Building a machine learning (ML) model to **automatically recognize vehicles for fuel ban enforcement** (based on age, fuel type, etc.) would involve several stages and components. Below is a structured guide outlining the **key parameters**, **data requirements**, and **modeling strategies**.

**🔍 Primary Objective**

Automatically **recognize a vehicle** at a fuel station and determine **whether it should be allowed to refuel or not**, based on:

* Vehicle age
* Fuel type
* Registration validity
* Compliance with emission norms

**🔧 Parameters & Features to Consider**

**1. Vehicle Identification**

* **License Plate Number** (via Automatic Number Plate Recognition - ANPR)
* **State of Registration** (to check jurisdiction)
* **Vehicle Type** (2W, 4W, commercial, private)

📦 Data Source: High-res camera footage + Government registration databases (e.g., Vahan API in India)

**2. Vehicle Attributes from Database**

Once the plate is recognized, fetch:

* **Make and Model**
* **Fuel Type** (Petrol / Diesel / CNG / EV)
* **Registration Date** (to compute age)
* **Emission Norm** (BS3/BS4/BS6)
* **Vehicle Class** (e.g., LMV, HMV, private, commercial)
* **Blacklist/Scrap status**

**3. Environmental Factors**

(Optional, for smart enforcement prioritization)

* **Air Quality Index (AQI)** at the station
* **Time of Day/Traffic Density**
* **Geolocation of Fuel Station** (is it inside the restricted zone?)

**4. Behavioral/Usage Data (if available)**

* Frequency of refueling
* Cross-border refueling patterns (to avoid ban)
* Violation history

**📊 Machine Learning Model Design**

**A. Model Type**

Depending on scope:

| **Task** | **Model Type** |
| --- | --- |
| License plate detection | Object Detection (YOLOv5, YOLOv8, Faster R-CNN) |
| Plate-to-text recognition | OCR (Tesseract, EasyOCR, CRNN) |
| Allow/Block Decision | Rule-based logic or classifier (if fuzzy data) |
| Vehicle detection (entire view) | Image Classification (ResNet, VGG) |

**B. Workflow Pipeline**

text

CopyEdit

1. Input → Image/Video Feed from camera

2. Preprocess → Clean frames, apply image enhancement

3. Detect → License Plate via object detection (YOLO etc.)

4. OCR → Extract vehicle number

5. API Call → Fetch vehicle details from government DB

6. Decision Logic → Check age, fuel type, registration

7. Output → Refuel: YES / NO | Raise alert if NO

**🗃️ Required Datasets**

* **ANPR dataset** (Indian license plates, various angles/lighting)
* **Vehicle registration dataset** (synthetic or via mock APIs)
* **Fuel station footage** (simulated if real data not available)
* **Blacklist dataset** (scrapped vehicles)

**📚 Tools & Libraries**

* **Computer Vision**: OpenCV, YOLO, TensorFlow/Keras
* **OCR**: Tesseract, EasyOCR
* **Backend/Database**: FastAPI + PostgreSQL or MongoDB
* **APIs**: VAHAN (for vehicle data) — can be simulated
* **Deployment**: NVIDIA Jetson (for edge deployment), or cloud via AWS Lambda/API Gateway

**🧠 Bonus: Smart Enhancements**

* **Real-time alerts to enforcement** team via dashboard
* **Face detection** of the driver (in future extensions)
* **Fraud detection** (e.g., tampered plates, spoofing via ML anomaly detection)

**✅ Decision Criteria Logic Example (Post-Data Fetch)**

python

CopyEdit

def should\_allow\_refueling(vehicle):

if vehicle['fuel\_type'] == 'CNG':

return True

if vehicle['fuel\_type'] == 'Diesel' and vehicle['age'] > 10:

return False

if vehicle['fuel\_type'] == 'Petrol' and vehicle['age'] > 15:

return False

if vehicle['scrap\_status'] == True:

return False

return True

Would you like help prototyping the pipeline (code level)? Or want a UI design for a dashboard to display the results to fuel station staff?