Technical Report: IoT Smoke/Fire Alarm Detection (ML)

Based on repository: hadif1999/iot_smoke_detection_ML (Colab notebook analysis)

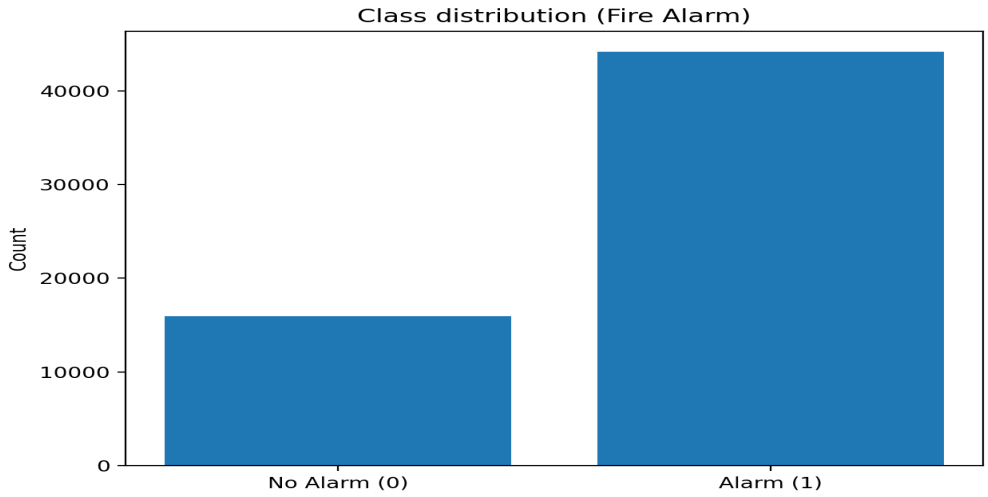
1. Problem definition

Goal: binary classification of **Fire Alarm** from multi-sensor IoT readings (temperature, humidity, VOC/CO2 proxies, ethanol and pressure).

2. Dataset analysis

CSV file: **smoke_detection_iot.csv**. Label: **Fire Alarm** (1=alarm, 0=normal).

Class distribution: Alarm=1 → 44,212, No-Alarm=0 → 15,972 (total 60,184). Majority baseline accuracy ≈ 0.735.



Selected features: Temperature[C], Humidity[%], TVOC[ppb], eCO2[ppm], Raw Ethanol, Pressure[hPa].

3. Pre-processing pipeline

3.1 Column removal. Dropped: Unnamed: 0, PM2.5, CNT, UTC, Raw H2, PM1.0, NC0.5, NC1.0, NC2.5.

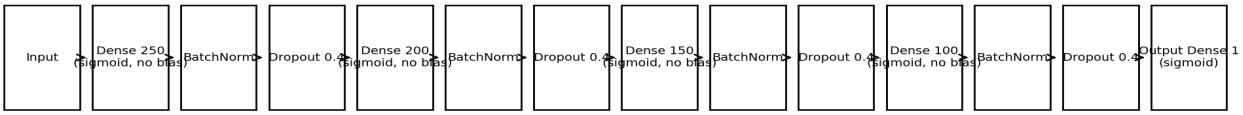
3.2 Outlier handling. IQR bands (Q1-1.5*IQR, Q3+1.5*IQR) are computed. But the filtering condition uses AND between “above upper band” and “below lower band” on the same row, so almost no rows are removed. Using OR or per-feature clipping would be more appropriate.

3.3 Train-test split. test_size=0.15, random_state=10.

3.4 Scaling. StandardScaler fit on train and applied to test.

4. Final model description

Final model: Keras MLP with 4 hidden Dense layers + BatchNorm + Dropout(0.4).



Architecture details (from model_gen):

• Input: 6 standardized numerical features.
• Hidden layers: [250, 200, 150, 100] neurons.
• Activation: sigmoid for hidden layers; sigmoid for output.
• Bias terms: disabled (use_bias=False) in Dense layers.
• Kernel initializer: glorot_normal.
• Regularization: BatchNormalization + Dropout(0.4) after each hidden layer.
• Output: Dense(1, sigmoid) returning probability P(alarm=1).

5. Training setup

• Loss: binary_crossentropy.
• Optimizer: Adam (default settings).
• Metrics: accuracy, binary_accuracy, AUC, Precision.
• Batch size: 24.
• Epochs: 7.
• Validation: 15% split from training set (validation_split=0.15).
• Model checkpointing/early stopping: not used in notebook.

6. Post-processing and inference

Output is probability P(alarm=1); post-processing uses round() (threshold=0.5).

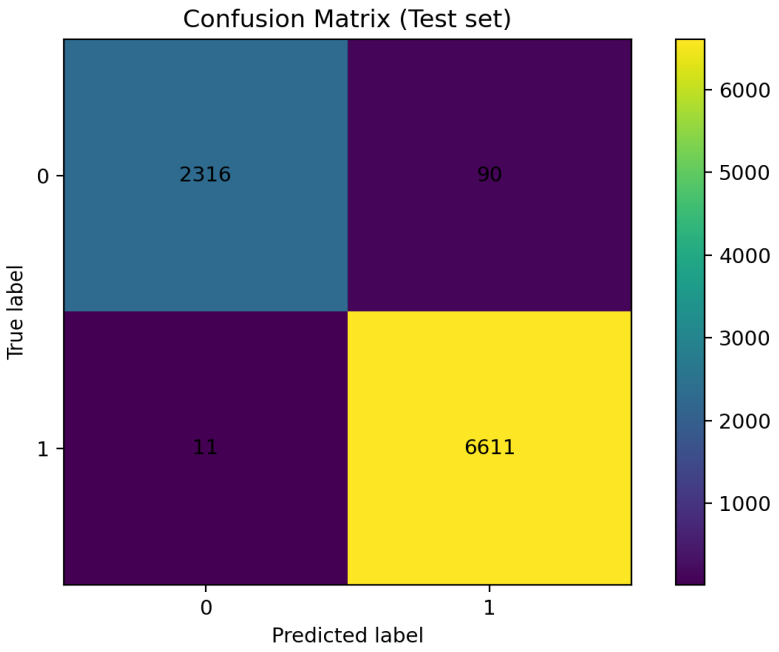
```
Demo (inference)
Input features (after StandardScaler):
  Temperature[C] : 0.12
  Humidity[%]    : -0.48
  TVOC[ppb]      : 1.05
  eCO2[ppm]      : 0.88
  Raw Ethanol    : 0.73
  Pressure[hPa]  : -0.10

Model output:
  P(Fire Alarm=1) = 0.997
  Predicted label  = 1 (Alarm)
```

Model is saved as **smoke_detection_iot.h5**; StandardScaler must be reused at inference.

7. Evaluation results

7.1 Holdout test (15%).

• Test accuracy: 0.9906
• Test AUC: 0.9995
• Confusion matrix (rows=true, cols=pred): [[2316, 90],[11, 6611]]



Label note: Correct mapping is 0=No Alarm, 1=Alarm; notebook target_names are swapped.

7.2 5-fold CV mean.

• Mean accuracy: 0.9808 (± 0.0027)
• Mean AUC: 0.9992 (± 0.0004)
• Mean precision: 0.9769 (± 0.0061)
• Mean loss: 0.0445 (± 0.0031)

8. Model comparison table

Model	Accuracy	AUC	Precision (pos=1)	F1 (pos=1)
Majority baseline (always 1)	0.7346	—	—	—
MLP (holdout test)	0.9906	0.9995	0.9866	0.9924
MLP (5-fold CV mean)	0.9808	0.9992	0.9769	—

9. Error analysis

Errors: 90 false positives (0→1) and 11 false negatives (1→0). Pos-class metrics: Precision=0.9866, Recall=0.9983, F1=0.9924.

Likely error sources: humidity/cooking fumes overlap, class imbalance, removing PM/NC features, and unfiltered outliers.

10. Recommendations

• Fix the outlier filtering logic (use OR for out-of-band checks; consider per-feature clipping).
• Correct class naming in evaluation plots/reports (0=No Alarm, 1=Alarm).
• Add EarlyStopping and ModelCheckpoint to reduce overfitting and improve reproducibility.
• Try simpler baselines (Logistic Regression, Random Forest, XGBoost) and compare calibration/latency for IoT deployment.
• Threshold tuning (e.g., optimize F-beta or minimize false positives) based on application requirements.
• Save and version the StandardScaler together with the model (e.g., with joblib) to avoid train-test leakage.

11. Conclusion

MLP achieves strong performance (~99% test accuracy, ~0.999 AUC) and stable CV (~98.1% mean accuracy). For deployment, tune threshold and package preprocessing + model together.