CLEANTECH: TRANSFORMING WASTE MANAGEMENT WITH TRANSFER LEARNING

MENTOR NAME – SRI L LAKSHMI NARAYANA

TEAM ID - LTVIP2025TMID40420

TEAM LEADER

NAME - M MANIKANTA

MAIL - maniroymulliqudi@gmail.com

ID NO: SBAP0040420

TEAM MEMBERS

MEMBER-1

NAME - M CHINNA REDDY

MAIL - youthreddy93@gmail.com

ID NO: SBAP0040482

MEMBER-2

NAME - MD AYUB

MAIL - mdayub37407@gmail.com

ID NO: SBAP0040440

MEMBER-3

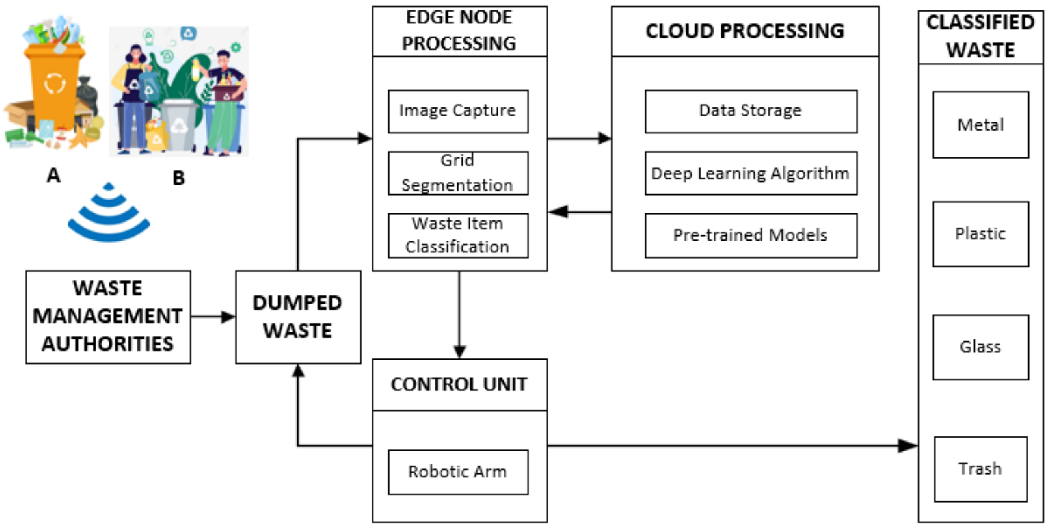
NAME - CH LAKSHMI GANAPATHI

MAIL - ganapathichelluriganapathi@gmail.com

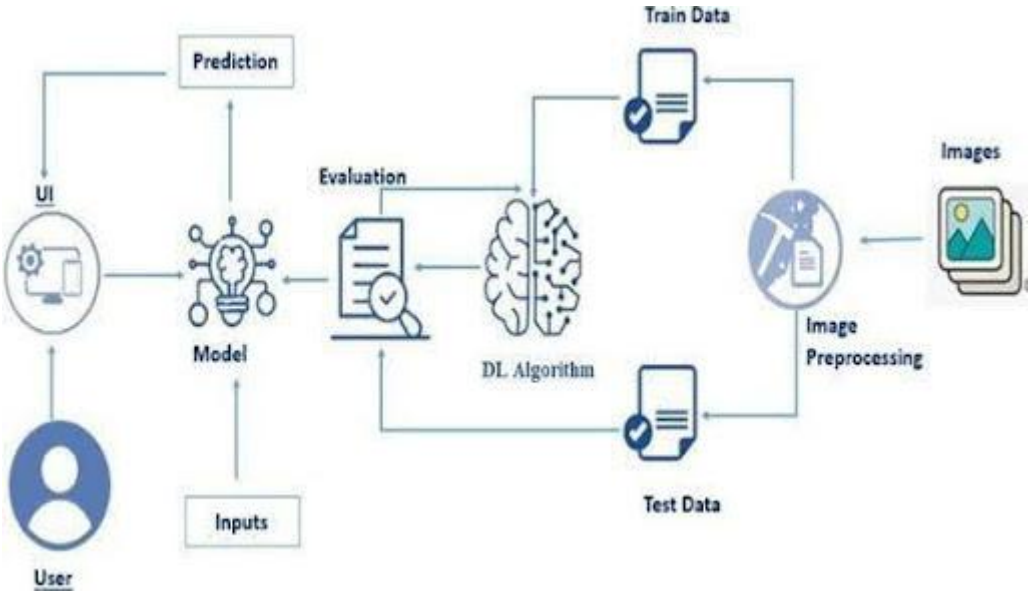
ID NO: SBAP0040446

INTRODUCTION:

Waste management is a growing global concern as urban populations expand and consumption increases. Traditional waste sorting and disposal methods are often inefficient, labor-intensive, and environmentally damaging. **CleanTech** leverages the power of artificial intelligence, particularly **Transfer Learning**, to revolutionize the way waste is identified, classified, and processed. By applying pre-trained deep learning models to the task of waste categorization, CleanTech enables faster, more accurate, and cost-effective waste management solutions. This approach not only improves recycling efficiency but also supports sustainability efforts by minimizing human error and reducing environmental impact.



ARCHITECTURE:



The image illustrates a typical **deep learning model pipeline** used in AI-based image classification tasks, such as smart waste management:

1. **User Interface (UI)**: The system starts with the user providing input through a user interface. This could be an image of waste to be classified.
2. **Inputs**: These are raw data inputs (e.g., images of waste items) sent into the model.
3. **Model**: This represents the core AI model that performs predictions using trained knowledge. In this case, it's a deep learning model enhanced by **Transfer Learning**.
4. **Prediction**: The model processes the inputs and makes predictions, such as classifying waste into categories (e.g., plastic, metal, organic).
5. **Evaluation**: The model's performance is evaluated using various metrics to ensure accuracy and reliability.
6. **DL Algorithm (Deep Learning Algorithm)**: This is the underlying neural network responsible for learning patterns in the data. It connects the evaluation process back to the training stage to fine-tune the model.
7. **Train and Test Data**: The system uses two main data sets — one for training and one for testing. The **training data** helps the model learn, while the **test data** evaluates how well it performs on unseen examples.
8. **Image Preprocessing**: Before feeding images to the model, they undergo preprocessing (e.g., resizing, normalization) to ensure consistency and efficiency.
9. **Images**: These are the actual visual data inputs, such as photos of waste items, used to train and test the model.

Prerequisites:

Prior Knowledge:

- **DL Concepts**
    - **Neural Networks**:: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
- **Deep Learning Frameworks**:: https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow
- **Transfer Learning**: https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a
- **VGG16**: https://www.geeksforgeeks.org/vgg-16-cnn-model/
- **Convolutional Neural Networks (CNNs)**: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- **Overfitting and Regularization:** https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/
- **Optimizers**: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
- **Flask Basics**: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

1. Fundamental Concepts and Techniques in Deep Learning

By the end of this project, you'll understand core deep learning concepts like:

- Neural Networks: Basics of how perceptrons combine into layers to model complex patterns.
- Activation Functions: Such as ReLU, sigmoid, and tanh, which help neural networks learn non-linear relationships.
- Backpropagation: A technique for training neural networks by adjusting weights to minimize loss.

- **Loss Functions: Such as Mean Squared Error (MSE) for regression or Cross Entropy for classification.**
- **Optimizers: Like SGD, Adam, and RMSProp, which help in minimizing the loss function effectively.**

These are the building blocks for creating any deep learning model.

2. Broad Understanding of Data

You'll gain insight into:

- **Types of Data: Structured vs. unstructured, labeled vs. unlabeled.**
- **Feature Engineering: Selecting and transforming variables to improve model performance.**
- **Data Splitting: Train/test/validation sets to ensure generalization.**
- **Class Imbalance Handling: Techniques like oversampling, undersampling, or synthetic data generation (e.g., SMOTE).**

Understanding your data is 80% of a successful machine learning pipeline.

3. Preprocessing and Outlier Handling Techniques

You will learn how to prepare data for modeling:

Preprocessing Techniques:

- **Normalization/Standardization: Bring data into the same scale (e.g., Min-Max Scaling, Z-score).**
- **Handling Missing Values: Imputation (mean, median, mode) or removal.**
- **Encoding Categorical Variables: Label encoding or one-hot encoding.**

Outlier Detection and Handling:

- **Z-score Method: Data points with Z-scores above a threshold are considered outliers.**
- **IQR Method: Outliers lie outside the interquartile range.**
- **Box Plots & Scatter Plots: Visual tools to detect outliers.**

Clean and preprocessed data improves model accuracy and reduces errors.

4. Visualization Concepts

Data visualization helps in understanding patterns and insights:

- Histograms: Show frequency distribution of features.
- Scatter Plots: Show relationships between two variables.
- Correlation Heatmaps: Show how features relate to each other.
- Pairplots (using seaborn): Useful for visualizing clusters and separability.

Visualization is a powerful tool for both EDA (Exploratory Data Analysis) and presenting results.

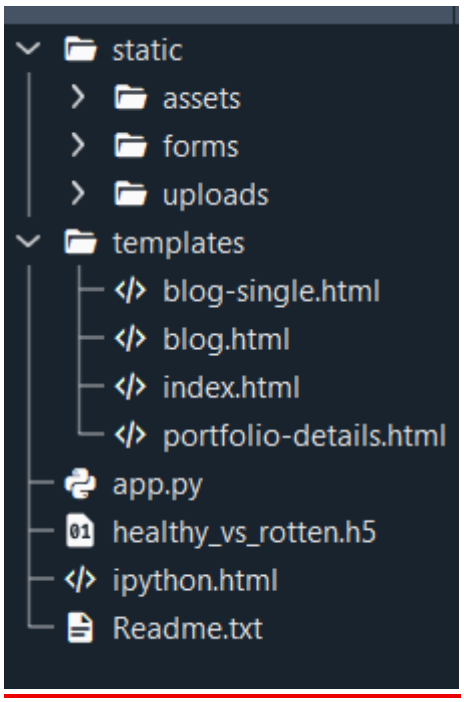
Project Flow:

- **The user interacts with the UI (User Interface) to choose the image.**
- **The chosen image is analyzed by the model which is integrated with the flask application.**
- **Once the model analyses the input the prediction is showcased on the UI.**

To accomplish this, we have to complete all the activities listed below,

- **Data Collection: Collect or download the dataset that you want to train.**
- **Data pre-processing**
  - **Data Augmentation**
  - **Splitting data into train and test**
- **Model building**
  - **Import the model-building libraries**
  - **Initializing the model**
  - **Training and testing the model**
  - **Evaluating the performance of the model**
  - **Save the model**
- **Application Building**
  - **Create an HTML file**
  - **Build python code**

Project Structure:



- **We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.**
- **Vgg16.h5 is our saved model. Further, we will use this model for flask integration**

Data Collection and Preparation

Collect the dataset:

In this project, we have used 3 classes of biodegradable, recyclable and trash images data. This data is downloaded from kaggle.com or can be connected by using API. Please refer to the link given below to download the dataset.

Link: Dataset

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

```
import os
import shutil
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

- **Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.**

At first, unzip the data and convert it into a pandas data frame.

```
# Set the path to the dataset
dataset_dir = '/content/Dataset'
classes = os.listdir(dataset_dir)

# Create directories for train, val, and test sets
output_dir = 'output_dataset'
os.makedirs(output_dir, exist_ok=True)
os.makedirs(os.path.join(output_dir, 'train'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'val'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'test'), exist_ok=True)

for cls in classes:
    os.makedirs(os.path.join(output_dir, 'train', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'val', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'test', cls), exist_ok=True)

    class_dir = os.path.join(dataset_dir, cls)
    images = os.listdir(class_dir)

    print(cls,len(images))

    train_and_val_images, test_images = train_test_split(images, test_size=0.2, random_state=42)
    train_images, val_images = train_test_split(train_and_val_images, test_size=0.25, random_state=42)  # 0.25 x 0.8 = 0.2

    # Copy images to respective directories
    for img in train_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'train', cls, img))
    for img in val_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'val', cls, img))
    for img in test_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'test', cls, img))

print("Dataset split into training, validation, and test sets.")
```

```
# Define directories
dataset_dir = '/content/output_dataset'
train_dir = os.path.join(dataset_dir, 'train')
val_dir = os.path.join(dataset_dir, 'val')
test_dir = os.path.join(dataset_dir, 'test')

# Define image size expected by the pre-trained model
IMG_SIZE = (224, 224)  # Common size for many models like ResNet, VGG, MobileNet

# Create ImageDataGenerators for resizing and augmenting the images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Load and resize the images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'  # Assuming binary classification for healthy vs rotten
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    shuffle=False  # Do not shuffle test data
)

# Print class indices for reference
print(train_generator.class_indices)
print(val_generator.class_indices)
print(test_generator.class_indices)
```

Output :

```
healthy 100
rotten 100
Dataset split into training, validation, and test sets.

{'healthy': 0, 'rotten': 1}
{'healthy': 0, 'rotten': 1}
{'healthy': 0, 'rotten': 1}
```

Data Visualization:

The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.

```
import random
from IPython.display import Image, display

# Specify the path to your image folder
folder_path = '/content/output_dataset/train/Biodegradable Images'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

In the above code, I used class biodegradable class for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as biodegradable image.

```
folder_path = '/content/output_dataset/test/Recyclable Images'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

In the above code, I used recyclable class for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as recyclable image.

```
folder_path = '/content/output_dataset/test/Trash Images'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

in the above code, I used trash class for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as trash image.

```
folder_path = '/content/output_dataset/train/Trash Images'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

In the above code, I used class for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as trash image.

Data Augmentation:

Data augmentation is a technique commonly employed in machine learning, particularly in computer vision tasks such as image classification, including projects like the healthy vs rotten Classification in fruits and vegetables. The primary objective of data augmentation is to artificially expand the size of the training dataset by applying various transformations to the existing images, thereby increasing the diversity and robustness of the data available for model training. This approach is particularly beneficial when working with limited labeled data.

In the context of the 28 class Classification, data augmentation can involve applying transformations such as rotation, scaling, flipping, and changes in brightness or contrast

Unseen data.

This is a crucial step but this data is already cropped from the augmented data so. this time it is skipped accuracy is not much affected but the training time increased.

Split Data and Model Building

Train-Test-Split:

In this project, we have already separated data for training and testing.

```
trainpath = "/content/output_dataset/train"
testpath="/content/output_dataset/test"

train_datagen = ImageDataGenerator(rescale = 1./255,zoom_range= 0.2,shear_range= 0.2)
test_datagen = ImageDataGenerator(rescale = 1./255)

train = train_datagen.flow_from_directory(trainpath,target_size =(224,224),batch_size = 20)
test = test_datagen.flow_from_directory(testpath,target_size =(224,224),batch_size = 20)
```

```
Found 234 images belonging to 3 classes.
Found 78 images belonging to 3 classes.
```

Model Building:

Vgg16 Transfer-Learning Model:

The VGG16-based neural network is created using a pre-trained VGG16 architecture with frozen weights. The model is built sequentially, incorporating the VGG16 base, a flattening layer, dropout for regularization, and a dense layer with SoftMax activation for classification into five categories. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss. During training, which spans 10 epochs, a generator is employed for the training data, and validation is conducted, incorporating call-backs such as Model Checkpoint and Early Stopping. The best-performing model is saved as "healthy_vs_rotten.h5 " for potential future use. The model summary provides an overview of the architecture, showcasing the layers and parameters involved.

vgg16

```
vgg = VGG16(include_top = False,input_shape = (224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
```

```
for layer in vgg.layers:
  print(layer)
```

Show hidden output

```
for layer in vgg.layers:
  layer.trainable = False
```

```
x= Flatten()(vgg.output)
```

```
output = Dense(3, activation ='softmax')(x)
```

```
vgg16 = Model(vgg.input,output)
```

```
vgg16.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
```

```
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
opt = Adam(lr=0.0001)

# Assuming you have defined your VGG16 model as vgg16

# Define Early Stopping callback
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True)

# Compile the model (you may have already done this)
vgg16.compile(optimizer = 'Adam' , loss='categorical_crossentropy', metrics=['accuracy'])


# # Train the model with early stopping callback
history = vgg16.fit(train, validation_data=test,
                    epochs=10,

                  steps_per_epoch=5,
                 callbacks=[early_stopping])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Epoch 1/10
5/5 [==============================] - 120s 24s/step - loss: 1.4950 - accuracy: 0.5300 - val_loss: 1.1408 - val_accuracy: 0.5513
Epoch 2/10
5/5 [==============================] - 111s 25s/step - loss: 0.7526 - accuracy: 0.6600 - val_loss: 0.7541 - val_accuracy: 0.6667
Epoch 3/10
5/5 [==============================] - 105s 23s/step - loss: 0.5088 - accuracy: 0.7766 - val_loss: 0.6151 - val_accuracy: 0.6923
Epoch 4/10
5/5 [==============================] - 107s 24s/step - loss: 0.3544 - accuracy: 0.8936 - val_loss: 0.6226 - val_accuracy: 0.7821
Epoch 5/10
5/5 [==============================] - 104s 23s/step - loss: 0.2693 - accuracy: 0.8936 - val_loss: 0.9809 - val_accuracy: 0.7051
Epoch 6/10
5/5 [==============================] - 108s 24s/step - loss: 0.2384 - accuracy: 0.9300 - val_loss: 0.8708 - val_accuracy: 0.7179
Epoch 7/10
5/5 [==============================] - 116s 26s/step - loss: 0.1812 - accuracy: 0.9500 - val_loss: 0.6773 - val_accuracy: 0.7436
```

Testing Model & Data Prediction:

Testing the model

Here we have tested with the Vgg16 Model With the help of the predict () function.

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
labels=[0,1,2]
```

Test-1

```
img_path ='/content/output_dataset/train/Biodegradable Images/TEST_BIODEG_HFL_10.jpeg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 1s 1s/step
array([[1., 0., 0.]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
0
```

Test-2

```
] img_path ='/content/output_dataset/test/Recyclable Images/cardboard134.jpeg'
```

```
] import numpy as np
  img = load_img(img_path, target_size=(224, 224))
  x = img_to_array(img)
  x = preprocess_input(x)
  preds = vgg16.predict(np.array([x]))
  preds
```

```
1/1 [==============================] - 1s 584ms/step
array([[1.9529088e-21, 4.3498831e-17, 1.0000000e+00]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
2
```

Test-3

```
] img_path='/content/output_dataset/train/Trash Images/TRAIN.4_NBIODEG_CCW_1491.jpg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 1s 532ms/step
array([[4.3134577e-20, 1.0559779e-13, 1.0000000e+00]], dtype=float32)
```

```
] labels[np.argmax(preds)]
```

```
2
```

Test-4

```
img_path='/content/output_dataset/test/Biodegradable Images/TRAIN.2_BIODEG_ORI_10149.jpg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 1s 985ms/step
array([[1., 0., 0.]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
0
```

Test-5

```
img_path='/content/output_dataset/train/Recyclable Images/paper123.jpeg'
```

```
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 1s 532ms/step
array([[1.1714678e-28, 4.7260136e-14, 1.0000000e+00]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
2
```

Saving the model:

Finally, we have chosen the best model now saving that model

```
vgg16.save('vgg16.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy.
  saving_api.save_model(
```

Application Building:

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

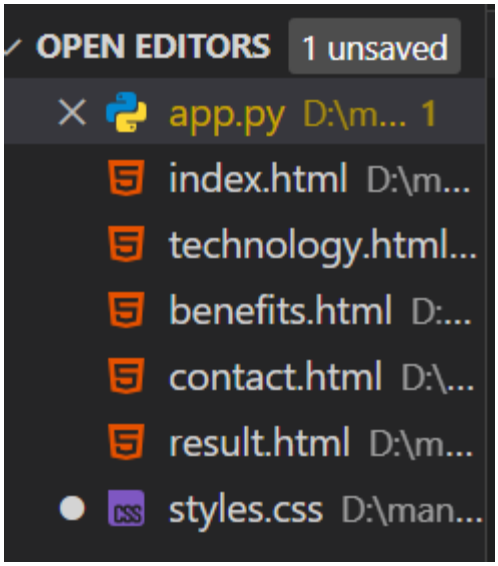
- **Building HTML Pages**
- **Building server-side script**

Building HTML Pages:

For this project create three HTML files namely

- **home.html**
- **index.html**
- **technology.html**
- **benefits.html**
- **contact.html**

Building HTML Pages:



For this project create three HTML files namely

- **index.html**
- **portofolio_details.html**

Build Python code:

Import the libraries

```
from flask import Flask, render_template,request,jsonify,url_for,redirect
from tensorflow.keras.preprocessing.image import load_img,img_to_array
from PIL import Image
import numpy as np
import os
import tensorflow as tf
```

Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (__name__) as argument.

```
app=Flask(__name__)
model = tf.keras.models.load_model('vgg16.h5')

@app.route('/')
def index():
    return render_template("index.html")
```

Here we will be using the declared constructor to route to the HTML page which we have created earlier.

In the above example, the '/' URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/predict', methods=['GET','POST'])
def output():
    if request.method =='POST':
        f=request.files['pc_image']
        img_path = "static/uploads/" + f.filename
        f.save(img_path)
        img=load_img(img_path,target_size=(224,224))
        # Resize the image to the required size
        # Convert the image to an array and normalize it
        image_array = np.array(img)
        # Add a batch dimension
        image_array = np.expand_dims(image_array, axis=0)
        # Use the pre-trained model to make a prediction
        pred=np.argmax(model.predict(image_array),axis=1)
        index=['Biodegradable Images (0)','Recyclable Images (1)','Trash Images(2)']
        prediction = index[int(pred)]
        print("prediction")
        #predict = prediction
        return render_template("portfolio-details.html", predict = prediction)
```

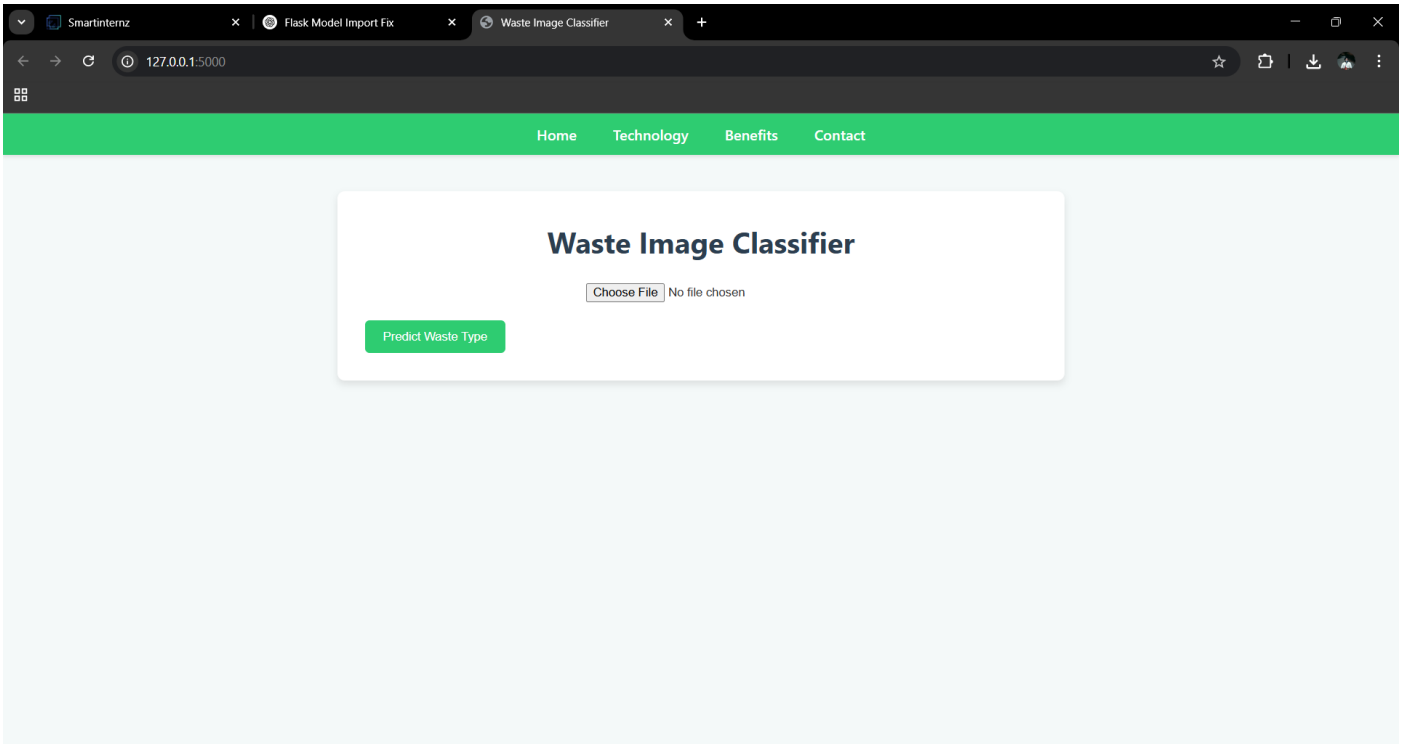
Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. This prediction value will rendered to the text that we have mentioned in the output.html page earlier.

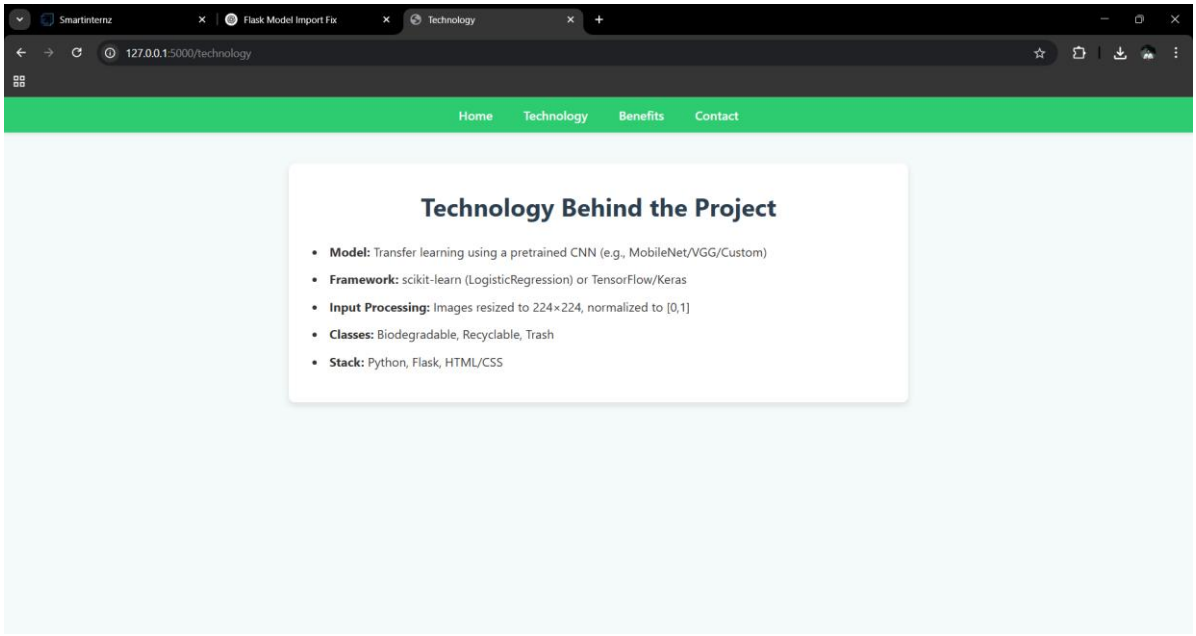
Main Function:

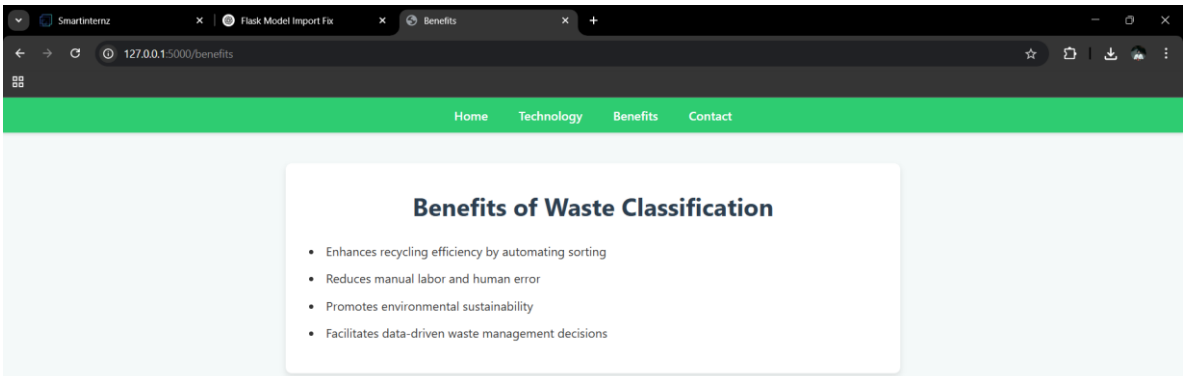
```
if __name__=='__main__':
    app.run(debug = True,port = 2222)
```

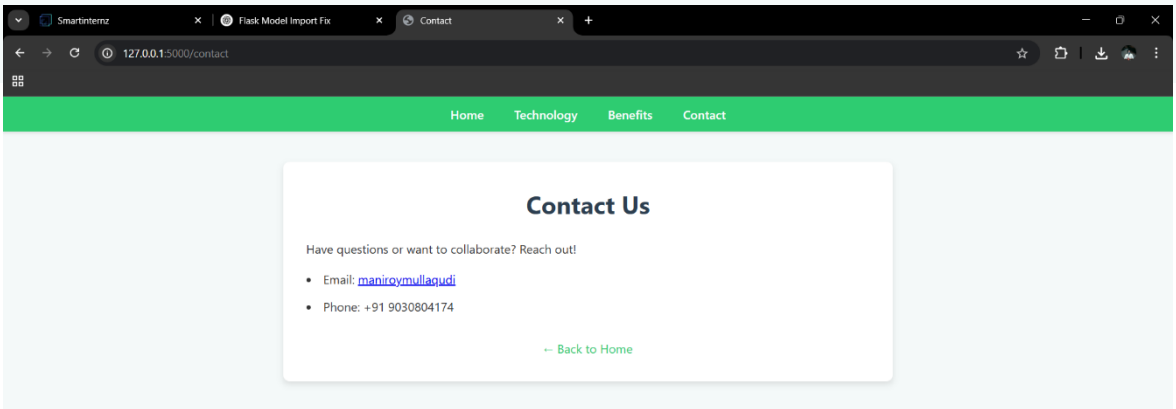
Run the web application:

- **Open Anaconda prompt from the start menu**
- **Navigate to the folder where your Python script is.**
- **Now type the "app.py" command**
- **Navigate to the local host where you can view your web page.**
- **Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.**

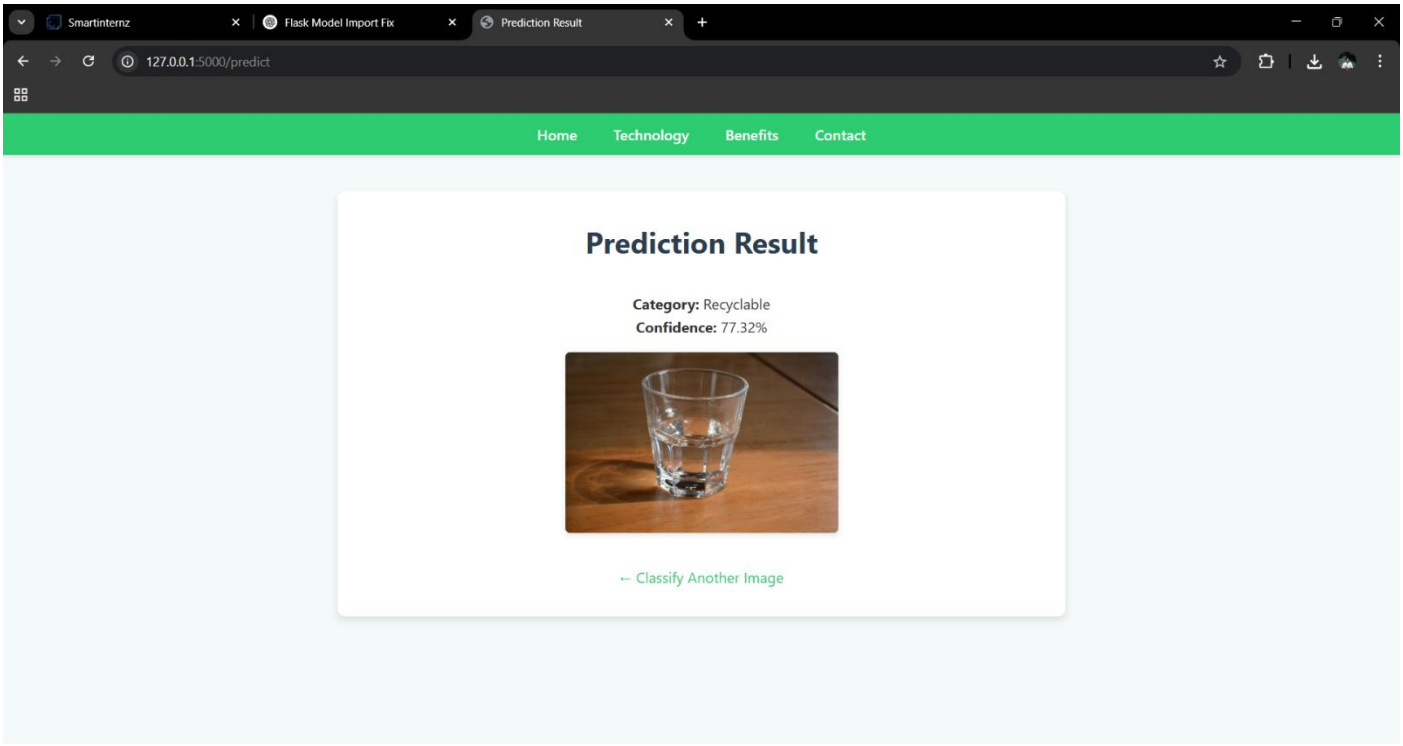


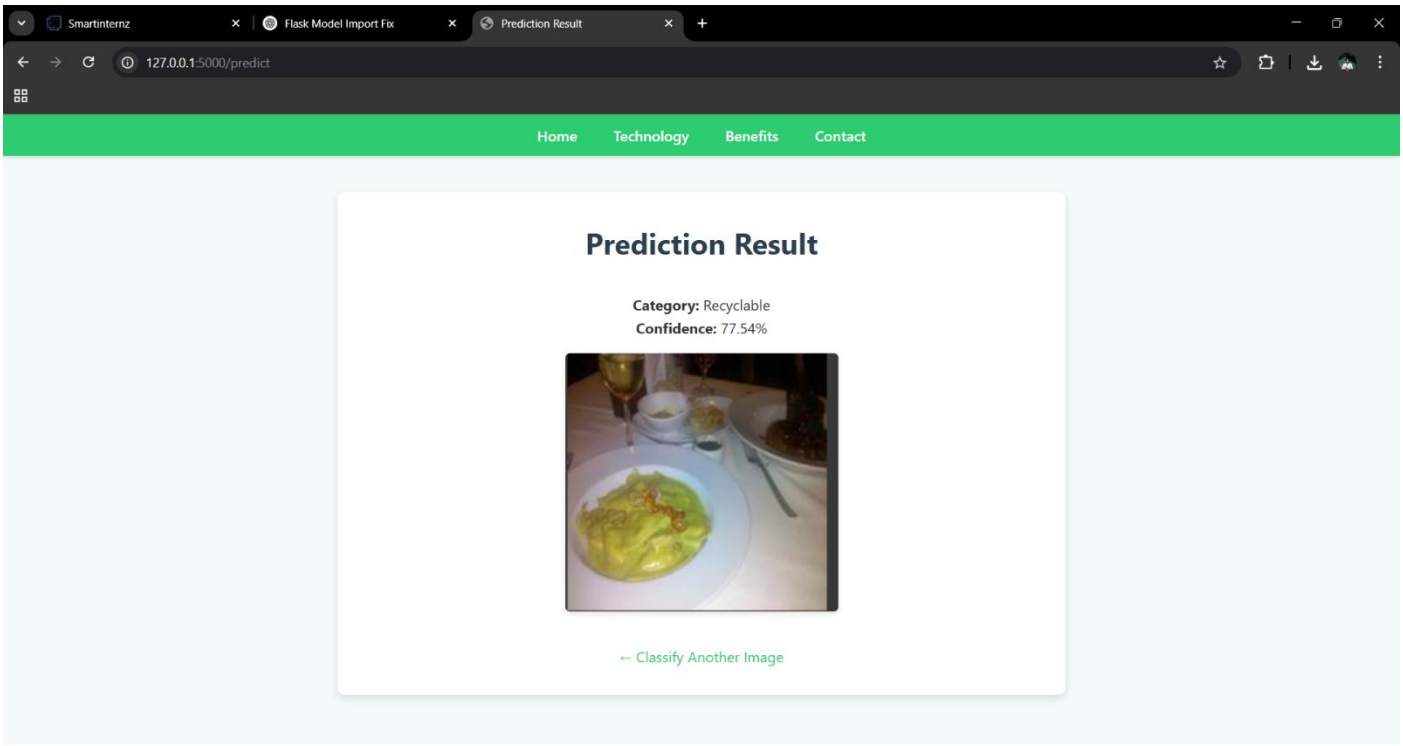

Smartinternz
Flask Model Import Fix
Technology
127.0.0.1:5000/technology
Home
Technology
Benefits
Contact
Technology Behind the Project
Model: Transfer learning using a pretrained CNN (e.g., MobileNet/VGG/Custom)
Framework: scikit-learn (LogisticRegression) or TensorFlow/Keras
Input Processing: Images resized to 224×224, normalized to [0,1]
Classes: Biodegradable, Recyclable, Trash
Stack: Python, Flask, HTML/CSS

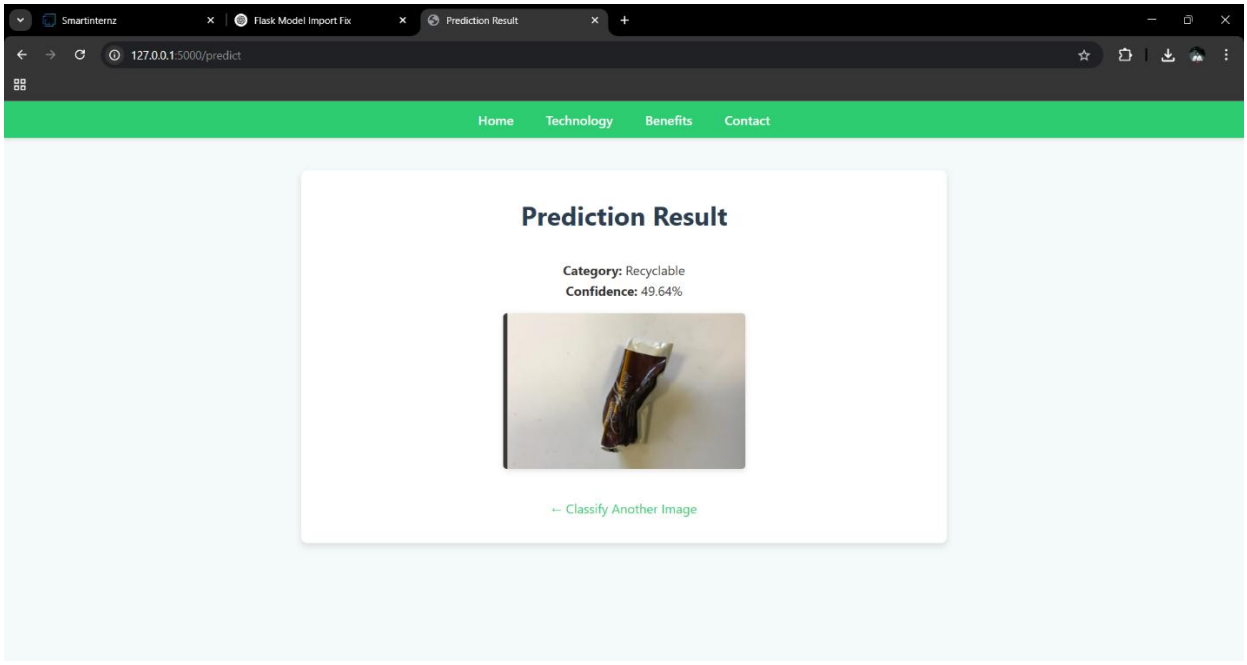

Smartinternz
Flask Model Import Fix
Benefits
127.0.0.1:5000/benefits
Home
Technology
Benefits
Contact
Benefits of Waste Classification
Enhances recycling efficiency by automating sorting
Reduces manual labor and human error
Promotes environmental sustainability
Facilitates data-driven waste management decisions

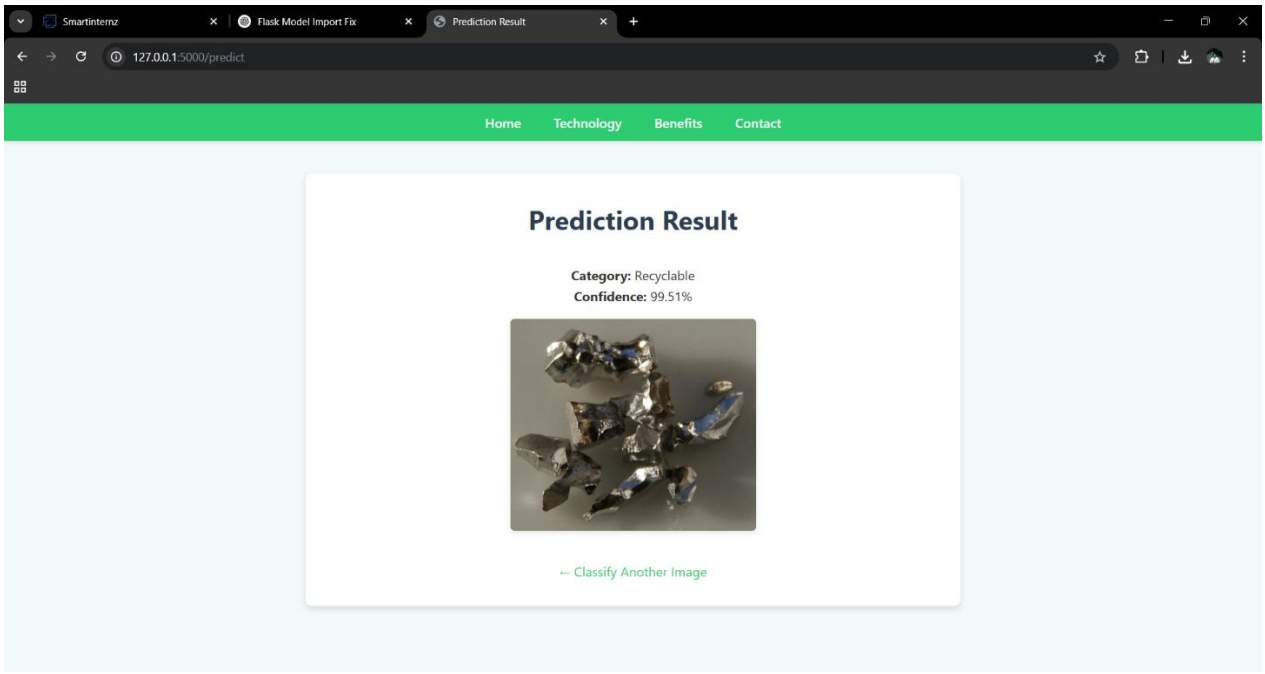

Smartinternz
Flask Model Import Fix
Contact
127.0.0.1:5000/contact
Home
Technology
Benefits
Contact
Contact Us
Have questions or want to collaborate? Reach out!
Email: maniroymullaqudi
Phone: +91 9030804174
← Back to Home

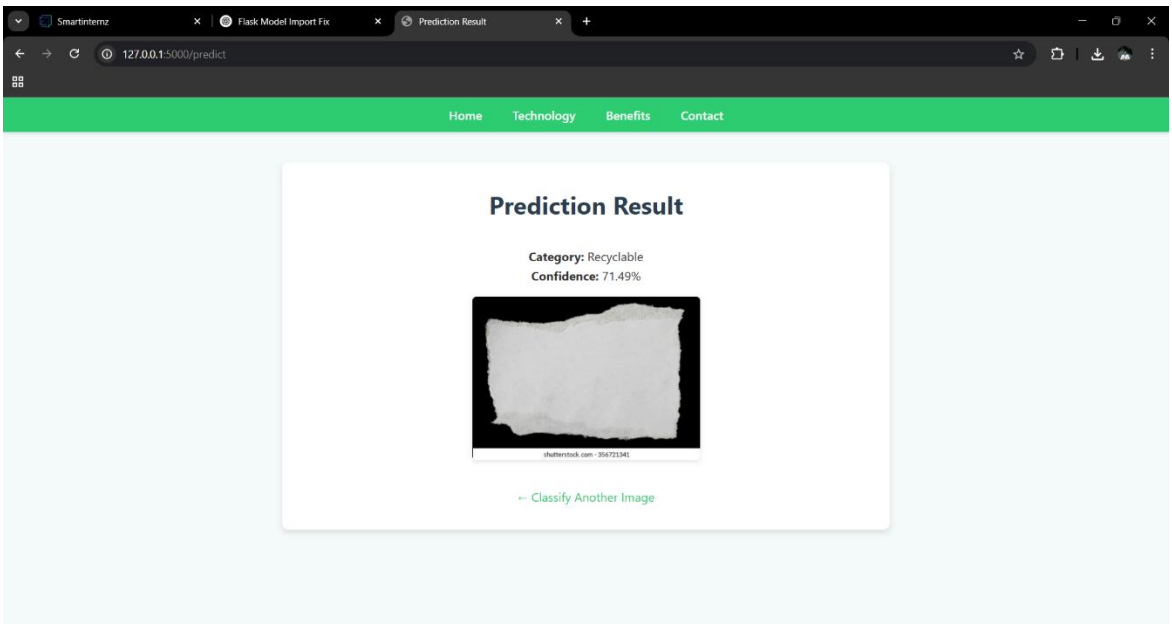
Demo input images and outputs:






Smartinternz
Flask Model Import Fix
Prediction Result
127.0.0.1:5000/predict
Home
Technology
Benefits
Contact
Prediction Result
Category: Recyclable
Confidence: 49.64%
← Classify Another Image


Smartinternz
Flask Model Import Fix
Prediction Result
127.0.0.1:5000/predict
Home
Technology
Benefits
Contact
Prediction Result
Category: Recyclable
Confidence: 99.51%
← Classify Another Image


Smartinternz
Flask Model Import Fix
Prediction Result
127.0.0.1:5000/predict
Home
Technology
Benefits
Contact
Prediction Result
Category: Recyclable
Confidence: 71.49%
← Classify Another Image

COMMENTS

FIRST AND FOREMOST, I SINCERELY GRATITUDE TO OUR ESTEEMED INISTITUTE SRI VASAVI DEGREE COLLEGE, FOR GIVING ME THIS OPPORTUNITY TO FULFILL OUR WARM DREAM TO BECOME A GRADUATE. OUR SINCERE GRADITUDE TO OUR SHORT TERM INTERNSHIP GUIDE **SRI L LAKSHMI NARAYANA**, LECTURER DEPARTMENT OF COMPUTER SCIENCE FOR TIMELY COOPERATION AND VALUABLE SUGGESTIONS WHILE CARRYING OUT THIS INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO **SRI L LAKSHMI NARAYANA**, HOD IN COMPUTER SCIENCE FOR PERMITTING ME TO DO MY PROJECT INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO **SRI M RAMA KRISHNA**, PRINCIPAL FOR PROVIDING A FAVOURABLE ENVIRONMENT AND SUPPORTING ME DURING THE DEVELOPMENT OF THIS INTERNSHIP.

THNAK YOU,SMART BRIDGE

----MULLAPUDI MANIKANTA

TEAM LEADER

THE END

SIGNATURE OF THE HOD

SIGNATURE OF THE PRINCIPAL