Material Identification System (ML Project)

1. Project Overview

This project implements a Machine Learning pipeline to identify recyclable materials (Glass, Metal, Paper, Plastic, Cardboard, Trash) from images. The system is designed to work both on a static dataset and in a real-time environment using a webcam.

The core objective was to build a robust classifier that balances accuracy with real-time performance, handling challenges like lighting variations and background noise.

2. Methodology & Pipeline

A. Feature Extraction

We avoided using raw pixels (which are sensitive to noise) and instead extracted two types of features:

1. **HOG (Histogram of Oriented Gradients):** Captures the *shape* and *texture* of the object (e.g., the edges of a bottle vs. the flat surface of cardboard).
2. **Color Histograms:** Captures the *color distribution* (e.g., the brown of cardboard vs. the transparency/white of glass).

- **Total Feature Vector:** ~8,000 dimensions per image.

B. Preprocessing

- **Label Encoding:** Converted class names strings to integers (0-5).
- **Stratified Split:** Used an 80/20 train-test split, ensuring balanced representation of all classes in both sets.
- **Standard Scaling:** Applied StandardScaler to normalize features to Mean=0 and Variance=1. This was critical because the Color features (0-255) had a different scale than HOG features (0-1), which would otherwise bias the distance calculations in SVM and k-NN.

3. Model Selection & Justification

We compared two supervised learning algorithms: **Support Vector Machines (SVM)** and **k-Nearest Neighbors (k-NN)**.

Experiment 1: k-Nearest Neighbors (k-NN)

- **Configuration:** k=5, Metric=Euclidean.
- **Result:** ~73% Accuracy.
- **Analysis:** k-NN struggled with the "Curse of Dimensionality." With over 8,000 features, the distance between data points became less meaningful, and the model was sensitive to noise in the training data.

Experiment 2: Support Vector Machine (SVM) - *Selected Model*

- **Configuration:** Kernel=RBF (Radial Basis Function), C=10, Probability=True.
- **Result:** ~76% Accuracy.
- **Justification:**
    - **Why SVM?** SVMs are generally more effective in high-dimensional spaces compared to k-NN.
    - **Why RBF Kernel?** Initial tests with a Linear kernel failed to distinguish between **Glass** and **Plastic** (both are shiny and smooth). The RBF kernel projects data into higher dimensions, allowing for non-linear separation boundaries between these similar classes.

4. Real-Time Deployment Challenges

Deploying the model to a webcam introduced a **Domain Gap** (the difference between clean dataset images and a noisy webcam feed). We implemented three specific strategies to solve this:

1. **Region of Interest (ROI):** We implemented a "Scan Zone" (Green Box) in the center of the frame. Features are extracted *only* from this box, ignoring the user's room background.
2. **Confidence Thresholding & "Unknown" Class:**
    - The model often outputs weak predictions (e.g., 40% confidence) for empty space or unknown objects.
    - **Logic:** If Max_Probability < 0.60, the system automatically assigns **Class ID 6 (Unknown)** instead of forcing a wrong prediction. This significantly reduces False Positives.
3. **Frame Skipping:** To maintain high FPS, we run the heavy ML inference only every 5th frame, while the lightweight UI updates every frame.

5. How to Run

1. **Install Requirements:**

Bash

pip install -r requirements.txt

2. **Train the Model:**

Bash

python src/train_svm.py

This generates: models/svm_model.joblib, models/scaler.joblib, models/label_encoder.joblib

3. **Run Real-Time App:**

Bash

python src/realtime_app.py

- *Usage:* Place object inside the green box. Press q to quit.

6. Conclusion

The SVM model with RBF kernel proved to be the superior architecture for this task, outperforming k-NN by ~3%. The addition of confidence thresholding allows the system to be "safe" in real-world use, preventing it from wildly guessing when it is unsure.