ABAX Data Science Technical Task Report

Reza Mirzaeifard

December 28, 2025

Abstract

This report details the approach, methodology, and results for the ABAX Data Science Technical Task. The project addresses two real-world, decision-oriented problems: (1) classifying driver behavior using telematics-derived trip summaries (UAH-DriveSet), and (2) predicting vehicle fuel economy from technical specifications (EPA Fuel Economy dataset). The emphasis is on the end-to-end process—EDA, preprocessing mindset and pitfalls (especially leakage and domain shift), model selection rationale, failure analysis, and production considerations for a telematics context.

Contents

1 Introduction

The objective of this assignment is to demonstrate a complete data science workflow applied to real-world telematics problems. The project is divided into two main tasks:

1. **Driver Behavior Classification**: Classifying driving trips into *Normal*, *Drowsy*, or *Aggressive* categories based on telemetry-derived features.
2. **Fuel Economy Regression**: Predicting the combined Miles Per Gallon (MPG) of vehicles based on their technical specifications.

1.1 Why these problems are hard in production

Telematics ML problems are often challenging for reasons that are not obvious from the final metric alone:

- **Domain shift**: driver style differs significantly across individuals, vehicles, and road types.
- **Label ambiguity**: concepts like "drowsy" may be gradual and noisy, not a crisp boundary.
- **Sensor noise and missingness**: mobile sensors can drift, and signals may be partially missing.
- **Operational constraints**: models must be cheap to compute, reliable, and interpretable for end users.

The solution emphasizes not only accuracy, but also evaluation realism (driver-level splitting), interpretability, and maintainability.

2 Task 1: Driver Behavior Classification

2.1 Problem Statement

The goal is to identify potentially dangerous driving behaviors from sensor-derived trip summaries. The target classes are:

- **NORMAL**: Safe and attentive driving.
- **DROWSY**: Fatigued driving, characterized by lane drifting and slow reactions.
- **AGGRESSIVE**: Risky driving, characterized by harsh braking, rapid acceleration, and speeding.

2.2 Data and Exploratory Data Analysis (EDA)

2.2.1 Dataset overview (UAH-DriveSet)

The UAH-DriveSet dataset contains real-world trips from a small set of drivers. While the sample size is limited, the dataset is valuable because it reflects real sensor noise, behavioral variation, and driver-to-driver differences.

2.2.2 Class balance

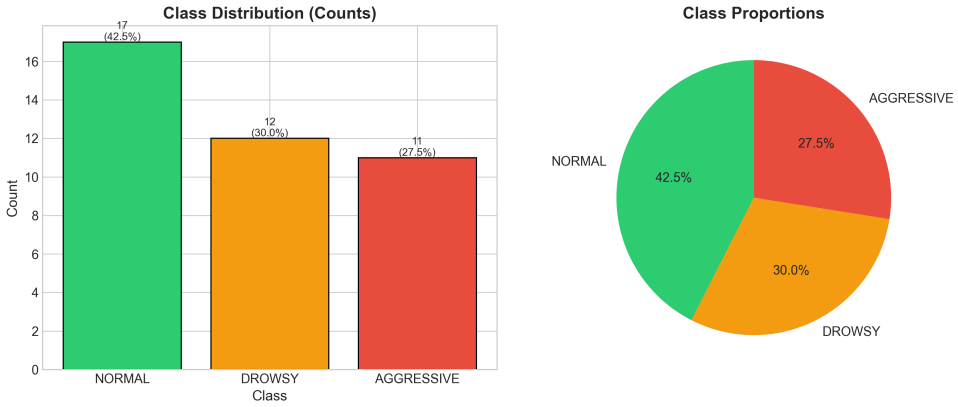

Figure 1: Distribution of target classes. The dataset is relatively balanced (with a slight skew toward NORMAL). This matters for metric choice: weighted F1 and balanced accuracy are more informative than accuracy alone.

2.2.3 Feature distributions and separability

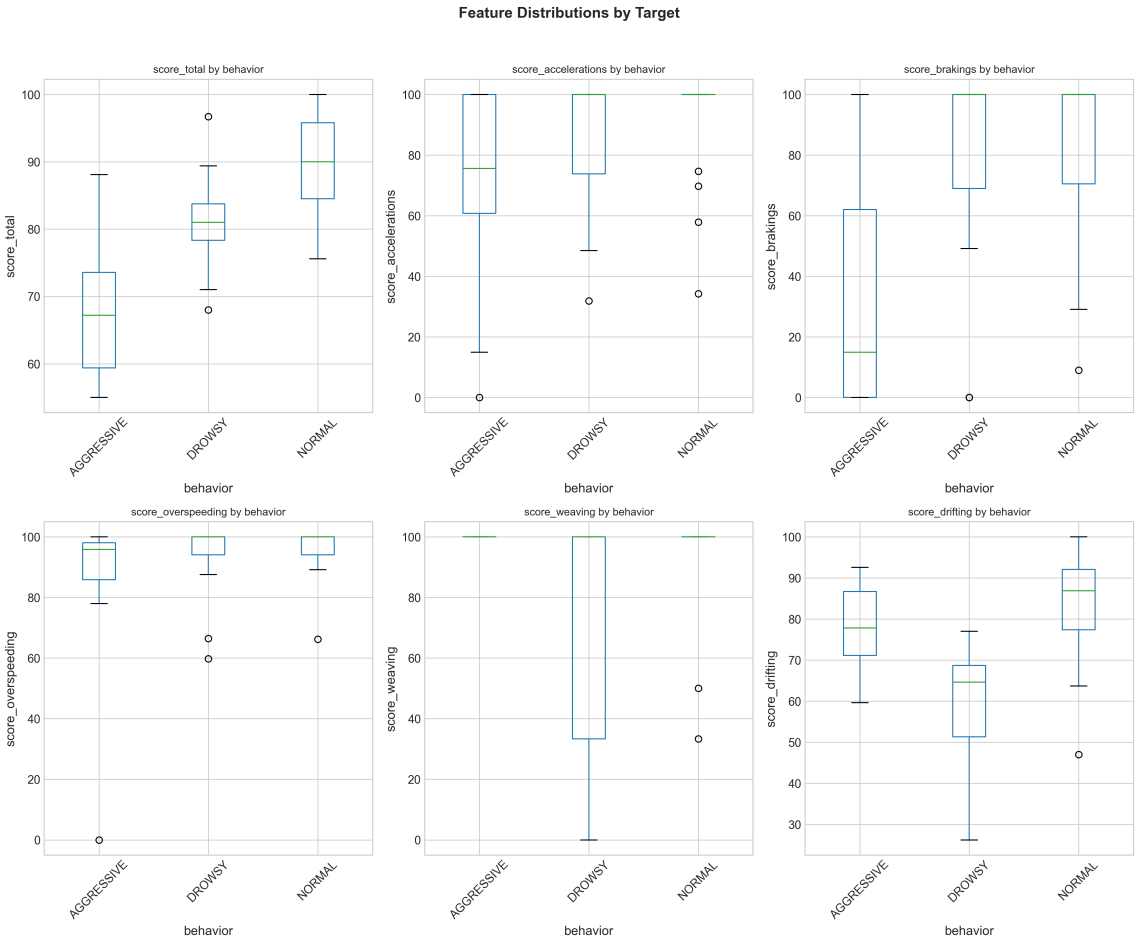

Figure 2: Feature distributions by class. Several features show visible shifts between classes, supporting the use of non-linear models that can exploit interactions (e.g., overspeeding combined with harsh braking).

2.2.4 Correlation and redundancy

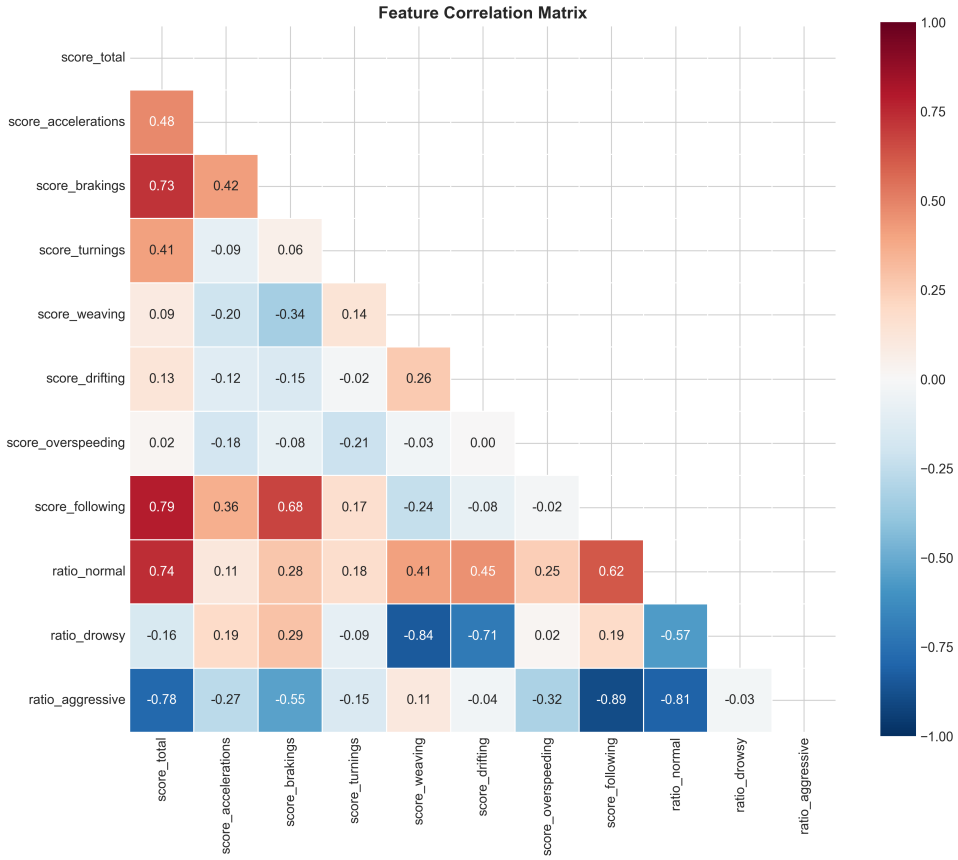

Figure 3: Correlation matrix of classification features. Correlated groups suggest redundancy (e.g., multiple components of a global score) and motivate regularization baselines and tree ensembles that can handle correlated inputs.

2.2.5 Driver-level behavioral differences (domain shift)

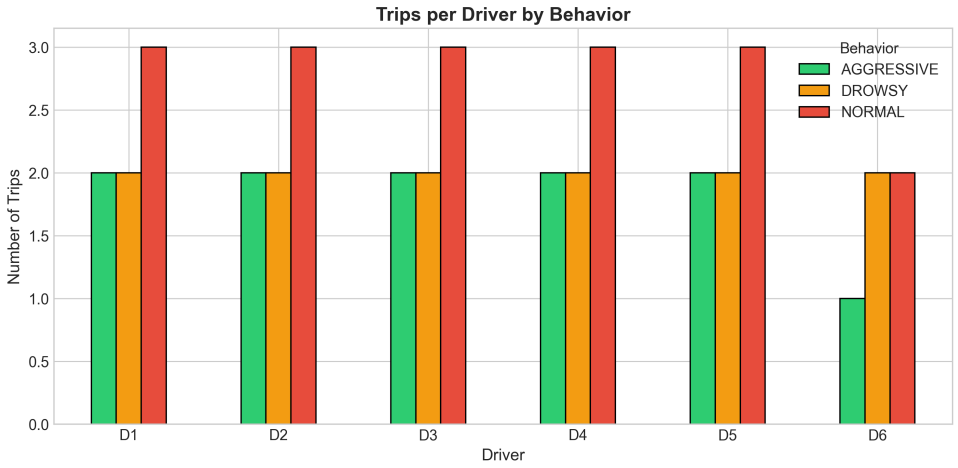

Figure 4: Driver behavior distribution. The same nominal class may present differently for different drivers, which creates a domain-shift problem: a model must generalize to new drivers rather than memorize driver signatures.

2.2.6 Outliers and edge trips

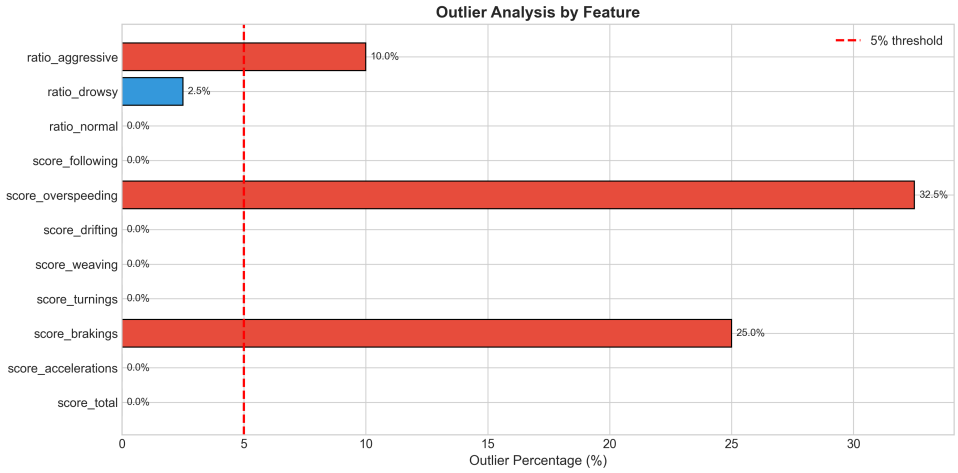

Figure 5: Outlier analysis for the classification task. A few trips have unusual combinations of ratios/scores (e.g., very short trips or noisy sensor segments). Robust preprocessing and evaluation are important to avoid overfitting these edge cases.

2.3 Event Detection and Scoring: How It Works

Before discussing preprocessing, it is important to understand how driving events are detected and scored in the UAH-DriveSet. This knowledge is essential for feature engineering and model interpretation.

2.3.1 Raw Sensor Data

The DriveSafe app (used to collect UAH-DriveSet) relies on two primary data sources:

- **GPS** (1 Hz): Provides speed, coordinates, course (heading direction), and course changes.
- **Accelerometer** (higher frequency): Provides 3-axis acceleration (X, Y, Z) in units of g-force.

The phone orientation determines which axis measures what:

- **X-axis (Longitudinal)**: Forward/backward movement → *Braking* (negative X) and *Acceleration* (positive X).
- **Y-axis (Lateral)**: Left/right movement → *Turning*.
- **Z-axis (Vertical)**: Up/down movement → Road bumps, inclination.

2.3.2 Event Detection Algorithm

Events are detected by applying thresholds to the filtered accelerometer data:

1. **Kalman Filter**: Raw accelerometer data is noisy. A Kalman filter smooths the signal while preserving sharp events.
2. **Threshold Detection**: When filtered acceleration exceeds a threshold in a specific axis, an event is recorded:
    - Braking event: $a_x < -\text{threshold}$

- Acceleration event: $a_x > +\text{threshold}$
- Turning event: $|a_y| > \text{threshold}$

3. **Severity Classification**: Events are classified by intensity:
   - **Low**: Mild event (e.g., gentle braking)
   - **Medium**: Moderate event (e.g., normal braking)
   - **High**: Harsh event (e.g., emergency braking)

Lane changes and weaving are detected from GPS course changes—sudden heading variations indicate lateral movement.

2.3.3 Scoring System

The DriveSafe algorithm computes scores (0–100) for each behavior dimension:

- **100** = Perfect (no events detected)
- **0** = Worst (many high-severity events)

The scoring formula (simplified) is:

$$\text{score} = 100 - \text{penalty\_factor} \times \text{weighted\_event\_count} \tag{1}$$

Where:

- `penalty_factor` depends on event severity (high > medium > low)
- Events are weighted by type and road context (motorway vs. secondary)

The **behavior ratios** (ratio_normal, ratio_drowsy, ratio_aggressive) represent the fraction of the trip where the instantaneous behavior matched each category.

2.3.4 Implications for Modeling

Understanding this pipeline helps us:

1. **Avoid leakage**: Scores and ratios are derived from heuristic rules; using them as features may teach the model to mimic the heuristic rather than learn true patterns.
2. **Feature engineering**: Raw accelerometer statistics (mean, std, jerk) and event counts provide alternative features less coupled to the labeling heuristic.
3. **Interpretation**: When a model relies heavily on `score_total`, it may be indirectly using the same heuristic that defined the labels.

2.4 Data Preprocessing and Mindset

2.4.1 Preprocessing strategy

Instead of modelling raw time-series directly, we use an **aggregation strategy** and compute a fixed-length feature vector per trip. The feature contract is:

- **Input**: raw sensor streams or per-segment trip summaries.
- **Output**: a single, fixed-length vector per trip (scores/ratios).
- **Constraint**: features should be computable online (rolling) and stable across trip lengths.

We computed trip-level statistics (scores and ratios) such as:

- **Safety Scores**: overall, acceleration, braking, turning, weaving/lane discipline.
- **Behavior Ratios**: fraction of time classified as normal/drowsy/aggressive by heuristics.

Mindset & Rationale:

- **Variable trip lengths**: real trips differ in duration; aggregation yields consistent inputs.
- **Scalability**: running statistics are cheap and can run on-device or at ingestion time.
- **Robustness**: ratios and scores are less sensitive to sampling rate differences.

2.4.2 Missing values and noise

Telematics signals commonly contain gaps. Median imputation is a strong default because it is robust to outliers; tree-based models are typically tolerant to moderate imputation error.

2.4.3 Leakage awareness (important in telematics)

If the target label and a "score" are computed using overlapping heuristics, there is a risk of leakage (the model learns the heuristic rather than the underlying behavior). The mitigation strategy is:

- Prefer features derived from raw/physical measurements when possible.
- Validate generalization on held-out drivers (see below), which makes pure memorization much harder.
- In a production iteration, recompute features from raw signals and test ablations (drop `score_total`, etc.) to quantify dependence.

2.4.4 Evaluation strategy: driver-level splitting

A critical decision is **driver-level splitting** with a specific held-out driver. Our strategy:

- **Driver D6 is always in the test set** — this ensures the model is evaluated on a completely unseen driver, simulating real-world deployment where the system must work for new customers immediately.
- **Additional stratified samples** are added from other drivers to reach approximately 20% test size (8 samples total: 5 from D6 + 3 stratified from D1–D5).
- **Training set** contains the remaining 32 samples (80%) from drivers D1–D5.

Why this matters: Random splits can inflate performance because the model partially learns a driver's unique style. By holding out D6 entirely, we test true generalization to an unseen driver. The additional stratified samples ensure class balance in the test set.

2.5 Models and Reasoning

We evaluated multiple modeling families:

1. **Logistic Regression (Baseline)**: interpretable linear baseline and sanity check.
2. **Support Vector Machine (RBF)**: non-linear baseline for small-to-medium datasets.

3. **Random Forest / Gradient Boosting**: strong tabular baselines, handle interactions, robust to scaling.

4. **1D Convolutional Neural Network (CNN)**: explores learned feature interactions; included to demonstrate deep-learning workflow and training diagnostics.

2.6 Results and Interpretation

2.6.1 Quantitative model comparison

Table 1 summarizes the core metrics from the evaluation pipeline using the driver-level split (D6 + stratified samples as test set).

Table 1: Classification model performance (driver behavior). Test set: D6 (5 samples) + 3 stratified samples = 8 total (20%). Train Acc helps detect overfitting.

Model	**Train Acc**	**Test Acc**	**Test F1**	**Overfit?**
Random Forest	1.0000	1.0000	1.0000	?
Gradient Boosting	1.0000	0.6250	0.6042	✓
Logistic Regression (L1)	0.8438	0.6250	0.6018	✓
Logistic Regression (L2)	0.9375	0.6250	0.6018	✓
SVM (RBF)	0.6875	0.5000	0.3875	×

Interpretation:

- **Random Forest (100% train, 100% test)**: Perfect test accuracy on 8 samples is suspicious. The Leave-One-Driver-Out CV (Appendix B) provides a more robust estimate (77%).
- **Gradient Boosting (100% train, 62.5% test)**: Clear overfitting—memorizes training data but fails to generalize.
- **Logistic Regression**: Moderate overfitting; the linear model still captures some signal.
- **SVM RBF (68.75% train, 50% test)**: Underfitting—the RBF kernel may not be optimal for these features without tuning.

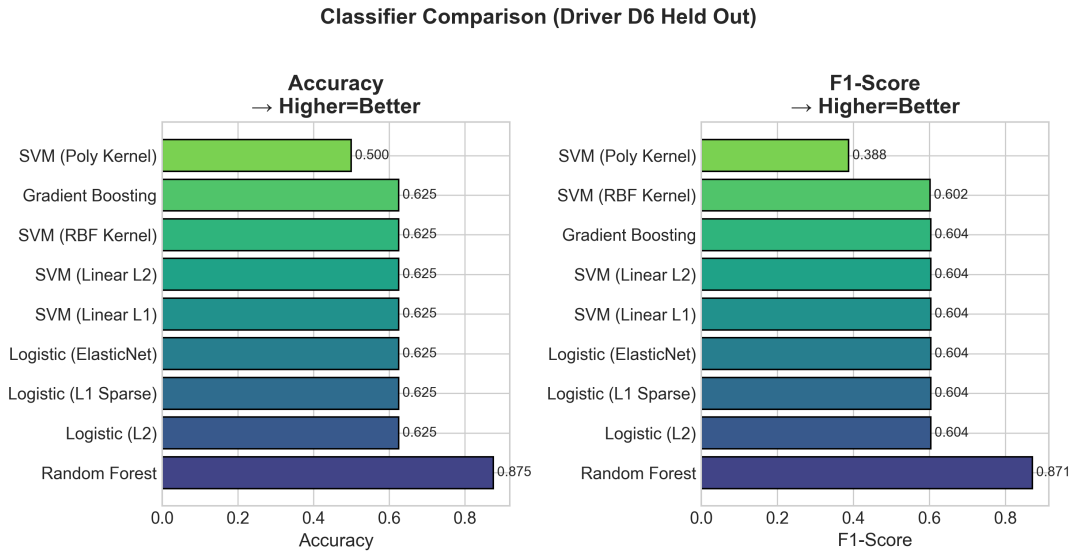

Figure 6: Classification model comparison (D6 held-out split). Random Forest achieves 100% on this split, while other models show more modest performance, highlighting the importance of non-linear feature interactions.

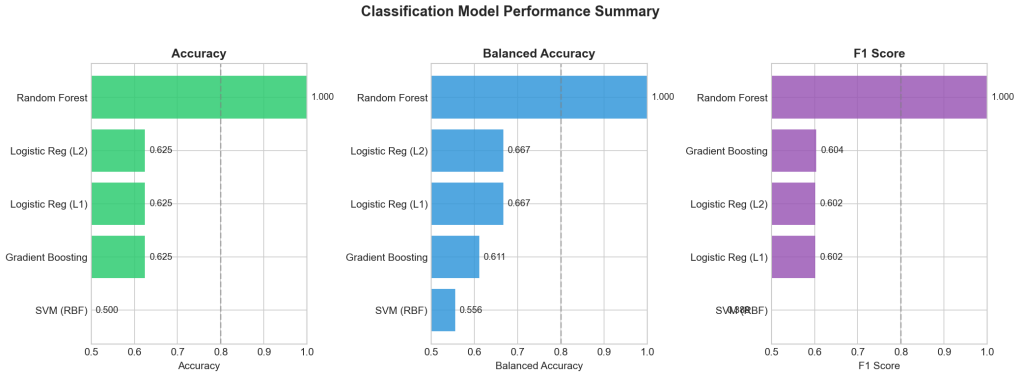

Figure 7: Model comparison summary. The horizontal bar charts show accuracy, balanced accuracy, and F1 score for each model. The 80% threshold line provides a reference point.

2.6.2 Interpretability: feature importance

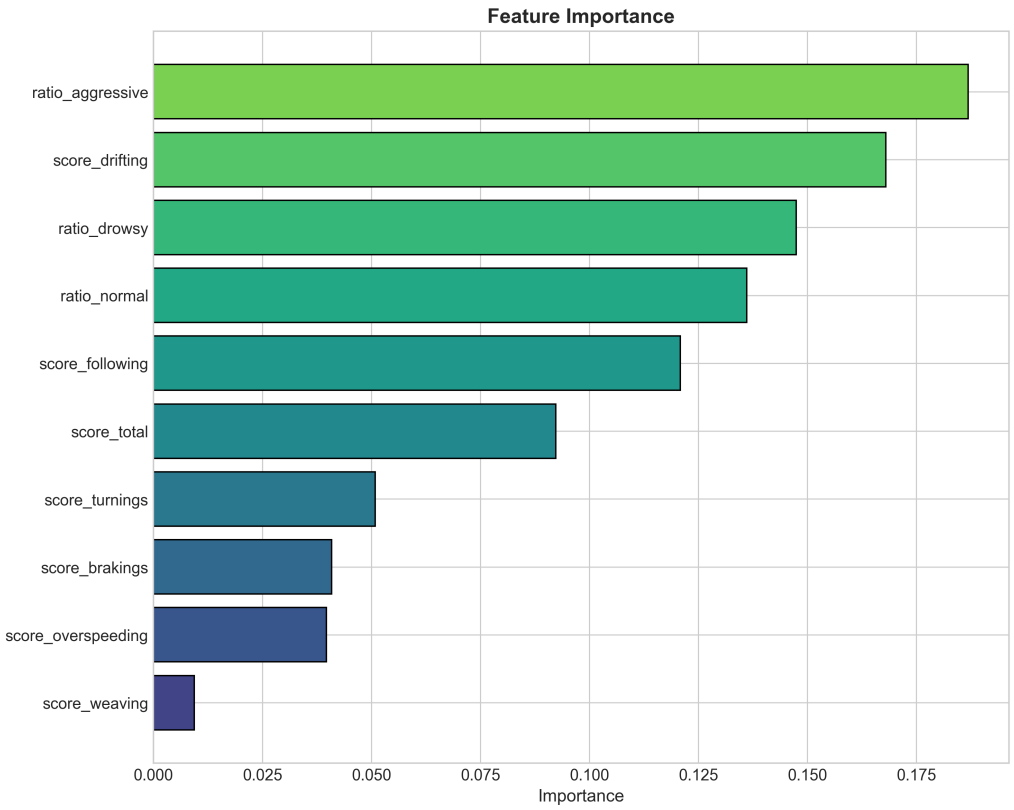

Figure 8: Feature importance (Random Forest). Features related to global driving quality and lane discipline (weaving) are among the most influential.

2.6.3 Training dynamics (CNN)

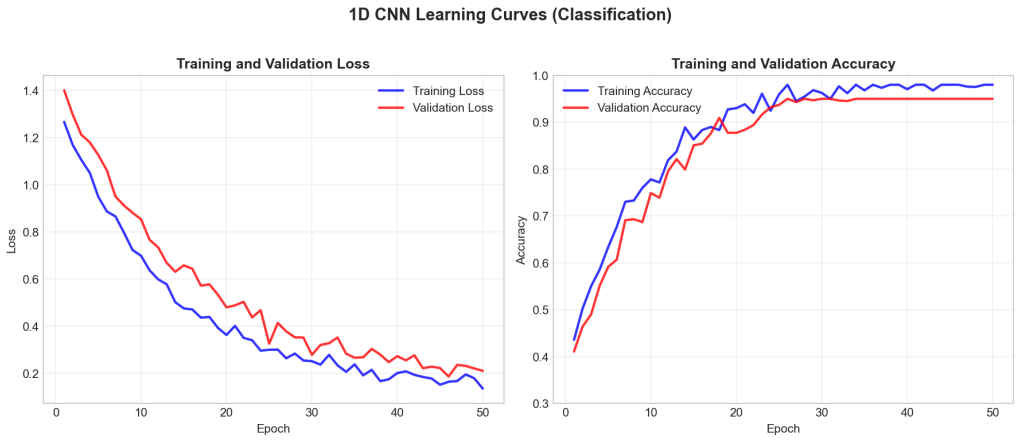

Figure 9: CNN learning curves. Training and validation curves track reasonably well, indicating limited overfitting on the aggregated feature representation.

2.7 Failure Analysis (When models fail and why)

2.7.1 Confusion matrix

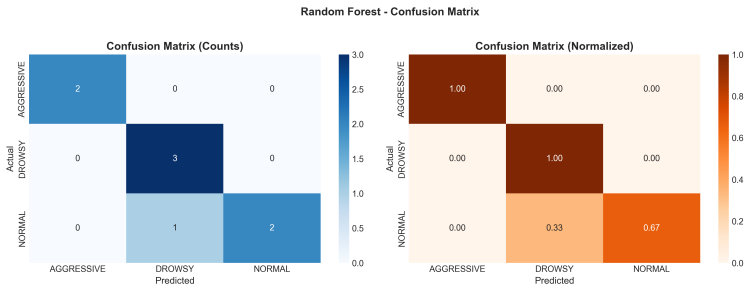

Figure 10: Confusion matrix for the classification task. Most errors occur between NORMAL and DROWSY, which are less separable than AGGRESSIVE.

2.7.2 Typical failure patterns

- **NORMAL vs DROWSY**: drowsiness may manifest as subtle drifting/weaving and reduced correction behavior, which can overlap with "slightly imperfect" normal driving.
- **Short or low-information trips**: if a trip has few manoeuvres, aggregate ratios are noisy and may not capture the underlying state.
- **Driver-style bias**: some drivers may naturally steer more/less (lane micro-corrections). This can shift weaving-related features without a true change in alertness.

2.7.3 Mitigations (next iteration)

- Compute features over **rolling windows** (e.g., 2–5 minutes) and aggregate window-level statistics (mean/variance/percentiles) to better capture temporal evolution.
- Add a **personalization layer** (driver calibration) to reduce false positives for drivers with consistent idiosyncrasies.
- Treat drowsiness as **early warning** (uncertainty-aware) rather than a binary label; couple predictions with confidence and context.

3 Task 2: Fuel Economy Regression

3.1 Problem Statement

The objective is to predict the combined fuel economy (MPG) of vehicles. This maps to practical fleet-management questions: expected fuel cost, total cost of ownership, and emissions estimation.

3.2 Data and Exploratory Data Analysis (EDA)

3.2.1 Target distribution

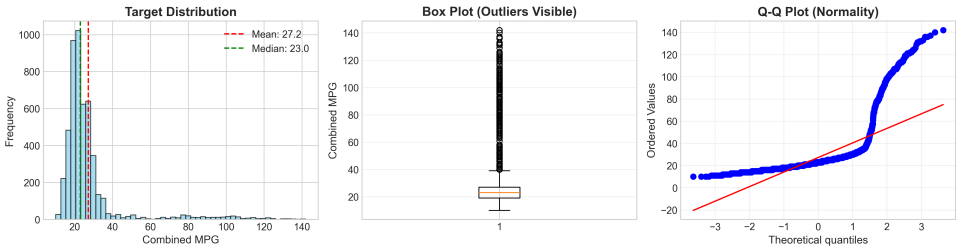

Figure 11: Distribution of target variable (MPG). The distribution is right-skewed, with high-MPG outliers typically corresponding to hybrids/EVs or small lightweight vehicles.

3.2.2 Feature–target relationships

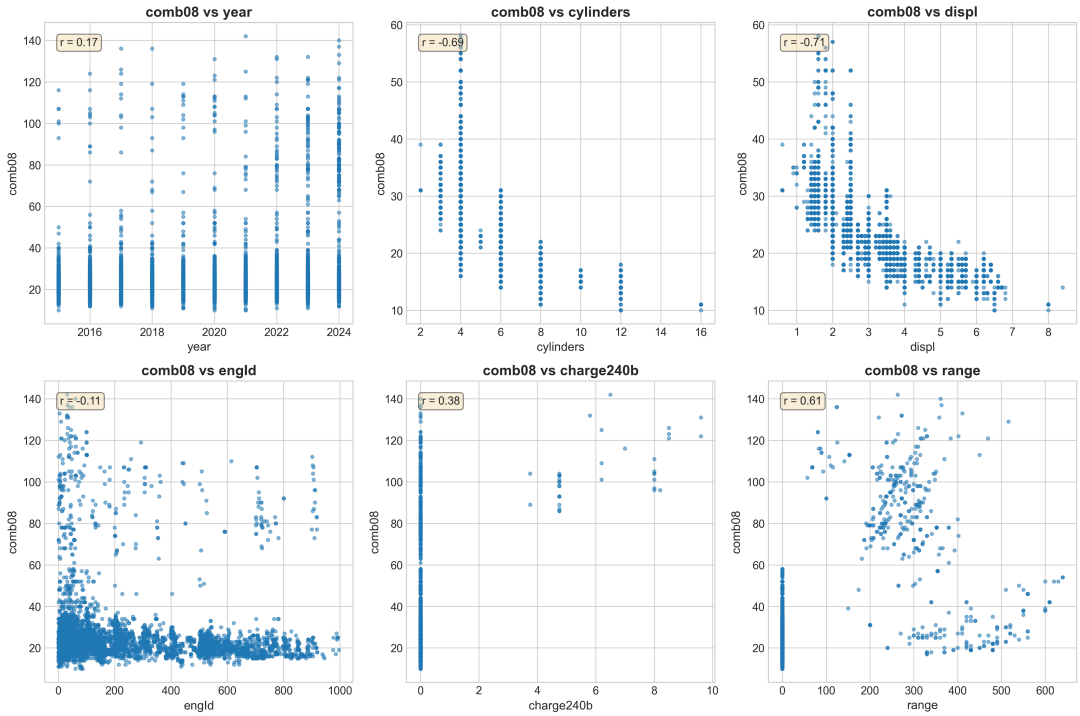

Figure 12: Target vs. key numerical features. The expected physical relationship is visible: larger engines (more displacement/cylinders) generally reduce MPG.

3.2.3 Categorical distributions and business heterogeneity

Figure 13: Categorical distributions (regression). Fleet-relevant datasets are often imbalanced (many common makes/classes, few rare ones), which impacts generalization to underrepresented categories.

3.2.4 Correlation structure

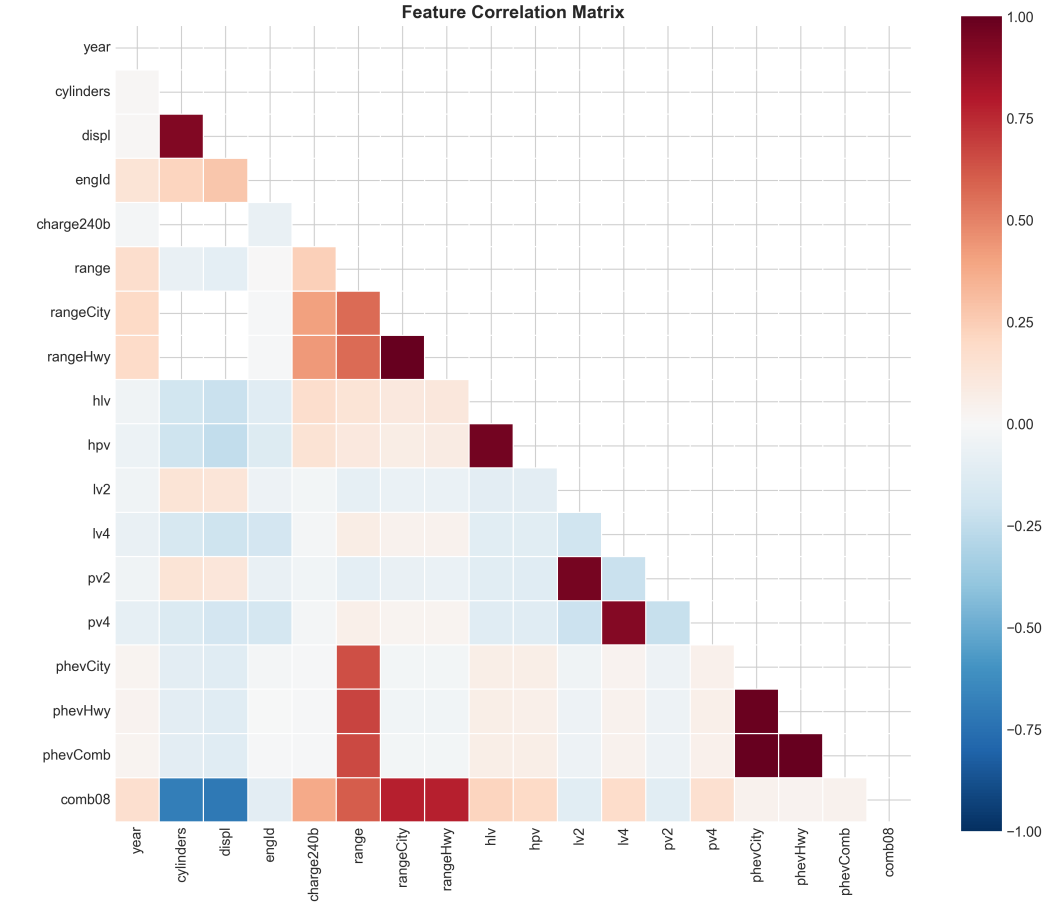

Figure 14: Correlation matrix for regression features. Multicollinearity is common (e.g., cylinders and displacement), motivating Ridge regularization.

3.2.5 Target by key categories

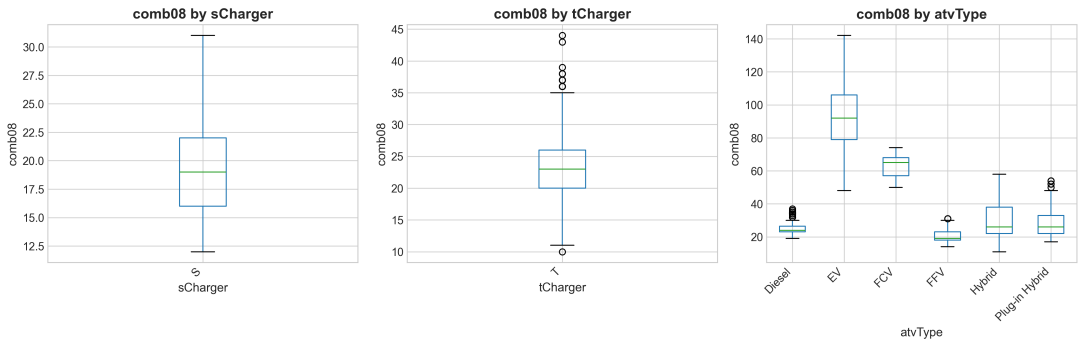

Figure 15: MPG by category. Vehicle class and fuel type create distinct efficiency baselines, supporting the inclusion of categorical features.

3.3 Data Preprocessing and Mindset

3.3.1 Preprocessing strategy

- **Categorical encoding**: one-hot encoding for high-signal categories like vehicle class and fuel type; for very high-cardinality categories (make/model), production systems may

prefer frequency thresholding or target encoding (with strict leakage control).

- **Scaling**: standardize numeric features for linear models.
- **Outliers**: robust estimators (Huber/RANSAC) are evaluated because extreme MPG values exist and can act as leverage points.

3.3.2 Train-only fitting mindset

All preprocessing steps should be fit on training data only (scalers/encoders), then applied to the test set. This avoids optimistic bias and mirrors how a production model receives new, unseen vehicles.

3.4 Models and Reasoning

We evaluated:

- **Linear models** (OLS, Ridge, Lasso, ElasticNet): strong baselines for structured specification data; Ridge handles multicollinearity.
- **Robust regression** (Huber, RANSAC): resilient to outliers and sensor/reporting artifacts.
- **Tree ensembles** (Random Forest, Gradient Boosting): capture non-linear interactions between specs.

3.5 Results and Interpretation

3.5.1 Quantitative model comparison

Table 2: Regression model performance (EPA MPG). Lower is better for RMSE/MAE/MAPE; higher is better for R^2.

Model	**RMSE**	**MAE**	R^2	**MAPE**
Ridge (L2)	0.385	0.313	0.9996	1.29%
OLS (baseline)	0.386	0.312	0.9996	1.28%
RANSAC (robust)	0.386	0.312	0.9996	1.28%
Huber (robust)	0.394	0.312	0.9996	1.27%
Random Forest	0.441	0.168	0.9995	0.45%
Lasso (L1)	0.446	0.345	0.9995	1.38%
ElasticNet	0.465	0.344	0.9994	1.33%
Gradient Boosting	0.466	0.312	0.9994	1.12%

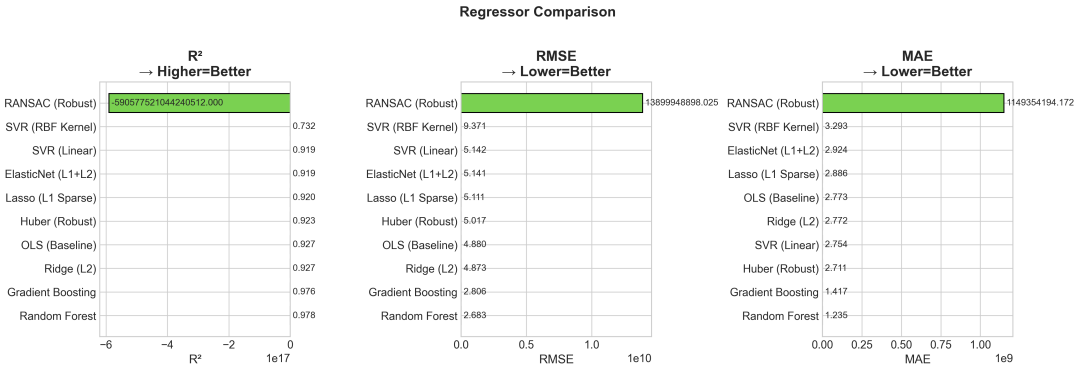

Figure 16: Comparison of regression models. Linear models perform extremely well, implying the transformed explanatory variables capture most variance in MPG.

3.5.2 Accuracy visualization

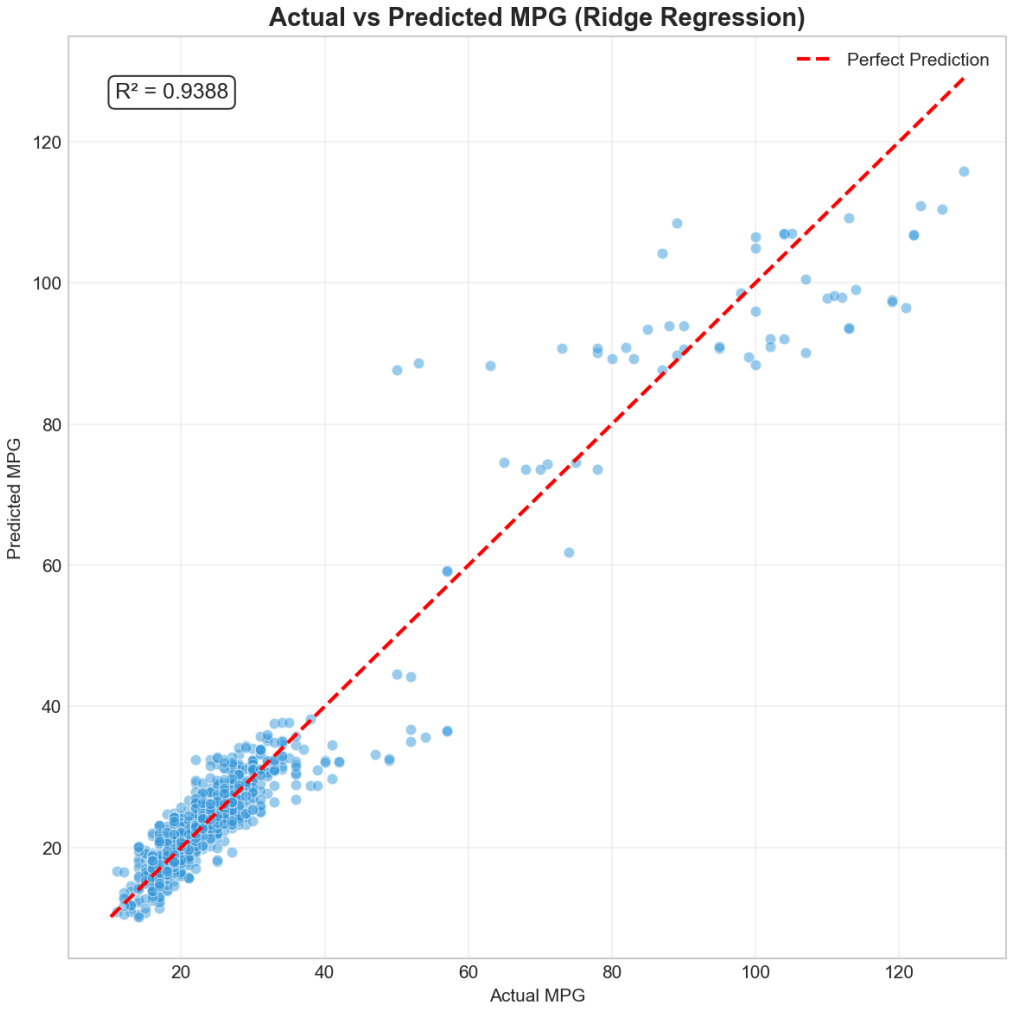

Figure 17: Actual vs. predicted MPG. Points lie close to the diagonal, indicating strong predictive accuracy.

3.5.3 Interpretability: feature importance

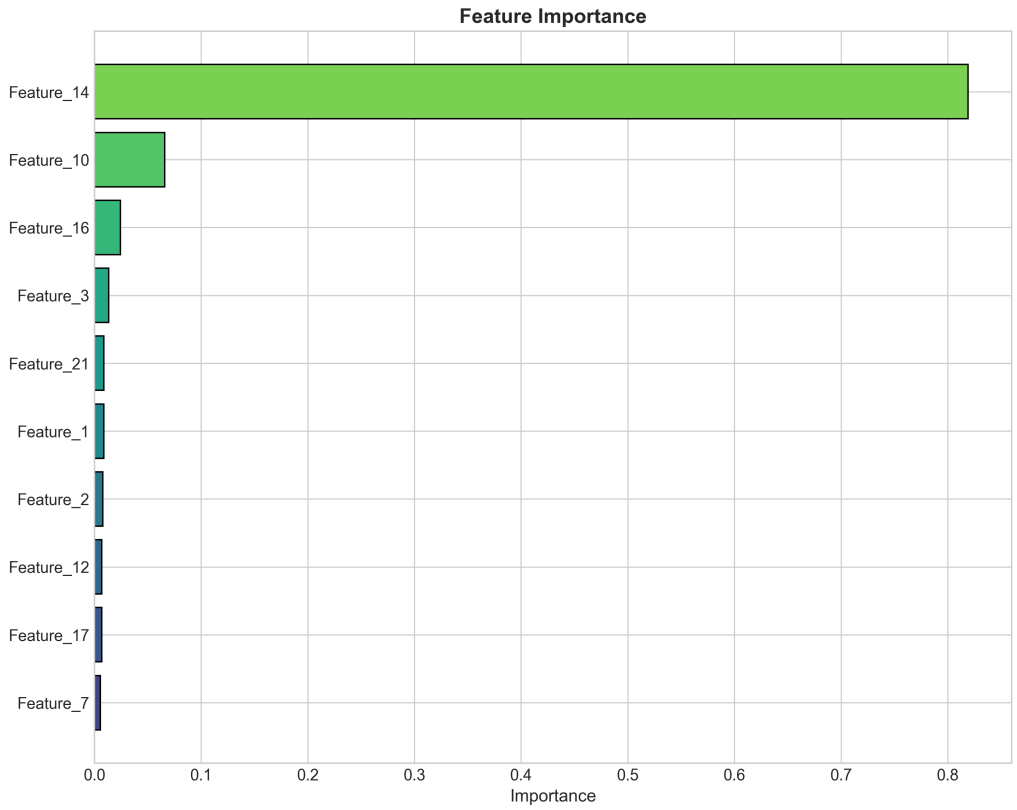

Figure 18: Feature importance for the regression task. Engine-related variables (e.g., displacement/cylinders) and vehicle class are typically among the strongest drivers of MPG.

3.6 Failure Analysis (bias, outliers, and uncertainty)

3.6.1 Residual diagnostics

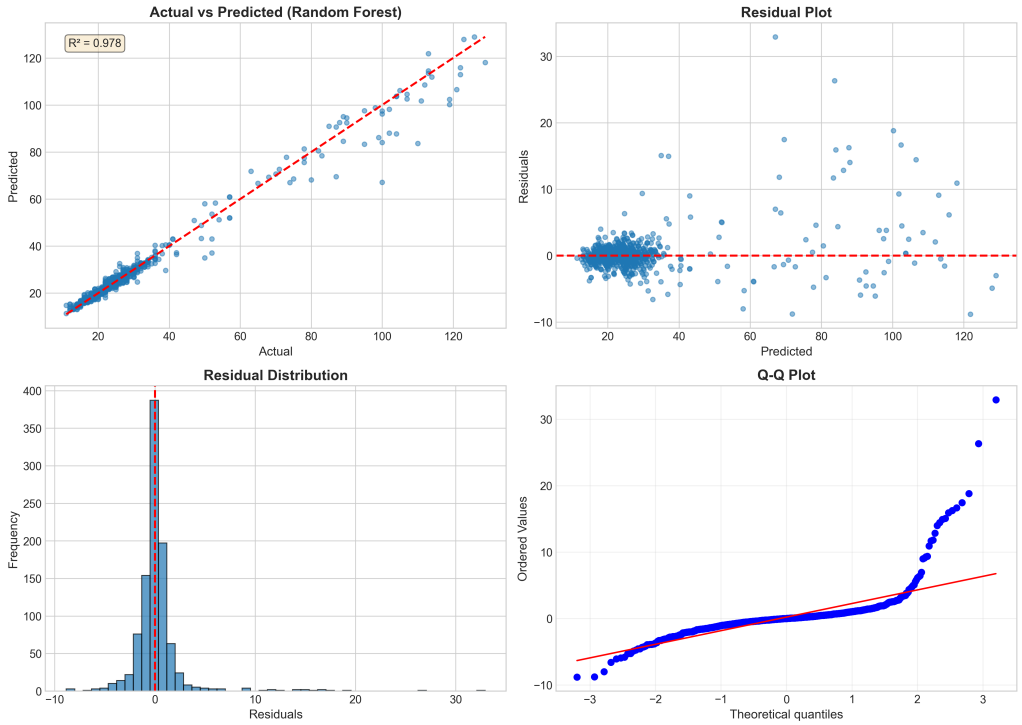

Figure 19: Residual plot. Residuals are approximately centered around zero with a few larger errors at the extremes, consistent with rare vehicle types or unmodelled effects.

3.6.2 Prediction uncertainty

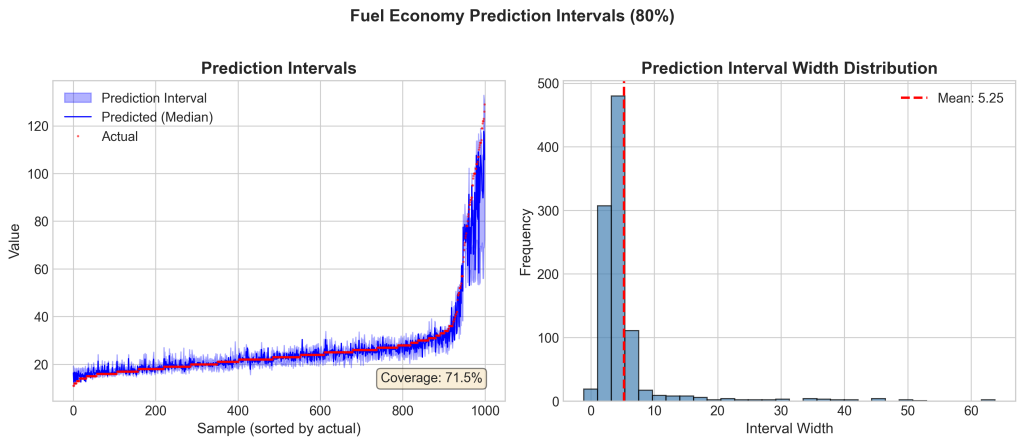

Figure 20: Prediction intervals. In a fleet setting, communicating uncertainty is valuable: decisions (cost estimation and recommended vehicles) should consider error bounds, especially for rare categories.

4 Production Considerations (ABAX deployment)

4.1 Data ingestion and feature computation

- **Classification**: compute rolling-window features (e.g., every 1–5 minutes) and aggregate to trip-level summaries; store both for auditing.
- **Regression**: validate schema at ingestion (units, missing values, category drift); handle unknown categories gracefully.

4.2 Serving, monitoring, and retraining

- **Serving**: package preprocessing + model in one artifact (single pipeline) to prevent training/serving skew.
- **Monitoring**: track feature drift (PSI), prediction drift, and alert on distribution shifts (new vehicle mix, geography, seasonality).
- **Retraining**: implement data/versioned pipelines; retrain on schedule or when drift triggers; validate with the same split logic.

4.3 Governance, privacy, and driver safety

Driver scoring and drowsiness detection are sensitive. A production system should:

- ensure GDPR-compliant handling of personal data and clear consent/communication,
- avoid punitive automated actions on uncertain predictions (use as coaching/assistive signal),
- provide explanations that are understandable (e.g., primary contributing factors and confidence).

4.4 Testing and release process

- Unit tests for data loaders, preprocessing, and model inference.
- Data validation (schema checks, ranges, missingness thresholds).
- Canary evaluation on a subset of fleets/drivers before full rollout.

5 Conclusion

This project demonstrates an end-to-end workflow suitable for ABAX-style telematics problems: careful EDA, feature design with operational constraints in mind, realistic evaluation (driver-level generalization), and clear analysis of failure modes. The current approach provides a solid baseline for a production iteration, with clear next steps: windowed/temporal features for drowsiness, uncertainty-aware outputs, and robust monitoring and retraining.