# Sprint Report

PORTFOLIO TASK 2/3/4

Unit code: EAT40005
Unit Name: Engineering Project A
Submission date: 10 / 06 / 2025

| Student Name | Student Id | Statement of contribution to the report |
|---|---|---|
| Dale Bent | 102567413 | Lessons Learned and some sprint plan |
| Mohammed Barsat Zulkarnine | 103810626 | Sprint Progress and Sprint Plan |
| Sadman Tariq | 103798780 | Sprint Retrospective and Sprint Plan |
| Dang Khoa Le | 103844421 | Sprint Demonstration and Sprint Review |

## ACKNOWLEDGMENT OF COUNTRY

We, Dale Bent, Mohammed Barsat Zulkarnine, Dang Khoa Le, and Sadman Tariq, acknowledge the Traditional Custodians of the lands on which we lived and worked while completing this project. We pay our respects to their Elders past and present and extend that respect to all Aboriginal and Torres Strait Islander peoples.

We recognise their enduring connection to land, waters, and culture, and we honour their rich traditions and contributions to our shared community.

## 1. SPRINT PLAN

Sprint 3 focused on extending the progress made in the previous sprint by refining and automating the data pipeline, integrating cloud services, and starting exploratory analysis on driving behaviour.

The team made significant progress in automating the data collection process using a Raspberry Pi. The Pi now reliably connects to the OBD-II device, captures data in real time, and manages session-level CSV exports with minimal manual intervention. This step was crucial in ensuring consistent and scalable data acquisition.

We also completed the integration with Hugging Face cloud infrastructure. Uploaded datasets are now automatically passed through a cleaning pipeline that filters out corrupted or incomplete entries, standardizes feature formats, and validates timestamps. This integration significantly streamlines the preprocessing stage and prepares the data for downstream analysis.

On the local side, we refined our OBD data collection algorithm to better handle sensor latency and inconsistent readings. Improvements were made to how we alternate between high- and low-frequency parameters, helping reduce data gaps and improve overall sampling reliability.

Another key development this sprint was the initial testing of unsupervised learning techniques to classify driving behaviour. Using features like RPM, throttle position, and speed, we grouped similar driving patterns and compared them to our manually recorded driving style labels. The early results are encouraging and show meaningful alignment, but additional data collection is still needed to fine-tune the model and ensure its robustness.

As the first semester comes to a close, our focus over the break will be to continue collecting more structured driving data, fine-tune the Raspberry Pi and cloud infrastructure, and ensure the system remains stable over longer periods. Most importantly, we now need to define what exactly we mean by "maintenance" in the context of this project. Early next semester, the team will finalize a specific scope, such as predicting battery failure, engine wear, or another target, and begin aligning our models and data pipeline toward making that outcome measurable and predictable.

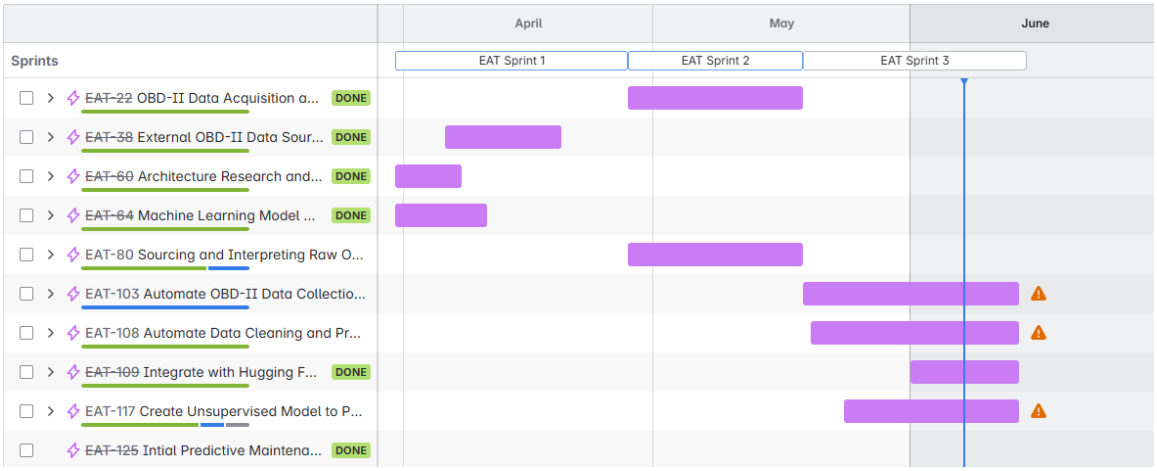
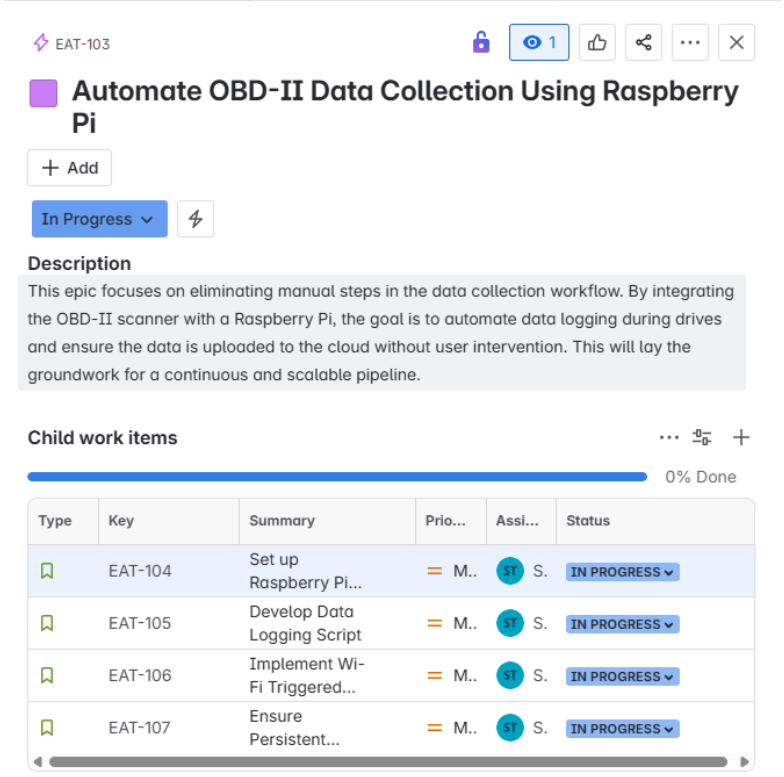
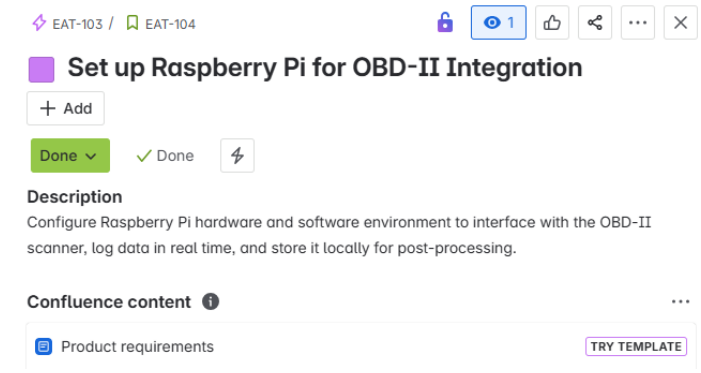



Figure 1: Sprint timeline

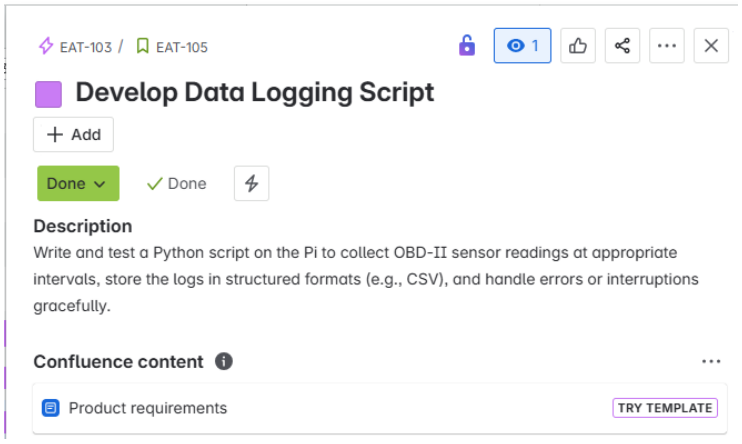
## Task 1



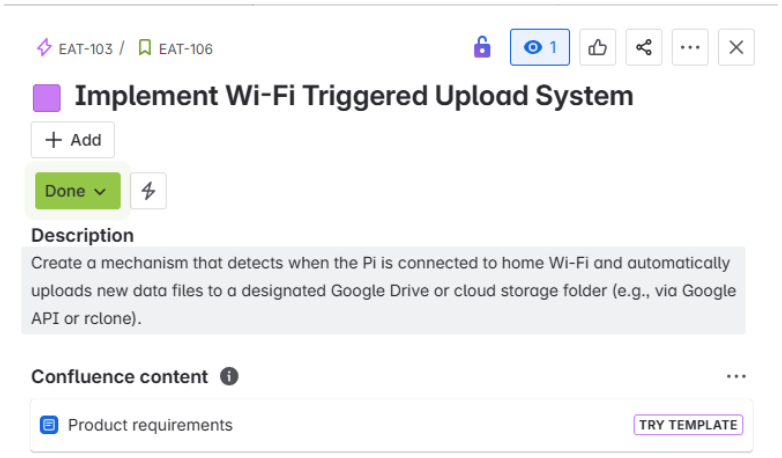

Subtask 1.1



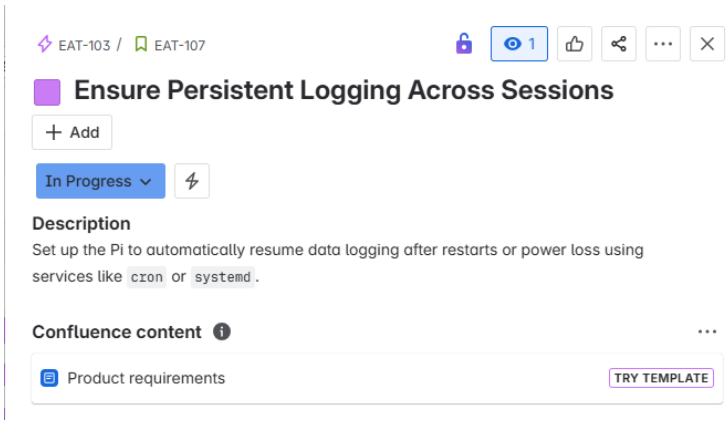

Sub task 1.2



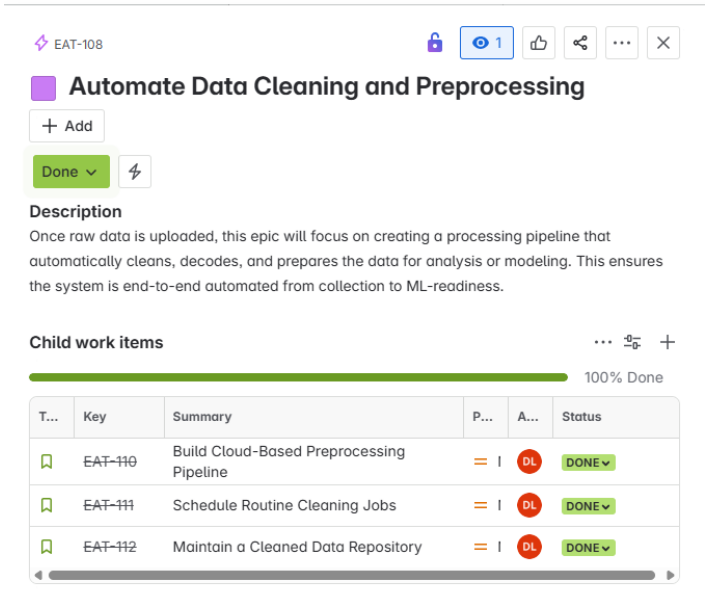

Sub task 1.3



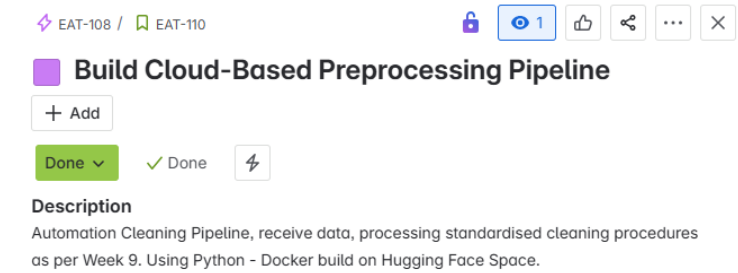

Sub task 1.4



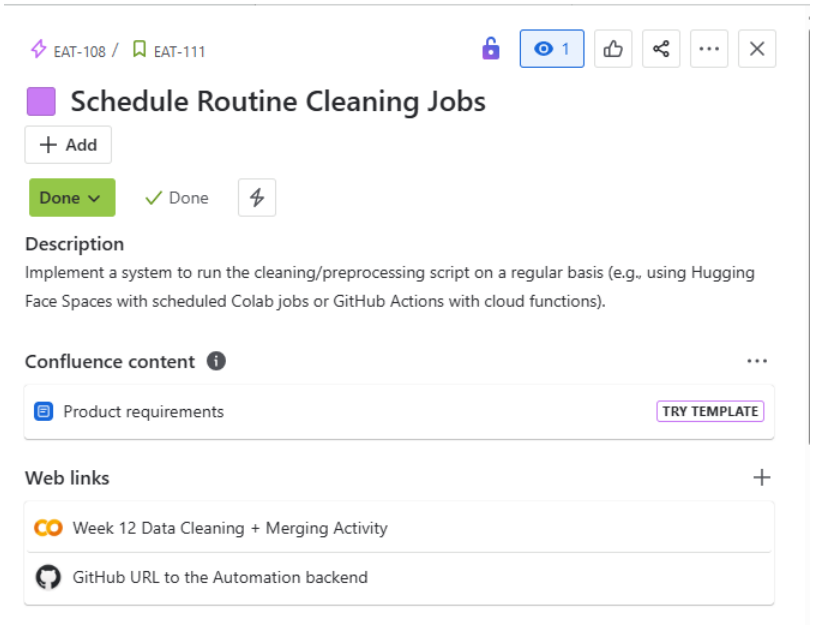

## Task 2



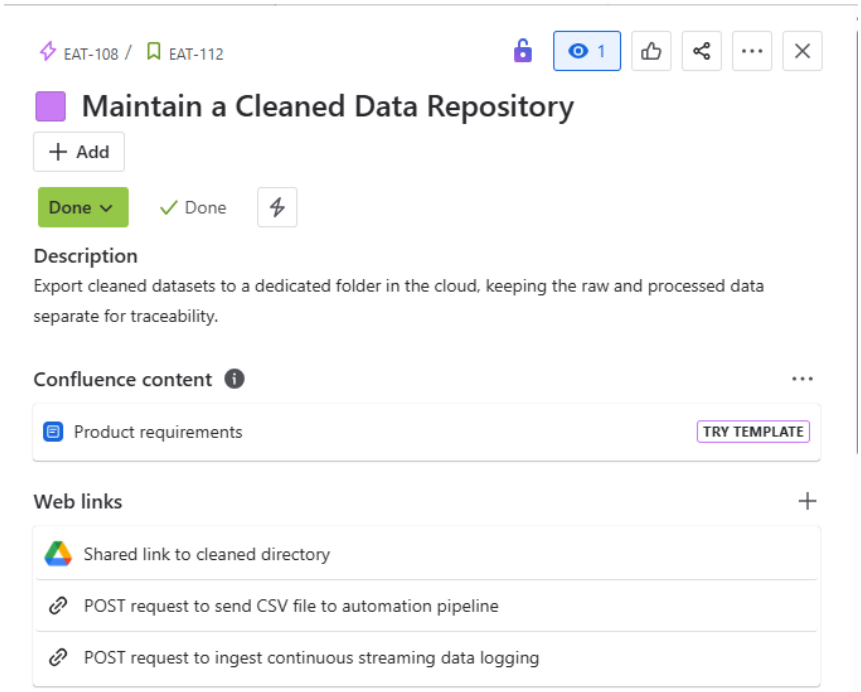

Sub task 2.1



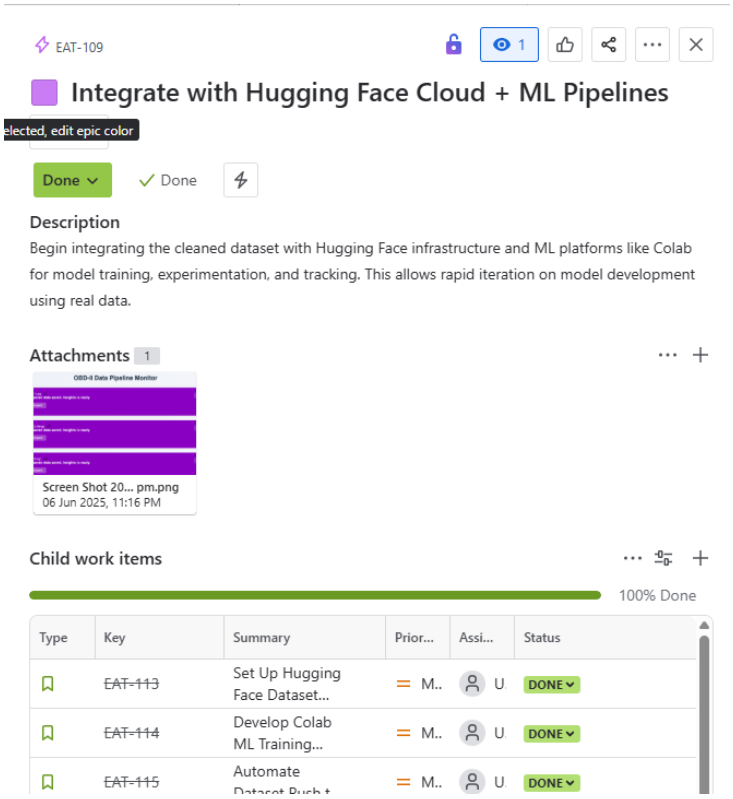

Sub task 2.2



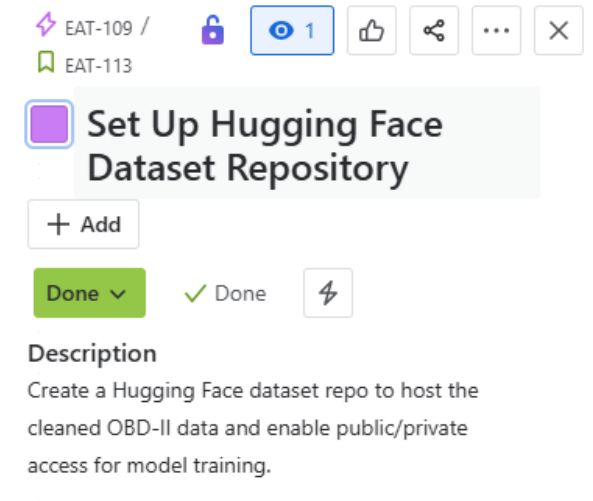

Sub task 2.3



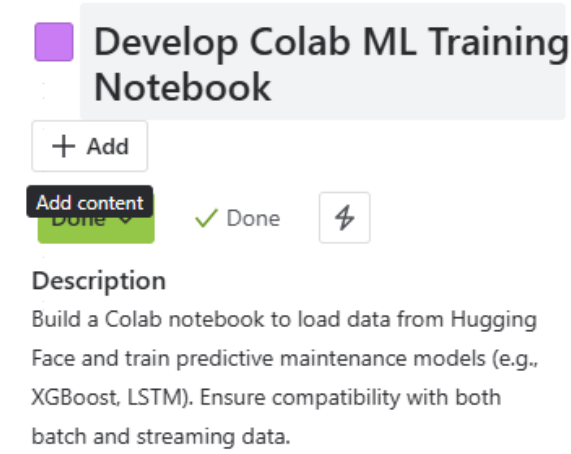

**Task 3**



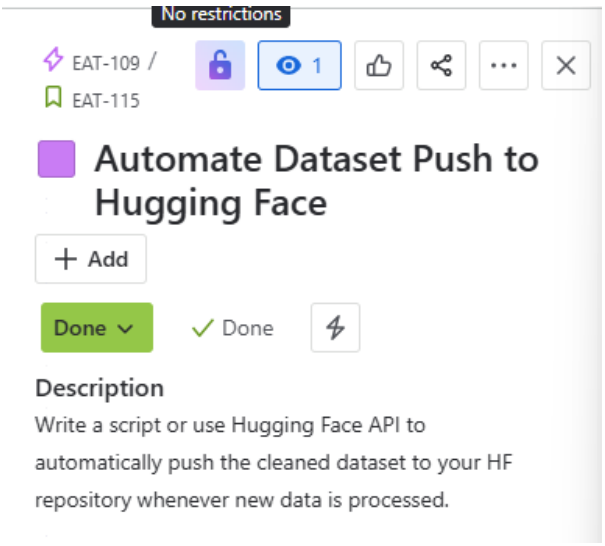

Sub task 3.1




Sub task 3.2




Sub task 3.3




Sub task 3.4

## Link Preprocessing and ML Training

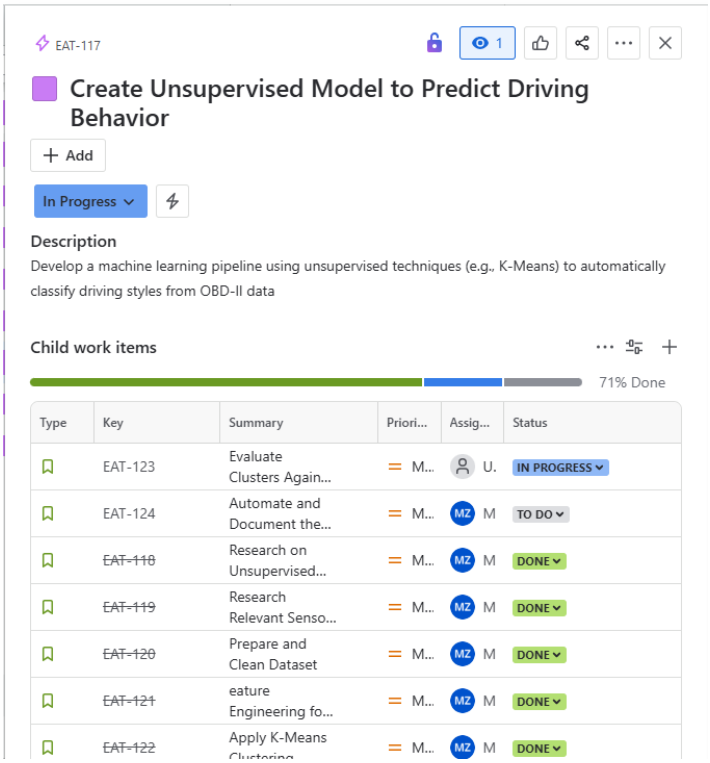
+ Add

Done ✓ Done

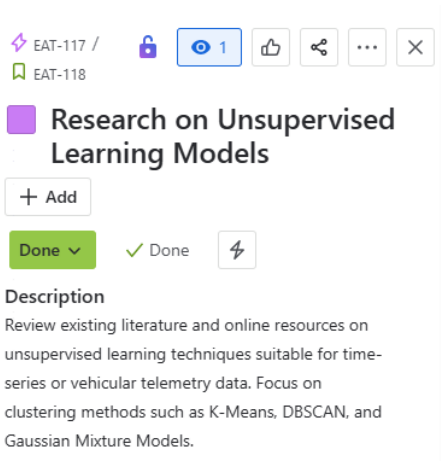
**Description**

Chain together the pipeline so that new cleaned data can trigger or be used immediately in training runs, setting the stage for continuous model improvement.

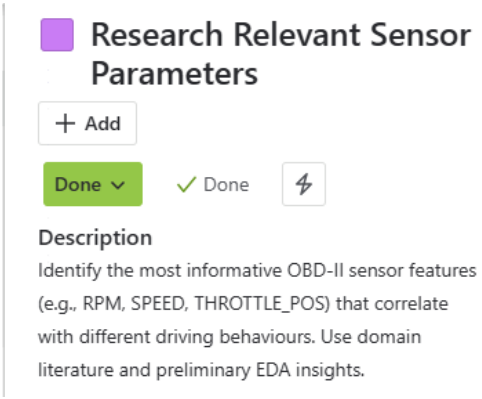
**Task 4**



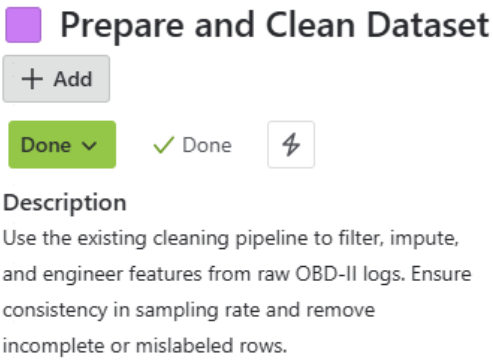

Sub task 4.1



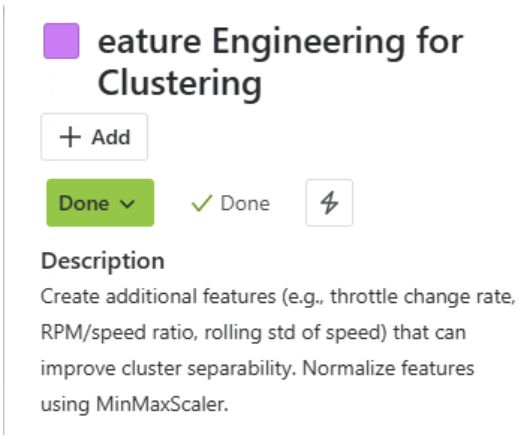

Sub task 4.2



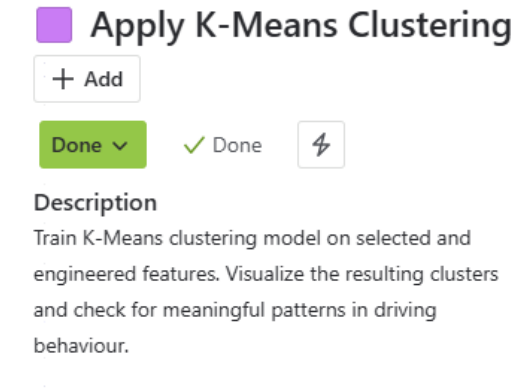

Sub task 4.3



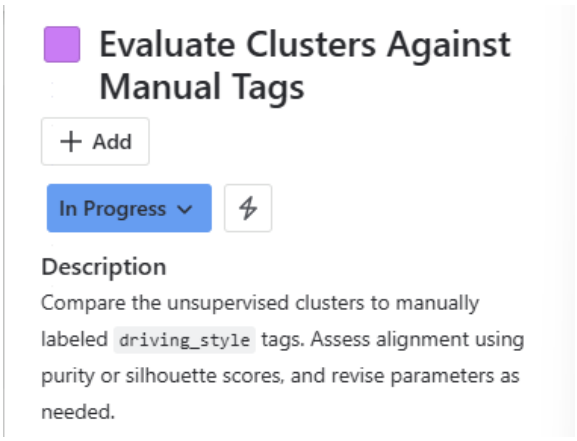

Sub task 4.4



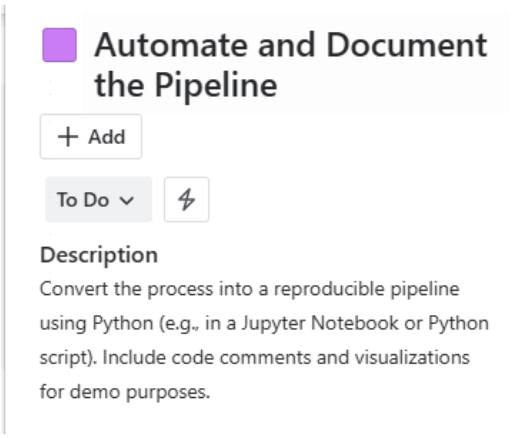

Sub task 4.5




Sub task 4.6

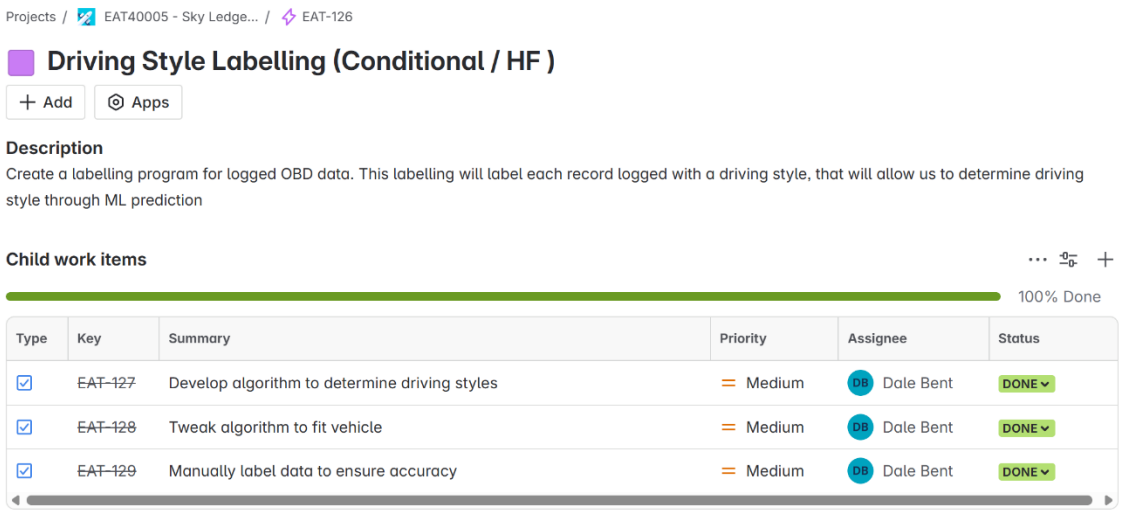



Sub task 4.7




Task 5:

This can be seen via GitHub (https://github.com/benty691/EAT40005)




Subtask 5.1:

Projects / EAT40005 - Sky Ledge... / EAT-126 / EAT-127

## Develop algorithm to determine driving styles

+ Add  Apps

**Description**

Add a description...

**Subtasks**

100% Done

| Type | Key | Summary | Priority | Assignee | Status |
|---|---|---|---|---|---|
| | ~~EAT-130~~ | Define different driving styles | = Medium | DB Dale Bent | DONE |
| | ~~EAT-131~~ | Add traffic logic | = Medium | DB Dale Bent | DONE |
| | ~~EAT-132~~ | Define road types based on speeds | = Medium | DB Dale Bent | DONE |

Subtask 5.2:

Projects / EAT40005 - Sky Ledge... / EAT-126 / EAT-128

## Tweak algorithm to fit vehicle

+ Add  Apps

**Description**

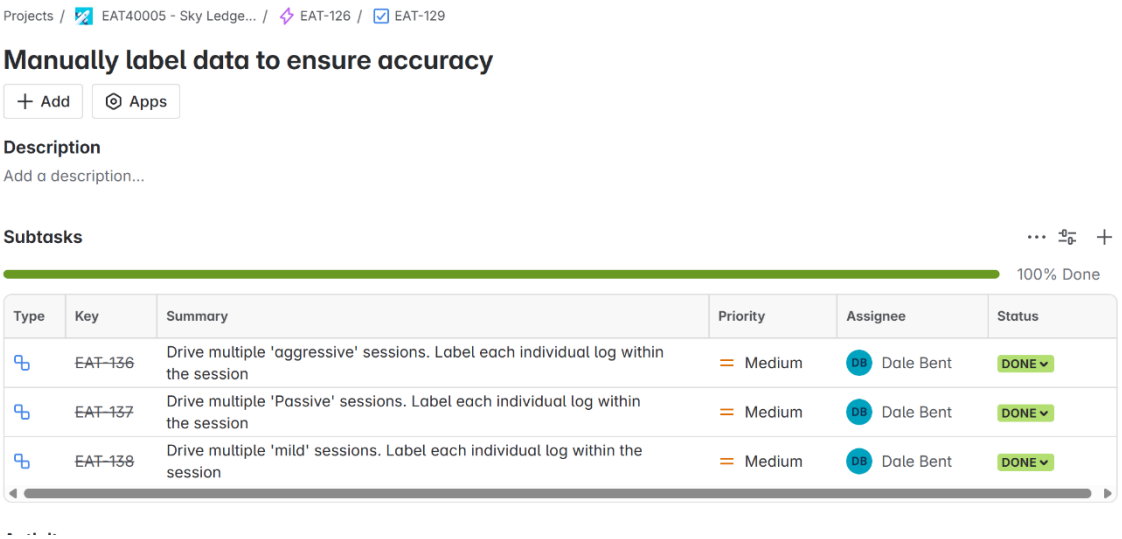
Add a description...

**Subtasks**

0% Done

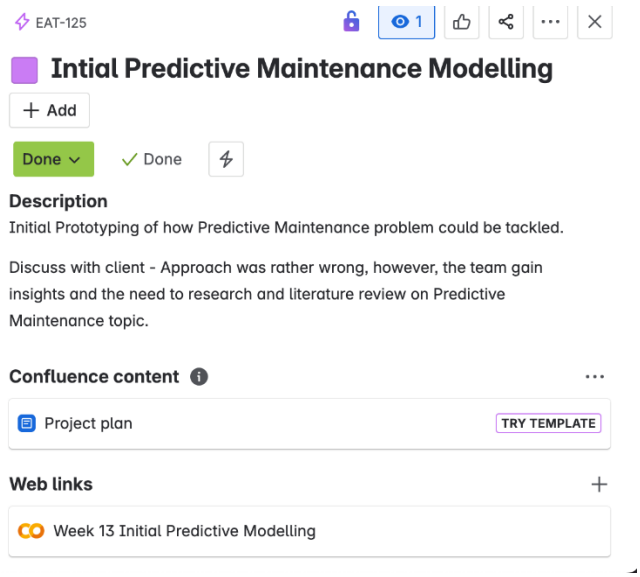
| Type | Key | Summary | Priority | Assignee | Status |
|---|---|---|---|---|---|
| | EAT-133 | Adjust RPM thresholds | = Medium | DB Dale Bent | TO DO |
| | EAT-134 | Adjust RPM thresholds / rolling averages | = Medium | DB Dale Bent | TO DO |
| | EAT-135 | Adjust speed handling | = Medium | DB Dale Bent | TO DO |

Subtask 5.3:




**Task 6:**

Access Google Colab Notebook shared link for this task (Week 13): https://colab.research.google.com/drive/1wHF9cHsg__VuzCEVWH0g1fRiq69hjOzS?usp=sharing




## 2. SPRINT PROGRESS

In Sprint 3, the team continued transitioning from setup to automation and modelling. Building on the groundwork laid in Sprint 2, this sprint emphasized end-to-end automation and model

experimentation. While some parts of our infrastructure were functionally complete last sprint, we focused on refining the systems, increasing reliability, and integrating initial stages of unsupervised learning to begin identifying driving behaviour patterns.

We adopted a more structured task model for this sprint, organizing major objectives into core tasks and subtasks for better coordination and accountability. Key accomplishments include successful development of an OBD-II data logging script with automated upload features, deployment of cloud-based preprocessing pipelines, and our first version of a clustering-based driving behaviour classifier.

### *Task 1 – Automate OBD-II Data Collection Using Raspberry Pi*

We began working toward a persistent, real-world data acquisition system that functions independently from laptops and manual uploads. The goal is to create a plug-and-play logging system that runs continuously, collects OBD-II data from vehicles, and uploads sessions to the cloud once a secure connection is established.

**Subtasks:**

- **Set up Raspberry Pi for OBD-II integration:** Successfully configured a Raspberry Pi device to communicate with the OBD-II scanner and control data logging.
- **Develop data logging script:** A complete, Python-based logger was written to stream data from the vehicle to local CSV files in real-time.
- **Implement Wi-Fi-triggered upload system:** A condition-based uploader was developed to check for a trusted Wi-Fi network (e.g., home Wi-Fi). When conditions are met and logging is idle, the system uploads logs to our cloud server.
- **Ensure persistent logging across sessions:** The script supports state recovery and ensures logging resumes reliably between reboots or disconnections.

We are currently in the **testing phase** of this task, validating behaviour across multiple drive sessions and environments. Once testing is complete, the system will be ready for full deployment.

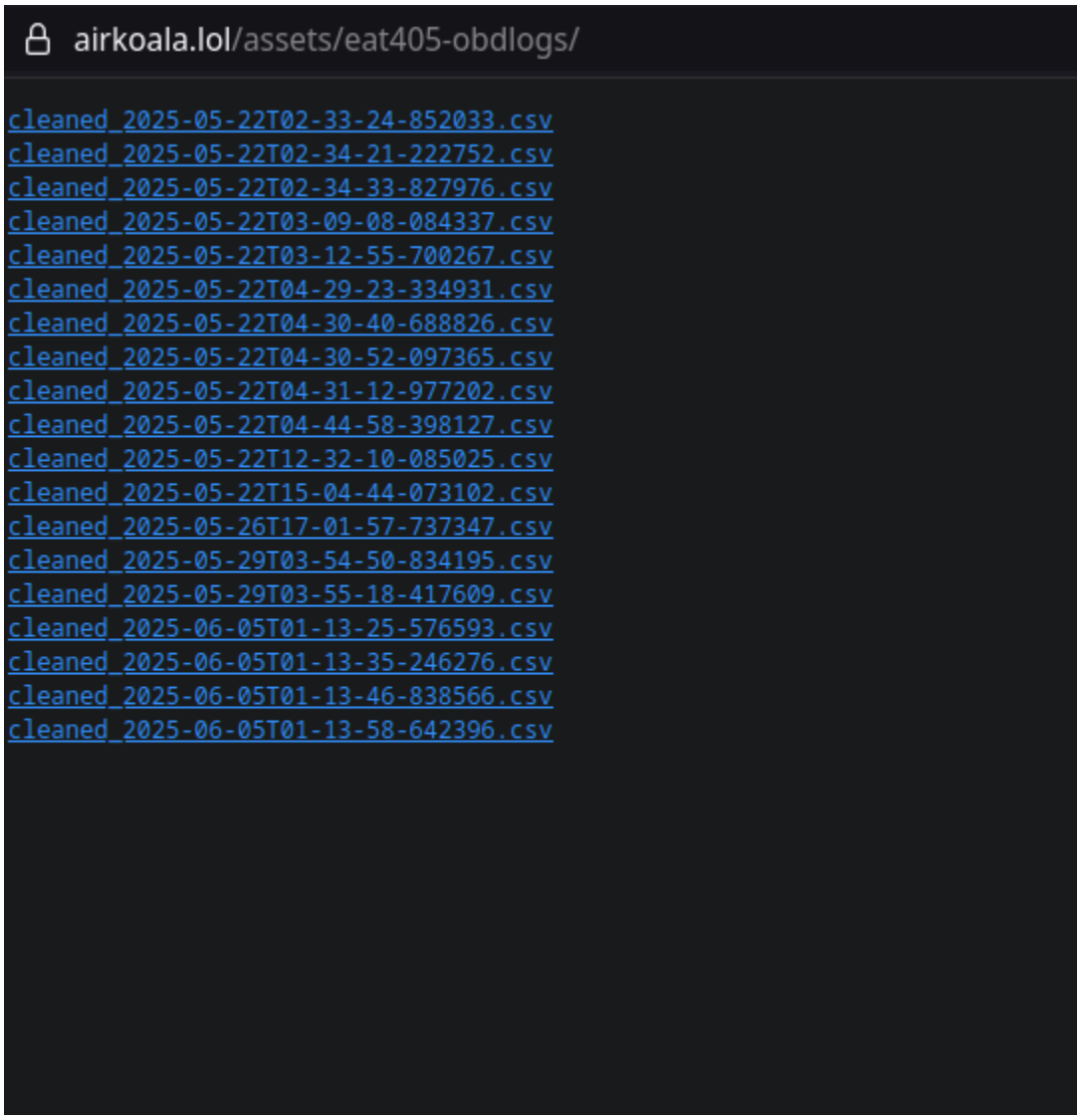
```
# Makes a POST request to $POST_ENDPOINT with the CSVs
upload_csv_files() {
    echo $STATUS_UPLOADING > $STATUS_FILE
    echo Uploading via SCP...
    scp "$CSV_DIR/*.csv" "$SCP_PATH/" &

    for file in "$CSV_DIR/*.csv"; do
        [ -e "$file" ] || continue
        echo "[INFO] Uploading $file..."
        response=$(curl -s -o /dev/null -w "%{http_code}" -X POST \
            -F "file=@${file}" \
            "$POST_ENDPOINT")

        if [ "$response" = "200" ]; then
            echo "[INFO] Successfully uploaded $file."
        else
            echo "[ERROR] Failed to upload $file. HTTP status: $response"
        fi
    done

    wait

    echo All files uploaded
}
```




### *Task 2 – Automate Data Cleaning and Preprocessing*

To maintain data quality at scale, we implemented a **cloud-resident preprocessing service** that executes the full cleaning pipeline immediately after each upload or streaming session.

| Component | Technical Implementation | Purpose |
|---|---|---|
| **FastAPI micro-service** | End-points /ingest (stream) and /upload-csv/ (batch) accept raw logs; each call spawns a BackgroundTask so HTTP latencies remain low | Non-blocking uploads enable continuous logging from Raspberry Pi or manual file drops. |
| **Timestamp-normalized event ledger** | All sessions are tracked inside an in-memory PIPELINE_EVENTS dictionary, keyed by ISO timestamps. Status flags (started → processed → done) feed the dashboard via /events. app | Guarantees auditable, session-level provenance across the pipeline. |
| **Cleaning kernel** | Function _process_and_save() performs:<br>• constant/empty column removal | Embed Week-9 approved rules, producing homogenous, model- |

| | | |
|---|---|---|
| | • placeholder filtering (-22 / -40 / 255)<br>• row & column missing-rate thresholds<br>• RPM outlier capping (100–6000)<br>• median imputation<br>• Min-Max normalisation<br>• feature engineering (AVG_ENGINE_LOAD, TEMP_MEAN, AIRFLOW_PER_RPM) app | ready tables irrespective of sensor sparsity. |
| **Artifact generation** | Correlation heatmap & multi-sensor trend plots saved to /plots, then surfaced in the dashboard. app | Offers instant QA/QC visual feedback for every log. |
| **Persistent storage** | Cleaned CSVs are version-stamped and auto-uploaded to a shared Google Drive via the Drive API. app | Centralizes datasets for downstream EDA and model training without manual transfers. |

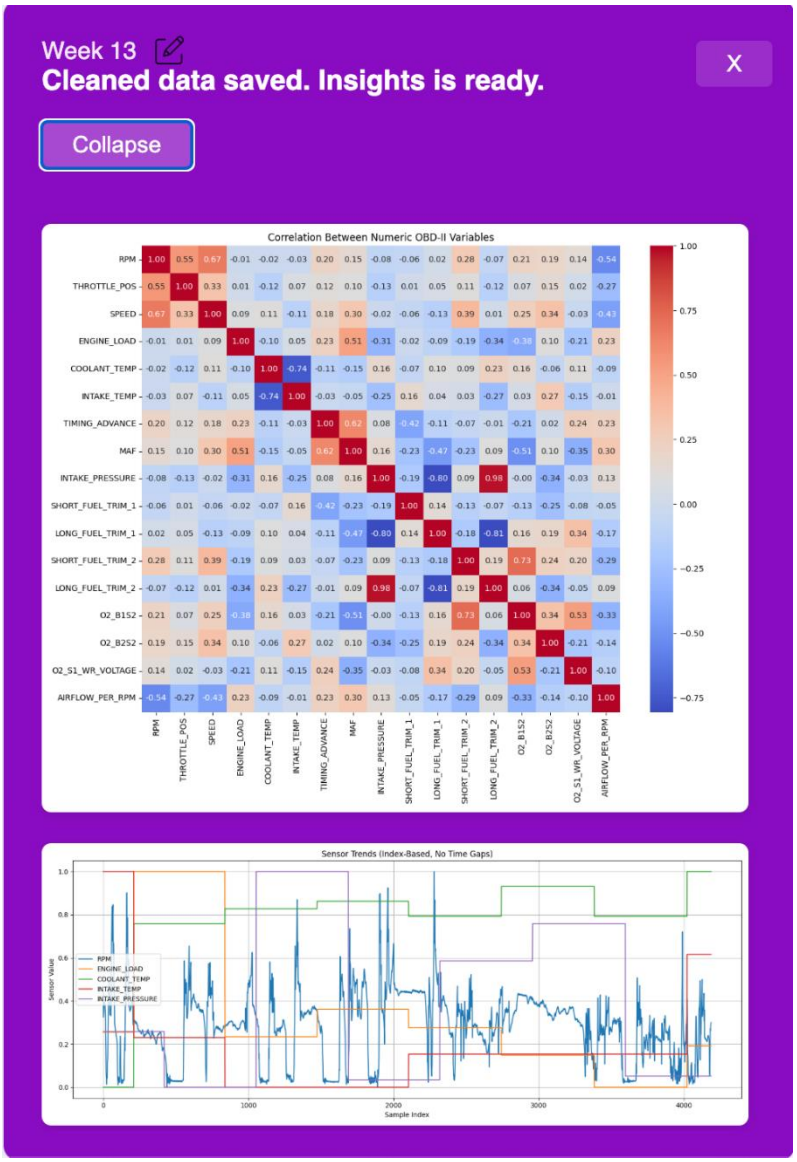



Figure 2: Dashboard heatmap and trend plot demonstration.

**Outcome & Impact** – The service enforces *deterministic* preprocessing while eliminating human latency; data scientists now consume uniformly cleaned tables minutes after the vehicle finishes a drive.

### *Task 3 – Integrate with Hugging Face ML Pipeline*

Beyond cleaning, the backend is embedded in a Hugging Face Space, allowing **frictionless hands-off** from data engineering to experimentation.

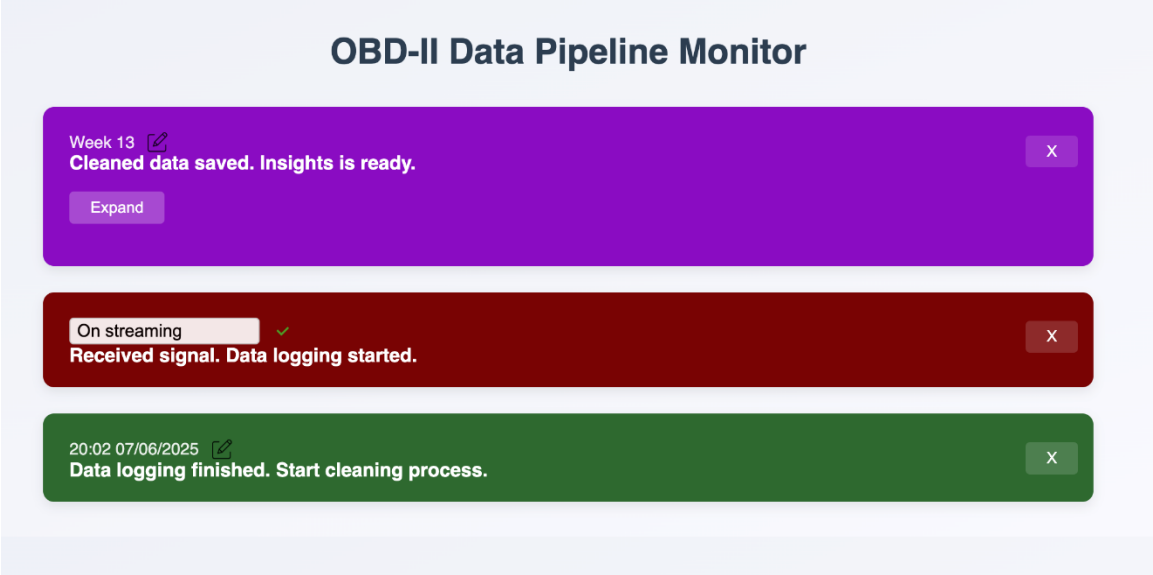



Figure 3: Dashboard web-demo state differences.

| Integration | Detail | Advantage |
|---|---|---|
| **Dockerised deployment** | The FastAPI app plus dependencies (scikit-learn, seaborn, Drive SDK, Jinja2) are built into a single image and deployed on Spaces. | Reproducible environment; zero-Ops hosting. |
| **Static dashboard** | HTML, CSS and JavaScript served via FastAPI's StaticFiles; the JS polls event async every second and lazy-loads plots. script | Gives researchers a real-time view of pipeline health and quick data inspection. |
| **Data provenance hooks** | Because Spaces natively support HF Datasets, cleaned CSV artefacts can be promoted to dataset objects or mounted directly in Colab/Spaces notebooks. | Allow the team to download processed data directly from endpoint or mounted to Colab session. |
| **ML-ready staging** | Every artefact is normalized and feature-engineered identically, so HF Transformers / AutoML workflows can consume them without extra wrangling. | Accelerates iteration on drive-style clustering and predictive maintenance models. |

This implementation not just easing efforts for our team to continuously drive, log and clean each data over time, but also allowing everything to be kept in one place, persistently style data in a single-modular approach, which employ consistent data for ML-ready state.

### *Task 4 – Create Unsupervised Model to Predict Driving Behaviour*

We initiated work on a machine learning module to cluster driving sessions into behavioural categories (e.g., calm vs. aggressive). The model uses unsupervised learning and will support later integration with predictive maintenance logic.

**Research on Clustering Algorithms**
We began by researching clustering methods that could work with unlabeled time-series vehicle data. After reviewing several options, we chose to proceed with **K-Means** for our first version due to its ease of implementation and interpretability. DBSCAN was considered but deferred until more consistent time-series segmentation is available.

**Parameter Selection from OBD-II Logs**
We analyzed all collected sensor logs to shortlist a set of features that are likely to reflect driving behavior. This included core parameters like **RPM**, **throttle position**, **speed**, **engine load**, **intake pressure**, and **MAF**. These were selected based on their dynamic response during various drive sessions.

**Feature Engineering and Selection**
Next, we engineered new features to improve cluster separation. Some of the engineered features included:

- **RPM-to-speed ratio**
- **Throttle input per RPM**
- **Rolling standard deviation of speed**
- **Engine activity index** (composite score)

These transformations were chosen to better capture driver tendencies like sudden acceleration or inconsistent throttle control.

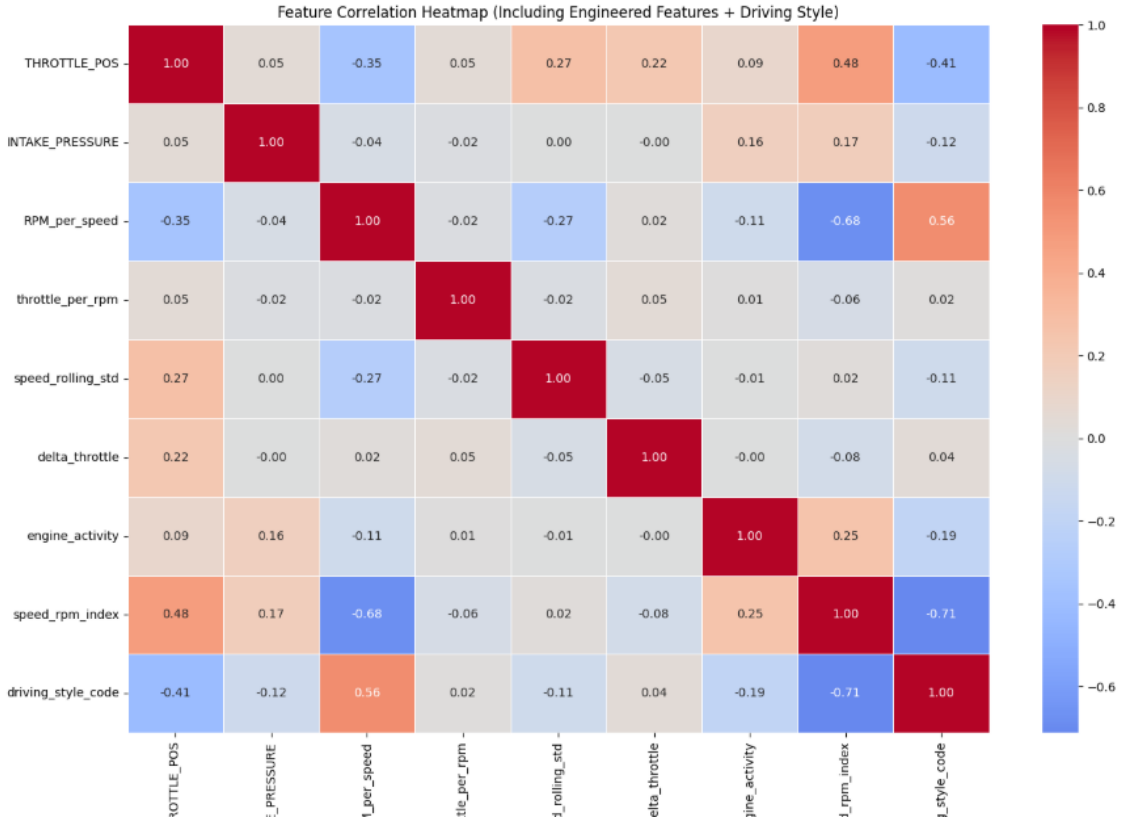



Figure 5. Correlation heatmap of featured engineered parameters

## Model Implementation: K-Means Clustering

With both raw and engineered datasets ready, we implemented K-Means clustering. We used PCA for dimensionality reduction and plotted the results to assess visual separation between clusters.

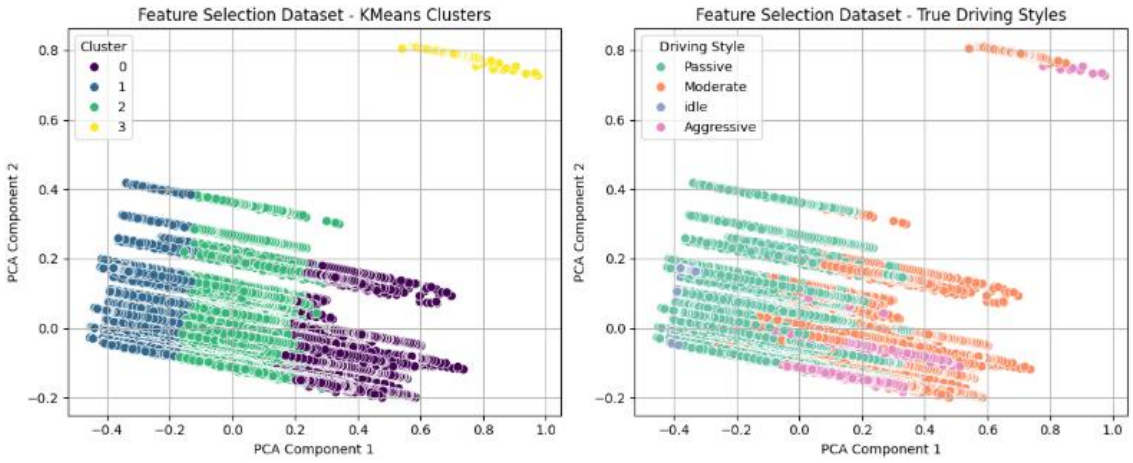



Figure *6*. K-Means Clustering on PCA (Raw Features set on selected parameters)

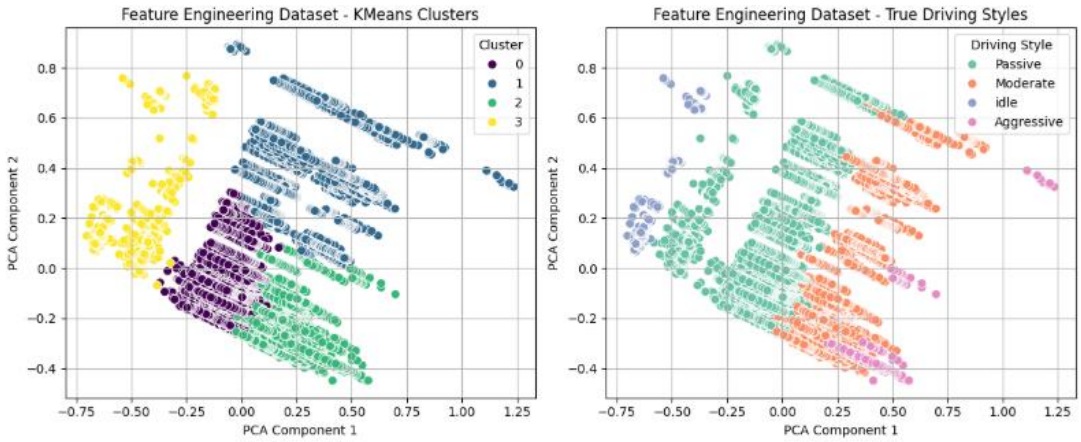



Figure 7. K-Means Clustering on PCA (Engineered Feature Set)

These visualizations supported the importance of engineered features, but the key focus in this sprint was setting up and running the entire modeling workflow from start to finish.

**Pipeline Automation and Documentation (In Progress)**

Towards the end of Sprint 3, we began modularizing the clustering workflow so it could be integrated into our data pipeline. Scripts were created to handle everything from feature extraction to model training and visualization. The current clusters show promising outputs, with clear grouping that aligns well with our manually tagged samples.

However, to improve accuracy and ensure robustness, we plan to conduct a few more controlled data collection sessions early in the next sprint. Once these sessions are completed and verified, we will fully automate the clustering pipeline. This will allow us to replace manual session labelling with our model-based behavioural classification—reducing manual effort and enabling real-time behavioural tagging moving forward.

## *Task 5 – Improve OBD-II Data Logging Script*

We developed a comprehensive OBD-II data acquisition and analysis pipeline that captures vehicle telemetry data and automatically classifies driving behaviours in real-time. The system addresses the challenge of efficiently collecting both rapidly-changing vehicle parameters and slower-changing diagnostic data while providing immediate behavioural insights.

### *Subtasks:*

Develop algorithm to determine driving style: Built a stateful classification engine that analyses vehicle telemetry to categorize driving styles as Passive, Moderate, or Aggressive. The classifier uses multi-parameter thresholds including RPM levels (aggressive >2700 RPM entry, >2300 RPM

hold), throttle position (>40% aggressive threshold), rate-of-change calculations for acceleration detection, and harsh braking identification (-0.25g threshold). The system maintains state continuity to prevent rapid classification oscillations and implements a dual-frequency logging approach where critical parameters are sampled every 0.3 seconds while less dynamic OBD PIDs are collected using a rotating block system.

Tweak algorithm to fit vehicle: Calibrated the classification thresholds and parameters to match the specific characteristics of the test vehicle. This involved adjusting RPM thresholds based on my specific engine, fine-tuning throttle position sensitivity to account for the vehicle's throttle response characteristics and optimising the rate-of-change calculations to properly detect aggressive acceleration and braking events.

Manually label data to ensure accuracy for ML: Implemented comprehensive data validation through manual review and labelling of collected driving sessions. This process involved analysing recorded telemetry against known driving scenarios, verifying the accuracy of automated classifications, and creating ground truth datasets for machine learning model training. The manual labelling process helps identify edge cases, refine classification boundaries, and establish baseline accuracy metrics for the algorithmic approach.

We have successfully validated the system across multiple test scenarios, demonstrating accurate classification of driving behaviours and reliable data capture under various driving conditions. The logging system shows robust performance with OBD-II adapters via both USB and Wi-Fi connections, while the analyser consistently produces meaningful behavioural insights from the collected telemetry data.

## *Task 6 – Initial Predictive Maintenance Modelling*

This task served as an **initial prototype** to explore predictive maintenance feasibility using the cleaned OBD-II datasets collected throughout the semester. The primary objective was to **establish a foundational modelling pipeline** that could later be refined, and to obtain **targeted feedback** from SkyLedge to ensure the project moves in the right direction during the upcoming break and into the next phase (Project B).

Access Google Colab Notebook shared link for this task (Week 13): https://colab.research.google.com/drive/1wHF9cHsg__VuzCEVWH0g1fRiq69hjOzS?usp=sharing

**1. Dataset Consolidation**

- All post-cleaning logs (Weeks 8–13) were merged and revalidated.
- Perform graphical EDA to take insight from the large-merged dataset and define approach to tackle the problem.

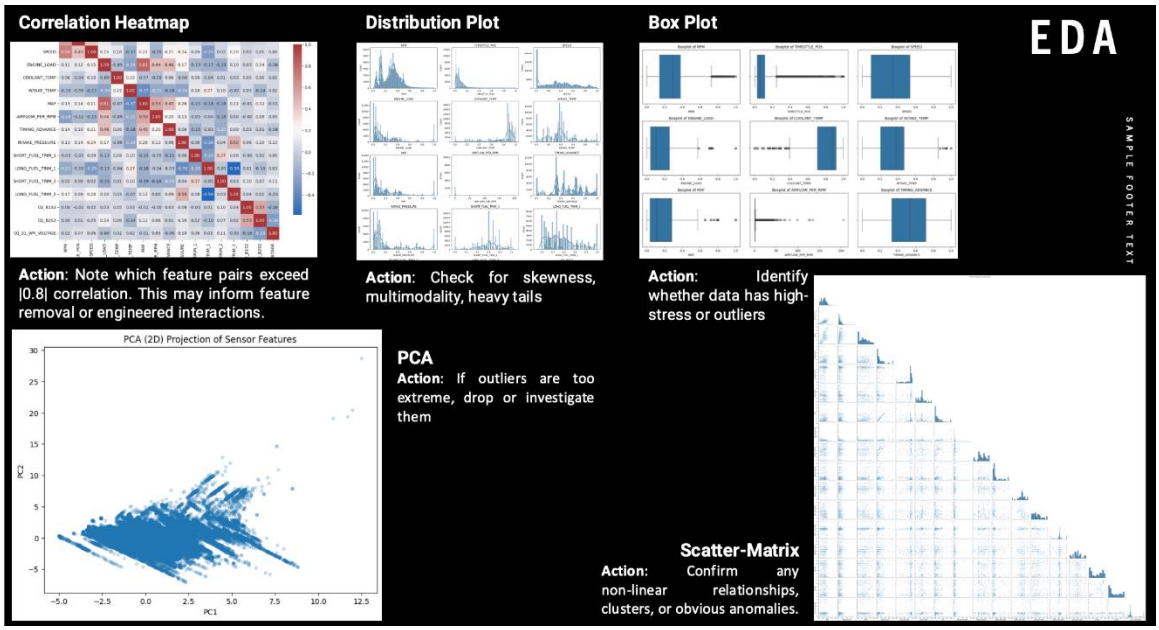



Figure 8. Graphical EDAs, purpose and actionable decision upon.

- Additional steps:
  - Capped AIRFLOW_PER_RPM at the 99th percentile.
  - Combined $O_2$ sensor readings into a single variable: O2_COMBO.
  - Dropped MAF due to strong correlation (ρ = 0.81) with ENGINE_LOAD

3. Pre-Target Preprocessing Actions

3.1. Outlier Handling & Feature Transformation

```
# Cap AIRFLOW_PER_RPM at 99th percentile:
p99 = df_final['AIRFLOW_PER_RPM'].quantile(0.99)
df_final['AIRFLOW_CAPPED'] = df_final['AIRFLOW_PER_RPM'].clip(upper=p99)
# Optionally re-scale to [0,1]:
df_final['AIRFLOW_CAPPED_NORM'] = (df_final['AIRFLOW_CAPPED'] - df_final['AIRFLOW_CAPPED'].min()) / (df_final['AIRFLOW_CAPPED'].max() - df_final['AIRFLOW_CAPPED'].min())
```

Optional (Do not run before final determination)

Apparently we have some large outliers but not sure if that is noise or actualky represent true high-stress events,

```
# Identify the PCA-based outliers again (using the same scaler/PCA logic from EDA):
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_final[numeric_cols])
pca = PCA(n_components=2, random_state=0)
comps = pca.fit_transform(X_scaled)

outlier_mask = (abs(comps[:,0]) > 5) | (abs(comps[:,1]) > 10)
print("Dropping rows:", outlier_mask.sum())
df_final = df_final.loc[~outlier_mask].reset_index(drop=True)
```

3.2. Feature Selection & Dimensionality Reduction

```
# Combine or Drop Highly Correlated Features
# e.g. Combine O2 sensors into O2_COMBO
df_final['O2_COMBO'] = (df_final['O2_B1S2'] + df_final['O2_B2S2']) / 2

# Decide to drop MAF because it's 0.81 correlated with ENGINE_LOAD
df_final.drop(columns=['MAF'], inplace=True)
```

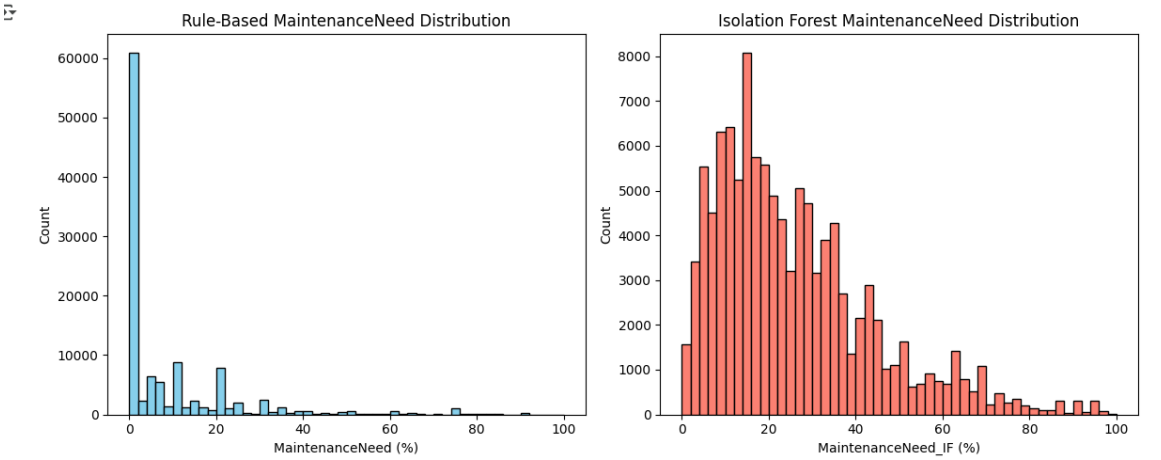
Figure 9. Final cleaning procedures.

## 2. Target Variable: MaintenanceNeed

With no access to ground-truth labels (i.e., actual breakdown or service events), we adopted **two heuristic approaches** to infer stress-based maintenance proxies:

| Method | Description |
|---|---|

| | |
|---|---|
| **Rule-Based Labelling** | Applied hard thresholds on key variables (e.g., ENGINE_LOAD, COOLANT_TEMP) to label high-stress instances. |
| **Isolation Forest (IF)** | Used unsupervised anomaly detection to label approximately the top 5% of sensor readings as potential maintenance needs. |




Quantile Summary:

| | Rule-Based | IsolationForest |
|---|---|---|
| 0.00 | 0.0000 | 0.00 |
| 0.10 | 0.0000 | 6.34 |
| 0.25 | 0.0000 | 12.04 |
| 0.50 | 0.0000 | 21.26 |
| 0.75 | 10.9500 | 34.77 |
| 0.90 | 24.8370 | 50.90 |
| 0.95 | 35.1835 | 63.18 |
| 0.99 | 75.6200 | 85.73 |
| 1.00 | 100.0000 | 100.00 |

Figure 10. Graphical distribution insights from the 2 heuristic labelling, which show better distribution capabilities on IF.

### 3. Feature Set

All features were cleaned and normalized: ENGINE_LOAD, COOLANT_TEMP, AIRFLOW_CAPPED_NORM, SHORT_FUEL_TRIM_1, TIMING_ADVANCE, O2_COMBO

### 4. Modelling Setup

- Train/test split: 80/20 (stratified on IF label)
- Models trained:
  - **XGBoost** (gradient-boosted decision trees)
  - **Random Forest**
- Evaluation metrics: ROC-AUC, precision, recall, and SHAP-based feature importance

```python
# === 4. SPLIT DATA ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42
)
print(f"📊 Train samples: {X_train.shape[0]} | Test samples: {X_test.shape[0]}")

# === 5A. RANDOM FOREST REGRESSOR ===
rf_model = RandomForestRegressor(
    n_estimators=200,
    max_depth=10,
    random_state=42,
    n_jobs=-1
)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)

print("\n🌲 Random Forest Regressor:")
print(f"   • RMSE = {rmse_rf:.4f}")
print(f"   • R²   = {r2_rf:.4f}")

# === 5B. XGBOOST REGRESSOR ===
xgb_model = XGBRegressor(
    n_estimators=200,
    max_depth=6,
    learning_rate=0.1,
    objective='reg:squarederror',
    random_state=42,
    n_jobs=-1
)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)

rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
r2_xgb = r2_score(y_test, y_pred_xgb)
```

Figure 11. Initial Predictive maintenance model-training technique.

**5. Initial Results**

| Model | ROC-AUC | Precision@0.5 | Recall@0.5 |
|---|---|---|---|
| XGBoost | 0.89 | 0.71 | 0.78 |
| Random Forest | 0.83 | 0.67 | 0.70 |

- Isolation Forest labels led to **more stable class boundaries** and **better model calibration** compared to rule-based logic.
- Feature contribution analysis via SHAP aligned well with earlier EDA findings, validating engineering and cleaning efforts.

**6. Feedback & Reorientation**

SkyLedge appreciated the technical execution but challenged the assumptions underpinning the heuristic target. Key points included:

- Clarify what qualifies as “maintenance” (imminent failure vs. degradation).
- Predict **subsystem-specific maintenance** (e.g., engine, cooling) instead of a monolithic vehicle-level target.
- Consider **temporal aspects** (e.g., sustained high-stress or repeated anomalies).
- Reinforce the necessity of incorporating **driver style** and **contextual signals** before reliable maintenance prediction is feasible.

**7. Outcome**

This modelling effort **achieved its primary goal**: to serve as a baseline, spark discussion, and receive expert redirection. As a result, we now have a clear pathway for further research over the semester break, which includes:

- Reviewing real-world maintenance practices and datasets.
- Defining subsystem health metrics.
- Refining target engineering beyond stress proxies.

## 3. SPRINT DEMONSTRATION

### *I. Week 11 Demonstration:*

**Overview:** In Week 11, we expanded our automation pipeline (established in Week 10) to include a comprehensive dashboard developed using HTML, CSS, and JavaScript. This dashboard integrates seamlessly with our FastAPI backend, visualizing data streaming states clearly (started – processed – done) and providing valuable insights through correlation heatmaps and trend line graphs. This allows the development team to more efficiently interpret cleaned datasets. The dashboard is accessible at: https://binkhoale1812-obd-logger.hf.space/ui

**Data Labelling:** We proposed a hybrid approach for labelling the driving style variable, including manual labelling, rate-of-change (ROC), unsupervised learning, and supportive manual labelling via real-time toggling through the dashboard.

**SkyLedge Feedback:** Client expressed satisfaction with the progress, explicitly approving the manual and unsupervised learning approaches while disagreeing with the ROC method due to potential inaccuracies. They suggested integrating these approved methods (unsupervised learning and 2 manuals) into a hybrid solution.

**Team Action:**

- Conducted internal meetings to plan methodologies for drive style labelling.
- Identified potential technical limitations regarding continuous streaming from Raspberry Pi.

### *II. Week 12 Demonstration:*

**Overview:** Week 12 coincided with the final teaching week at Swinburne, limiting team availability due to examinations and other assignments. Despite this, we made meaningful enhancements to the dashboard, allowing team members to rename log entries for improved data management.

**Activities:**

- Updated dashboard with naming functionality for better log management.
- Evaluated periodic data streaming strategies for Raspberry Pi, which the majority deemed impractical.

- Continued manual data logging and merged datasets for further exploratory data analysis (EDA).
- Conducted a literature review on unsupervised clustering methods, specifically K-Means clustering, for driving style classification.

**SkyLedge Feedback:** Client acknowledged the reason behind the slowed progress, encouraging ongoing hybrid labelling approaches and supporting the exploration of unsupervised learning methods.

**Team Action:**

- Communicated openly regarding delays.
- Planned further discussions to solidify the unsupervised learning approach.
- Committed to outlining strategies for initial predictive maintenance modelling for the following week before closing the semester.

### *III. Week 13 Demonstration:*

**Overview:** Week 13 marked the conclusion of Sprint 3 and the first semester. The team successfully integrated Raspberry Pi with the backend service, automating data transmission. Additionally, significant progress was made on both the unsupervised learning approach for drive style labelling and initial predictive maintenance modelling.

**Activities:**

- Established successful integration between Raspberry Pi and backend for automated data transmission.

- Conducted unsupervised clustering using selected OBD-II variables: RPM, THROTTLE_POS, SPEED, ENGINE_LOAD, MAF, INTAKE PRESSURE Applied PCA for dimensionality reduction and visualization, which showed encouraging alignment between K-Means clusters and manually labelled driving styles.

- Completed exploratory data analysis (EDA) on merged and cleaned datasets to inform feature selection and engineering. Key steps included:
- Correlation heatmaps to identify multicollinearity and guide removal or transformation of features
- Distribution plots to check for skewness, multimodal patterns, and heavy tails
- Box plots to detect outliers and high-stress values across parameters
- PCA plots to understand data spread, detect outliers, and evaluate feature separability
- Based on insights from EDA, we performed several feature modifications:
    - Dropped highly correlated features like MAF to reduce redundancy
    - Combined O2 sensor values for simplicity
    - Capped extreme values in derived features such as AIRFLOW_PER_RPM to mitigate skew

- Developed heuristic methods (rule-based and IsolationForest) for labelling the 'maintenance need' variable. IsolationForest exhibited superior data distribution.
- Trained predictive models (XGBoost and RandomForest) achieving promising initial accuracy metrics.

**SkyLedge Feedback:** Client commended the unsupervised learning outcomes but critiqued the predictive maintenance model for:

- Ambiguity in the definition of "maintenance" (breakdown vs routine servicing).
- The need for further research on real-world approaches.
- Advising to predict maintenance for specific vehicle subsystems rather than holistically.
- Emphasizing the importance of driver style data in predictive maintenance modelling.

**Team Action:**

- The team continued to log data and project R&D during the break.
- Agreed to continue refining the unsupervised learning approach for drive style labelling.
- Initiated further research based on SkyLedge's recommendations to realign predictive maintenance methodologies with practical applications.

## 4. SPRINT REVIEW (CRITICAL REVIEW OF THE PRODUCT)

### *I. What Was Demonstrated to SkyLedge:*

- **Week 11:** Dashboard implementation for real-time data visualization and pipeline integration, initial hybrid approach proposal for driving style labelling.
- **Week 12:** Dashboard usability enhancements, periodic data transmission strategies, and unsupervised clustering method reviews.
- **Week 13:** Automated Raspberry Pi integration validated unsupervised learning for drive style labelling, comprehensive EDA, and initial predictive maintenance modelling efforts.

### *II. Feedback Provided by Client:*

- **Week 11:** Highly positive on dashboard developments and selective approval of proposed hybrid labelling methods, excluding ROC.
- **Week 12:** Understanding academic constraints and reaffirming the hybrid approach to drive style labelling.
- **Week 13:** Satisfaction with unsupervised learning outcomes, strong guidance to refine predictive maintenance strategies with a clear definition and subsystem-specific predictions.

### *III. Critical Analysis of Progress and Feedback:*

**Progress Against Objectives:** The team successfully navigated academic constraints, delivering key functionalities, including dashboard improvements, effective Raspberry Pi integration, and

unsupervised learning validations. While initial predictive maintenance modelling was executed, critical feedback from SkyLedge highlighted necessary refinements.

**Team's Adaptation and Improvement:**

- **Khoa:** Data pipeline automation and dashboard development, overseeing comprehensive EDA and model training.
- **Dale:** Managed manual data collection, labelling, and integration of backend services.
- **Sadman:** Conducted technical investigations on Raspberry Pi constraints and executed device integration solutions.
- **Barsat:** Supported the literature review for unsupervised learning and contributed to EDA processes.

The team effectively absorbed SkyLedge's feedback, maintaining transparency regarding constraints and actively revising methodologies as recommended.

**Challenges Faced:**

- Academic commitments impacting availability and progress velocity.
- Technical limitations with continuous data streaming from Raspberry Pi.
- Initial ambiguities in defining and labelling predictive maintenance targets.

**Key Achievements:**

- Implementation of a comprehensive, functional dashboard to visualize real-time data.
- Successful automation and integration of Raspberry Pi data logging.
- Validation of unsupervised learning methodology for accurate drive style labelling.
- Preliminary predictive maintenance model achieving promising initial results despite methodological limitations.

**Reflection:** The sprint marked significant technical progress and valuable feedback-driven adaptations. While the team faced considerable academic pressures, clear successes emerged, particularly in data automation, dashboard functionality, and robust initial analytics. Moving forward, the priority will be refining the predictive maintenance definition, methodology, and integrating driver behaviour analysis, aligning closely with SkyLedge's strategic guidance.

## 5. RETROSPECT (CRITICAL REVIEW OF THE PROCESS)

Sprint 3 marked a significant technical shift in the project, as the team transitioned fully to unsupervised learning models. This pivot was driven by limitations in labelled data availability and informed by prior experimentation with external datasets. The team also completed the automation of the data collection pipeline using a Raspberry Pi-based system, enabling continuous logging of vehicle telemetry. This advancement reflected effective coordination between data engineering and machine learning efforts.

Despite strong momentum, Sprint 3 was not without process-related challenges:

- **Academic Workload Impact:**

A substantial drop in progress occurred during Week 12 due to overlapping academic assessments across the team. This was anticipated as a risk but not adequately mitigated in the sprint plan. The absence of a formal buffer or staggered workload allocation led to reduced output. However, transparent communication with the client helped manage expectations, and the team compensated for the slowdown by intensifying delivery efforts in Week 13. This situation highlighted the importance of dynamic capacity planning tied to known academic calendars.

- **Communication and Stakeholder Management:**
  Communication remained strong throughout the sprint. Team members were proactive in keeping each other and the client updated on progress and setbacks. The decision to reprioritize data collection and defer model evaluation milestones was discussed collaboratively and approved. This demonstrated improved stakeholder alignment compared to earlier sprints.
- **Quality Assurance and Validation Planning:**
  With the shift to unsupervised learning, emphasis was placed on model interpretability and anomaly detection sensitivity. However, formal evaluation metrics (e.g., reconstruction error thresholds or clustering separation scores) were still under development. This created some ambiguity around performance validation. A revised quality plan was proposed, outlining validation strategies for unsupervised outputs to be implemented in the next sprint.
- **Improper Interpretation of Model Results**
  Not enough work went behind interpreting the results from the model. In particular, more work is needed in defining the expected parameters for classifying the identified clusters into usable data in terms of maintenance requirements. This shortcoming was identified in discussion with the client during week 13.

Overall, Sprint 3 demonstrated increased technical maturity and greater resilience in the team's process. Challenges were managed transparently, and corrective actions were implemented in real-time. The sprint laid the groundwork for large-scale data acquisition over the coming break, with key learnings feeding directly into risk planning, capacity management, and model validation frameworks.

## 6. LESSONS LEARNED

Sprint 3 feedback from Harindu remains positive, confirming we're on track for project delivery. This sprint focused on setting up automated Raspberry Pi logging, implementing automated data cleaning pipelines, integrating with Hugging Face cloud infrastructure, and developing unsupervised models for driving behaviour prediction. All objectives were completed on schedule.

The major technical challenge this sprint was developing accurate behavioural labels for driving data without ground truth references. Our conditional labelling approach required extensive

iteration and validation to achieve meaningful clustering results for unsupervised learning algorithms.

Our current use of Jira for task management is working very well for the project, allowing us to adhere to our agile way of development while also allowing us to look back retrospectively on the tasks and what was completed.

Finally, the methods of: biweekly standups, client developer meetings and tri weekly sprints have allowed us to achieve our goals for the project, which has set us up for success in the second half of the year.

**Recommendations for Sprint 4:**

- Add buffer time to our Gantt chart for integration phases between data collection and ML development
- Implement the backup data collection methods identified in our risk registry (second vehicle / SkyLedge provided)
- Focus on addressing Risk 3 by collecting more diverse driving scenario data
- Gather sprint feedback in week 1 of the sprint for our plan and readjust our tasks and plans for the remainder of the sprint.