**Data Preparation/Feature Engineering**

**1. Overview**

Data preparation and feature engineering are crucial phases in a machine learning project. They involve collecting, cleaning, and transforming data into a format suitable for model training. Feature engineering enhances the predictive power of the model by creating relevant features.

**2. Data Collection**

The dataset contains 2 folders:

* Infected
* Uninfected

And a total of 27,558 images.

This Dataset is taken from the official NIH Website: <https://ceb.nlm.nih.gov/repositories/malaria-datasets/>

**3. Data Cleaning**

As image data is used, typical data cleaning operations (e.g., handling missing values or outliers) might not apply.

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

There's no explicit EDA in our project.However, the code generates visualizations of augmented images during the preprocessing stage, which could be considered a form of exploratory visualization.

**5. Feature Engineering**

In image-based machine learning, direct feature engineering might not be necessary.However, the **ImageDataGenerator** performs various transformations (e.g., rescaling, shear, zoom, flip) that augment and create variations in the image dataset, potentially improving model generalization.

**6. Data Transformation**

The ImageDataGenerator is responsible for data transformations:

* Images are resized to a target size of (128, 128).
* Pixel values are rescaled to be within the range of 0 to 1.

**Model Exploration**

**1. Model Selection**

**Model Type:** Sequential CNN using TensorFlow/Keras.

**Rationale**: CNNs are effective for image classification tasks due to their ability to capture spatial hierarchies in data.

**2. Model Training**

**Architecture:** CNN architecture consists of Conv2D, MaxPooling2D, Flatten, and Dense layers.

**Hyperparameters:** Training occurs over 10 epochs with an Adam optimizer, categorical cross-entropy loss, and batch size of 32.

**3. Model Evaluation**

**Evaluation Metrics:**Accuracy and loss metrics are utilized to assess the model's performance on both training and validation sets.

**4. Code Implementation**

* **Visualizations of augmented images**

**# Create an ImageDataGenerator**

**datagen = ImageDataGenerator(rescale=1./255)**

**# Flow from the directory and load a few samples**

**data\_flow = datagen.flow\_from\_directory(**

**test\_path,**

**target\_size=(64, 64), # Adjust the target size as needed**

**batch\_size=20, # Number of samples to plot**

**class\_mode='categorical', # If you're doing classification**

**shuffle=True,**

**seed=42,**

**subset='training' # Use 'training' subset to control the number of samples**

**)**

**# Plot the loaded samples**

**plt.figure(figsize=(15, 8))**

**num\_samples\_to\_plot = 15 # Adjust this based on your preference**

**for i in range(num\_samples\_to\_plot):**

**img, label = data\_flow.next()**

**class\_name = "Positive" if label[0][0] == 1 else "Negaive" # Modify this based on your folder names**

**plt.subplot(1, num\_samples\_to\_plot, i + 1)**

**plt.imshow(img[0])**

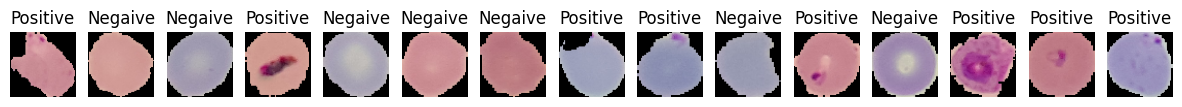
**plt.title(class\_name) # Adjust this if needed**

**plt.axis('off')**

**if i + 1 == num\_samples\_to\_plot:**

**break # Exit the loop after plotting the desired number of samples**

**plt.show()**

****

* **Data augmentation**

**# Create an ImageDataGenerator for preprocessing**

**datagen = ImageDataGenerator(**

**rescale=1./255, # Normalize pixel to be between 0 and 1**

**shear\_range=0.2, # Shear intensity for data augmentation**

**zoom\_range=0.2, # Zoom intensity for data augmentation**

**horizontal\_flip=True, # Horizontal flip for data augmentation**

**validation\_split=0.2 # Split the dataset into training and validation sets**

**)**