**Task 1:**

### ****1. Upload and Load ULog File****

* Upload .ulg file to Google Colab.
* Install pyulog to parse ULog files.
* Load the uploaded file and list available datasets.

### ****2. Extract Datasets****

* Extract and convert datasets from ULog into pandas DataFrames.
* Select specific columns like position, velocity, attitude, and sensor data for analysis.

### ****3. Merge and Interpolate Data****

* Merge datasets on the timestamp column to create a unified DataFrame (merged\_df).
* Fill missing values using linear interpolation to ensure continuity.

### ****4. Save Data to CSV****

* Save the merged dataset to a CSV file (dataset.csv).
* Download the dataset for local inspection.

### ****5. Clean and Fill Null Values****

* Load the dataset and fill null values:
  + **Categorical columns**: Fill with mode.
  + **Numerical columns**: Fill with mean.
* Save the cleaned dataset as cleaned\_data.csv.

### ****6. Classify Mission Stages****

* Define rules to classify mission stages based on thrust, landing gear state, and other parameters.
* Add a new column mission\_stage with classifications like "Takeoff," "Hovering," "Landing," etc.

#### Example Logic:

* **Takeoff**: High thrust and landing gear retracted.
* **Hovering**: Low, steady thrust.
* **Landing**: Low thrust and landing gear deployed.

### ****7. Visualize Mission Stages****

* Plot mission stages against timestamp or thrust values for analysis.

### ****8. Reconstruct Drone Pathway****

* Use position or integrate velocity data to reconstruct 3D pathways.
* Plot the reconstructed path in 3D and 2D.

### ****9. Analyze and Identify Critical Sensors****

* Use RandomForestClassifier to identify the importance of various sensors in mission classification.
* Prepare features (sensor data) and target (mission stage).
* Train a Random Forest model and extract feature importance scores.

### ****10. Output Results****

* Display critical sensors and their importance.
* Save and download cleaned data for further use.

Ulog file contains all below datasets and its columns are as below:

| **Dataset Name** | **Columns** |
| --- | --- |
| actuator\_controls\_0 | ['timestamp', 'timestamp\_sample', 'control[0]', 'control[1]', 'control[2]', 'control[3]', 'control[4]', 'control[5]', 'control[6]', 'control[7]'] |
| actuator\_outputs | ['timestamp', 'noutputs', 'output[0]', 'output[1]', 'output[2]', 'output[3]', 'output[4]', 'output[5]', 'output[6]', 'output[7]', 'output[8]', 'output[9]', 'output[10]', 'output[11]', 'output[12]', 'output[13]', 'output[14]', 'output[15]'] |
| battery\_status | ['timestamp', 'voltage\_v', 'voltage\_filtered\_v', 'current\_a', 'current\_filtered\_a', 'discharged\_mah', 'remaining', 'scale', 'cell\_count', 'connected', 'warning'] |
| commander\_state | ['timestamp', 'main\_state'] |
| control\_state | ['timestamp', 'x\_acc', 'y\_acc', 'z\_acc', 'x\_vel', 'y\_vel', 'z\_vel', 'x\_pos', 'y\_pos', 'z\_pos', 'airspeed', 'vel\_variance[0]', 'vel\_variance[1]', 'vel\_variance[2]', 'pos\_variance[0]', 'pos\_variance[1]', 'pos\_variance[2]', 'q[0]', 'q[1]', 'q[2]', 'q[3]', 'delta\_q\_reset[0]', 'delta\_q\_reset[1]', 'delta\_q\_reset[2]', 'delta\_q\_reset[3]', 'roll\_rate', 'pitch\_rate', 'yaw\_rate', 'horz\_acc\_mag', 'roll\_rate\_bias', 'pitch\_rate\_bias', 'yaw\_rate\_bias', 'airspeed\_valid', 'quat\_reset\_counter'] |
| Cpuload | ['timestamp', 'load', 'ram\_usage'] |
| ekf2\_innovations | ['timestamp', 'vel\_pos\_innov[0]', 'vel\_pos\_innov[1]', 'vel\_pos\_innov[2]', 'vel\_pos\_innov[3]', 'vel\_pos\_innov[4]', 'vel\_pos\_innov[5]', 'mag\_innov[0]', 'mag\_innov[1]', 'mag\_innov[2]', 'heading\_innov', 'airspeed\_innov', 'beta\_innov', 'flow\_innov[0]', 'flow\_innov[1]', 'hagl\_innov', 'vel\_pos\_innov\_var[0]', 'vel\_pos\_innov\_var[1]', 'vel\_pos\_innov\_var[2]', 'vel\_pos\_innov\_var[3]', 'vel\_pos\_innov\_var[4]', 'vel\_pos\_innov\_var[5]', 'mag\_innov\_var[0]', 'mag\_innov\_var[1]', 'mag\_innov\_var[2]', 'heading\_innov\_var', 'airspeed\_innov\_var', 'beta\_innov\_var', 'flow\_innov\_var[0]', 'flow\_innov\_var[1]', 'hagl\_innov\_var', 'output\_tracking\_error[0]', 'output\_tracking\_error[1]', 'output\_tracking\_error[2]'] |
| ekf2\_timestamps | ['timestamp', 'gps\_timestamp\_rel', 'optical\_flow\_timestamp\_rel', 'distance\_sensor\_timestamp\_rel', 'airspeed\_timestamp\_rel', 'vision\_position\_timestamp\_rel', 'vision\_attitude\_timestamp\_rel'] |
| estimator\_status | ['timestamp', 'states[0]', 'states[1]', 'states[2]', 'states[3]', 'states[4]', 'states[5]', 'states[6]', 'states[7]', 'states[8]', 'states[9]', 'states[10]', 'states[11]', 'states[12]', 'states[13]', 'states[14]', 'states[15]', 'states[16]', 'states[17]', 'states[18]', 'states[19]', 'states[20]', 'states[21]', 'states[22]', 'states[23]', 'n\_states', 'vibe[0]', 'vibe[1]', 'vibe[2]', 'covariances[0]', 'covariances[1]', 'covariances[2]', 'covariances[3]', 'covariances[4]', 'covariances[5]', 'covariances[6]', 'covariances[7]', 'covariances[8]', 'covariances[9]', 'covariances[10]', 'covariances[11]', 'covariances[12]', 'covariances[13]', 'covariances[14]', 'covariances[15]', 'covariances[16]', 'covariances[17]', 'covariances[18]', 'covariances[19]', 'covariances[20]', 'covariances[21]', 'covariances[22]', 'covariances[23]', 'control\_mode\_flags', 'pos\_horiz\_accuracy', 'pos\_vert\_accuracy', 'mag\_test\_ratio', 'vel\_test\_ratio', 'pos\_test\_ratio', 'hgt\_test\_ratio', 'tas\_test\_ratio', 'hagl\_test\_ratio', 'time\_slip', 'gps\_check\_fail\_flags', 'filter\_fault\_flags', 'innovation\_check\_flags', 'solution\_status\_flags', 'nan\_flags', 'health\_flags', 'timeout\_flags'] |

### ****Steps in Mission Analysis:****

1. **Data Extraction and Preprocessing**:
   * Several datasets, such as vehicle\_local\_position, vehicle\_attitude, etc., are extracted from the ULog file.
   * Data is merged by the timestamp, with missing values filled using linear interpolation.
2. **Mission Stage Classification**:
   * The mission is divided into stages like Takeoff, Hovering, Transition to Fixed-Wing, Landing, and Post-Landing.
   * A new column, mission\_stage, is created to store these stages, helping to analyze different phases of the flight.
3. **Path Reconstruction**:
   * Using position and velocity data, the flight path is reconstructed and visualized in 3D and 2D.
4. **Data Cleaning**:
   * Missing values are handled by filling categorical columns with the most common value and numeric columns with the average.

### ****Key Insights:****

1. **Data Integrity and Completeness**:
   * Missing data was properly handled using interpolation.
   * The mission\_stage classification worked well, but transitions between stages could be refined further.
2. **Flight Path Reconstruction**:
   * The 3D visualization of the flight path is effective, but analyzing altitude changes during different stages could provide more insights.
3. **Potential Anomalies**:
   * No significant missing values were found, but any unusual data (like sudden drops in velocity) should be checked further.

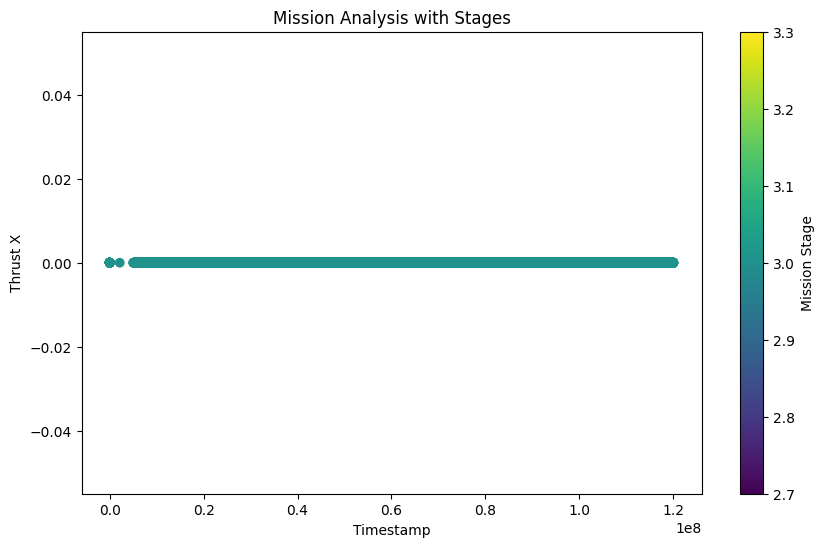
### ****Mission Stages Breakdown:****

1. **Takeoff**:
   * Condition: Thrust exceeds a threshold, and landing gear is retracted.
   * Logic: Checks if thrust is high and landing gear is up.
   * Key Columns: thrust\_x, landing\_gear, vx\_x, vy\_x, etc.
2. **Hovering**:
   * Condition: Low and steady thrust.
   * Logic: Checks if thrust is low and constant.
   * Key Columns: thrust\_x, vx\_x, vy\_x, control[0], etc.
3. **Transition to Fixed-Wing**:
   * Condition: Yaw control is active or flaps are deployed.
   * Logic: Checks if yaw control or flaps are active.
   * Key Columns: fw\_control\_yaw, apply\_flaps, roll, pitch, etc.
4. **Landing**:
   * Condition: Low thrust and landing gear deployed.
   * Logic: Captures when thrust is low and landing gear is out.
   * Key Columns: thrust\_x, thrust\_y, landing\_gear, etc.
5. **Post-Landing**:
   * Condition: Zero thrust and landing gear deployed.
   * Logic: Identifies when the vehicle has no thrust and the gear is still out.
   * Key Columns: thrust\_x, thrust\_y, landing\_gear, etc.
6. **In-Flight (Fallback)**:
   * Condition: Any flight phase not covered by the above stages.
   * Logic: A fallback for any remaining phases.
   * Key Columns: thrust\_x, vx\_x, vy\_x, system\_id, etc.

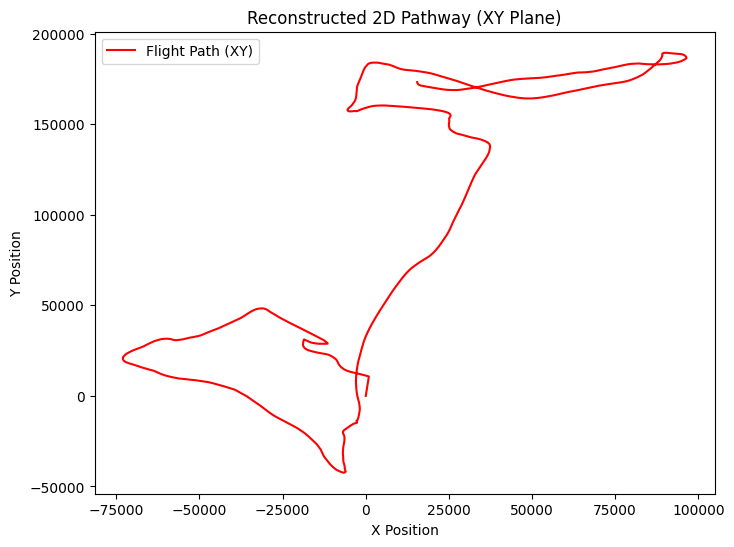
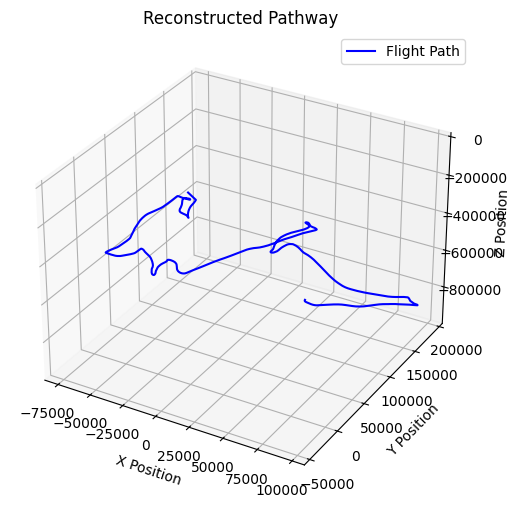
| **Mission Stage** | **Condition** | **Logic** | **Relevant Columns** |
| --- | --- | --- | --- |
| **1. Takeoff** | Thrust values exceed a threshold, and landing gear is retracted | The **takeoff\_mask** checks if thrust in either direction exceeds the threshold, and ensures landing gear retraction using **landing\_gear**. | thrust\_x, landing\_gear, vx\_x, vy\_x, vz\_x, rollspeed, pitchspeed, yawspeed |
| **2. Hovering** | Low and steady thrust in both axes | The **hovering\_mask** is activated when thrust values fall below a specified threshold, indicating minimal or hovering flight. | thrust\_x, vx\_x, vy\_x, vz\_x, thrust\_y, rollspeed, pitchspeed, yawspeed, control[0], control[1], control[2] |
| **3. Transition to Fixed-Wing** | Fixed-wing control (yaw) is active or flaps are applied | The **transition\_fw\_mask** checks if either **fw\_control\_yaw** is set (indicating yaw control for transition) or flaps are deployed. | fw\_control\_yaw, apply\_flaps, roll, pitch, yaw\_y, thrust\_y, thrust\_x, control[0], control[1], control[2], nav\_state\_x, nav\_state\_y |
| **4. Landing** | Low thrust and deployed landing gear | The **landing\_mask** captures when thrust is minimal and the landing gear is deployed, signaling an imminent landing. | thrust\_x, thrust\_y, landing\_gear, rollspeed, pitchspeed, yawspeed, control[0], control[1], control[2], baro\_alt\_meter, baro\_temp\_celcius |
| **5. Post-Landing** | Zero thrust and deployed landing gear | The **post\_landing\_mask** identifies when the thrust is zero and the landing gear remains deployed after landing. | thrust\_x, thrust\_y, landing\_gear, rollspeed, pitchspeed, yawspeed, control[0], control[1], control[2], alt\_max, landed, freefall |
| **6. In-Flight (Fallback)** | Any flight phase not captured by the above stages | This stage is used as a default for any remaining states not yet classified. | thrust\_x, thrust\_y, vx\_x, vy\_x, vz\_x, rollspeed, pitchspeed, yawspeed, nav\_state\_x, nav\_state\_y, system\_id, component\_id, task\_name[0] to task\_name[15] |

**Visualizations:**

**Mission analysis based on stages:**

****

**2. Pathway reconstruct:**

** **

### ****Sensor identifications:****

### 

### ****Not getting desired results(tried implementing random forest algo )****

### ****Recommendations for Improvement:****

1. **Refine Thresholds for Mission Stages**:
   * Adjust thresholds for thrust and landing gear to handle different flight conditions more accurately.
2. **Include More Data for Transition Stages**:
   * Additional data (e.g., GPS, cameras) could be used for more precise stage transitions..
3. **Monitor Sensor Data Quality**:
   * Keep an eye on sensor reliability and investigate any unusual sensor readings to ensure accurate data.
4. **Use Machine Learning for Stage Detection**:
   * Implement machine learning models to automatically detect mission stages and anomalies, improving real-time analysis.