Waste Classification code:

```
import os
import pickle
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tkinter import Tk, Text, Button, Label, filedialog, END
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from skimage.io import imread
from skimage.transform import resize

main = Tk()
main.title("ML-Driven Waste Classification")
main.geometry("1300x1200")

X, Y, model, accuracy, x_train, x_test, y_train, y_test, y_pred = None, None, None, None, None, None,
None, None, None
Categories = ['NONORGANIC', 'ORGANIC']

def uploadDataset():
    global filename
    filename = filedialog.askdirectory(initialdir=".")
    text.insert(END, 'Dataset loaded\n')

def Preprocessing():
    global x_train, x_test, y_train, y_test
    datadir = "DATASET"
    flat_data_file = os.path.join(datadir, 'flat_data.npy')
    target_file = os.path.join(datadir, 'target.npy')
```

```
    if os.path.exists(flat_data_file) and os.path.exists(target_file):
        flat_data = np.load(flat_data_file)
        target = np.load(target_file)
    else:
        flat_data, target = [], []
        for category in Categories:
            path = os.path.join(datadir, category)
            for img in os.listdir(path):
                img_resized = resize(imread(os.path.join(path, img)), (150, 150, 3))
                flat_data.append(img_resized.flatten())
                target.append(Categories.index(category))
        flat_data, target = np.array(flat_data), np.array(target)
        np.save(flat_data_file, flat_data)
        np.save(target_file, target)

    x_train, x_test, y_train, y_test = train_test_split(flat_data, target, test_size=0.4, random_state=77)
    sns.countplot(x=target)
    plt.show()
    text.insert(END, f"Classes: {Categories}\n")

def RFModel():
    global model, y_pred
    model_file = 'RF_Classifier.pkl'

    if os.path.exists(model_file):
        with open(model_file, 'rb') as file:
            model = pickle.load(file)
    else:
        model = RandomForestClassifier()
        model.fit(x_train, y_train)
```

```
        with open(model_file, 'wb') as file:
            pickle.dump(model, file)

    y_pred = model.predict(x_test)
    acc = accuracy_score(y_test, y_pred) * 100
    text.insert(END, f"Random Forest Accuracy: {acc:.2f}%\n")

def predict():
    filename = filedialog.askopenfilename(initialdir="testing")
    img = resize(imread(filename), (150, 150, 3)).flatten().reshape(1, -1)
    prediction = model.predict(img)[0]
    category = Categories[prediction]
    text.insert(END, f"Predicted Category: {category}\n")

def graph():
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", xticklabels=Categories,
yticklabels=Categories)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title("Confusion Matrix")
    plt.show()

def close():
    main.destroy()

font = ('times', 15, 'bold')
Label(main, text='ML-Driven Waste Classification', font=font).place(x=0, y=5)

font1 = ('times', 12, 'bold')
Button(main, text="Upload Dataset", command=uploadDataset, font=font1).place(x=20, y=100)
```

```
Button(main, text="Preprocessing", command=Preprocessing, font=font1).place(x=20, y=150)
Button(main, text="Build RF Model", command=RFModel, font=font1).place(x=20, y=200)
Button(main, text="Predict Image", command=predict, font=font1).place(x=20, y=250)
Button(main, text="Graph", command=graph, font=font1).place(x=20, y=300)
Button(main, text="Exit", command=close, font=font1).place(x=20, y=350)

text = Text(main, height=30, width=85, font=font1)
text.place(x=500, y=100)

main.config(bg='skyblue')
main.mainloop()
```