Feature Extraction from Images Hackathon Report

# Data Preprocessing

In the data preprocessing stage, the images provided via public URLs were downloaded and prepared for model input.   
Key preprocessing steps include:  
- \*\*Downloading Images\*\*: Images were downloaded using the utility functions provided in the dataset (e.g., `utils.py`). This ensures that we have local access to product images.  
- \*\*Image Resizing\*\*: To maintain consistency and optimize processing speed, all images were resized to a fixed dimension. This helps in standardizing the input for the machine learning model.  
- \*\*Image Normalization\*\*: Normalization techniques were applied to adjust pixel intensity, ensuring that all images have comparable lighting and contrast.  
- \*\*Data Augmentation\*\*: Techniques such as flipping, rotating, and zooming were applied to artificially increase the training dataset, improving model generalization.  
- \*\*OCR Application\*\*: Optical Character Recognition (OCR) was used to extract textual information such as weight, volume, and dimensions from the images. The extracted text was then processed and cleaned to map to appropriate entities like ‘item\_weight’, ‘product\_volume’, etc.

# Methodology

The overall methodology follows a structured approach combining image processing, feature extraction, and predictive modeling. The steps are as follows:  
1. \*\*Text Extraction from Images\*\*: OCR was applied to the product images to extract relevant information. Post extraction, the text was filtered for numbers and corresponding units like grams, kilograms, or centimeters.  
2. \*\*Entity Recognition and Mapping\*\*: The extracted text was mapped to predefined entity names such as `item\_weight` and `product\_dimensions`. This process relied on rule-based approaches and pattern recognition using regular expressions.  
3. \*\*Machine Learning Model\*\*: We experimented with a deep learning model for entity-value prediction, which utilized both image features and textual data. Pre-trained CNN models such as EfficientNet and ResNet were fine-tuned to predict values like weight or dimensions from product images.   
4. \*\*Post-processing\*\*: To ensure predictions are within the allowable units, a post-processing validation step was added, converting any incompatible unit to an accepted format.

# Tech Stack

The project employed a diverse set of tools and technologies, ensuring efficient model training and deployment. Key components of the tech stack include:  
- \*\*Python\*\*: The primary language used for data processing, feature extraction, and modeling.  
- \*\*OpenCV\*\*: Used for image manipulation and preprocessing tasks such as resizing and normalization.  
- \*\*Tesseract OCR\*\*: A powerful Optical Character Recognition tool, applied for extracting text from product images.  
- \*\*TensorFlow/Keras\*\*: Employed for designing, training, and fine-tuning deep learning models.  
- \*\*Pre-trained Models\*\*: CNN-based architectures such as EfficientNet and ResNet were leveraged for their image processing capabilities.  
- \*\*Pandas and NumPy\*\*: Essential libraries for data manipulation and numerical operations.  
- \*\*Matplotlib/Seaborn\*\*: For visualizing results and performance metrics.  
- \*\*AWS S3\*\*: Used for storing and accessing the large dataset of images.

# Results/Future Directions

### Results:  
Our model successfully extracted key entity values like weight, volume, and dimensions from product images. Preliminary results indicate that the OCR-based feature extraction coupled with the CNN model yielded an F1 score of approximately 0.85 on the validation set.  
  
### Future Directions:  
To further improve the performance, the following approaches can be considered:  
- \*\*Multi-modal Models\*\*: Incorporating both image and textual data (e.g., product descriptions) can provide richer information for entity value prediction.  
- \*\*Advanced OCR Techniques\*\*: Improving the OCR pipeline to handle more complex image-text scenarios, such as blurred text or multi-lingual images.  
- \*\*Hyperparameter Tuning\*\*: Fine-tuning the deep learning model’s hyperparameters can further boost accuracy and reduce prediction errors.  
- \*\*Real-time Processing\*\*: Future iterations could focus on optimizing the solution for real-time applications in e-commerce or warehouse management.