# Unveiling the Power of Data-Driven Fault Detection in HVAC Systems: A Journey Through Advanced Machine Learning

**Authors:** ***Masoud Kishani Farahani, Alireza Ghadertootoonchi, Mohammad Talaei***

As we increasingly rely on technology to optimize energy efficiency and maintain comfort in modern buildings, heating, ventilation, and air conditioning (HVAC) systems stand at the forefront of this revolution. HVAC systems play a pivotal role in ensuring indoor air quality, comfort, and energy efficiency in buildings. However, undetected faults in these systems can lead to increased energy consumption, suboptimal performance, and expensive maintenance. Fault Detection and Diagnosis (FDD) is an essential aspect of modern building management, helping facility operators detect and rectify anomalies before they escalate into significant issues. In this post, we explore a data-driven approach to FDD in multi-zone Variable Air Volume (VAV) Air Handling Units (AHUs) using a publicly available dataset provided by Lawrence Berkeley National Laboratory.

## The Dataset: A Treasure Trove from Lawrence Berkeley National Laboratory for HVAC Research

The dataset used in this project was provided by Lawrence Berkeley National Laboratory and published in the journal Scientific Data. Their paper offers an in-depth explanation of the dataset's structure, parameters, and significance, titled "A Labeled Dataset for Building HVAC Systems Operating in Faulted and Fault-Free States" [1]. You can access the dataset in Ref [2]. This dataset is one of the most comprehensive sources available for analyzing HVAC system behavior, recording detailed operational states of a multi-zone variable air volume (VAV) air-handling unit (AHU) system under both faulted and fault-free conditions over an entire year. This extensive data allows for the development of machine learning models capable of identifying patterns that signify potential faults. You can see a diagram of this system in Figure 1, which illustrates the interplay of components such as heating coils, cooling coils, reheat coils, fans, and local controllers.

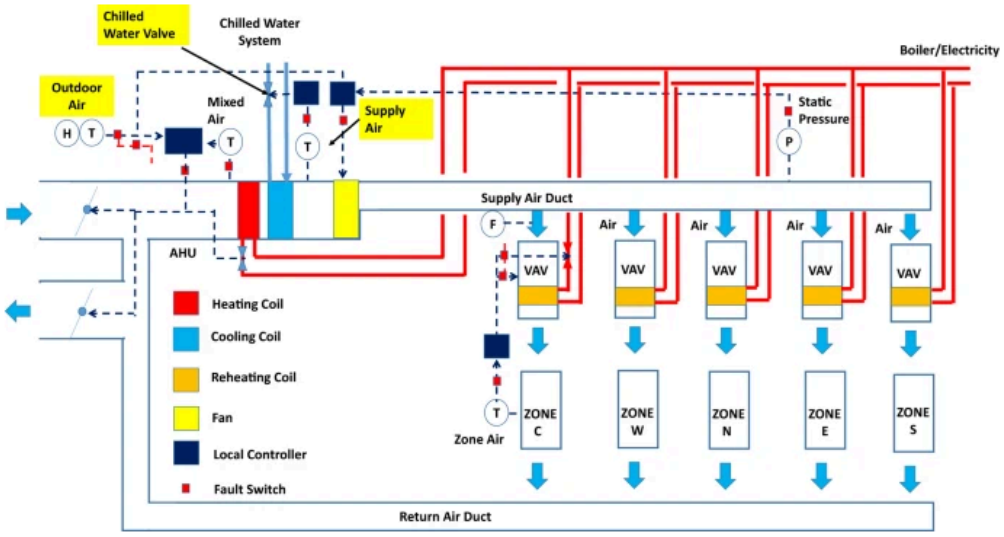


Figure 1. Schematic diagram of multi-zone VAV AHU.

This dataset provides labeled data for various operating states of HVAC systems, making it an excellent resource for developing and testing FDD models. It's a treasure trove for researchers and engineers aiming to develop intelligent solutions for building management.

**The Challenge: Detecting Faults in Complex HVAC Systems**

Faults in HVAC systems—whether due to sensor malfunctions, valve issues, or improper airflow—can be subtle yet costly. Traditional fault detection relies on rule-based methods, which often miss nuanced patterns or generate false alarms. Enter machine learning: a data-driven approach that can uncover hidden anomalies by learning from historical data. In this post, we will explore a Python-based project that leverages machine learning to detect faults in HVAC systems and build a robust FDD framework using this dataset. We will address real-world challenges such as imbalanced data while ensuring high accuracy in identifying faults. We will walk through the code, explore the data, and discuss how machine learning models can be trained to predict and identify potential issues before they become critical.

**Building the Models: A Step-by-Step Journey**

Let's begin by importing the dataset into a Python environment using Google Colab, leveraging libraries like NumPy, Pandas, Matplotlib, Plotly, Scikit-learn, XGBoost, and CatBoost. The dataset is stored in a CSV file, and contains various features related to the operation of AHUs, such as supply air temperature, outdoor air temperature, mixed air temperature, and more.

**********

```
# Import Library
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import warnings
import plotly.express as px
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from  sklearn.metrics  import  classification_report,  confusion_matrix,
ConfusionMatrixDisplay
from sklearn.utils import resample
warnings.filterwarnings('ignore')
%matplotlib inline

!pip install xgboost
!pip install catboost

# Load the dataset
df = pd.read_csv(r'/content/drive/MyDrive/MZVAV-1 (Sim-Full year).csv')
df = pd.DataFrame(df)
```

**********

You can see the first 5 rows of the dataset below. The Fault Detection Ground Truth refers to the labeled data that explicitly identifies whether the HVAC system is operating in a fault-free state or under specific fault conditions. This ground truth is crucial for training and evaluating fault detection models because it provides a clear benchmark for what constitutes normal operation versus various types of faults.

| | Datetime | AHU: Supply Air Temperature | AHU: Supply Air Temperature Set Point | AHU: Outdoor Air Temperature | AHU: Mixed Air Temperature | AHU: Return Air Temperature | AHU: Supply Air Fan Status | AHU: Return Air Fan Status | AHU: Supply Air Fan Speed Control Signal | AHU: Return Air Fan Speed Control Signal | AHU: Outdoor Air Damper Control Signal | AHU: Return Air Damper Control Signal | AHU: Cooling Coil Valve Control Signal | AHU: Heating Coil Valve Control Signal | AHU: Supply Air Duct Static Pressure Set Point | AHU: Supply Air Duct Static Pressure | Occupancy Mode Indicator | Fault Detection Ground Truth |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2017-01-30 00:00:00 | 68.00 | 55.04 | 32.00 | 68.00 | 75.20 | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.04 | 0.0 | 0 | 0 |
| 1 | 2017-01-30 00:01:00 | 67.99 | 55.04 | 32.00 | 66.71 | 75.20 | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.04 | 0.0 | 0 | 0 |
| 2 | 2017-01-30 00:02:00 | 67.99 | 55.04 | -5.93 | 66.71 | 74.45 | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.04 | 0.0 | 0 | 0 |
| 3 | 2017-01-30 00:03:00 | 67.99 | 55.04 | -5.88 | 66.71 | 74.04 | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.04 | 0.0 | 0 | 0 |
| 4 | 2017-01-30 00:04:00 | 67.99 | 55.04 | -5.83 | 66.71 | 73.56 | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.04 | 0.0 | 0 | 0 |

The dataset is rich with features that capture the state of the HVAC system at any given moment. Our goal is to use these features to predict whether the system is operating normally or if a fault has occurred.

**Data Preprocessing**

Before diving into model development, it's essential to preprocess the data. Fortunately, the dataset doesn't contain any missing values, so we can proceed without any imputation.

```
**************************************************************************
********
# check for missin values
print(f'total number of missing values: {df.isnull().sum().sum()}')
**************************************************************************
********
```

*Total number of missing values: 0*

The dataset revealed a wealth of features; however, some non-essential features should be dropped to focus on the most relevant variables for our FDD models. Non-essential features are those that are not commonly used in existing building automation systems for monitoring purposes.

```
**************************************************************************
********
# Drop non-essential features
nonbasic_points = ['AHU: Supply Air Temperature Set Point', 'AHU: Supply
Air Fan Status',
       'AHU: Return Air Fan Status',
       'AHU: Supply Air Fan Speed Control Signal',
       'AHU: Return Air Fan Speed Control Signal',
       'AHU: Outdoor Air Damper Control Signal  ',
       'AHU: Return Air Damper Control Signal',
       'AHU: Cooling Coil Valve Control Signal',
       'AHU: Heating Coil Valve Control Signal',
       'AHU: Supply Air Duct Static Pressure Set Point']
df = df.drop(nonbasic_points, axis=1)
**************************************************************************
********
```

In the next step, we should check the distribution of the target variable (Fault Detection Ground Truth) to see if the dataset is balanced. Unfortunately, the initial dataset was imbalanced, with fewer instances of faults compared to normal operations. An imbalanced dataset with an unequal segmentation of the classes will significantly impact data-driven FDD models in various ways, such as poor performance in minority classes and bias towards the majority class [3].

```
# Check class distribution
plt.figure(figsize=(8, 4))
plt.bar(df['Fault Detection Ground Truth'].value_counts().index, df['Fault Detection Ground Truth'].value_counts().values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution')
plt.show()

if sum(df['Fault Detection Ground Truth'].value_counts().values)/len(df['Fault Detection Ground Truth'].value_counts().values) == df['Fault Detection Ground Truth'].value_counts().values[0]:
   print('The dataset is balanced')
else:
   print('The dataset is not balanced')
```

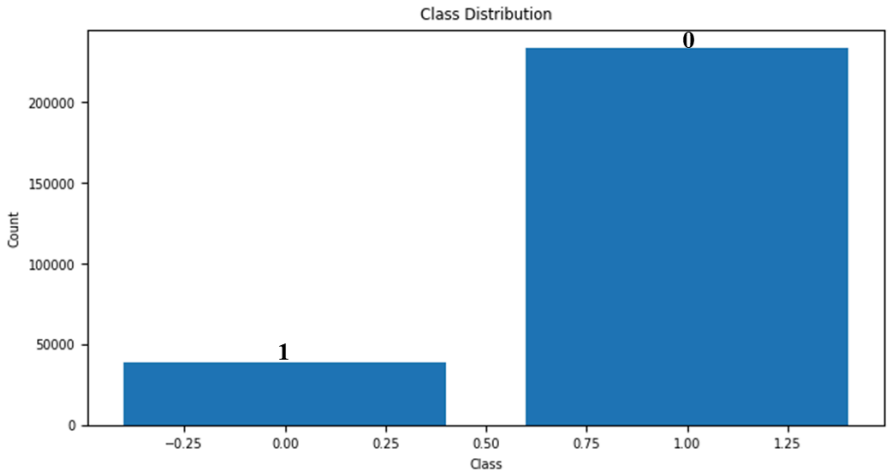


Figure 2. The distribution of classes in the dataset.

Therefore, it is necessary to apply an appropriate technique to equalize the number of different classes of faults within the dataset to balance the dataset. One of the common techniques for balancing is SVMSMOTE which generates synthetic samples of the minority (faulted) class, ensuring a balanced dataset for training.

.

```
**************************************************************************
********
df = df.drop('Datetime', axis=1)
from imblearn.over_sampling import SVMSMOTE
X = df.drop('Fault Detection Ground Truth', axis=1)
y = df[['Fault Detection Ground Truth']]

# Initialize SVM_SMOTE
svm_smote = SVMSMOTE(random_state=42)

# Fit and resample the dataset
X_resampled, y_resampled = svm_smote.fit_resample(X, y)

# Combine X_resampled and y_resampled into a new DataFrame
resampled_df = pd.DataFrame(X_resampled, columns=X.columns)
resampled_df['Fault Detection Ground Truth'] = y_resampled

# Now resampled_df is your new balanced dataset
df = resampled_df
**************************************************************************
********
```

**Developing Data-Driven FDD Models**

In the final step, with a balanced dataset in hand, we can apply three tree-based machine learning algorithms (Random Forest, XGBoost, and CatBoost) as classifiers to construct the data-driven FDD models. The preprocessed dataset was split into training (75%) and testing (25%) sets. Each model was then trained, and their performance was evaluated using metrics such as accuracy, confusion matrices, and classification reports.

```
**************************************************************************
********
from sklearn.metrics import accuracy_score, confusion_matrix,
ConfusionMatrixDisplay, classification_report
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import lightgbm as lgb
from sklearn.preprocessing import label_binarize
from itertools import cycle
```

```
# Initialize the models
models = {
    'RandomForest': RandomForestClassifier(),
    'XGBoost': XGBClassifier(),
    'CatBoost': CatBoostClassifier(verbose=0)
}

X = df.drop('Fault Detection Ground Truth', axis=1)
y = df['Fault Detection Ground Truth'].values.ravel()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

# Dictionary to store predictions and true values
predictions = {}
classification_reports = {}

# Train, predict, and store results for each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    predictions[name] = y_pred
     classification_reports[name] = classification_report(y_test, y_pred,
digits=4)

# Plot confusion matrices in subplots
fig, axes = plt.subplots(1, 3, figsize=(20, 5))
for i, (name, y_pred) in enumerate(predictions.items()):
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(ax=axes[i], cmap='Blues')
    axes[i].set_title(f'{name} Confusion Matrix')
plt.tight_layout()
plt.show()

# Bar plot for F1 score
f1_scores = [f1_score(y_test, y_pred, average='weighted') for y_pred in
predictions.values()]  # Calculate F1 score
plt.figure(figsize=(10, 5))
```

```
plt.bar(models.keys(), f1_scores)
for i, f1 in enumerate(f1_scores):
    plt.text(i, f1, f'{f1:.2f}', ha='center', va='bottom')  # Display F1
score on top of the bars
plt.xlabel('Model')
plt.ylabel('F1 Score')
plt.title('F1 Score Comparison of Models')
plt.ylim(0, 1)  # Set y-axis limits for better visualization
plt.show()
```

**********

**Key Findings**

The results indicate that all three data-driven models effectively detected faults in HVAC systems with impressive performance, achieving high accuracy in distinguishing between fault-free and faulted states. Notably, the Random Forest model achieved an F1-score exceeding 100%, outperforming the other models in this metric, which positions it as a strong candidate for real-world deployment. Figure 3 demonstrates that these models could reliably detect faults, even in a complex multi-zone variable air volume (VAV) system.

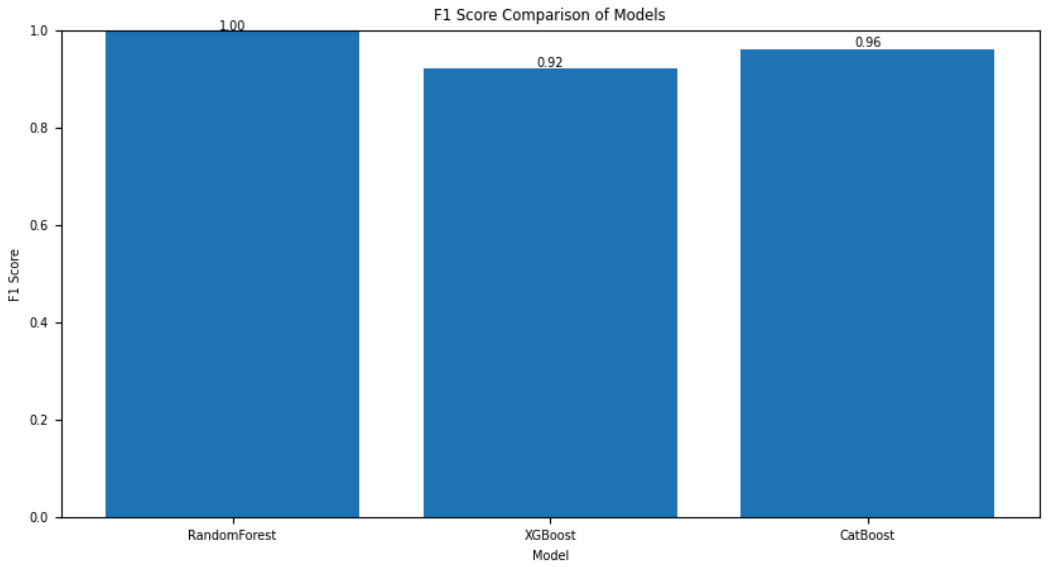


Figure 3. The performance of constructed data-driven FDD models based on F1-score.

The confusion matrices provide a clear visualization of how well each model distinguishes between faulted and fault-free states.

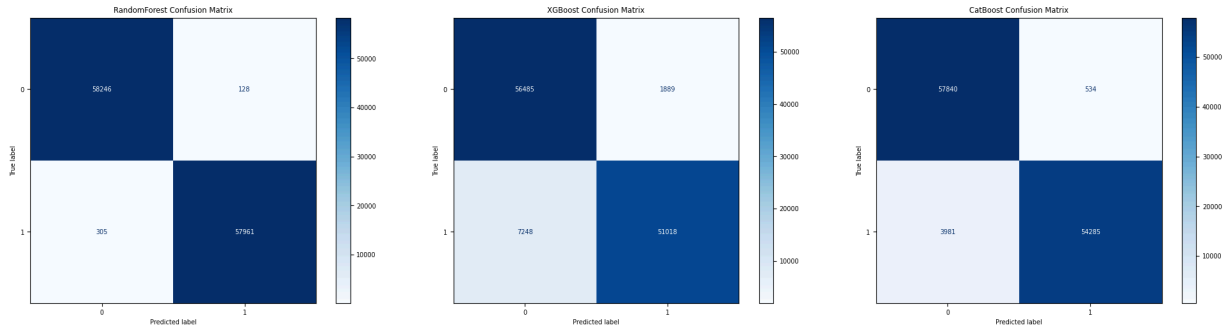


Figure 4. Confusion matrix for the constructed FDD models.

**Why This Matters**

Fault detection in HVAC systems is a game-changer in the journey toward smarter and more sustainable buildings. By leveraging data-driven techniques, facility managers can identify and address issues before they lead to system failures, ultimately reducing operational costs and improving energy efficiency. The availability of labeled datasets allows researchers and industry professionals to refine their models further, improving accuracy and adaptability across different HVAC configurations.

**Looking Ahead**

The success of these models opens the door for real-time fault detection systems integrated into building management platforms. Future work could explore additional features, such as weather data or occupancy patterns, and refine models with deep learning techniques for even greater precision. The possibilities are endless, and I'm excited to see how this research can evolve to support smarter, greener buildings.

**Get Involved**

If you're passionate about HVAC optimization, machine learning, or sustainable building technologies, we encourage you to explore the Lawrence Berkeley National Laboratory dataset and replicate or expand on this work. The code and methodology are openly shared, inviting collaboration and innovation in this critical field.

Let's harness the power of data to build a more efficient and comfortable future—one zone at a time. If you'd like to dive deeper or discuss potential applications, feel free to reach out or leave a comment below. Happy modeling!

**References**

[1] Granderson, Jessica, Guanjing Lin, Yimin Chen, Armando Casillas, Jin Wen, Zhelun Chen, Piljae Im, Sen Huang, and Jiazhen Ling. "A labeled dataset for building HVAC systems operating in faulted and fault-free states." Scientific data 10, no. 1 (2023): 342.
[2] figshare https://doi.org/10.6084/m9.figshare.c.6486349.v1.
[3] Movahed P, Taheri S, Razban A. A bi-level data-driven framework for fault detection and diagnosis of HVAC systems. Appl Energy 2023;339:120948.