Date: 14/06/2024

Capstone Project Data Preparation & Feature Engineering and Model Exploration

The Data-Bears Team Members:

1. Ilyas Nayle
2. KAOUBARA DJONG-MON
3. DAGIMAWI MOGES WAKUMO

EcoLens: A Visionary Solution for a Plastic-Free World

Data Preparation/Feature Engineering

Project Overview

In the machine learning project, the data preparation and feature engineering are very important and have vital stages that involve converting raw data into a workable format or correcting the images resolution that can effectively be used by an algorithm. These stages are crucial as they enhance model accuracy and efficiency by ensuring that the data is clean, relevant and structured properly.

Data Collection

The dataset might be sourced from multiple origins such as public repositories, internal company records, or real-time data streams. Our dataset has been obtained from Kaggle. We made sure of the user privacy and obtained the necessary data for our project. This dataset contains images which are related to plastic and non-plastic objects.

Data Cleaning

The data set that we obtained has an imbalanced number of images, it contained images such as: Cardboard, Glass, Metal, Paper, Plastic and trash. The following shows the total number of each set.

1. cardboard: 403 images

2. glass: 501 images
3. metal: 410 images
4. paper: 594 images
5. plastic: 482 images
6. trash: 137 images

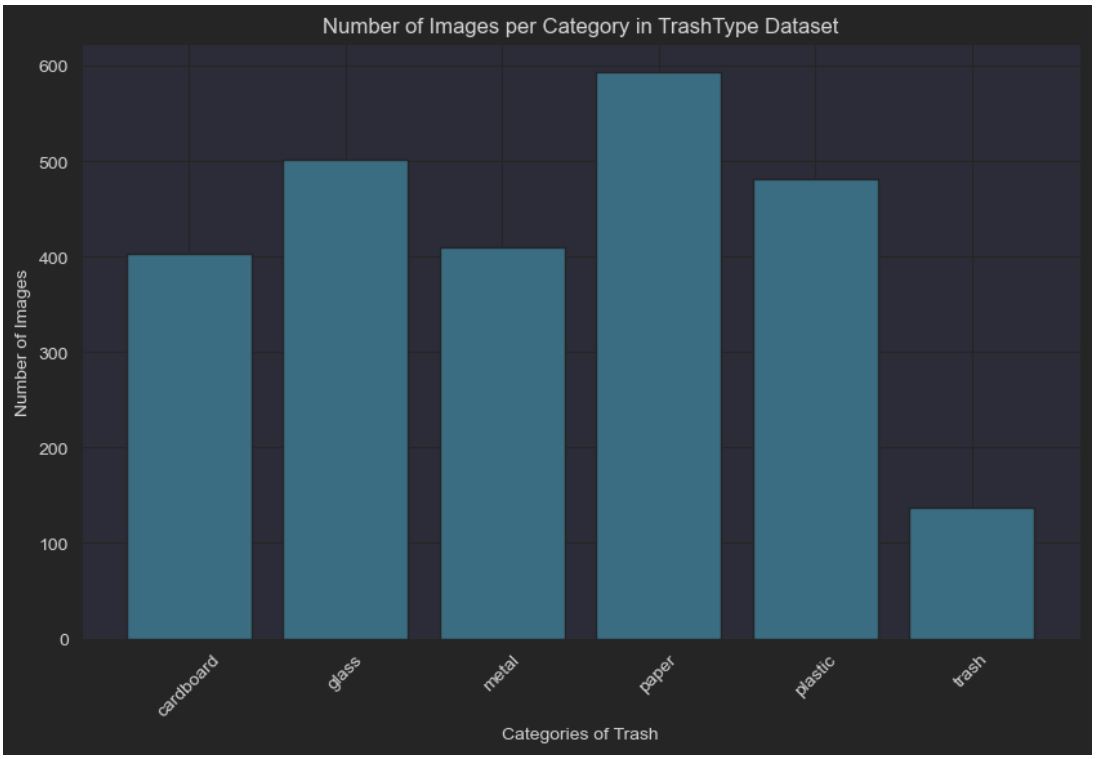
Since the number of papers images are more than others, while training the model might lean toward the paper and that makes it biased. For this reason, the number of images in all datasets has decreased and increased accordingly.

Exploratory Data Analysis (EDA)

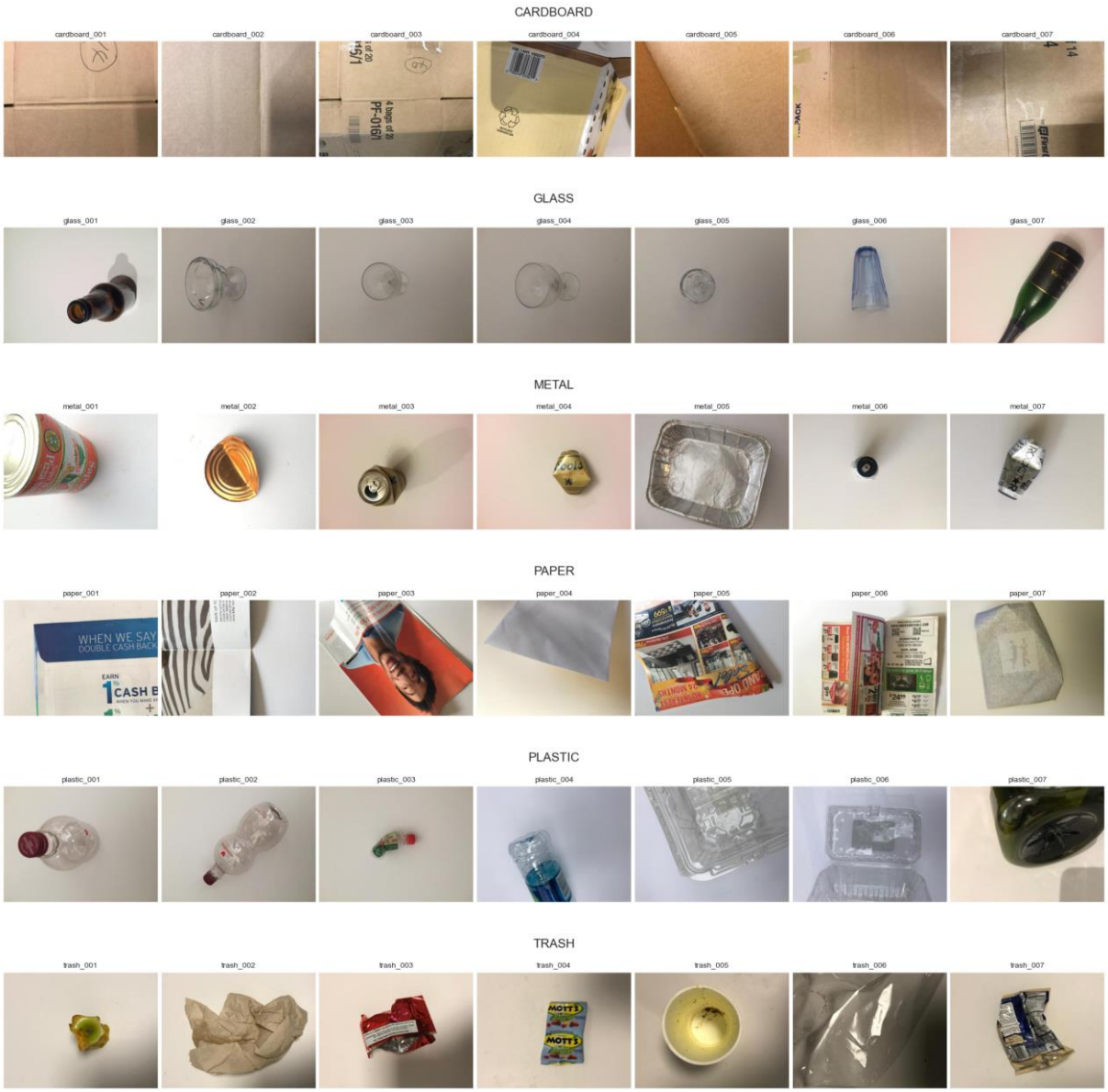
Before starting, we conducted a thorough exploration of the dataset to gain a proper understanding of the images. The following is the information about the total number of images in each of the folders.

```
The number of images in the CARDBOARD folder is: 403
The number of images in the GLASS folder is: 501
The number of images in the METAL folder is: 410
The number of images in the PAPER folder is: 594
The number of images in the PLASTIC folder is: 482
The number of images in the TRASH folder is: 137

All images in the dataset have the same dimensions: 512x384 with 3 color channels.
```



We are also going to visual each category of the images below:



The above images have been randomly selected from the dataset to visualize the images for better understanding along with their titles.

Feature Engineering & Transformation

To simplify the procedure and enhance model training, we addressed the dataset imbalance by utilizing class weights. The featuring of the images for category is as follows:

1. 'cardboard' is mapped to → 0
2. 'glass' is mapped to →1
3. 'metal' is mapped to →2
4. 'paper' is mapped to →3
5. 'plastic' is mapped to →4
6. 'trash' is mapped to →5

Date: 14/06/2024

Model Exploration

Model Selection

For our project, we're tapping into the power of deep learning with a tool called ResNet-50. It's a smart type of neural network that's good at recognizing images. The standout feature of ResNet50 architecture is the incorporation of "skip" or "shortcut" connections. These ingenious connections allow layers to skip intermediate layers and connect directly to subsequent ones. This design is pivotal in addressing the vanishing gradient problem, enabling the efficient training of much deeper neural networks than previously possible. This means ResNet-50 can handle complex tasks without getting bogged down, making it a great choice for our needs.

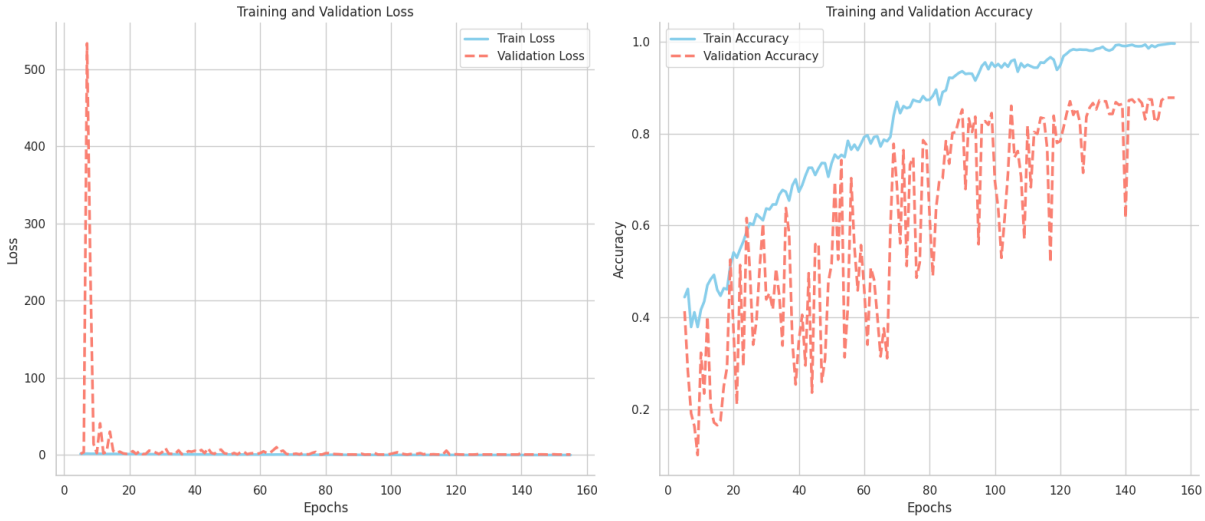
Model Training

For model training, we chose the Adam optimizer, known for adapting learning rates. We used the categorical_crossentropy loss function, suitable for multi-class classification tasks, and measured performance using accuracy for simplicity

```
CNN_ResNet50_model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

Model Evaluation

After the training of the model has been completed. We obtained the Training and Validation loss chart along with the accuracy to see the result. Which is as below:

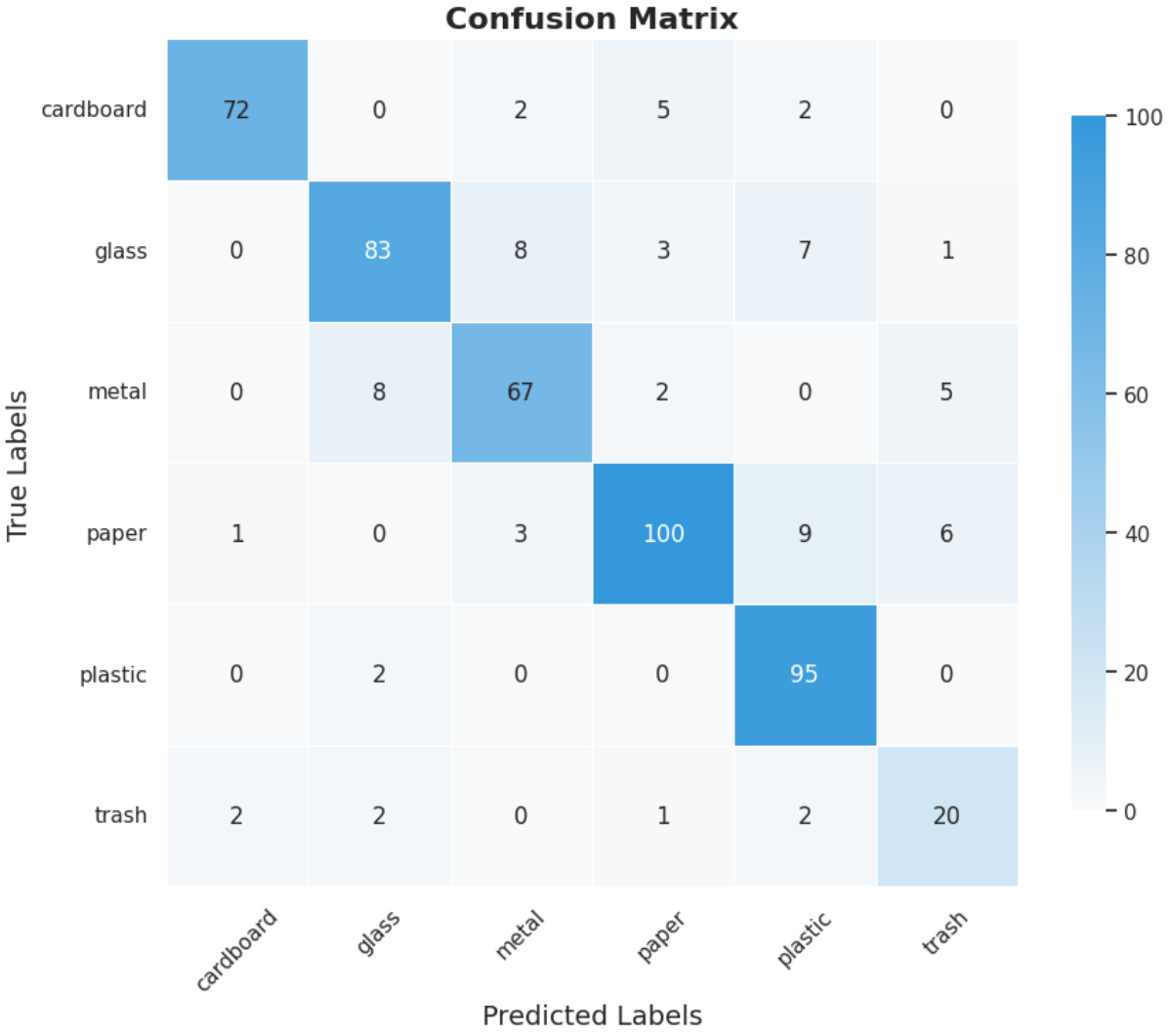


Date: 14/06/2024

Our model in the first step has an accuracy ranging between 84 – 86 for all the images type. And also the confusion matrix is shown below as well for the model result.

Classification Report:

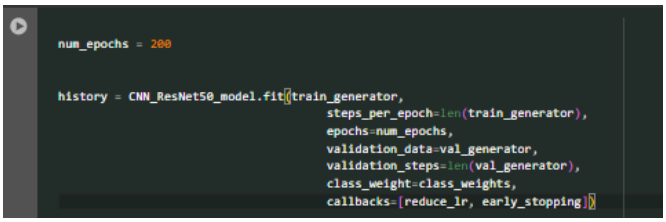
	precision	recall	f1-score	support
cardboard	0.96	0.89	0.92	81
glass	0.87	0.81	0.84	102
metal	0.84	0.82	0.83	82
paper	0.90	0.84	0.87	119
plastic	0.83	0.98	0.90	97
trash	0.62	0.74	0.68	27
accuracy			0.86	508
macro avg	0.84	0.85	0.84	508
weighted avg	0.87	0.86	0.86	508



Date: 14/06/2024

Code Implementation

The following code snippet trains a CNN based on the ResNet-50 architecture for image classification:



```
num_epochs = 200

history = CNN_ResNet50_model.fit(train_generator,
                                 steps_per_epoch=len(train_generator),
                                 epochs=num_epochs,
                                 validation_data=val_generator,
                                 validation_steps=len(val_generator),
                                 class_weight=class_weights,
                                 callbacks=[reduce_lr, early_stopping])
```

- num_epochs: Sets the total number of training cycles through the entire dataset to 200.
- train_generator and val_generator: Load batches of training and validation data respectively, useful for handling large datasets efficiently.
- steps_per_epoch and validation_steps: Determine how many batches are processed per epoch for training and validation, respectively.
- class_weight: Adjusts the model's sensitivity to classes that are underrepresented in the data.
- callbacks:
- reduce_lr: Lowers the learning rate when improvement stalls, helping refine model weights.
- early_stopping: Ends training early if the validation score stops improving, preventing overfitting.

This following function, **plot_learning_curves**, visualizes the training and validation loss and accuracy from a machine learning model's training history the image of the plot is shown below here:

```
def plot_learning_curves(history, start_epoch=5):
    df = pd.DataFrame(history.history)


    df = df.iloc[start_epoch-1:]
    df.index = range(start_epoch, len(df) + start_epoch)

    sns.set(style="whitegrid")
    plt.figure(figsize=(16, 7))

    plt.subplot(1, 2, 1)
    sns.lineplot(data=df, x=df.index, y='loss', label='Train Loss', color='skyblue', linewidth=2.5)
    sns.lineplot(data=df, x=df.index, y='val_loss', label='Validation Loss', color='salmon', linewidth=2.5, linestyle='--')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()


    plt.subplot(1, 2, 2)
    sns.lineplot(data=df, x=df.index, y='accuracy', label='Train Accuracy', color='skyblue', linewidth=2.5)
    sns.lineplot(data=df, x=df.index, y='val_accuracy', label='Validation Accuracy', color='salmon', linewidth=2.5, linestyle='--')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    sns.despine()
    plt.tight_layout()
    plt.show()
```

Date: 14/06/2024