained and evaluated multiple models. We trained a model based on the full 12-features of the dataset, a model based on the top 4 features, a model that included the NC0.5 feature for redundancy. We then compared the performance of these three models to determine if the feature elimination was necessary, and how adding back the NC0.5 feature affected this performance.

We also trained a fully connected model on the imputed error data, and a simulation of a federated learning system. The performance of the federated model was then compared to the fully connected model.

2) Metrics

The performance of each model was evaluated using the following metrics:

- Recall: Measures the proportion of actual fire cases detected by the model to minimize false negatives. This is the most important metric, as it is crucial that the model detects as many fires as possible.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

- Precision: The proportion of correctly identified fire alarms

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

- F1-Score and Accuracy: Implements the harmonic mean of precision and recall with overall correctness of the model.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- Confusion Matrix: A visualization tool that lists the true positive/negative and false positive/negatives

3) Honeywell FS90 Simulation

The 2013 study mentioned in the introduction I only listed the Honeywell FS90's accuracy with no mention of precision, recall, or the raw data necessary to calculate these values. To provide a more accurate comparison, we developed a simple simulation of the FS90:

First, we created a copy of the target column from the training data. Then we randomly selected 12.5% of the datapoints in the data to flip, creating the FS90's 87.5% accurate 'prediction'. To simulate sensor error, we assumed that the FS90 had a 20% chance of experiencing a sensor error during any given sample. This error rate was chosen to match the rate of error in the AlarmNet model. We then assumed that during a sensor error, the FS90 would indicate that it did not detect a fire. To apply this to the simulation, we randomly selected 20% of the prediction data. If the FS90 predicted a detected a fire during any of these samples, we flipped the prediction to indicate that it did not detect a fire. This simulation allowed us to calculate the recall and precision of the FS90 with and without sensor error, which we then compared to the AlarmNet models.

B. Model Architectures

For the model trained on the 12-feature model, the fully connected neural network included an input layer of 12 neurons, two hidden layers of 64 and 32 neurons respectively, and a layer consisting of one output neuron. The 4 and 5-feature models used input layers of 4 and 5 neurons respectively, two hidden layers both consisting of 64 neurons, and one output neuron. Between each layer, a ReLU activation function was used. This introduced nonlinearity into the model, allowing it to learn complex patterns in the data and ensuring its loss function was differentiable. After the output layer, a sigmoid activation function was applied to ensure that the output was between 0 and 1.

To allow the fully connected model trained on the data with errors inserted into it, the architecture was increased to a model with three hidden layers each consisting of 256 neurons. This allowed the model to learn more complex patterns in the data and ensured that it could handle the noisy and incomplete data.

The model used in the federated learning simulation used three hidden layers consisting of 32, 16, and 8 neurons. This model includes fewer neurons than any other model to ensure that it is less computationally intensive. This model was trained on three clients, each with a copy of the global model. The clients trained their local models on their own data, and then sent the weights of their models to the global model. The global model then averaged these weights to create a new global model.

[1]The joint probability formula $P(A \cap B = P(A) \times P(B))$ implies that this probability should actually be 16%. We chose to use 20% instead for the added challenge.

C. Model Architecture and Training

In the real world sensor data, is often due to hardware communication limitations and errors. It can lead to incomplete or noisy environmental factors. The reliability of the model can go in such conditions in error handling using robust strategies to incorporate into the model and training process. These strategies allowed the model to maintain high accuracy and recall during the sensor error process. In simulating real-world scenarios, errors were intentionally introduced into the dataset. In the model run, 20 percent was applied across all features, with particulate matter (PM) sensors likely to encounter errors due to insufficient redundancy. The dataset showed four sensors in their categories, temperature and Humidity, Pressure, (TVOC, CO2, Ethanol, H2), and Particulate Matter (NC0.5, NC1.0, NC2.5, PM1.0, PM2.5). erroneous values were represented as missing data (NaN) or extreme outliers to emulate hardware failures or any communication drops. Missing values were handled using median imputation, which replaces erroneous values with the median of the corresponding feature calculated from the training dataset. The median was chosen against outliers suitable for skew distributions in sensor data. Imputation implementation ensures the dataset remains intact, allowing it to learn effectively. The feature was further improved by identifying the most correlated features with fire detection. Key features such as TVOC, Pressure, Raw Ethanol, and Humidity were retained. The training process specialized adjustments for handling noisy and incomplete data. During the deep ANN error handling architecture of [(256,256,256)] for better resolution. Extending training epochs were used, enabling the model to generalize effectively. In the error handling stage, errors were introduced to replicate real-world conditions challenging the robustness of the architecture. The implementation strategy introduced erroneous or missing values were replaced with the median of each feature to maintain data integrity. The imputation approach was particularly effective for skewed feature distributions which enchanted the model ([256,256,256]) and demonstrated resilience to noisy and incomplete data. The imputed error model showcased its ability to handle a 20 percent sensor error rate with minimal performance degradation and showcasing minimized false negatives under challenging conditions. Performance metrics were involved in all models to evaluate using recall, precision, F1-Score, and confusion matrices. The model was benchmarked against the Honeywell FS90 fire detection system. Comparison revealed superior recall and resilience in the proposed model, under simulated noisy conditions, such as histograms of missing values and bar charts comparing recall and recall without errors showing the model's robustness. Furthermore, the size and fast inference of the final model exported in ONNX format ensure the deployment in real-time IoT scenarios of fire detection systems. These error-handling strategies demonstrate the model's ability to adapt to real-world challenges for a reliable solution in fire detection in different environments. Simulating error handling, imputation, and targeted features in correlation highlights the model's resilience and practical applications.

V. Results and Analysis

In a comparative analysis, SVM(Support Vector Machine) and NN (Neural Network) were both considered for fire detection within the 12-feature test bench. Neural networks are ultimately chosen for their superior flexibility and robustness. SVMs are known for their simplicity and effectiveness within smaller datasets due to their ability to model linear and non-linear relationships using kernel functions. However, in this dataset, the project had a large size of 62,000 samples which limited the SVM training model as memory requirement scale quadratically with the number of samples. On the other hand neural network demonstrates superior performance and adaptability. They efficiently scaled with large datasets by leveraging batch size gradient descent and GPU acceleration with mixed precision and gradient scaling. It can learn hierarchical features representing them as particularity effective for multi-sensor data its architecture model is tailored to handle noisy and missing data, as seen in the imputed error model under results., It can maintain high recall and precision under a 20 percent error rate. The full feature model utilizes all 12 -features that achieved high precision and recall but its computational complexity made it less efficient. After using a correlation technique ranking them by target variable it reduced to 4 and 5 feature sets retaining only the top four features (Humidity, Raw Ethanol, Pressure, and TVOC), achieved performance to the full feature model with minimal precision loss of 0.1 percent. It significantly improved making it more suitable for different environments. The redundant feature model added the NC0.5 enchanting redundancy and improved robustness maintaining efficiency. The imputed error model designed to handle noisy and missing data shows a resilience with 20 percent error rate. Thus outperforming the Honeywell FS90 system. In addition to accuracy and recall, time was evaluated to determine real-time applicability. The proposed models achieve an average time of 2e-2 seconds per sample making them viable for deployment in fire detection systems. Evaluation metrics showed visual comparison including confusion matrices, and bar charts illustrating the superior performance. The reduced features and imputed error and aggregation demonstrated the ability to balance accuracy, efficiency, and robustness for real-world fire detection applications. The Federated model trains the local copy of a global model using local data. The training sends model weights (weights of the neurons in the neural network) to the global mode. The federate and AlarmNet achieve near identical accuracy. These bar graphs show the comparison of recall with and without error and precision with 20 percent error and without. With Machine Learning algorithms, it was able to provide excellent validation against Honeywell FS90 highlighting the potential ANN that can provide for enchanting better real-world safety protocols (fig 18 and 19).

VI. Lessons Learned

This machine learning model successfully demonstrated the development an efficient fire detection system using sensor data. By leveraging a custom neural network architecture, the system achieved high precision and recall even with real-world scenarios such as noisy or missing sensor data. Feature selections and target preparation, including median imputation and error handling, ensure the model's exploration. The reduced feature model maintained comparable performance to the 12-feature model significantly reducing computational requirements and making it good for constrained environments. Comparative analysis highlights the proposed neural network over the traditional such as SVM. The neural network can handle large datasets, capture complex features tolerate sensor errors underscores the effects in IoT-base fire detection systems. Optimizing time and compact model size ensure viability for read-time devices. The project object results emphasize the importance of designing models that are balanced, efficient, accurate, and robust for safety and critical protocol applications. Future work can be explored further optimizing the learning, advancing imputation, and improving the federated model weights between three clients to enhance system performance. The project demonstrates the potential for machine learning to revolutionize fire detection technology to challenge high-end companies such as Honeywell to ensure the power of ANN to contribute to improving safety and reliability in diverse environments.

VII. Contributions

- **Matthew Anderson:**
  - Federated Learning Class
  - Edited Report and Presentation
- **Jaskin Kabir:**
  - Project Manager
  - Dataset Selection, Preparation, and Analysis
  - AlarmNet Class that contained the fully connected model
  - Error insertion and imputation
  - Edited Report and Presentation
- **Axel Leon Vasquez:**
  - Wrote Report
  - Created Presentation
  - Data Visualization

A. References

References

[1] N. k. Fong, "Investigation of the performance and improvement of optical smoke detectors," *Procedia Engineering*, vol. 62, 12 2013.

[2] G. Proulx, J. C. Latour, J. W. MacLaurin, J. Pineau, L. E. Hoffman, and C. Laroche, "Housing evacuation of mixed abilities occupants in highrise buildings," National Research Council of Canada. Institute for Research in Construction, Tech. Rep., Aug 1995. [Online]. Available: https://doi.org/10.4224/20375582

[3] H. Prasad, A. Singh, J. Thakur, C. Choudhary, and N. Vyas, "Artificial intelligence based fire and smoke detection and security control system," in *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*, 2023, pp. 1–6.

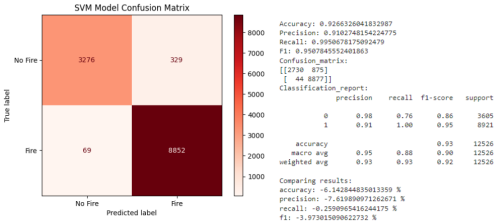

Fig. 1. SVM Metric

```
Accuracy: 0.9999201660546064
Precision: 1.0
Recall: 0.999887904943392
F1: 0.9999439493301945
Confusion_matrix:
[[3605    0]
 [   1 8920]]
Classification_report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3605
         1.0       1.00      1.00      1.00      8921

    accuracy                           1.00     12526
   macro avg       1.00      1.00      1.00     12526
weighted avg       1.00      1.00      1.00     12526
```

Fig. 2. 12 - Feature Metrics

[4] D. K. Dewangan and G. P. Gupta, "Explainable ai and yolov8-based framework for indoor fire and smoke detection," in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, 2024, pp. 1–6.

[5] H. Zhong, Z. Wang, Z. Chen, W. Chen, and Y. Li, "A novel fire monitoring system for electric bicycle shed based on yolov8," in *2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2023, pp. 142–147.

[6] S. Blattman, "Smoke detection dataset version 1," Aug 2022. [Online]. Available: https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset

[7] S. Blattmann, "Real-time smoke detection with ai-based sensor fusion," Aug 2022. [Online]. Available: https://www.hackster.io/stefanblattmann/real-time-smoke-detection-with-ai-based-sensor-fusion-1086e6#toc-background-1

B. Results

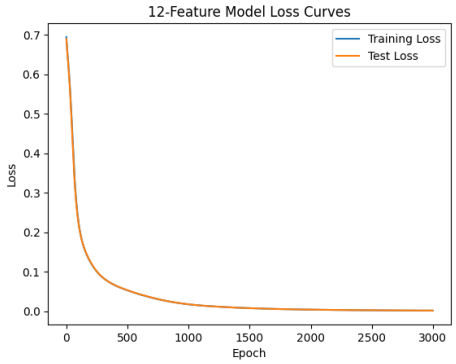

Fig. 3. 12 - Feature Loss Curve

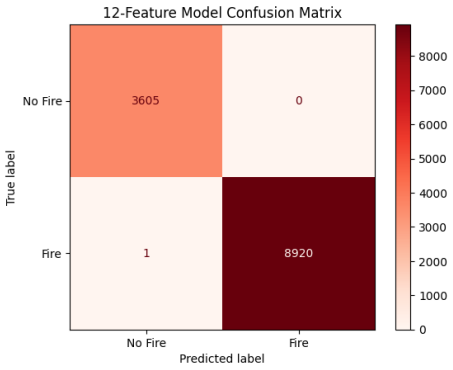

Fig. 4. 12 - Feature Confusion Matrix

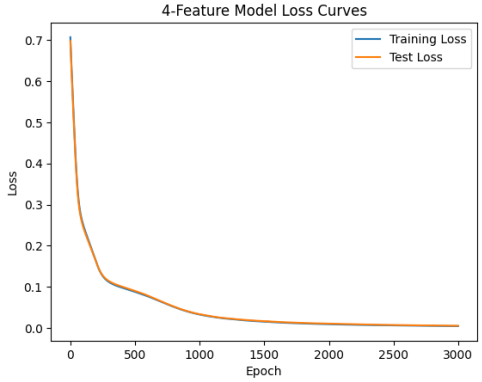

Fig. 7. 4 - Feature Loss Curves

```
Humidity[%]        0.399846
Raw Ethanol        0.340652
Pressure[hPa]      0.249797
TVOC[ppb]          0.214743
Temperature[C]     0.163902
NC0.5              0.128118
PM1.0              0.110552
Raw H2             0.107007
eCO2[ppm]          0.097006
PM2.5              0.084916
NC1.0              0.082828
NC2.5              0.057707
Name: Fire Alarm, dtype: float64
```

Fig. 5. Features

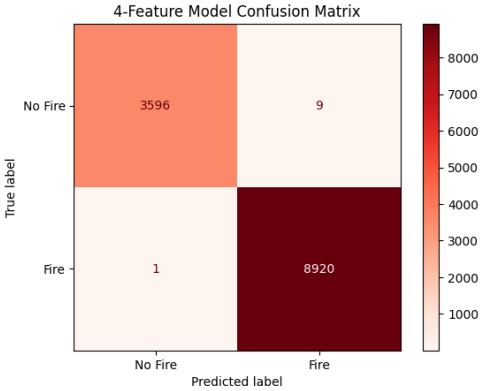

Fig. 8. 4 - Feature Confusion Matrix

```
Accuracy: 0.9992016605460642
Precision: 0.9989920483816777
Recall: 0.999887904943392
F1: 0.9994397759103641
Confusion_matrix:
[[3596    9]
 [   1 8920]]
Classification_report:
              precision    recall  f1-score   support

         0.0       0.98      0.93      0.95      3605
         1.0       0.97      0.99      0.98      8921

    accuracy                           0.97     12526
   macro avg       0.97      0.96      0.97     12526
weighted avg       0.97      0.97      0.97     12526

Comparing results:
accuracy: -0.0719079578139993 %
precision: -0.10089686098654951 %
recall: 0.0 %
f1: -0.05044560282495212 %
```

Fig. 6. 4 - Feature Metrics

```
Accuracy: 0.9998403321092129
Precision: 1.0
Recall: 0.9997758098867839
F1: 0.9998878923766816
Confusion_matrix:
[[3605    0]
 [   2 8919]]
Classification_report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3605
         1.0       1.00      1.00      1.00      8921

    accuracy                           1.00     12526
   macro avg       1.00      1.00      1.00     12526
weighted avg       1.00      1.00      1.00     12526

Comparing results:
accuracy: 0.06387735547748555 %
precision: 0.10079516183223447 %
recall: -0.011212019284675844 %
f1: 0.044816670922210436 %
```

Fig. 9. 5 - Feature Metrics

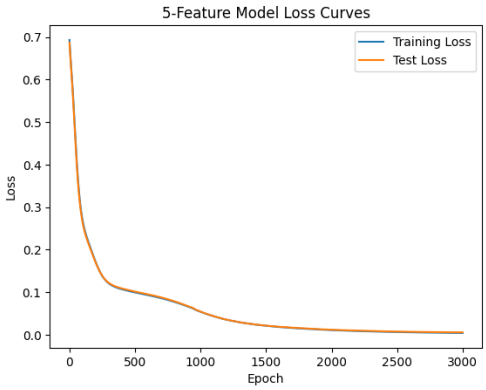

Fig. 10. 5 - Feature Loss Curves

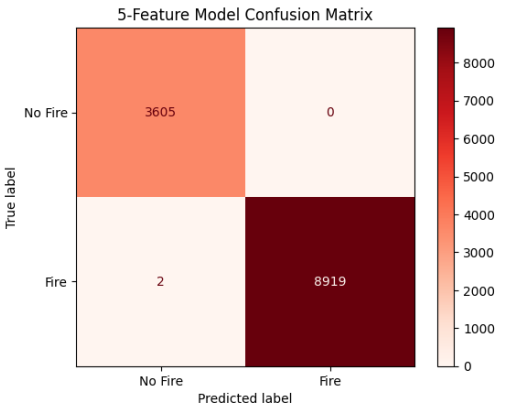

Fig. 11. 5 - Feature Confusion Matrix

```
Accuracy: 0.9835542072489223
Precision: 0.9796367638965328
Recall: 0.9976460038112319
F1: 0.9885593690991892
Confusion_matrix:
[[3420  185]
 [  21 8900]]
Classification_report:
              precision    recall  f1-score   support

         0.0       0.98      0.97      0.98      3605
         1.0       0.99      0.99      0.99      8921

    accuracy                           0.98     12526
   macro avg       0.99      0.97      0.98     12526
weighted avg       0.98      0.98      0.98     12526

Comparing results:
accuracy: -1.6558441558441537 %
precision: -2.078651685393257 %
recall: -0.213483146067411 %
f1: -1.1459628659243208 %
```

Fig. 12. Imputation Metrics

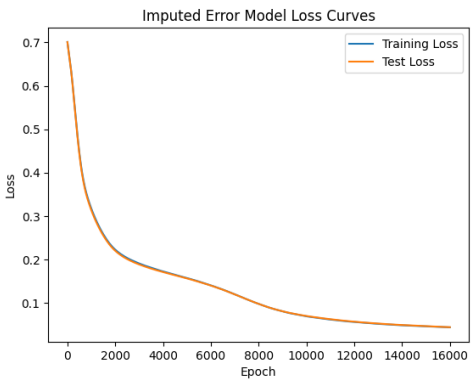

Fig. 13. Imputation Loss Curves

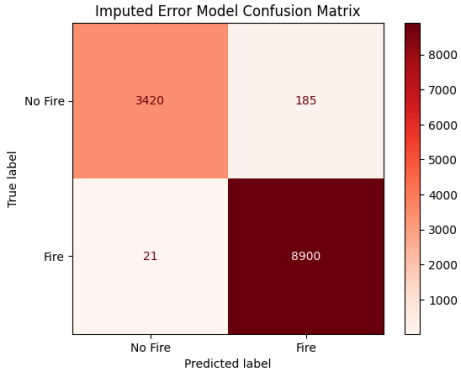

Fig. 14. Imputation Confusion Matrix

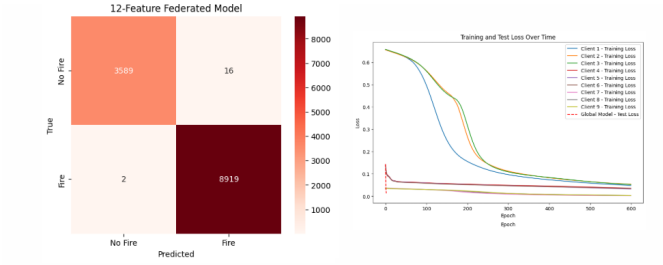

Fig. 15. Federarted Learning: 12-Features

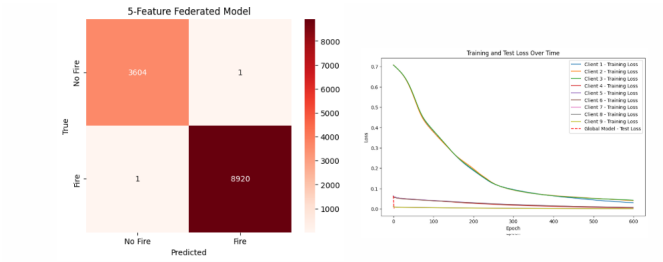

Fig. 16. Federated Learning: 5-Features

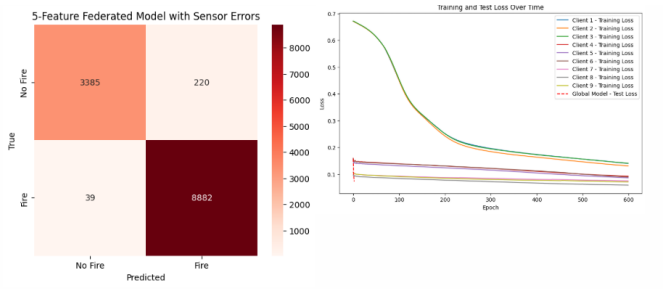

Fig. 17. Federated Learning: Error handling

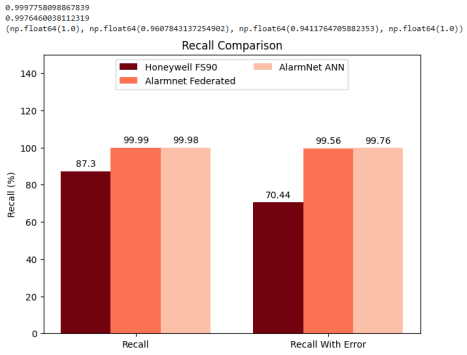

Fig. 18. Recall Comparison Honeywell FS90 Vs AlarmNet ANN Vs Federated

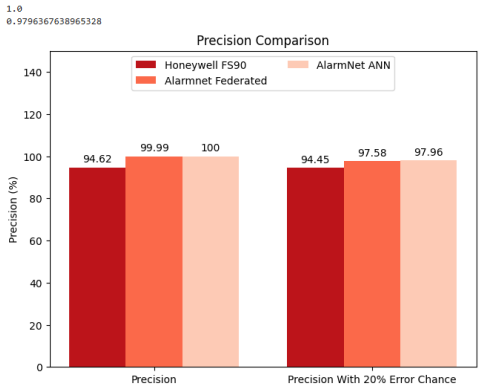

Fig. 19. Precison Comparison Honeywell FS90 Vs AlarmNet ANN Vs Federated