### **1. Project Overview**

**Overview:**This project focuses on detecting household and personal care items using image uploads. By utilizing OCR (Optical Character Recognition) and machine learning models, the application extracts text from product images and classifies them by brand and category. The solution leverages Streamlit for user interaction and integrates easyOCR for text extraction.

### **2. Technologies Used**

List of the tools, frameworks, and technologies we used in the project:

* **Python**: Main programming language.
* **Streamlit**: For building the web interface.
* **EasyOCR**: For text extraction from images.
* **Pillow (PIL)**: For image handling and manipulation.
* **Pickle**: For saving and loading machine learning models.
* **Difflib**: For string matching and similarity scoring.

### **3. Model Training and Dataset**

Here are details about how the models were trained and the datasets used:

* **Personal Care Model**: Trained on a dataset of personal care items, such as skin care and hair care products.
* **Household Model**: Trained on a dataset of household items, such as cleaning supplies and cooking oils.

### **4. OCR Implementation Details**

In-depth explanation of how the OCR process works:

* The project uses **EasyOCR** to extract text from uploaded images.
* Optionally include details about preprocessing techniques (e.g., resizing images, improving image clarity) to enhance OCR accuracy.

### **5. Brand and Category Matching Logic**

Explain how brand and category are identified:

* **String Matching**: Describe how extracted text is compared to predefined brand names using the **difflib** library.
* **Category Assignment**: Elaborate on how different brands are mapped to their respective categories based on a dictionary.

### **6. Challenges and Improvements**

You can include some challenges encountered during the development of the project:

* **OCR Limitations**: Issues related to noisy images, poor resolution, or incorrect text extraction.
* **Model Misclassification**: Mention how certain brands or products might be misclassified and steps to improve the detection process.

**Future Improvements**:

* Expanding the brand and category dictionaries.
* Improving the accuracy of text extraction and classification(We can switch to **Paddle-OCR)**

### **7. User Experience**

* Simplicity of the user interface and how Streamlit helps users easily upload images and view results in real-time.
* Mention additional features such as:
  + Timestamped detections.
  + Suggestions for further actions, like visiting a brand’s website or product page.

### **8. Deployment**

Explain how the project is deployed (e.g., on **Streamlit Cloud**, **Heroku**, or any other platform). Mention any deployment challenges, such as package compatibility issues or environment setup.

* We faced certain issue for Deploying using the Streamlit Cloud Platform hence it was working in my Local Host , I have attached the working in the video( )

**GITHUB CODE**:<https://github.com/ipsita-kar/Brand-Detection>

**VIDEO LINK OF WORKING:**

**CODE DOCUMENTATION:**

import streamlit as st

from PIL import Image

import os

import easyocr

import pickle

import difflib

from datetime import datetime

# Load the OCR models (for classification)

def load\_model(model\_path):

with open(model\_path, 'rb') as model\_file:

return pickle.load(model\_file)

# Load models for personal care and household items

personal\_care\_model = load\_model('personal\_care\_ocr\_model (1).pkl')

household\_model = load\_model('household\_ocr\_model (1).pkl')

# Comprehensive brand and category data

brand\_dict = {

"Dove": "Personal Care: Skin Care",

"Nivea": "Personal Care: Skin Care",

"Fortune": "Food: Cooking Oil",

"L'Oreal": "Personal Care: Hair Care",

"Patanjali": "Personal Care: Ayurvedic Products",

"Colgate": "Oral Care: Toothpaste",

"Pantene": "Personal Care: Hair Care", # Added Pantene

"Ariel": "Household: Laundry Detergent",

"Rin": "Household: Laundry Detergent",

"Vim": "Household: Dishwashing Soap",

"Haldiram's": "Household: Food Products",

"Tata Salt": "Household: Salt",

"Dettol": "Household: Disinfectant",

"Amul": "Food: Dairy Products",

"Maggi": "Food: Instant Noodles",

# Add more brands as needed...

}

# Define categories based on the brand dictionary

category\_dict = {

"personal care": ["cream", "moisturizing", "lotion", "shampoo", "soap", "toothpaste", "deodorant", "bar"],

"household": ["cleaner", "detergent", "spray", "cooking oil"],

"food": ["salt", "chips", "snacks", "oil", "biscuits", "milk", "yogurt", "cheese", "butter"],

# Add more categories as needed...

}

# Dictionary for additional details

details\_dict = {

"Dove": "Dove is a personal care brand specializing in moisturizing products, soaps, and body wash.",

"Nivea": "Nivea offers a wide range of skin care products, including lotions, creams, and deodorants.",

"L'Oreal": "L'Oreal is a leading beauty and cosmetics brand offering hair care and skincare products.",

"Ariel": "Ariel is a popular laundry detergent brand known for its stain-removing properties.",

"Rin": "Rin is a laundry detergent brand that provides effective cleaning for clothes.",

"Vim": "Vim is a well-known brand offering dishwashing soap and cleaners.",

"Fortune": "Fortune is a leading brand of cooking oils, including sunflower and mustard oil.",

"Dettol": "Dettol offers disinfectants, antiseptic liquids, and health hygiene products.",

# Add more brand details as needed...

}

# OCR text extraction using EasyOCR with optional preprocessing

def extract\_text(image\_path):

reader = easyocr.Reader(['en'])

result = reader.readtext(image\_path, detail=0) # Extract text from image

extracted\_text = " ".join(result).lower() # Convert to lowercase for consistency

return extracted\_text

# Function to match extracted text with known brands

def identify\_brand(extracted\_text):

extracted\_words = extracted\_text.split() # Split the text into words

for word in extracted\_words:

for brand in brand\_dict.keys():

# Check for close matches on a word-by-word basis

if difflib.get\_close\_matches(brand.lower(), [word.lower()], n=1, cutoff=0.6):

return brand, brand\_dict[brand]

# If no brand is matched, return extracted text as brand

return extracted\_text, "Not Recognized"

# Function to fetch additional details from the details\_dict

def fetch\_details(item):

# Check if item is in the details dictionary

if item in details\_dict:

return details\_dict[item]

else:

return "Please visit Flipkart for more information."

# Main Streamlit app function

def run():

st.title("Household and Personal Care Item Detection")

# Create the upload directory if it doesn't exist

upload\_dir = './uploaded\_images'

if not os.path.exists(upload\_dir):

os.makedirs(upload\_dir)

# Upload image

img\_file = st.file\_uploader("Choose an Image", type=["jpg", "png", "webp"])

if img\_file is not None:

img = Image.open(img\_file).resize((250, 250))

st.image(img, use\_column\_width=False)

save\_image\_path = os.path.join(upload\_dir, img\_file.name)

with open(save\_image\_path, "wb") as f:

f.write(img\_file.getbuffer())

# OCR to extract text from image

extracted\_text = extract\_text(save\_image\_path)

st.write("Extracted Text: ", extracted\_text) # Debug output for extracted text

# Identify the brand and category

brand, category = identify\_brand(extracted\_text)

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S") # Add timestamp

st.write(f"Brand Identification: {brand}, {category} (Detected at: {timestamp})") # Debug output for brand identification

if brand and category != "Not Recognized":

st.success(f"\*\*Brand Detected:\*\* {brand}")

st.info(f"\*\*Category:\*\* {category}")

st.write(f"Detected at: {timestamp}")

# Fetch additional details from dictionary

details = fetch\_details(brand)

st.warning(f"\*\*Details:\*\* {details}")

else:

st.warning(f"\*\*Extracted Text (No Brand Detected):\*\* {brand}")

if \_\_name\_\_ == "\_\_main\_\_":

run()