Fuel Rate (L/hr)

- Based on the Tune, S.S., & Wolf, M. (2015). Estimation of Fuel Consumption from Mass Air Flow and Fuel Trim Data.
- Formula

$$\text{Fuel Rate (L/hr)} = \frac{\text{MAF (g/sec)} \times (1 + \text{STFT} + \text{LTFT}) \times 3600}{\text{AFR} \times \text{Fuel Density (g/L)}}$$

Where:

- **MAF** = Mass Air Flow rate in grams per second (g/sec)
- **STFT** = Average short-term fuel trim (expressed as a decimal, e.g. 5% → 0.05)
- **LTFT** = Average long-term fuel trim (also in decimal form)
- **AFR** = Air–Fuel Ratio (typically **14.7** for gasoline)
- **Fuel Density** = Density of the fuel in g/L (typically **740 g/L** for gasoline)
- **3600** = Seconds in an hour (to convert from g/sec to g/hr)

Explanation:

1. Adjust the MAF using fuel trims:

$$\text{Corrected MAF} = \text{MAF} \times (1 + \text{STFT} + \text{LTFT})$$

2. Convert air mass to fuel mass by dividing by AFR.
3. Convert g/sec → g/hr by multiplying by 3600.
4. Convert fuel mass (g/hr) → volume (L/hr) by dividing by fuel density.

- For normalization technique: Uses Login transformation with MinMaxScaler, because it helps reduces right skewness, compresses high values, and spreads low values—helpful when fuel rates vary drastically. It also transforms values into a fixed [0, 1] range, which is beneficial for models like neural networks.

MAF[g/sec]

- MAF[g/sec] stands for Mass Air Flow, measured in grams per second. It quantifies the amount of air entering the engine per second and is a crucial input for determining fuel injection and combustion efficiency in internal combustion engines.
- According to the eVED paper:
  - The dataset is collected via CAN-bus signals using On-Board Diagnostic (OBD-II) ports.
  - MAF is obtained directly from the vehicle's electronic control unit (ECU) when available.
  - Sampling is done at 1 Hz frequency, i.e., one value per second.
  - If the MAF signal is not directly available, it may be calculated from other engine parameters or remain missing (NaN).
- Linear interpolation to fill Nan is often used for sensor time series data like MAF for several good reasons:

    - Temporal Continuity:
        - Vehicle sensor data is continuous in time. Linear interpolation assumes that the change between two known values is linear — a reasonable assumption over short intervals.
    - Smooth Transitions:
        - Linear interpolation avoids sudden jumps in the values, which preserves the dynamics of the engine airflow pattern without introducing artificial spikes or drops.
    - Preserves Trends:
        - It captures the natural upward or downward trends between measured points — crucial for derived calculations like fuel rate or load.
    - Minimally Invasive:
        - Unlike polynomial or spline interpolation, it doesn't introduce curvature that may not be physically meaningful.

For normalization: The recommendation technique is Min-max scaling

- MAF[g/sec] (Mass Air Flow) is:
    - A non-negative continuous sensor value (typically > 0)
    - Sensitive to vehicle load, engine size, and throttle position
    - Typically bounded within a known range (e.g., 0–200 g/sec for most passenger vehicles)
- Thus, Min-Max Normalization is preferred because:
    - It scales the values between 0 and 1 (or another custom range like [0, 100])
    - Preserves the original distribution shape
    - Ensures interpretability and uniform contribution to ML models (especially neural networks like Transformers)

Vehicle speed

- Vehicle Speed[km/h] represents the instantaneous speed of a vehicle at a given timestamp.
- It is typically measured in kilometers per hour (km/h).
- Values range from 0 (idle) to over 100 km/h depending on road conditions and driving behavior.
- How Is It Recorded? (Based on eVED paper)
    - In the eVED dataset (European Vehicle Emission Dataset), Vehicle Speed is:
    - Collected directly from the vehicle's On-Board Diagnostics (OBD-II) interface, or
    - Derived from GPS-based measurements in some cases, especially if OBD-II is not available.
    - Logged at 1 Hz frequency (i.e., one reading per second) over the duration of a trip.
- Linear interpolation is a good choice for filling missing speed values because:
    - Temporal Continuity
        - Vehicle speed typically changes smoothly over time, especially under normal driving.
    - Short Gaps
        - Missing values in vehicle telemetry are usually short (a few seconds), where linear changes are realistic.

    - Avoids Unrealistic Jumps
        - Linear interpolation ensures speed transitions remain physically plausible.
- As for normalization, the recommended technique is Min-max scalling
    - Vehicle speed naturally falls within a known range (e.g., 0–130 km/h on most roads). Min-max scaling preserves this interpretable range.
    - ```
      Useful when the original distribution is important (e.g., for
      time series or transformers).
      ```
    - Especially helpful when using ReLU-based architectures like Transformers, where 0–1 ranges help gradients propagate better.

Duration

- Duration represent the time of each trip, this is calculated by grouping the trip and vehicle id (Note that each combination of trip and vehicle id is unique), then get the last timestamp - the first timestamp then convert to hours.
- Each timestamp is collected along with the OBD sensors combining with the GPS, which indicates the location of the vehicle at that timestamp.
- For nomarlization, we are using robust scaler because most trip in the data are short trips and right-skewed, the range is moderate, with a long tail due to some very long trips and, outliers may exist due to the few very long trips.

Distance_km

- Distance is calculated by providing the first lattitude and first longtitude, last lattitude and last longtitude then calculate the distance between these points. Due to limited resources, the formula I'm using is Haversine formula, which calculates the great-circle distance between two points on the Earth's surface given their latitudes and longitudes, which is the "as-the-crow-flies" or "sky distance" between two points.

$$\text{distance} = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

🥏 **Where**:

- ϕ_1, ϕ_2: latitudes of the two points (in radians)
- λ_1, λ_2: longitudes of the two points (in radians)
- $\Delta\phi = \phi_2 - \phi_1$
- $\Delta\lambda = \lambda_2 - \lambda_1$
- R: radius of the Earth (typically 6371 km)
- Result: distance in **kilometers**

Why Use the Haversine Formula?

- It accounts for Earth's curvature (unlike Euclidean distance).
- It's accurate for small to medium distances.
- Used widely in navigation and GPS-based distance calculations.
      ```

- No payment needed

Limitations of the Haversine Formula:

- It does not follow road networks.
- It ignores terrain, turns, speed limits, and traffic conditions.
- It underestimates real driving distance, especially in urban or non-grid road layout

Normalization technique used here is RobustScalar because the data contains outlier or skewed distribution, like duration which is expected as shorter time means shorter distance to go

HourofDay

- This columns show when does the time of the journey start.
- Since this can only be a 24hr time, we will normalize it with Min-max scaler

Engine RPM[RPM]

- Engine RPM stands for Engine Revolutions Per Minute — it's a measure of how many times the engine's crankshaft makes a full rotation every minute. This value gives a direct indication of the engine's workload and is critical for analyzing fuel consumption, engine stress, and driving behavior.
- According to the paper and dataset description, data collection includes:
  - Sampling rate is 2 sec
  - Real-time readings from the car's ECU (Engine Control Unit)
  - Logged alongside GPS, fuel rate, speed, MAF, and throttle/load information
- As for filling Nan values, first if the Vehicle speed at that time is 0 than RPM is 0 as in most cars, idling RPM is low (around 600–1000 RPM), but for conservative estimation, setting it to 0 avoids overestimation.
- As for the other Nans, I filled with Linear Interpolation, since
  - RPM changes gradually as the driver accelerates or decelerates.
  - For time-series data like trip-based engine logs, linear interpolation assumes a smooth progression, which is reasonable unless sudden gear shifts or engine braking occur.
  - Grouping by 'Trip' ensures interpolation doesn’t leak values across trips, maintaining logical separation.
- For normalization technique, I am using RobostScaler because:
  - The data appears right-skewed, with some extreme values well above the mean.
  - The standard deviation is relatively large, indicating some spread and possible outliers.
  - The data isn't normally distributed.
  - Retain the influence of central tendency without being distorted by extreme values.

OAT[DegC]

- It measures the ambient temperature outside the vehicle.
- This is not cabin temperature — it reflects real environmental conditions, which can affect:
  - Engine performance
  - Fuel combustion efficiency

    - HVAC (heating, ventilation, air conditioning) usage
    - Emission levels
- The eVED dataset, used in multiple research works (e.g., emissions estimation and eco-driving studies), records OAT as part of on-road sensor logs which records every 60 seconds
- Therefore to fill Nan values in OAT, we uses Linear Interpolation
    - OAT Changes Gradually Over Time
    - Time-ordered Data (like a Trip) Benefits from Sequential Filling
    - Interpolation Maintains Physical Consistency
    - Low Volatility Means Low Error
    - Efficient and Non-intrusive
- As for normalization technique, we will be using Robustscaler
    - Robust to outliers like extreme cold (-40°C) or heat (60°C), which can distort StandardScaler or MinMaxScaler.
    - Centers on median and scales by IQR (Interquartile Range), which is useful since your data shows a skew.

Absolute Load[%]

- Absolute Load is a parameter measured from the Engine Control Unit (ECU) via OBD-II that reflects the actual engine load compared to its theoretical maximum under current conditions. It's usually expressed as a percentage (%).
- In the dataset (e.g. eVED or similar OBD-II datasets), Absolute Load [%] is logged by:
    - The vehicle's OBD-II interface
    - Measured in real-time from the ECU via PID 43 (per SAE standard)
    - Typically recorded once every 5 second, alongside other engine parameters (e.g., RPM, speed)
- As for filling Nan:
    - When the vehicle is stationary (speed = 0), the engine typically runs at idle, resulting in low or near-zero load.→ This rule is physically meaningful and avoids artificial spikes when filling NaNs.
    - Next in linear Interpolation for each trip as Absolute Load changes smoothly during driving (except during sudden throttle changes). Interpolating linearly between known values preserves the trend. Grouping by Trip ensures that interpolation is done within continuous journeys, not across unrelated events.
- As for normalization technique, we will be using RobustScaler
    - RobustScaler uses the median and interquartile range (IQR), making it resistant to outliers

Speed limit and Speed limit direction

- Both indicates the speed limit of that road, but speed limit direction include only the speed limit of the direction the car was driving along whereas speed limit indicate the max speed of the road.
- All nan values here are fill with 0 indicating no speed limit
- As for normalization, we use RobustScaler for both features to ensures the outliers dont effect the result

Intersection and bus stop

- The features are from counting the number of intersection and bus stop of each group by of trip and vehicle ID
- Since the distribution is not normalize, we also applied robust scaler to avoid outliers effecting the result

Elevation Raw[m] and Elevation Snoothed[m]

- This parameters measure how high the road is compare to sea level.
- The integrating of elevation information in the eVED dataset is intuitive since it has a direct impact on vehicle behavior, e.g., traversing up an incline requires more energy than a flat surface or decline.
- Due to different sampling intervals in the VED dataset,part of the recorded GPS coordinates are duplicated across several consecutive records because of the lower polling rate of GPS. Furthermore, duplication of coordinates induces the extracted elevation data to be stepped.
- To mitigate such impact, we smooth the extracted elevation using a “five-point” average approach. This result the elevation smoothed column.
- We will be using both in our training.
- As for normalization, since there is no extreme scaling in elevation we will be using min-max scaling
    - Min-Max Scaler will preserve the distribution shape while scaling values to [0, 1].
    - Ideal when no extreme outliers are present

Generalized weght

- This column provides the vehicle weights
- Normalization technique use here is also robust scaler sine the data is right schkew

Finally is Fuel_consump

- This is an additional feature to get the fuel consumption in Liters of a trip, it gets from formula:
- Fuel_consump = Fuel Rate * Duration
- For normalization, we will be using robust scaler to avoid outliers, it is also worth mention that both fueal rate and duration also uses this normalize method

Engine Configuration & Displacement

- This column represent teh engine and displacement of the vehicle
- Since Encoding it all together is not sufficent, we will split this column into 3 columns:
    - Engine Config: This is the number of engine cylinders. More cylinders often mean higher power output but lower fuel efficiency. This usually is the first digit before hyphen.

    - Engine_Tags: The type of fuel delivery or engine configuration. Indicates fuel efficiency, engine responsiveness, or hybrid type. From the middle part (after the hyphen and before displacement).
    - Displacement_L: Total volume of all engine cylinders (in liters), Affects power output and fuel consumption. It extracts from the numeric value with the L suffix
- As for the Nans values in Displacement_L, this could be due to a fully electric engine (no displacement), or parsing issue (unstructured entry).
    - To fix it, I first check if the engine is electric, if yes then fill with 0, the others is then filled with median
- As for Engine_Config Missing likely due to poor parsing (e.g., odd string format like 3.0L 6cyl 4A or H-4 2.0 L/122).
    - To fix it, we first try a second-pass parsing (regex fallback), else we just filled with Unkown.
- To now we apply Robust Scaler for Displacement_L and Label Encoder which for both Engine_Config and Engine_Tags

Vehicle Type

- This is the type of the vehicle to be either ICE, HEV, PHEV or EV. We encode this with Label Encoder as well