User Manual: Vehicle Sensor Fault Detection System

Utility 1.0: Dual-Method Anomaly Detection

October 28, 2025

Abstract

This manual provides instructions for deploying and running the Vehicle Sensor Fault Detection System. The system employs a **Dual-Methodology** approach: 1) A machine learning model (`IsolationForest`) for statistical anomaly scoring, and 2) A **Physics-Based Model** utilizing pre-calculated proportionality constants (k) to detect deviations from expected physical relationships between sensor readings (e.g., Acceleration vs. GPS Speed).

1 Overview and Purpose

The Fault Detection System is designed to identify sensor readings that deviate significantly from a baseline of healthy operation. It outputs a comprehensive fault status based on two criteria:

1. **Statistical Anomaly Score:** An Isolation Forest model determines if the feature vector falls outside the cluster of normal operating data.
2. **Physical Deviation Check:** A reading is flagged if the relationship between sensors (e.g., GPS speed and accelerometer magnitude) violates a tolerance threshold (0.5) defined by the pre-calculated proportionality constants.

This dual approach minimizes false positives and provides diagnostic context for detected faults.

2 Setup and Prerequisites

2.1 System Requirements

- Python 3.x Environment (Tested with 3.8+)
- Execution Platform: Jupyter Notebook or script environment.

2.2 Required Libraries

The following Python packages must be installed prior to execution:

Library Installation

```
pip install pandas numpy scikit-learn
```

Note: The `pickle` library, used for model loading, is standard with Python and does not require a separate installation.

2.3 Deployment Assets (MANDATORY)

The following files, generated during the model training phase, must be present in a local folder named `deployment_assets/`:

- `iforest_model.pkl`: The trained Isolation Forest model.
- `scaler.pkl`: The fitted `StandardScaler` object for data preprocessing.
- `constants.pkl`: A dictionary containing all physics constants (k) and the model's anomaly threshold.

3 Operational Guide: Loading and Prediction

The following steps demonstrate how to load the required assets and prepare the system to process new sensor readings.

1. **Initialize Environment and Define Paths**

   Import the libraries and define the folder path where the deployment assets are stored.

   Python Code 1.1

```
import pandas as pd
import numpy as np
import pickle
import os

DEPLOYMENT_FOLDER = 'deployment_assets'
```

2. **Load Model Assets**

   Use `pickle.load` to retrieve the necessary components from disk. This initializes the system state with the trained parameters.

   Python Code 1.2

```
print("Loading model components...")
try:
    with open(os.path.join(DEPLOYMENT_FOLDER, 'iforest_model.pkl'), 'rb') as f:
        iforest_model = pickle.load(f)
    with open(os.path.join(DEPLOYMENT_FOLDER, 'scaler.pkl'), 'rb') as f:
        scaler = pickle.load(f)
    with open(os.path.join(DEPLOYMENT_FOLDER, 'constants.pkl'), 'rb') as f:
        constants = pickle.load(f)
    print("Models and constants loaded successfully.")
    print(f"k1 (GPS/Accel constant): {constants['k1']:.2f}")

except FileNotFoundError:
    print("ERROR: Deployment files not found. Ensure 'deployment_assets' folder exists.")
```

3. **Define Sensor Reading Format**

   The system expects input data in a specific structure. A new reading must be a NumPy array or list containing the feature values in the required order.

   Python Code 1.3

```
# Feature order MUST match training order:
SENSOR_COLS = ['Accel_X', 'Accel_Y', 'Accel_Z', 'GPS_Speed', 'Engine_RPM']
```

```
# Example of a new, single sensor reading (e.g., 0.1g X, 0.1g Y, 9.8g Z, 50kph, 2500RPM)
new_reading = np.array([[0.1, 0.1, 9.8, 50.0, 2500.0]])
```

4. **Execute Fault Prediction**

   The core prediction process involves scaling the input, calculating the ML score, and performing physics checks. While the full implementation is outside this manual, the logical execution steps are as follows:

Python Code 1.4

```
# 1. Scale the raw reading using the loaded scaler
scaled_reading = scaler.transform(new_reading)

# 2. Get the Isolation Forest anomaly score
anomaly_score = iforest_model.decision_function(scaled_reading)[0]

# 3. Apply the Anomaly Threshold
is_ml_fault = anomaly_score < constants['anomaly_threshold']

# (Additional steps here for physics checks using constants['k1'], etc.)

# 4. Final verdict (simplified check)
if is_ml_fault:
    print(f"\n--- FAULT DETECTED ---")
    print(f"ML Anomaly Score: {anomaly_score:.4f}")
else:
    print(f"\n--- System NORMAL ---")
    print(f"ML Anomaly Score: {anomaly_score:.4f}")
```

4 Troubleshooting

4.1 File Not Found Error

If the script fails to load the '.pkl' files, verify:

- The folder `deployment_assets` exists in the directory where the script is being run.
- The file names (`iforest_model.pkl`, `scaler.pkl`, `constants.pkl`) are spelled correctly.

4.2 Incorrect Scaling

If the anomaly scores are erratic, ensure the input features in your `new_reading` array are provided in the exact same order as they were during training (defined by `SENSOR_COLS`).