**Data Preparation/Feature Engineering**

**1. Overview**

The data preparation and feature engineering phase are critical in machine learning projects. For this project, the focus was on preparing an image dataset to classify waste into categories such as plastic, paper, metal, and organic. This phase ensures the dataset is clean, well-organized, and ready for model training, directly influencing the model's performance.

**2. Data Collection**

The dataset used in this project is the Garbage Classification v2 Dataset from Kaggle. It contains over 15,000 labeled images of waste items divided into categories like plastic, paper, metal, and organic. Data preprocessing steps included resizing all images to 128x128 pixels, applying data augmentation, and normalizing pixel values to ensure consistent input to the deep learning model.

**3. Data Cleaning**

Since the dataset was well-organized, minimal cleaning was required. Duplicate images were removed, and corrupted files were excluded. Missing labels were verified and corrected where necessary.

**4. Exploratory Data Analysis (EDA)**

EDA was performed to understand the dataset better:

* Class Distribution: The dataset's class distribution was visualized using bar plots to ensure a balanced representation of categories.
* Sample Images: Random samples from each category were visualized to verify the dataset's diversity.
* Augmentation Impact: Data augmentation techniques such as rotation, zoom, and horizontal flipping were applied and visualized to show their impact on the dataset.

**Code Snippet for Class Distribution Visualization:**

import seaborn as sns

import matplotlib.pyplot as plt

sns.barplot(x=class\_labels, y=class\_counts)

plt.title("Class Distribution in Dataset")

plt.xlabel("Waste Categories")

plt.ylabel("Number of Samples")

plt.show()

**5. Feature Engineering**

Feature engineering included:

* Resizing images to 128x128 pixels.
* Applying data augmentation to enhance generalization.
* Normalizing pixel values to the range [0, 1] by dividing by 255.

**6. Data Transformation**

* Scaling: Pixel values were scaled to a [0, 1] range.
* Encoding: Labels were one-hot encoded for compatibility with categorical cross-entropy loss.

**Code Snippet for Data Augmentation:**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(

rescale=1.0 / 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'

)

**Model Exploration**

**1. Model Selection**

ResNet50, a pretrained convolutional neural network, was selected for this project due to its robust architecture, proven effectiveness in image classification tasks, and transfer learning capabilities. Its ability to extract hierarchical features makes it suitable for this multi-class classification problem.

**2. Model Training**

The model was fine-tuned on the dataset with the following configurations:

* Optimizer: Adam
* Learning Rate: 0.001
* Loss Function: Categorical Cross-Entropy
* Batch Size: 32
* Epochs: 10

**Code Snippet for Model Training:**

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(128, 128, 3))

model = Sequential([

base\_model,

Flatten(),

Dense(128, activation='relu'),

Dense(len(class\_labels), activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_generator, validation\_data=validation\_generator, epochs=10)

**3. Model Evaluation**

Evaluation metrics included:

* Accuracy: Achieved over 90% accuracy on the validation set.
* Confusion Matrix: Visualized to analyze misclassifications.
* ROC Curve: Demonstrated model performance across all classes.

**Code Snippet for Confusion Matrix:**

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

val\_labels = validation\_generator.classes

predictions = model.predict(validation\_generator)

predicted\_classes = predictions.argmax(axis=1)

conf\_matrix = confusion\_matrix(val\_labels, predicted\_classes)

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class\_labels, yticklabels=class\_labels)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.show()

**4. Code Implementation**

All code, from data preparation to model evaluation, was implemented in Python using TensorFlow and Keras. Clear comments were added to explain each step.

Visualizations

* Class Distribution Bar Plot
* Sample Images with Augmentation
* Training and Validation Accuracy/Loss Curves
* Confusion Matrix

