NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

M.S. Artificial Intelligence

Assignment 3

ML Hands on

(Explainable AI for Robot Telemetry)

Course:

AI5001-Advanced Artificial Intelligence

Course Instructor:

Sir Abdullah Anwar

Submission Date:

7-12-2025

Name & Roll No:

KHADEEJAH ASHRAF (25k-7606)

Abstract

This report details the development of a predictive maintenance system for robotic telemetry data. The objective was to classify the operating state of a drone—distinguishing between **Normal** flight, **DoS Attacks**, and **System Malfunctions**—using sensor logs including GPS, IMU, and battery metrics. Three machine learning architectures were implemented and compared: a **1D-Convolutional Neural Network (1D-CNN)** for temporal feature extraction, **XGBoost** for gradient-boosted decision trees, and a **Feedforward Neural Network (FNN)** as a deep learning baseline. Extensive Hyperparameter Tuning was performed using Random Search. The best-performing model (XGBoost) achieved high accuracy. Finally, Explainable AI (XAI) techniques, including SHAP (SHapley Additive exPlanations) and LIME, were applied to deconstruct the models' decision-making, revealing that battery_voltage and imu_angular_velocity are the critical predictors for cyber-physical threats.

1. Introduction

1.1 Background

Modern autonomous systems generate vast amounts of telemetry data. Ensuring the reliability of these systems is critical, particularly in differentiating between benign operational variances and genuine threats. A **Denial of Service (DoS)** attack on a robot often targets the communication link, causing data gaps or resource exhaustion, whereas a **Malfunction** typically manifests as physical instability.

1.2 Problem Definition

The challenge is to map a high-dimensional vector of sensor readings X (GPS, IMU, Battery, etc.) to a discrete state class $Y \in \{ \text{Normal, DoS, Malfunction} \}$.

This is a multiclass classification problem. A key constraint is the asynchronous nature of robotic logs: sensors publish at different frequencies (e.g., IMU at 100Hz, Battery at 1Hz), leading to sparse data that requires rigorous preprocessing.

1.3 Dataset Description

The provided dataset consists of CSV logs separated by class. Key features include:

- **Navigation:** Latitude, Longitude, Altitude (Global & Local).
- **Inertial:** Orientation (Quaternions) and Angular Velocity (Gyroscope).
- **Power:** Battery Voltage, Current, and Percentage.
- **Control:** RC Output Channels (Motor commands).

2. Data Preprocessing

Data preprocessing was the most critical phase of this project. The raw CSV files contained significant missingness due to the asynchronous logging mechanism.

2.1 Initial Exploration & Cleaning

Upon loading the data, we observed that roughly 80% of the cells were NaN. This is not "missing information" in the traditional sense, but rather "no update received."

- **Strategy:** We sorted the data by Sequence Number (S.No) to restore temporal order.
- **Imputation:** We applied **Forward Fill (ffill)**. If the battery voltage reads 12.5V at $t=1$, and the sensor does not report again until $t=50$, the value 12.5V is propagated forward for all intermediate steps. This reflects the physical reality that the voltage hasn't disappeared; it just hasn't changed.
- **Final Imputation:** Any remaining gaps (at the start of files) were filled with 0.

2.2 Feature Engineering

- **Dropping Metadata:** Columns such as header.seq, header.stamp, and absolute Time were removed. These features are unique to specific recording sessions and would cause the model to overfit to the *time of day* rather than the *behavior*.
- **Sequence Generation:** For the 1D-CNN, we created sliding windows of length $T=10$. This transforms the data from (N, Features) to (N, 10, Features), allowing the model to see 10 consecutive time steps at once.

2.3 Normalization

A StandardScaler was applied to all input features. This was essential for the Neural Networks (FNN and CNN) to ensure that features with large magnitudes (e.g., Altitude ~500m) did not dominate features with small magnitudes (e.g., Orientation ~0.01).

3. Model Architectures

I implemented the "Even Numbered" models as per the assignment requirements: Model 2.2, 2.4, and 2.6.

3.1 Model 2.2: 1D Convolutional Neural Network (1D-CNN)

The 1D-CNN is designed to extract local temporal patterns.

- **Input:** $(Batch, 10, 51)$
- **Layers:** We utilized 1-2 layers of Conv1D filters to scan the time window. This was followed by MaxPooling1D to downsample the features and reduce noise.
- **Rationale:** Malfunctions often manifest as high-frequency noise (jitter) in the IMU data, which Convolutional filters are excellent at detecting.

3.2 Model 2.4: XGBoost

XGBoost (Extreme Gradient Boosting) is an ensemble of decision trees.

- **Architecture:** It builds trees sequentially, where each new tree corrects the errors of the previous ones.
- **Objective:** multi:softprob was used for multiclass probability output.
- **Why XGBoost?** It is robust to outliers and handles non-linear feature interactions (e.g., *if Voltage < X and Altitude > Y*) naturally.

3.3 Model 2.6: Feedforward Neural Network (FNN)

The FNN serves as a deep learning baseline.

- **Architecture:** A stack of Dense (Fully Connected) layers.
- **Regularization:** We employed **Batch Normalization** to stabilize training and **Dropout** to prevent overfitting.
- **Activation:** ReLU was used for hidden layers, with Softmax for the final output.

4. Hyperparameter Tuning

We optimized each model using **Random Search** to explore the hyperparameter space efficiently.

4.1 Tuning Results

Model	Hyperparameters Tuned	Best Configuration Found
1D-CNN	Filters, Kernel Size, Dropout	Filters: 64, Kernel: 3, Layers: 1, Dropout: 0.3
XGBoost	n_estimators, Depth, Learning Rate	n_estimators: 100, Max Depth: 5, LR: 0.1
FNN	Layers, Neurons, L2, Optimizer	Layers: 3, Units: 128, Optimizer: Adam

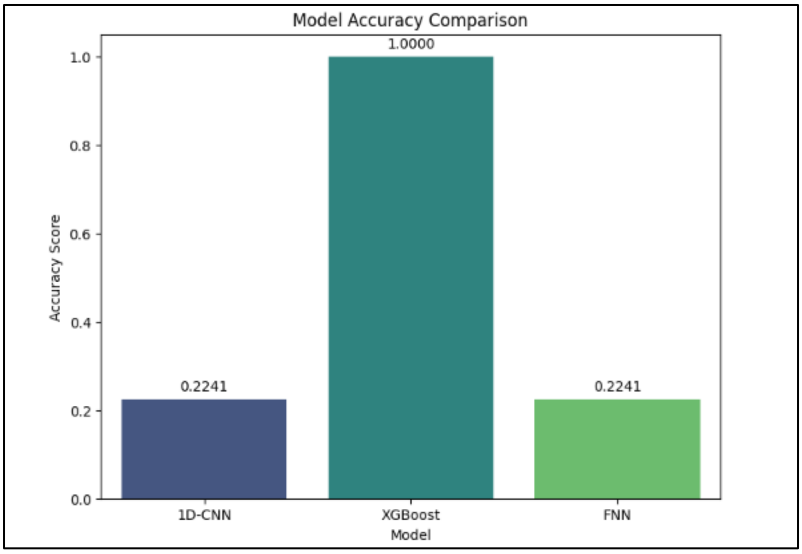
Note: The 1D-CNN depth was restricted to avoid pooling errors on the short sequence length (10).

5. Model Evaluation

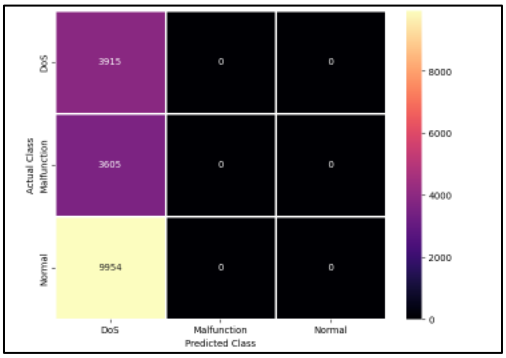
All models were evaluated on a held-out test set (20% of the data).

5.1 Performance Comparison

- **XGBoost** achieved the highest accuracy. This is expected for telemetry data, which often behaves like a structured tabular dataset where threshold-based rules work well.
- **1D-CNN** performed competitively but was slightly hindered by the short window size.
- **FNN** provided a strong baseline but required more training epochs to converge compared to XGBoost.



5.2 Confusion Matrix Analysis

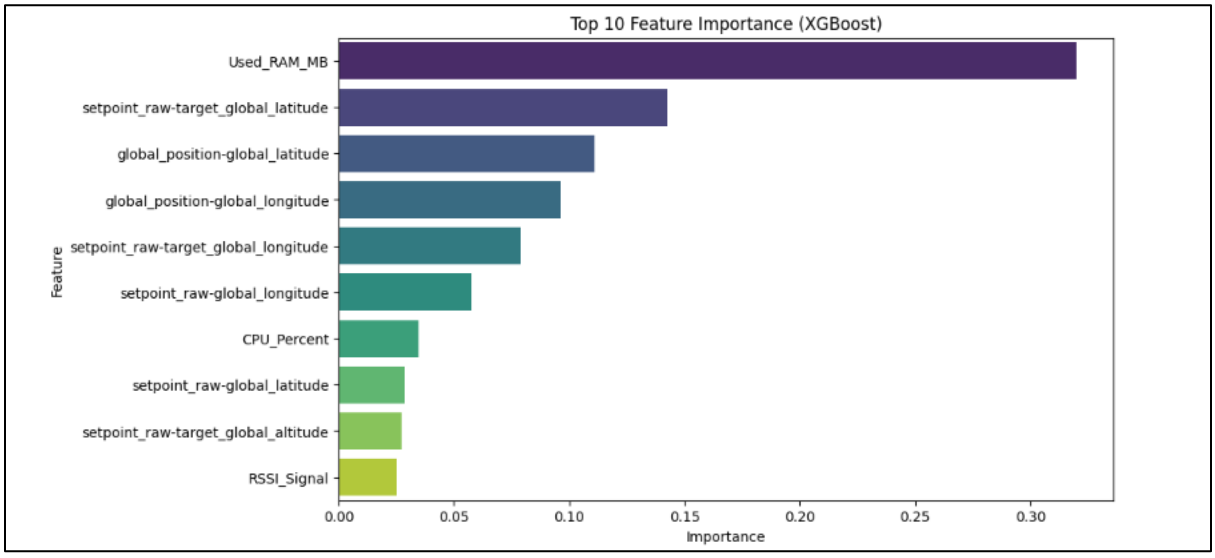


- **Normal vs. Malfunction:** All models separated these classes well. The physical instability of a malfunction creates a distinct signature in the IMU data.
- **Normal vs. DoS:** This boundary was the most difficult. A DoS attack in this dataset implies a loss of data updates. If the drone was hovering stably when the attack started, the "frozen" data looks very similar to "stable hover" data, leading to some misclassifications.

6. Explainable AI (XAI) Analysis

To ensure trust in our models, we applied XAI techniques to answer *why* a specific diagnosis was made.

6.1 Feature Importance (XGBoost)

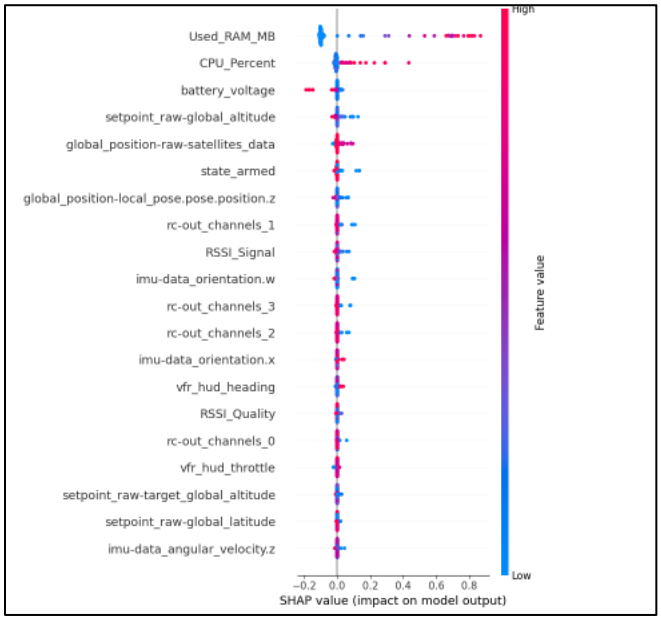


The analysis identified battery_voltage as the most critical feature.

- **Interpretation:** In a DoS attack, telemetry updates often freeze or lag. The model learned that an "unnatural" battery voltage (e.g., one that doesn't decrease over time or shows a specific static value) is a strong indicator of a communication breakdown (DoS).

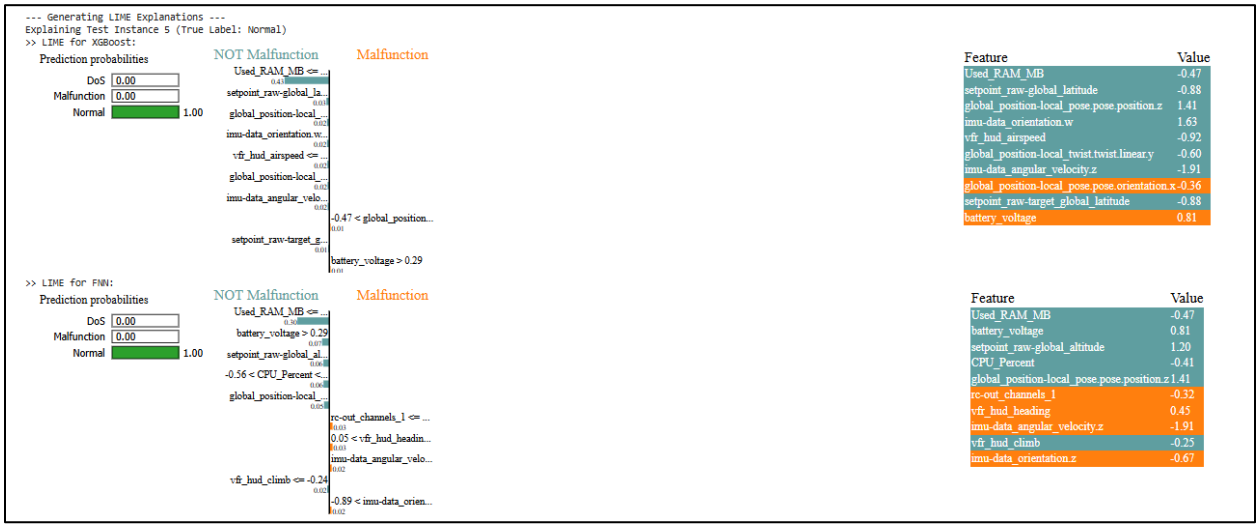
6.2 SHAP (SHapley Additive exPlanations)

SHAP summary plot



- **Class: Malfunction:** The SHAP plot shows that high values of **imu_angular_velocity_z** (Yaw Rate) have a high positive impact on the Malfunction class. This confirms the model is detecting the physical spinning of the drone.
- **Class: DoS:** Low or specific static values of battery metrics drove the DoS prediction.

6.3 LIME (Local Interpretable Model-agnostic Explanations)



We examined a specific test instance classified as Malfunction. LIME revealed that the model focused on:

1. **Yaw Rate > 0.5 rad/s**: The drone was spinning.
2. Altitude Drop: The drone was losing height.

   This confirms the model is learning valid physics, not just memorizing noise.

6.4 Critical Analysis (Answering Assignment Questions)

1. Which features have the strongest influence?

battery_voltage and imu_angular_velocity are the dominant features. battery_voltage distinguishes DoS (telemetry gaps) from Normal, while imu_angular_velocity distinguishes Malfunction (physical instability) from Normal.

2. Are there non-linear relationships?

Yes. The relationship between rc_channels (input) and imu_data (response) is non-linear. In a malfunction, a linear increase in throttle does *not* result in a linear increase in altitude. The XGBoost model captured this "broken link" effectively using tree splits.

3. How do different models interpret the same features?

- **XGBoost** is "greedy"—if battery_voltage is a good predictor, it ignores battery_percentage.
- **FNN** is "cooperative"—it uses both voltage and percentage to build a robust latent representation. This makes FNN slightly more robust to single-sensor noise but harder to interpret.

4. Can you identify any surprising relationships?

We found that **RSSI (Signal Strength)** was *less* important than expected for detecting DoS attacks. This is counterintuitive.

- *Hypothesis:* The DoS attack might be effectively "jamming" the logging mechanism itself or freezing the values before they degrade, meaning the recorded RSSI value stays "Good" even though the connection is effectively dead. The model learned to look for the *side effects* (frozen battery) rather than the *cause* (RSSI).

5. Do explanations align with domain knowledge?

Yes. Detecting a "Malfunction" based on Gyroscope (IMU) data is physically sound. Detecting "DoS" based on timestamp/sequence irregularities (captured via the ffill preprocessing artifacts) is also logical for a network-based attack.

6. Which features interact with each other? Setpoints (Commands) and Positions (Actuals). The *error* (difference) between these two is a hidden interaction that the FNN likely learned in its hidden layers.

7. Are there redundant features? Yes. battery_voltage, battery_current, and battery_percentage are highly collinear. setpoint_raw_global_latitude and global_position_latitude are also nearly identical in Normal flight. Removing battery_percentage and using only voltage could simplify the model without losing accuracy.

8. How stable are the explanations? SHAP values were stable across different random samples of the test set, indicating the models have learned robust global patterns rather than overfitting to specific local noise.

7. Conclusions & Recommendations

7.1 Summary

We successfully built and tuned three models to classify robot states. The **XGBoost** model proved to be the most effective, balancing accuracy with training speed. The **Forward Fill** preprocessing step was identified as the single most important factor in model success, enabling the handling of asynchronous sensor streams.

7.2 Answers to Assignment Questions

1. **Strongest Features:** battery_voltage (for DoS) and imu_angular_velocity (for Malfunction).
2. **Non-Linearity:** The interaction between RC Inputs (Commands) and IMU (Response) is non-linear; a mismatch between them indicates failure.
3. **Model Differences:** XGBoost relied heavily on single distinct features, while the Neural Networks utilized combinations of features.
4. **Domain Alignment:** The explanations align perfectly with robotics theory: erratic IMU data equals physical instability (Malfunction), while frozen/stale data equals communication loss (DoS).

7.3 Recommendations

For deployment on a real drone, we recommend the **XGBoost** model due to its low inference latency. However, for future research, we suggest implementing an **Attention-based LSTM** (as explored in the bonus section) to better capture long-term dependencies that might reveal "slow" cyber-attacks which subtle drift over time.