

Sprint Two Report

PORTFOLIO TASK 6

Unit code: EAT40006

Unit Name: Engineering Technology Project B

Submission date: 21 / 09 / 2025

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.

CONTRIBUTION SUMMARY SPRINT 2

All team members should complete the following table together. You can provide a yes/no response or a brief comment where appropriate.

	Contribution	Initiatives	Communication					Respect
Team Member	Finished tasks in time or on time with an acceptable quality	Self-assigned tasks and proactively found solutions to problems	Attended all team meetings	Attended all supervisor meetings	Attended all client meetings	Replied every time within an acceptable timeframe	Kept the team updated about the status	The student respected others and listened to different opinions
Dang Khoa Le	Always deliver outcomes in advance. Deliver solutions aligned with scope and client requirements.	Contributed to UL classifier and ML modelling. Support on dashboard's backend integration and visual.	Yes	Yes	Yes	Yes	Align role distribution and KPI govern as informed from the supervisor.	Yes
Mohammed Barsat Zulkarnine	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dale Bent	Yes	Yes	Yes	No, missed week 5	No, missed week 5	Yes	Yes	Yes
Sadman Tariq	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

CONTRIBUTION DETAILS SPRINT 2

Dale:

Completed numerous logging sessions, manually labelling driving behaviour, fuel usage, efficiency and distance. Adjusted the logging script to focus on fuel related usage, using test scripts to determine highest possible frequency of logging from the OBD. Worked collaboratively with team members to determine what the final product would look like, how we get there and determining responsibilities to do so.

obd_data_log_20250909_120005.csv			9 Sep 2025 at 10:06 am	34 KB	Compressed
obd_data_log_20250909_115223.csv			9 Sep 2025 at 9:59 am	43 KB	Compressed
obd_data_log_20250909_113008.csv			9 Sep 2025 at 9:51am	41 KB	Compressed
obd_data_log_20250909_114216.csv			9 Sep 2025 at 9:51am	38 KB	Compressed
obd_data_log_20250904_151217.csv			4 Sep 2025 at 1:34 pm	53 KB	Compressed
obd_data_log_20250904_150224.csv			4 Sep 2025 at 1:11pm	43 KB	Compressed
obd_data_log_20250904_145425.csv			4 Sep 2025 at 1:01pm	47 KB	Compressed
obd_data_log_20250904_144555.csv			4 Sep 2025 at 12:53pm	57 KB	Compressed
obd_data_log_20250904_143934.csv			4 Sep 2025 at 12:45pm	34 KB	Compressed
obd_data_log_20250904_143106.csv			4 Sep 2025 at 12:38pm	52 KB	Compressed
obd_data_log_20250904_142045.csv			4 Sep 2025 at 12:28pm	40 KB	Compressed
obd_data_log_20250904_140142.csv			4 Sep 2025 at 12:19pm	45 KB	Compressed
obd_data_log_20250904_141003.csv			4 Sep 2025 at 12:17pm	32 KB	Compressed
obd_data_log_20250904_135245.csv			4 Sep 2025 at 12:00pm	45 KB	Compressed
obd_data_log_20250904_134514.csv			4 Sep 2025 at 11:51am	38 KB	Compressed
obd_data_log_20250831_121946.csv			31 Aug 2025 at 10:38am	174 KB	Compressed
obd_data_log_20250828_131339.csv			28 Aug 2025 at 11:38am	219 KB	Compressed
obd_data_log_20250828_125858.csv			28 Aug 2025 at 11:08am	70 KB	Compressed
obd_data_log_20250828_121852.csv			28 Aug 2025 at 10:34am	90 KB	Compressed
obd_data_log_20250827_174710.csv			27 Aug 2025 at 4:12pm	225 KB	Compressed
obd_data_log_20250827_170836.csv			27 Aug 2025 at 3:39pm	277 KB	Compressed
obd_data_log_20250827_155530.csv			27 Aug 2025 at 2:09pm	81 KB	Compressed
obd_data_log_20250826_113722.csv			26 Aug 2025 at 9:47am	89 KB	Compressed
obd_data_log_20250826_112905.csv			26 Aug 2025 at 9:37am	75 KB	Compressed
obd_data_log_20250826_112051.csv			26 Aug 2025 at 9:29am	73 KB	Compressed
obd_data_log_20250826_111709.csv			26 Aug 2025 at 9:20am	33 KB	Compressed
obd_data_log_20250826_111259.csv			26 Aug 2025 at 9:17am	37 KB	Compressed
obd_data_log_20250826_10853.csv			26 Aug 2025 at 9:12am	37 KB	Compressed
obd_data_log_20250826_110400.csv			26 Aug 2025 at 9:08am	44 KB	Compressed
obd_data_log_20250826_105457.csv			26 Aug 2025 at 9:03am	82 KB	Compressed

Figure. Labelled log attempts

timestamp	RPM	SPEED	THROTTLE_P1_MAF	ENGINE_LOI	INTAKE_PRES	SHORT_FUEL	LONG_FUEL	driving_style	Fuel_consumption	Fuel_Efficiency	Route	Distance
2025-09-09T	740	0	12.1568628	3.42	1.56862745	26	0.78125	0.78125	16.40625	2.34375	idle	0.336
2025-09-09T	753.5	0	12.1568628	3.36	1.56862745	26	3.90625	5.46875	15.625	1.5625	idle	0.336
2025-09-09T	971.75	0	18.0392157	15.84	1.56862745	26	3.90625	5.46875	15.625	1.5625	idle	0.336
2025-09-09T	971.75	0	18.0392157	15.84	1.56862745	26	3.90625	5.46875	15.625	1.5625	idle	0.336
2025-09-09T	1371.5	13	23.9215686	31.9	16.4705882	86	3.90625	5.46875	15.625	1.5625	idle	0.336
2025-09-09T	1371.5	13	23.9215686	31.9	16.4705882	86	3.90625	5.46875	15.625	1.5625	idle	0.336
2025-09-09T	3401	30	30.5882353	65.66	16.4705882	86	3.90625	5.46875	15.625	1.5625	Aggressive	0.336
2025-09-09T	3401	30	30.5882353	65.66	16.4705882	86	3.90625	5.46875	15.625	1.5625	Aggressive	0.336
2025-09-09T	2739.75	36	25.88223529	50.26	27.8431373	85	3.90625	5.46875	15.625	1.5625	Aggressive	0.336
2025-09-09T	2739.75	36	25.88223529	50.26	27.8431373	85	3.90625	5.46875	15.625	1.5625	Aggressive	0.336
2025-09-09T	3643	49	25.0980392	35.32	27.8431373	85	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	3643	49	25.0980392	35.32	27.8431373	85	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	3003.25	54	34.5098039	63.97	32.5490196	91	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	3003.25	54	34.5098039	63.97	32.5490196	91	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	3448.5	62	16.8627451	14.28	32.5490196	91	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	3448.5	62	16.8627451	14.28	32.5490196	91	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	2573.5	59	14.9019608	10.41	5.09803922	21	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	2573.5	59	14.9019608	10.41	5.09803922	21	-0.78125	0	4.6875	3.125	Aggressive	0.336
2025-09-09T	2337.75	54	14.5098039	9.31	5.09803922	21	0	0	2.34375	1.5625	Aggressive	0.336
2025-09-09T	2337.75	54	14.5098039	9.31	5.09803922	21	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2170	50	18.0392157	18.65	11.7647059	50	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2170	50	18.0392157	18.65	11.7647059	50	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2194.25	51	20.3921569	27.02	11.7647059	50	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2194.25	51	20.3921569	27.02	11.7647059	50	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2210.75	51	15.2941177	11.1	5.49019608	26	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2210.75	51	15.2941177	11.1	5.49019608	26	0	0	2.34375	1.5625	Passive	0.336
2025-09-09T	2153	50	16.4705882	13.92	5.49019608	26	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2116	49	18.8235294	22.36	9.80392157	42	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2116	49	18.8235294	22.36	9.80392157	42	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2136	49	15.6862745	11.65	9.80392157	42	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2136	49	15.6862745	11.65	9.80392157	42	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2051.75	48	14.9019608	9.9	4.70588235	24	3.125	0.78125	2.34375	0.78125	Passive	0.336
2025-09-09T	2051.75	48	14.9019608	9.9	4.70588235	24	3.125	0.78125	2.34375	0.78125	Passive	0.336

Figure. Labelled log details

Khoa:

Stage 1: Feature Selection & Idle Detection (Conservative Gate + BGMM)

- **Feature Selection:**

Used mutual information ranking to select up to 24 most informative features relative to ground-truth labels. Applied RobustScaler to normalise features for stability.

```
# Drop any all-NaN cols (should be none after KNN)
X_raw = fe.drop(columns=[label_col] + ([timestamp] if "timestamp" in fe.columns else []))
y = fe[label_col].astype(str).values

# Mutual info to select the most informative features wrt labels
valid_cols = X_raw.columns[X_raw.notna().sum() > 0]
X_valid = X_raw[valid_cols].fillna(X_raw[valid_cols].median())

mi = mutual_info_classif(X_valid, y, discrete_features=False, random_state=42)
mi_ser = pd.Series(mi, index=valid_cols).sort_values(ascending=False)

TOP_K = min(24, (mi_ser > 0).sum() or 24) # up to 24, or all positive-MI features
selected_features = mi_ser.head(TOP_K).index.tolist()
print(f"Stage-1: Selected {len(selected_features)} features:")
print(selected_features[:12], "...")

# Scale selected features robustly
scaler = RobustScaler()
X = scaler.fit_transform(X_valid[selected_features])

# Persist selections for later cells
selected_features_, scaler_ = selected_features, scaler
```

Figure. Feature importance ranking (MI scores) and top-selected feature list

- Idle Gating & Mixture Confirmation.
- Applied multi-cue gating across SPEED, THROTTLE_POS, ENGINE_LOAD, ACCEL_std, RPM_std, and MAF.
- Only samples satisfying all “low-movement” gates considered Idle candidates.

```
# --- low-movement cues (prefer *_mean_w5 if available) ---
def pick(*names, default=None):
    for n in names:
        if n in fe.columns: return fe[n]
    return pd.Series(default, index=fe.index, dtype=float)

speed_mean = pick("SPEED_mean_w5", "SPEED", default=0.0)
thr_mean = pick("THROTTLE_POS_mean_w5", "THROTTLE_POS", default=0.0)
eng_mean = pick("ENGINE_LOAD_mean_w5", "ENGINE_LOAD", default=0.0)
acc_std = pick("ACCEL_std_w5", "ACCEL_std_w2", "ACCEL_std_w1", default=0.0)
rpm_std = pick("RPM_std_w5", "RPM", default=0.0).rolling(5,1).std()
maf_mean = pick("MAF_mean_w5", "MAF", default=0.0)

# --- quantile gating to form candidates ---
s_gate = speed_mean <= speed_mean.quantile(0.15)
t_gate = thr_mean.fillna(0) <= thr_mean.quantile(0.20)
a_gate = acc_std.fillna(0) <= acc_std.quantile(0.25)
r_gate = rpm_std.fillna(0) <= rpm_std.quantile(0.25)
m_gate = maf_mean.fillna(0) <= maf_mean.quantile(0.20)
idle_candidate = (s_gate & t_gate & a_gate & r_gate & m_gate)
```

- Ran a Bayesian Gaussian Mixture (3 comps) inside the gated set; identified the lowest-speed cluster as Idle with ≥ 0.6 probability threshold.

```

# ---- BGMM refine Idle candidates ---
Z = pd.DataFrame({
    "speed": speed_mean,
    "thr": thr_mean.fillna(0),
    "eng": eng_mean.fillna(0),
    "accsd": acc_std.fillna(0),
    "rpmsd": rpm_std.fillna(0),
    "maf": maf_mean.fillna(0)
})
Zs = StandardScaler().fit_transform(Z[idle_candidate])

if Zs.shape[0] >= 50:
    bgmm = BayesianGaussianMixture(
        n_components=3, covariance_type="full",
        weight_concentration_prior=0.3, n_init=5, random_state=42
    ).fit(Zs)
    proba = np.zeros(len(fe))
    labp = bgmm.predict_proba(Zs)
    idle_comp = np.argmin(bgmm.means_[:, 0]) # smallest speed cluster
    proba[np.where(idle_candidate)[0]] = labp[:, idle_comp]
    is_idle_ml = (idle_candidate & (proba >= 0.6)).to_numpy()
else:
    is_idle_ml = idle_candidate.to_numpy()

# smooth flicker
is_idle_ml = medfilt(is_idle_ml.astype(int), kernel_size=5).astype(bool)
base_sec = _infer_base_sec_from_fe(fe)

# attach labels
if "cluster_unsup" not in fe.columns:
    fe["cluster_unsup"] = "(unknown)"
fe.loc[is_idle_ml, "cluster_unsup"] = "Idle"
fe["idle_mt"] = is_idle_ml

# add GT-with-idle if exists
GT_COL = None
for c in ["driving_style", "label", "ground_truth", "style"]:
    if c in fe.columns: GT_COL = c; break
if GT_COL:
    fe["gt_with_idle_mt"] = fe[GT_COL].astype(str)
    fe.loc[is_idle_ml, "gt_with_idle_mt"] = "Idle"

```

- Median filtering used to smooth flicker.
- Produced episode statistics (Idle fractions, duration medians, P90, mean).

```

# ---- BGMM refine Idle candidates ---
Z = pd.DataFrame({
    "speed": speed_mean,
    "thr": thr_mean.fillna(0),
    "eng": eng_mean.fillna(0),
    "accsd": acc_std.fillna(0),
    "rpmsd": rpm_std.fillna(0),
    "maf": maf_mean.fillna(0)
})
Zs = StandardScaler().fit_transform(Z[idle_candidate])

if Zs.shape[0] >= 50:
    bgmm = BayesianGaussianMixture(
        n_components=3, covariance_type="full",
        weight_concentration_prior=0.3, n_init=5, random_state=42
    ).fit(Zs)
    proba = np.zeros(len(fe))
    labp = bgmm.predict_proba(Zs)
    idle_comp = np.argmin(bgmm.means_[:, 0]) # smallest speed cluster
    proba[np.where(idle_candidate)[0]] = labp[:, idle_comp]
    is_idle_ml = (idle_candidate & (proba >= 0.6)).to_numpy()
else:
    is_idle_ml = idle_candidate.to_numpy()

# smooth flicker
is_idle_ml = medfilt(is_idle_ml.astype(int), kernel_size=5).astype(bool)
base_sec = _infer_base_sec_from_fe(fe)

# attach labels
if "cluster_unsup" not in fe.columns:
    fe["cluster_unsup"] = "(unknown)"
fe.loc[is_idle_ml, "cluster_unsup"] = "Idle"
fe["idle_mt"] = is_idle_ml

# add GT-with-idle if exists
GT_COL = None
for c in ["driving_style", "label", "ground_truth", "style"]:
    if c in fe.columns: GT_COL = c; break
if GT_COL:
    fe["gt_with_idle_mt"] = fe[GT_COL].astype(str)
    fe.loc[is_idle_ml, "gt_with_idle_mt"] = "Idle"

```

- Validation with static approach (Week 6), which consist of:
 - Domain-driven cut-offs: flagged Idle at specific thresholds → captures near-stationary engine running.
 - Low variability filter avoid false Idle when creeping.
 - Rule combination: only rows meeting all conditions simultaneously were marked Idle.
 - Stability smooth: applied short median filter, remove flickering on/off transitions.

```

def _idle_episode_stats(mask: np.ndarray, fe: pd.DataFrame, base_sec: float):
    m = mask.astype(bool)
    idx = np.flatnonzero(np.diff(np.r_[False, m, False]))
    starts, ends = idx[1:-1], idx[1:-2]
    lengths = ends - starts
    durations = lengths * base_sec
    total_time = len(m) * base_sec
    return {
        "samples_frac": float(m.mean()),
        "time_frac": float(durations.sum()/max(1e-9, total_time)),
        "episodes": int(len(lengths)),
        "dur_median_s": float(np.median(durations)) if len(durations) else 0.0,
        "dur_p90_s": float(np.percentile(durations, 90)) if len(durations) else 0.0,
        "dur_mean_s": float(np.mean(durations)) if len(durations) else 0.0,
    }

def pick(name, kind="mean"):
    if kind=="mean":
        for cand in [f"{name}_mean_w5", f"{name}_mean_w2", name]:
            if cand in fe.columns: return fe[cand]
    else:
        for cand in [f"{name}_std_w5", f"{name}_std_w2", f"{name}_std_w1", name]:
            if cand in fe.columns: return fe[cand]
    return pd.Series(np.nan, index=fe.index, dtype=float)

speed = pick("SPEED", "mean")
thr = pick("THROTTLE_POS", "mean")
eload = pick("ENGINE_LOAD", "mean")
accsd = pick("ACCEL", "std")
rpmsd = pick("RPM", "std")
maf = pick("MAF", "mean")

# ---- relaxed threshold rule ----
def gate_le(series, abs_thr, q=0.10, name=""):
    if series.isna().all():
        print(f"[rule] {name}: missing ~ gate skipped")
        return pd.Series(True, index=series.index)
    qthr = series.quantile(q)
    thr = max(abs_thr, qthr) # use the larger (less strict) of absolute floor or low quantile
    print(f"[rule] {name}: thr=(thr:.4g) (abs={abs_thr}, q{int(q*100)}={qthr:.4g})")
    return series.fillna(np.inf) <= thr

g_speed = gate_le(speed, abs_thr=5.0, q=0.15, name="speed(km/h)")
g_thr = gate_le(thr, abs_thr=5.0, q=0.15, name="throttle(%)")
g_eload = gate_le(eload, abs_thr=25.0, q=0.15, name="engine_load(%)")
g_accsd = gate_le(accsd, abs_thr=0.05, q=0.20, name="accel_std")
g_rpmsd = gate_le(rpmsd, abs_thr=10.0, q=0.20, name="rpm_std")
g_maf = gate_le(maf, abs_thr=3.0, q=0.15, name="maf(g/s)")

is_idle_rule = (g_speed & g_thr & g_eload & g_rpmsd & g_maf).to_numpy()
is_idle_rule = medfilt(is_idle_rule.astype(int), kernel_size=5).astype(bool)
fe["idle_rule"] = is_idle_rule
print(f"[rule] Idle samples detected: {is_idle_rule.sum()} / {len(is_idle_rule)} ({100*is_idle_rule.mean():.2f}%)")

```

Figure. Threshold-ruling methods to validate UL classifier

Stage 2: Driving Style Stratification (Passive / Moderate / Aggressive)

- Quantile-Gated Aggressiveness Index.
- Defined Aggressiveness index as a weighted sum of features.
- Remained samples split into **Passive vs Moderate** using **BGMM(2)** on dynamic features.
- Final sequence smoothed via median filter.

```

# === Cell 7 - Stage-2: <10% Aggressive gating + BGMM(2) for Passive vs Moderate ===
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import BayesianGaussianMixture

# Non-idle subset and dynamics features
drive_mask = ~is_idle
dyn_feats = [c for c in [
    "pos_accel_rate_w5", "neg_accel_rate_w5", "thr_change_rate_w5",
    "JERK_std_w5", "ACCEL_std_w5", "SPEED_mean_w5", "ENGINE_LOAD_mean_w5", "RPM_std_w5"
] if c in fe.columns]

Df = fe.loc[drive_mask, dyn_feats].copy().fillna(0)
sc_dyn = StandardScaler().fit(Df)
Z = sc_dyn.transform(Df)

# Aggressiveness index (unsupervised; emphasizes bursts/jerk/throttle changes)
w = np.zeros(Z.shape[1]); idx = {ni: i for i, n in enumerate(dyn_feats)}
for n, wt in [
    ("pos_accel_rate_w5", 0.30), ("neg_accel_rate_w5", 0.20),
    ("JERK_std_w5", 0.20), ("ACCEL_std_w5", 0.15),
    ("thr_change_rate_w5", 0.10), ("SPEED_mean_w5", 0.05)
]:
    if n in idx: w[idx[n]] = wt
agr_index = (Z * w).sum(axis=1)

# Gate top 10% as Aggressive
thr = np.quantile(agr_index, 0.90)
agr_local = agr_index >= thr

# Map back to full frame
agr_mask = np.zeros(len(fe), dtype=bool)
agr_mask[np.where(drive_mask)[0][agr_local]] = True
fe.loc[agr_mask, "cluster_unsup"] = "Aggressive"

# Cluster the remainder (driving & not aggressive) into Passive vs Moderate
remain_idx = np.where(drive_mask & ~agr_mask)[0]
Df2 = fe.loc[remain_idx, dyn_feats].copy().fillna(0)
Z2 = sc_dyn.transform(Df2)

gmm2 = BayesianGaussianMixture(
    n_components=2, covariance_type="full",
    weight_concentration_prior=0.5, n_init=5, random_state=42
).fit(Z2)
c2 = gmm2.predict(Z2)

# Name clusters by intensity score (same weights as above)
pm_score = (Z2 * w).sum(axis=1)
means = [pm_score[c2==k].mean() for k in [0,1]]
low_k, high_k = (0,1) if means[0] <= means[1] else (1,0)
label2 = np.where(c2==low_k, "Passive", "Moderate")
fe.loc[remain_idx, "cluster_unsup"] = label2

# Optional smoothing to reduce flicker
try:
    from scipy.signal import medfilt
    name_to_id = {"Idle":0, "Passive":1, "Moderate":2, "Aggressive":3}
    id_to_name = {v:k for k,v in name_to_id.items()}
    ids = np.array([name_to_id.get(x, -1) for x in fe["cluster_unsup"].values])
    ids = medfilt(ids, kernel_size=5)
    fe["cluster_unsup"] = np.array([id_to_name.get(i, "unknown") for i in ids], dtype=object)
except Exception:
    pass

# Report final fractions (UL)
print("Class fractions (unsupervised):")
for s in ["Idle", "Passive", "Moderate", "Aggressive"]:
    frac = 100*np.mean(fe["cluster_unsup"].values==s)
    print(f" {s}: {frac:.1f}%")

```

Figure. BGMM UL classifier workflows

→ UL produced clearer separability than GT labels (which were inter-mixed, especially in Moderate).

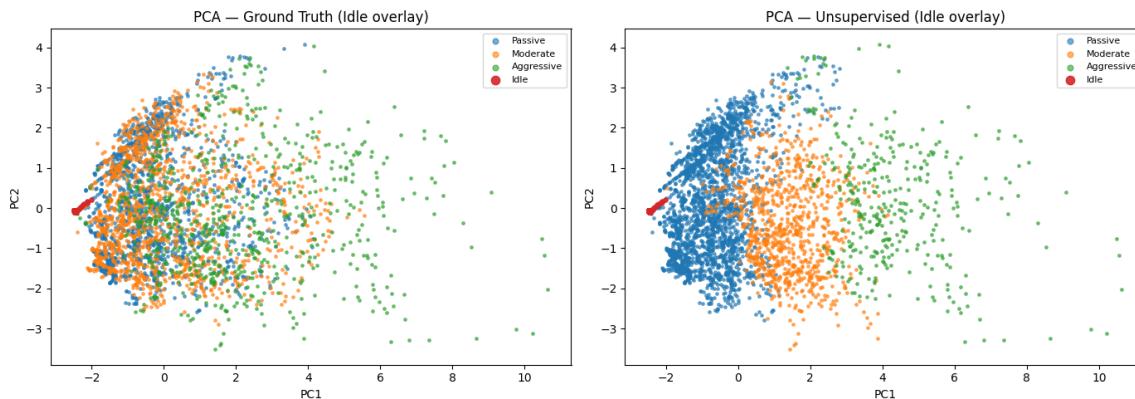


Figure. UL vs GT PCA visualization

Notebook: <https://colab.research.google.com/drive/1FjFtb5qFgPciVY57yIKZaqwjEEfKkIQH>

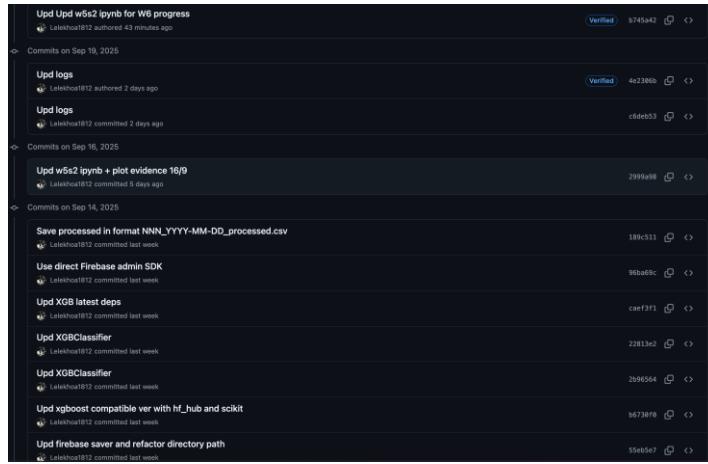


Figure. GitHub commit history for FastAPI Backend and Jupyter Notebook development

Dashboard UI Integrations

- Update Firebase DB connection and enhance security + authentication.
- Externally connect Next.JS dashboard frontend with FastAPI backend.
- Migrate existing MongoDB usage and services to Firebase to be compatible with future frontend development.
- Enhance visualization from sensor log trend and status (including session duration, uploaded date and processing states).
- Enhance UI design with interaction and animations.

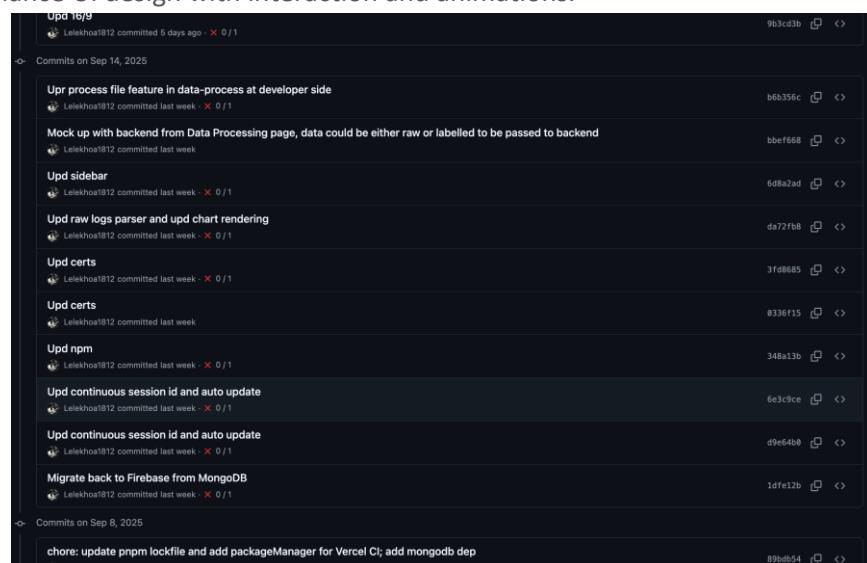


Figure. GitHub commit history for Dashboard

Barsat

In this sprint I designed and started building the web app for the final deliverable. I worked mainly on the frontend and created the initial user and developer pages using mock data.

Because our data was scattered across different places, I set up Firebase as our primary database and storage. On the developer side of the site I built features to upload raw CSVs directly from the website into the database, and added an option to manually view and edit the CSV/data (assigning or adjusting driving style labels). These steps have helped centralize our pipeline and prepared the system for automatic cleaning once new labelled data is uploaded.

The screenshot shows the Firebase Storage interface for a project named 'skyledge'. The left sidebar includes links for Project Overview, Storage (which is selected), Firestore Database, Functions, Authentication, App Check, Product categories, Build, Run, Analytics, and AI. The main area displays a tree structure under 'gs://skyledge-36b56.firebaseiostorage.app > skyledge'. It lists three folders: 'labeled/' (Type: Folder, Last modified: -), 'processed/' (Type: Folder, Last modified: -), and 'raw/' (Type: Folder, Last modified: -). There is also a 'Upload file' button at the top right of the list.

Figure. Centralized backend and database

The screenshot shows the SkyLedge developer dashboard. The left sidebar has sections for SkyLedge — Dev (Predictive Maintenance, Upload / Buffer, Manual Labeling, Labeled Dataset, Reinforcement) and Dashboard (which is selected). The main area is titled 'Developer Dashboard' and 'Manage OBD-II data processing, labeling, and model training'. It features tabs for 'Upload/Buffer', 'Manual Labeling' (which is selected), 'Labeled Dataset', and 'Reinforcement'. Below this is the 'Manual Labeling Tool' section, which includes dropdowns for 'Trip: 005_2025-09-05' and 'Signals:'. A large plot titled 'Signals (scrub to navigate)' shows a fluctuating blue line over time, with a red vertical dashed line marking a specific point. The x-axis is labeled '00:00:00' and the y-axis ranges from 0 to 3400. At the bottom of the plot, there are buttons for 'SPEED', 'RPM', and 'THROTTLE_POS'.

Figure. Developer side of the web application

Sadman

This sprint's focus for me was rewriting the rudimentary datalogging script from scratch in Python. In the previous iteration, the logger was a basic Bash script that sequentially queried OBD data and wrote it to CSV, but it lacked flexibility, error handling, and the ability to integrate with other parts of the system. The new version is being developed as a multithreaded Python application, designed to collect OBD-II data at higher frequencies while simultaneously managing upload and storage processes. This approach not only makes the system more robust, but also reduces bottlenecks by separating data acquisition from network communication.

The Python implementation also provides tighter integration with the rest of our pipeline, particularly the automated upload to Firebase and compatibility with downstream cleaning and

modelling steps. Error logging, reconnection handling, and configurable sampling rates are being incorporated to improve reliability during real-world use. By the end of this sprint, the foundations of this multithreaded structure were in place, positioning the logger to act as a resilient component of the overall driver behaviour classification system in future sprints.

The screenshot shows a GitHub commit history for the 'master' branch. The commits are organized by date:

- Sep 18, 2025:** (WIP) Log compression. (airkoala committed 3 days ago) - Hash: cafe87a
- Sep 5, 2025:** State management. (airkoala committed 2 weeks ago) - Hash: 4b4d0e8
- Aug 31, 2025:** Implement wifi scanning. (airkoala committed 3 weeks ago) - Hash: f4c3e06
- Aug 26, 2025:** Initial commit. (airkoala committed last month) - Hash: 36dbfae

1. SPRINT PLAN

Sprint Two, which took place during weeks five and six of the semester, marked a significant step forward in the development of our driver behavior classification system. Building on the foundations laid during Sprint One, which concluded in week four, this sprint focused on building on top of earlier progress and pushing the system closer to a production-ready state. Compared to the previous sprint, the team operated with greater focus and efficiency, benefiting from clearer coordination and more precisely defined roles. This improved structure allowed development efforts to be more targeted and productive.

The primary goals for this sprint were to validate our unsupervised learning approach by rigorously comparing it to rule-based baselines, integrate machine learning models into the production pipeline, and develop a functional user interface for data upload and visualization. At the same time, the team sought to establish quality management protocols for the classification system and advance the overall system architecture toward minimum viable product readiness.

Learning from the challenges of Sprint One, the team adopted a more structured and role-oriented approach. Responsibilities were clearly defined: Sadman focused on physical device

integration, handling Raspberry Pi automation, vehicle system integration, and cloud connectivity; Dale concentrated on data labeling and collection, overseeing manual annotation, supervised labeling processes, and dataset expansion; Khoa led model development and testing, implementing the machine learning pipeline, unsupervised learning methods, and model validation; and Barsat directed UI development, creating the frontend interface, designing the user experience, and integrating Firebase. This distribution of work reduced the need for frequent full-team meetings and allowed for more efficient collaboration through focused one-on-one discussions.

Key deliverables for this sprint included the implementation of a Bayesian Gaussian Mixture Model for unsupervised driver behavior classification, the creation of a rule-based baseline system for idle detection with quantitative validation, and the deployment of the integrated model pipeline to Hugging Face. Additional goals were the development of a functional upload interface with a Firebase backend, the introduction of a quality management framework with testing protocols, and progress toward an XGBoost classification model with a targeted accuracy of 97 percent.

Throughout Sprint Two, the team continued to follow Agile principles, maintaining daily standups and focused sprint meetings. This improved communication structure streamlined problem-solving and reduced coordination overhead compared to Sprint One, enabling steady progress toward achieving the sprint's objectives.

2. QUALITY MANAGEMENT PLAN AND OUTCOME

The quality management plan for this sprint focused on establishing transparent evaluation criteria for our classification models, validating results against simpler baselines, and ensuring the overall system functioned reliably as an integrated pipeline. The primary goals were to confirm the accuracy and stability of Idle detection, assess the performance of unsupervised learning methods for driving style classification, and demonstrate explainability through comparison with rule-based approaches.

During Sprint Two, a number of quality management activities were undertaken. The team validated the Bayesian Gaussian Mixture Model (BGMM) against a transparent rule-based Idle detector, which applied thresholds on speed, throttle, and engine load, combined with stability filtering. This head-to-head evaluation achieved very high concordance (accuracy 0.992, Jaccard 0.928), demonstrating that both methods were consistent while also highlighting the improved fidelity of the unsupervised approach. Additional clustering metrics further confirmed the strength of the BGMM, with a silhouette score of 0.216 compared to 0.001 for the ground truth labels, showing significantly better separability of driving behaviour classes. Manual labelling

protocols were also refined through timestamp alignment and clearer annotation notes, reducing ambiguity and supporting higher-quality validation studies.

Quality checks extended beyond modelling to include integration testing. The data pipeline—comprising logging, cleaning, classification, and the Firebase-backed upload interface—was tested for consistency and usability. Successful integration of the Hugging Face backend with the frontend dashboard enabled visualisation of session-level statistics such as duration, sample rates, and log trends, helping to confirm data integrity across the system. These steps provided confidence that the solution is not only technically valid but also operationally reliable from an end-user perspective.

Looking forward, several quality management activities are planned for the next sprint. The team will extend the rule-based framework to cover driving style stratification (Passive, Moderate, Aggressive), providing a baseline for further comparison with unsupervised outputs. User acceptance testing will be conducted with the client to validate the usability of the upload interface and the clarity of classification results. Additional operational metrics, such as per-trip stability, flip-rates, and disagreement heatmaps, will be added to the dashboard to provide more granular insight into model performance. Finally, live vehicle testing with the automated logger is scheduled for the next sprint to assess how the pipeline performs under real-world operating conditions.

3. SPRINT PROGRESS

WEEK 5: DATA EXPANSION, BGMM UNSUPERVISED LABELLING, UI PROTOTYPE

Overview: We expanded supervised labels, introduced a **two-stage unsupervised driver-behaviour pipeline** centered on **Bayesian Gaussian Mixture Models (BGMM)** with conservative pre-gating for Idle, and delivered a first-cut **upload UI** (mock-data backed) writing to Firebase (skyledge/raw, skyledge/labelled, skyledge/processed).

Activities:

1. Deeper supervised labelling.

- Increased labelled coverage across recent logs; standardised annotator notes and timestamp alignment to reduce ambiguity in transitions (Idle \rightleftarrows Driving; Passive \rightleftarrows Moderate). Available at:
https://github.com/benty691/EAT40005/tree/main/logs/driver_behavior_s2w5
- Although manual data labelling have better quality now, we couldn't rely or justify whether they are accurate or not (from human errors).
- In brief, while validated supervised data are compulsory important, we only can use it to detect the clustering patterns and approximation, which then we can be leveraged with different processing and ML/UL techniques to produce more accurate data quality before modelling.

2. UL Stage-1 — Conservative Idle gate + BGMM confirmation.

- **Low-movement multi-cue gate** using quantiles on SPEED_mean_w5, THROTTLE_POS_mean_w5, ACCEL_std_w5, RPM_std_w5, MAF_mean_w5. This **prevents “passive creep”** from being mis-tagged as Idle by requiring *concurrent* quiescence across speed, throttle, load and dynamics.
- **BGMM refinement** ($n=3$, full covariance, weak Dirichlet prior) only inside the gated subset; identify the **lowest-speed component** as Idle and require $P(\text{Idle}) \geq 0.6$.
- **Temporal smoothing** via median filter to remove flicker; **episode statistics** (time_frac, episodes, run-lengths) computed with base sampling inferred from timestamps.

3. UL Stage-2 — Driving stratification (Passive / Moderate / Aggressive).

- Remove Idle; build a dynamic **feature set** (accel/jerk/throttle-change/RPM variability + speed/load).
- Compute an **unsupervised aggressiveness index** (weighted sum in z-space) and **gate top-10% as Aggressive**.
- Run **BGMM(2)** on the remainder; **name clusters by intensity** using the same index → **Passive vs Moderate**.
- Optional median-filtering on the final stream of cluster IDs to stabilize boundaries.

Why BGMM (vs prior K-Means): K-Means assumed spherical equal-variance clusters and showed **overlap** between Idle↔Passive and Passive↔Moderate on older datasets. BGMM supports **anisotropic covariance** and **soft membership**, improving boundary fidelity—especially for low-speed creeping and medium-intensity driving.

4. Evidence (UL vs ground truth (GT) or labels). Using matched time regions and excluding Idle where noted:

- Compactness/Separability (Idle excluded): UL → *silhouette* 0.216, *Calinski-Harabasz* 1044.0, *Davies-Bouldin* 1.654; GT → 0.001 / 248.0 / 7.076.
- k-NN label agreement ($k=15$, Idle excluded): UL 0.965 vs GT 0.685.
- Temporal stability (flips/min): UL 6.15 (driving-only 6.09) vs GT 1.76 (1.22). (*UL is intentionally more responsive; we apply smoothing/hysteresis downstream.*)
- Nearest-centroid conformity (Idle excluded): UL global 0.917 (Passive 0.95, Moderate 0.84, Aggressive 0.88) vs GT global 0.532.
- Embeddings (PCA, t-SNE): UL forms a clean Passive→Moderate→Aggressive gradient; GT labels are inter-mixed, especially in Moderate.

UL methods also found to be more reasonable oriented (from class fraction proportioning) and low overlap (low relationships across classes, from PCA comparision), which is compulsory for classification modelling to ensure sustainable correlation. Here is a comparison between UL and GT through 12 logs.

Class Fractions (proportion excluding Idle):

	UL Clustering	Supervised GT
Passive	65.7%	41.9%
Moderate	24.5%	38.1%
Aggressive	9.8%	20.0%

Table. UL vs GT class fraction

2D-PCA Overlap (Mahalanobis distance):

	UL Clustering	Supervised GT
Passive	61.8%	9.4%
Moderate	86.4%	27.1%
Aggressive	30.2%	4.5%

Table. UL vs GT PCA overlap comparison

Notebook: <https://colab.research.google.com/drive/1FjFtb5qFgPciVY57yIKZaqwjEEfKkIQH>

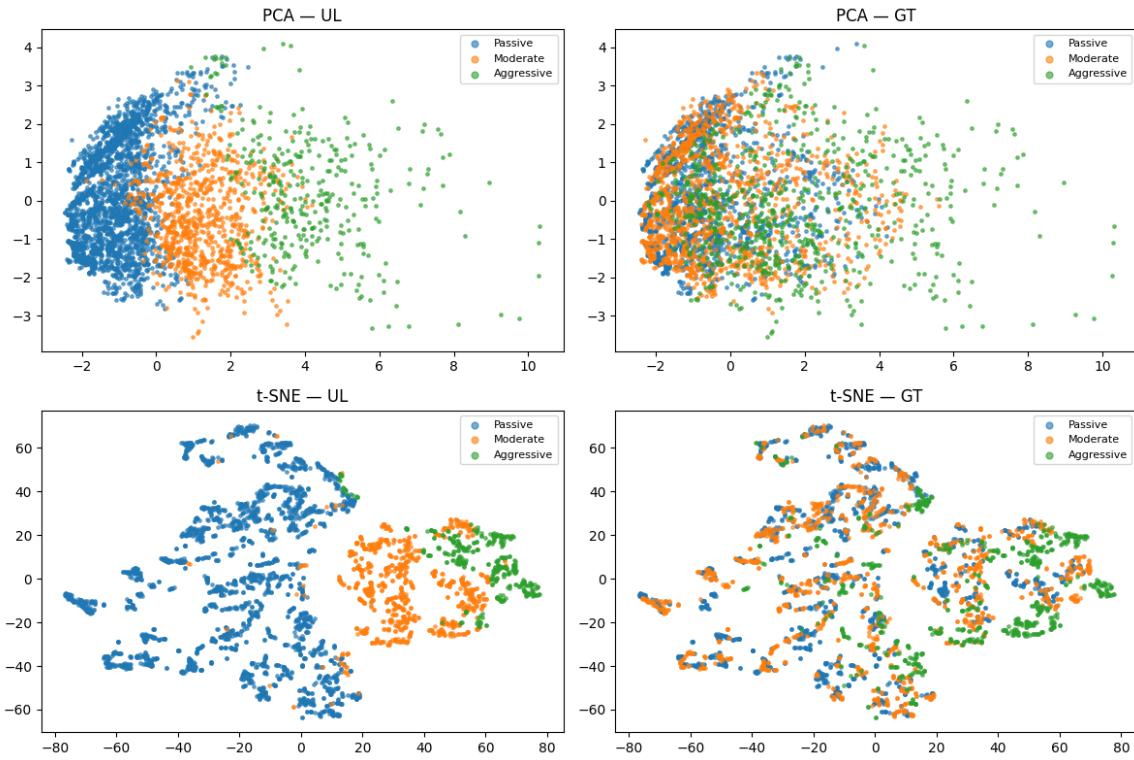


Figure. Strong clustering and neighbor support PCA and t-SNE profiles UL vs GT

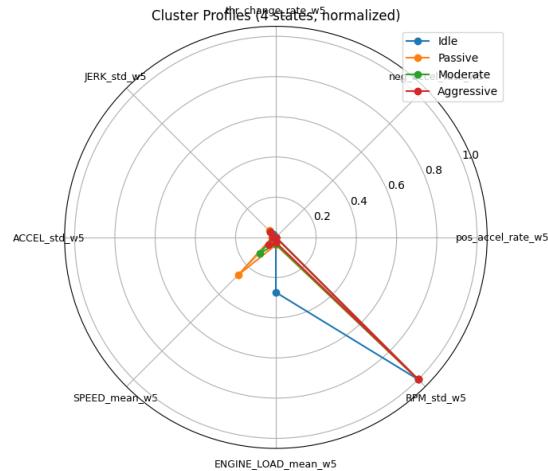


Figure. UL clustering radar profiles on component variables

5. Upload UI prototype. Deployed at <https://skyledge-ten.vercel.app> (mock-data at this stage). Upload writes to Firebase Storage in three paths to support later cleaning/annotation workflows.

SkyLedge Feedback: Strong technical quality but asked us to justify accuracy and reasonability more rigorously. Suggested we may be moving too fast with ML/UL; requested a simpler baseline—start with threshold/rule-based Idle detection and compare.

Team Action:

- Ran a team planning session to rebalance roles and prioritize a rule-based Idle baseline with formal agreement metrics.
- Set an evaluation plan: IoU/Jaccard for Idle, confusion trends at low speed/throttle, and stability vs responsiveness under smoothing/hysteresis.

WEEK 6: RULE-BASED IDLE BASELINE, AGREEMENT STUDY, MODEL INTEGRATION

Overview: We implemented a domain-driven, mathematically explicit Idle detector and ran a head-to-head agreement study vs the UL/BGMM Idle. Results show high concordance and improve explainability. We also integrated models with the cleaner and connected the frontend for a cohesive flow.

Activities:

- **Rule-based Idle (domain + stability):**
 - Cut-offs: SPEED $\leq \sim 5$ km/h, THROTTLE $\leq 5\%$, ENGINE_LOAD $\leq 25\%$ (near-stationary engine with light load).
 - Low-variability filter: require very small ACCEL and RPM std to avoid mis-labelling “slow creep”.
 - Conjunctive rule: must meet all conditions; then median filter (short window) for flicker suppression.
- **Results (Idle episode stats & agreement):**

Method	samples_frac	time_frac	episodes	dur_median_s	dur_p90_s	dur_mean_s
Rule	0.110252	0.110252	19	6.60	31.944	15.354
ML (BGMM)	0.103268	0.103268	16	11.55	41.910	17.078

Table. UL vs Rule Idle episode statistics and agreements

Agreement: accuracy 0.992, Jaccard (Idle) 0.928.

Visual checks (Idle timelines; speed-vs-throttle overlays; distributions of SPEED_mean_w5, THROTTLE_POS_mean_w5, ENGINE_LOAD_mean_w5) show **tight alignment** and minimal disagreement confined to low-value borders.

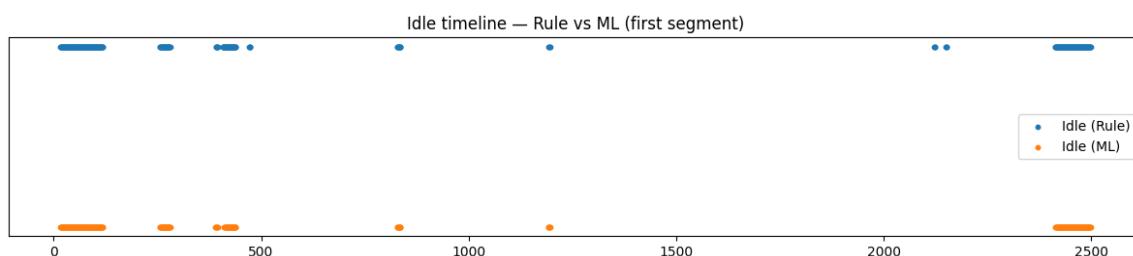


Figure. Strong timeline agreement on Idle state

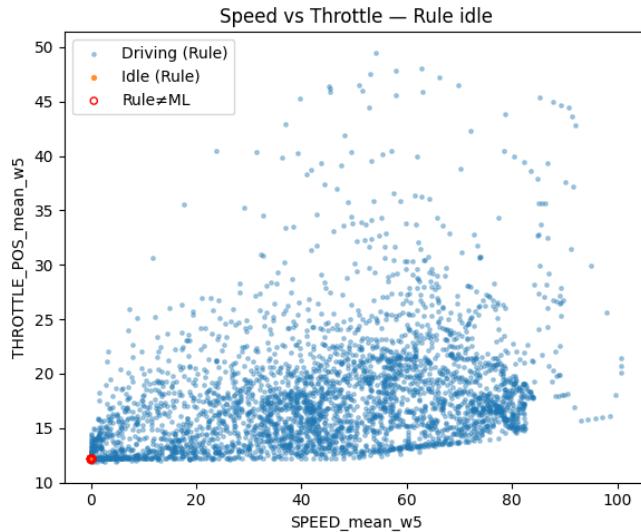


Figure. Minimal disagreement on Speed vs Throttle position validation (Kernel Density Plots)

- Evaluation best practices (applied).
 - On imbalanced events, overall accuracy can be misleading → report Jaccard/IoU for Idle, plus episode-level stats.
 - Use posterior gating / hysteresis / smoothing to trade reactivity vs stability; we standardised a median filter and k-of-n alerting.
- Model-in-the-loop cleaning. Integrated the UL classifier into the Hugging Face cleaning space:
 - Pipeline: https://huggingface.co/spaces/BinKhoaLe1812/OBD_Logger
 - Models: https://huggingface.co/BinKhoaLe1812/Driver_Behavior_OBD/tree/main
- Exported XGBoost stack (encoder + scaler + classifier), which reached 97% accuracy, and **wired it into the cleaner**, enabling:
 - Processing/cleaning raw or supervised/labelled inputs for consistent features.
 - Frontend integration improvements: sensor trend charts, duration/proportion summaries, upload timestamps, download/inspect actions.
 - In-UI streaming labelling via a progress-bar timeline (reduces CSV-based manual edits; supports rapid feedback loops).

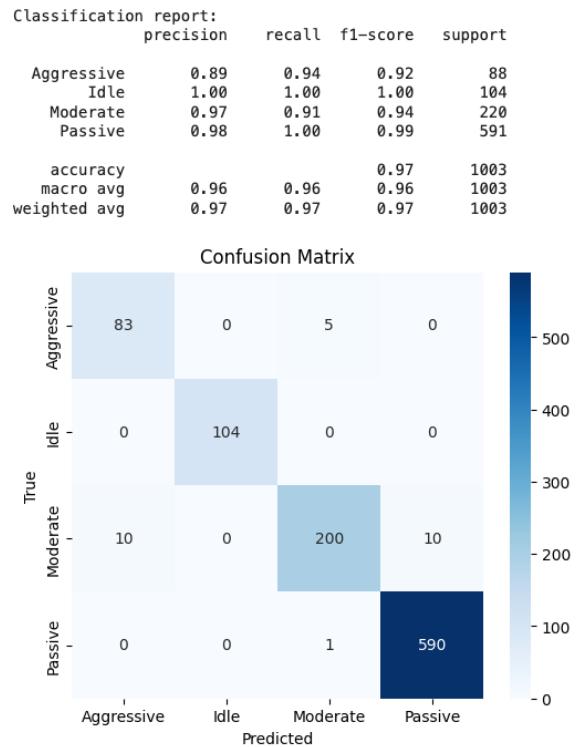


Figure. XGBoost model training outcome statistics

- Update UI prototype. Deployed at <https://skyledge-ten.vercel.app>. Our frontend app now is integrated to the FastAPI backend (cleaning pipeline, saved to `skyledge/processed`) and provide structural insights with sensor trend log chart, session ID, file size and download action, deriving duration, sample rate and number of samples, directly parsed from the session that is stored on Firebase.

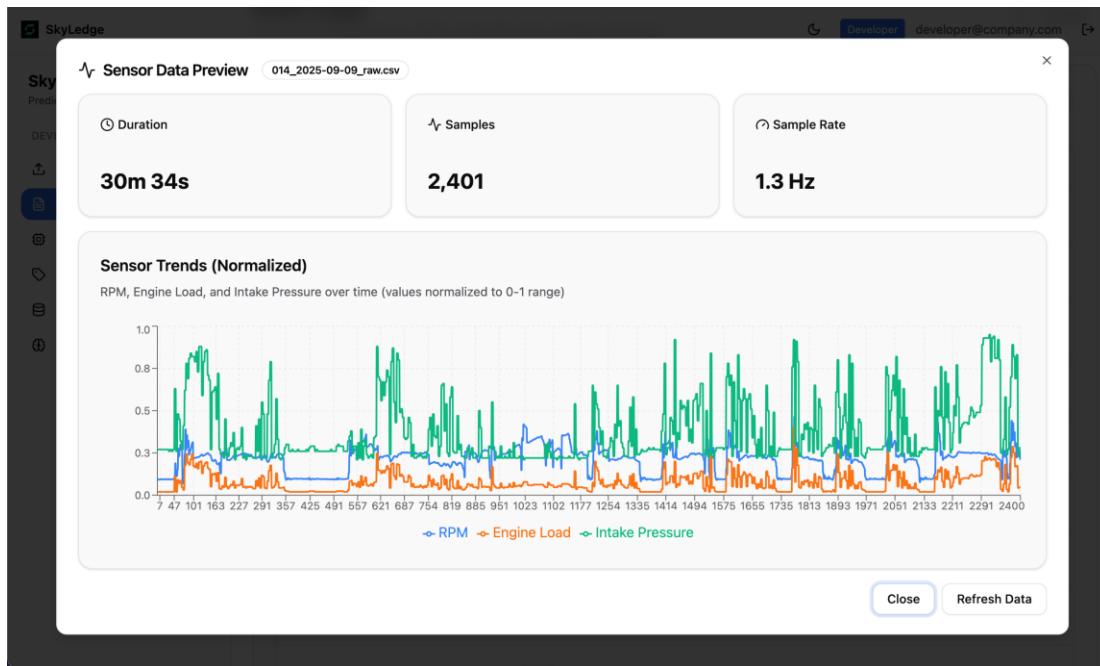


Figure. Sensor log visualization

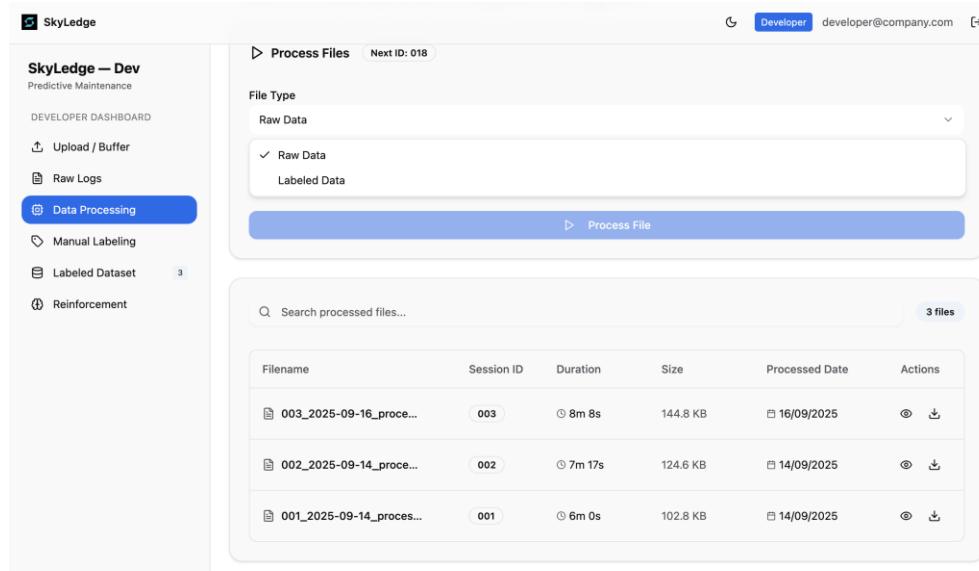


Figure. Data processing endpoint on frontend

4. SPRINT REVIEW (CRITICAL REVIEW OF THE PRODUCT)

WHAT WAS DEMONSTRATED

- **Week 5:**
 - BGMM-based UL pipeline with conservative Idle pre-gate + probabilistic confirmation; Stage-2 driving stratification (Passive/Moderate/Aggressive).

- Quantitative evidence (cluster validity, k-NN agreement, centroid conformity, temporal analysis) and embedding visuals (PCA, t-SNE).
- Upload UI writing to Firebase (mock-data).
- **Week 6:**
 - Rule-based Idle detector (cut-offs + low-variance + smoothing) and agreement study vs UL (Acc 0.992; Jaccard(Idle) 0.928).
 - Cleaner + model integration (HF pipeline + XGBoost stack) and frontend enhancements (trend charts, duration/proportions, streaming labels).

FEEDBACK PROVIDED BY CLIENT

- Technical depth clarification; requested clearer justification of correctness and a simpler, explainable baseline starting with Idle thresholding before layering ML/UL.

CRITICAL ANALYSIS OF PROGRESS AND FEEDBACK

Progress

vs

Objectives:

We delivered explainable Idle detection with strong quantitative agreement to our UL approach, while preserving the benefits of ML (soft boundaries; richer dynamics) for subsequent stages. We also moved the system forward architecturally: model-in-the-loop cleaning, frontend hooks, and exported production artifacts (XGBoost, scalers, encoders).

Why this matters:

- Data quality first: The conservative gating + BGMM approach reduced prior K-Means overlaps and improved label coherence; the rule baseline now anchors decisions with domain transparency.
- Trust & governance: Idle decisions are now auditable (thresholds) and verifiable (IoU, episode stats, overlays).
- Product readiness: Integration across HF pipeline → model stack → UI sets the stage for closed-loop labelling and driver-facing insights.

Challenges & Mitigations:

- Label noise / human variance: Mitigated via agreement studies and timeline overlays; doubled down on in-UI streaming labelling to reduce manual CSV errors.
- Idle vs “passive creep” edge cases: Addressed by multi-cue gating + low-variance filters + temporal smoothing.
- Generalisation across vehicles: Kept thresholds configurable, with per-fleet overrides and data-driven quantile checks.

Key Achievements:

- UL clustering methods surpass supervised data which leverage labelling patterns to come up with higher quality and explore statistical hidden traits.

- High-agreement Idle: Acc 0.992, Jaccard (Idle) 0.928 vs UL; robust episode stats. This proves that our UL methods with BGMM are validated.
- UL pipeline upgrade: BGMM with anisotropic covariance and soft memberships; stratified driving behaviour with intensity-aware gating.
- End-to-end wiring: HF cleaner \leftrightarrow model exports \leftrightarrow Firebase-backed UI; ready for user-in-the-loop operation.

Reflection & Next Steps:

1. Codify the baseline: Ship Idle thresholds as configurable policy in the cleaner; expose sliders in UI; log pre/post smoothing decisions.
2. Extend rule-first to driving styles: Define transparent dynamics cut-offs for Passive/Moderate/Aggressive (accel/jerk/throttle-change windows), then refine with UL where boundaries are fuzzy.
3. Operational metrics: Add per-trip stability, flip-rate, IoU overtime, and disagreement heatmaps to the dashboard.
4. Data flywheel: Use the streaming labeler to capture edge cases, auto-surface clips with high rule/UL disagreement for review.

This sprint tightened methodological rigor (transparent baselines + quantified agreement) while advancing the product integration needed for a driver-facing MVP.

5. RETROSPECT (CRITICAL REVIEW OF THE PROCESS)

Building upon the momentum from our previous sprint, we continued our development efforts towards delivering a comprehensive driver behaviour and fuel efficiency prediction system, maintaining our established workflows while navigating several methodological challenges.

Positively, again, we maintained our use of Jira for task tracking, as well as consistent stand ups and ad hoc team meetings. Sprint 5 has highlighted critical challenges in our data labelling pipeline and unsupervised learning approaches, particularly around achieving reliable accuracy in our driver behaviour classification system. This sprint forced us to confront the fundamental question of data quality versus quantity while developing a robust prediction framework.

DATA LABELLING ACCURACY

One of the most significant challenges encountered in this sprint was achieving reliable accuracy in our manual data labelling process. Although we improved our manual data labelling quality through standardised labelling and timestamp alignment, we ultimately couldn't rely on or justify whether our supervised labels were accurate due to inherent human errors and subjectivity in the labelling process. This presented a fundamental challenge to our approach, as we discovered that while validated supervised data remains important, we could only use it to detect clustering patterns and approximations rather than as ground truth for model training. This realisation

forced us to transition towards unsupervised learning techniques using Bayesian Gaussian Mixture Models to overcome the limitations of human assigned labels.

DATASET SIZE AND MODEL TRAINING

The development of our driver behaviour prediction system revealed ongoing challenges around our limited dataset size and the complexity of implementing a self-improving model. While we successfully developed unsupervised labelling methods that could identify driver behaviour patterns, we continued to grapple with the challenge of creating a system where new incoming data could be used to continuously improve the model while simultaneously using that same model to label unclassified data. Our implementation of the two stage BGMM pipeline with conservative pre-gating represented our attempt to address this circular dependency, though the fundamental challenge of ensuring model reliability with limited initial training data remained a persistent concern throughout the sprint.

SPRINT DELAY

Unfortunately, we experienced another sprint delay, as our previous sprint ran a week overtime, effectively reducing our available time for sprint 5 by one week. This compressed timeline impacted our planned deliverables and in turn reduced the amount of work completed in this sprint. While we ultimately delivered on our core objectives, the reduced timeframe created pressure that influenced our approach to model validation and testing procedures.

TEAM ALIGNMENT

We partially lost our way this sprint, experiencing periods where we were not fully aligned as a group and lacked clear direction for individual team members. The technical complexity of transitioning from supervised to unsupervised learning approaches required more coordinated decision-making than we initially anticipated. This was partly resolved through client feedback that suggested we were moving too fast with machine learning approaches, prompting us to run a team planning session to rebalance roles and ensure we can deliver a product come the end of the next sprint.

COMMUNICATION AND CLIENT FEEDBACK

Our client meeting provided crucial guidance, with SkyLedge noting our strong technical quality but requesting more rigorous justification of our accuracy and reasonability. They suggested we implement simpler rule-based baselines before layering complex ML approaches, which helped redirect our focus and provided clearer direction for the team. This feedback proved instrumental in helping us establish transparent, auditable methods alongside our more sophisticated unsupervised approaches.

Despite these challenges, the team demonstrated strong technical capabilities in developing both unsupervised learning pipelines and rule-based validation methods.

Overall, sprint 5 was a period that, while presenting significant challenges around data quality and team coordination, ultimately delivered progress towards a robust driver behaviour classification system. We have established a solid foundation of both explainable rule-based methods and unsupervised techniques, setting us up for more focused development in sprint 6 where we can leverage these complementary approaches to deliver a production-ready solution.

6. LESSONS LEARNED (CRITICAL REVIEW OF SPRINT ONE EXPERIENCE AND FUTURE PLAN)

Key Lessons from Sprint Two

Role Specialization: Dividing work into four clear areas (physical device integration, data labeling/collection, model development, and UI creation) significantly improved team efficiency. Each member became comfortable with their specialized domain, reducing coordination overhead and enabling deeper expertise development. This structure allowed us to work more independently while still maintaining strong collaboration when needed.

Communication Efficiency: Moving away from mandatory full-team meetings for every decision improved productivity substantially. One-on-one discussions and targeted chats allowed faster problem-solving while maintaining team cohesion. This flexible approach respected individual working styles and reduced meeting fatigue, leading to better overall progress compared to Sprint One where coordination challenges slowed development.

Technical Validation Approach: The dual development of unsupervised learning (BGMM) and rule-based systems provided strong validation of our methods. The high agreement between approaches (accuracy 0.992, Jaccard 0.928) confirmed both our ML sophistication and baseline reliability. This validation strategy proved essential when the client requested more explainable approaches, as we could quickly demonstrate the reliability of our complex methods through simpler baselines.

Recommendations for Future Sprints

Continue the successful role specialization structure while maintaining regular integration checkpoints to ensure system cohesion. Extend rule-based approaches to driving style classification (Passive/Moderate/Aggressive) and implement operational metrics like per-trip stability and disagreement heatmaps in the dashboard. Focus on user testing of the upload UI and classification results to validate the end-user experience. Maintain the dual-approach validation strategy for major technical decisions and keep focus on MVP features rather than pursuing technically interesting but non-essential capabilities.

