PLANT DISEASE DETECTION & RECOMMENDATION SYSTEM

(3. Preprocess Data & 6. Evaluation Metrics)

1) ***DEFINE SCOPE***
2) ***COLLECT DATA***
3) ***PREPROCESS DATA***
4) ***CHOOSE MODEL***
5) ***MODEL TRAINING***
6) ***EVALUATE THE MODEL***
7) ***DEPLOYMENT***
8) ***FARMER USUABILITY***
9) ***GATHER FEEDBACK***

1. **What is image processing and why it is important?**
    - Image preprocessing is the process of manipulating raw image data into a usable and meaningful format.
    - It allows you to eliminate unwanted distortions and enhance specific qualities essential for computer vision applications.
    - clean up the image, standardize its format, and highlight important features.

2. **List all the preprocessing steps.**
    - **Noise reduction**:
        - Smoothing, blurring, and filtering techniques can be applied to remove unwanted noise from images.
        - The GaussianBlur () and medianBlur () methods are commonly used for this.
    - **Resizing**:
        - Adjusting the image dimensions to a uniform size, crucial for training ML models.
        - Use OpenCV's resize () method to resize images.
        - ML algos typically require a standard size, to a square dimension, often 224x224 or 256x256 pixels.
    - **Normalization**:
        - Scaling pixel values (intensity values of pixels) to a specific range, often between 0 and 1, to ensure consistency.
        - This makes the images more suitable for analysis and allows machine learning models to learn patterns independent of lighting conditions.
        - Normalize () from scikit-image can be used for this.
        - 3 methods-rescaling to the 0–1 range, histogram equalization, and standardization

- **Contrast enhancement**:
    - The contrast of images can be adjusted using histogram equalization.
    - The equalizeHist () method enhances the contrast of images.
- **Binarization:**
    - Binarization converts grayscale images to black and white by thresholding.
    - The threshold () method is used to binarize images in OpenCV.
- **Grayscale conversion**:
    - converting a coloured image into a black-and-white image,
    - where each pixel has a shade of gray ranging from black (0 intensity) to white (maximum intensity).
    - removes colour info & represents the image using only brightness (luminance) values
    - simplifying data and reducing computational load.
    - The cvtColor() method can be used
- **Image segmentation**: Dividing an image into distinct regions based on features like color or intensity, useful for object identification.
- **Edge detection**: Identifying sharp transitions in pixel intensity, highlighting boundaries of objects within an image.
- **Feature extraction**: Identifying and extracting specific characteristics from an image that are relevant for the task at hand.
- **Data augmentation**: Generating variations of an image (e.g., rotating, flipping) to increase training data diversity for ML models.
- **Image thresholding**: Converting an image to a binary image by setting pixel values above a certain threshold to one value and below to another.
- **K-means clustering:**
    - Used for grouping similar data points into clusters.
    - Used for image segmentation by clustering similar pixel intensities (e.g., foreground/background separation).
    - Can be used to detect objects in images by clustering similar pixel regions.
    - Recommended for Medical imaging (tumor detection), Object segmentation (e.g., separating sky, land, and water in satellite images), Image compression
- **Fuzzy C-Means:**
    - Used for assigning degrees of membership to each data point instead of a hard assignment like K-Means.
    - Can be used for segmentation in image preprocessing (Commonly in medical imaging).
    - Works similarly to K-Means but allows soft clustering, meaning each pixel can belong to multiple clusters with varying degrees.
- **PCA**

    - Used for reducing the number of features while preserving most of the variance in the data.
    - Dimensionality reduction: Compresses image data while keeping important features.
    - Noise reduction: Removes minor variations (useful in preprocessing noisy images).
    - Used as a feature extraction step before applying classification models.
    - Recommended for Face recognition (Eigenfaces technique), Image compression (reducing file size), Enhancing patterns in images.
- Class Balancing: Class Imbalance occurs in classification when we have datasets with an unequal ratio of data points in each class.

Reference:

- The Complete Guide to Image Preprocessing Techniques in Python | by Maahi Patel | Medium
- A survey on the utilization of Superpixel image for clustering-based image segmentation | Multimedia Tools and Applications

3. When should I resize an image?

- The image is too large to process efficiently. Reducing size can speed up processing.
- The image needs to match the input size of a machine learning model.
- The image needs to be displayed pn a screen or webpage at a specific. size.

4. If normalized pixels are between 0 and 1, why grayscale?

- Grayscale Conversion: Converts a colored image (RGB) into a single channel representing intensity (brightness) levels. This is often done to reduce complexity when color information is not critical.
- Normalization: Rescales pixel values, whether it's grayscale or RGB, to a range like 0–1. This helps the model process the data more efficiently.
- So, you can normalize both RGB and grayscale images—the range (0–1) is independent of whether the image is in color or grayscale.

5. Why convert to 0–1 instead of keeping 0–255?

Here's why 0–1 is preferred:

1. Machine Precision:
    - Computers handle floating-point values like 0.01 or 0.5 very efficiently.
    - Keeping the range 0–255 involves larger numbers, which can cause slower computations or numerical instability.

- For example, multiplying or dividing large numbers like 231 and 123 repeatedly can increase rounding errors.

2. Gradient Descent Works Better:
   - Deep learning models use gradient descent to adjust weights.
   - Smaller input values (like 0–1) ensure gradients remain stable and help the optimizer converge faster.
3. Standard Practice: Many pre-trained models and frameworks (like TensorFlow, PyTorch) are designed to work with normalized input data, so sticking to 0–1 ensures compatibility.
4. Why not 0–255? While humans find 0–255 easier to understand, computers process normalized values more efficiently in deep learning models because of how matrix operations are optimized for smaller, consistent ranges.

6. Which One to Use for normalization 'divide by 255' or 'normalize from scikit learn'?

- For Simplicity and Speed: Stick with dividing by 255 if you're working with regular RGB images where pixel values are already in the range [0, 255].
- For Advanced Use Cases: Use Normalize() if you need to handle:
  1. Images with unknown or non-standard ranges (e.g., [-1, 2] or [0, 4096]).
  2. Custom normalization ranges other than [0, 1].

7. What is Histogram Equalization?

In data/image processing, histogram equalization is used to:

a. In a low-contrast image, most pixels have similar intensity values, meaning the image looks dull, hazy, or dim instead of having a good distinction between bright and dark areas.
b. A "narrow range" means that instead of using the full range of intensity values (0 to 255), the pixels are clustered within a small subset of values, such as 100 to 150, making the image look washed out or too dark.
c. For example, Hazy Image, A foggy scene where most pixels have mid-gray intensity instead of deep blacks and bright whites.
d. Histogram equalization spreads out these narrow-range intensity values across a wider range (closer to 0-255), improving contrast and making details more visible.

8. How Does Histogram Equalization Work?

Here's a step-by-step explanation of how histogram equalization works:

1. **Create a Histogram:**
   - Compute the histogram of the image, which shows the frequency of each intensity value (e.g., for a grayscale image, values range from 0 to 255).
2. **Compute the Cumulative Distribution Function (CDF):**

- Calculate the cumulative sum of the histogram values.
- Normalize the CDF to scale it between the range of intensity values (e.g., 0 to 255 for an 8-bit grayscale image).

3. **Map Old Intensities to New Intensities:**
   - Use the normalized CDF to map the original pixel intensity values to new values that are distributed more evenly across the intensity range.
4. **Generate the Equalized Image:**
   - Replace each pixel's intensity in the original image with the corresponding new intensity value.

9. When to Use Different Image Preprocessing Techniques?

1. Resizing → When input image sizes vary
   - Use when images have different dimensions to standardize input for models.
   - Necessary for CNNs, which require a fixed input size (e.g., 224×224 for ResNet).
   - The image is too large to process efficiently. Reducing size can speed up processing.
   - The image needs to match the input size of a machine learning model.
   - The image needs to be displayed on a screen or webpage at a specific. size.
2. Colour Conversion → When colour is not a key feature
   - Use grayscale conversion when colour doesn't provide meaningful information (e.g., leaf texture analysis).
   - Use black-and-white (thresholding) when you need to segment objects clearly.
3. Noise Reduction → When images contain unwanted distortions
   - Use Gaussian blur for general noise reduction.
   - Use Median blur if the image has salt-and-pepper noise (random white & black dots).
   - Use Bilateral filter if you need to remove noise while preserving edges (useful for plant veins & disease spots).
4. Normalization → When pixel values need to be in a standard range
   - Use when models require standardized input values (0–1 or -1 to 1).
   - Helps avoid issues where high pixel values dominate learning in neural networks.
5. Contrast Enhancement → When details are unclear or low contrast
   - Use histogram equalization when images look dull, washed out, or have poor contrast.
   - Helps in highlighting disease spots or lesions on plant leaves.
6. Edge Detection → When identifying shapes or contours is important
   - Use when analyzing leaf shape, disease spots, or cracks.

- Helps in extracting important boundaries for feature detection.

10. How to Choose the Right Preprocessing Steps?

- If images are inconsistent in size → Resize.
- If colors are not necessary → Convert to grayscale.
- If noise affects clarity → Apply blurring techniques.
- If pixel values are too large → Normalize them (0–1).
- If details are lost in shadows or lighting issues → Enhance contrast.
- If object boundaries are important → Use edge detection.

11. Image Augmentation

Categories of Image Augmentation:

- Model-based algorithms
- Model-free approach
- Optimizing policy-based algorithms

Table 2
Taxonomy with relevant methods.

Categories			Relevant methods
Model-free	Single-image	Geometrical transformation	translation, rotation, flip, scale, elastic distortion.
		Color image processing	jittering.
		Intensity transformation	blurring and adding noise, Hide-and-Seek [23], Cutout [24], Random Erasing [25], GridMask [26].
	Multiple-image	Non-instance-level	SamplePairing [27], Mixup [28], BC Learning [29], CutMix [30], Mosaic [22], AugMix [31], PuzzleMix [32], Co-Mixup [33], SuperMix [34], GridMix [35].
		Instance-level	CutPas [36], Scale and Blend [37], Context DA [38], Simple CutPas [39], Continuous CutPas [40].
Model-based	Unconditional		DCGAN [41], [42–44]
	Label-conditional		BDA [45], ImbCGAN [46], BAGAN [47], DAGAN [48], MFC-GAN [49], IDA-GAN [50].
	Image-conditional	Label-preserving	S+U Learning [51], AugGAN [52], Plant-CGAN [53], StyleAug [54], Shape bias [55].
		Label-changing	EmoGAN [56], δ-encoder [57], Debiased NN [58], StyleMix [59], GAN-MBD [60], SCIT [2].
Optimizing policy-based	Reinforcement learning-based		AutoAugment [61], Fast AA [62], PBA [63], Faster AA [64], RandAugment [65], MADAO [66], LDA [67], LSSP [68].
	Adversarial learning-based		ADA [69], CDST-DA [70], AdaTransform [71], Adversarial AA [72], IF-DA [73], SPA [74].

Table 3
Challenges in computer vision tasks from the perspectives of datasets and deep learning models.

Challenges	Descriptions	Strategies and related studies
Images variations	The following basic variations exist in many datasets and applications, including illumination, deformation, occlusion, background, viewpoint, and multiscale, as shown in Fig. 1.	Geometrical transformation and color image processing improve the majority of the variations. Occlusion: Hide-and-Seek [23], Cutout [24], Random Erasing [25], GridMask [26]. Background or context: CutMix [30], Mosaic [22], CutPas [36]. Multiscale: Scale and Blend [37], Simple CutPas [39].
Class imbalance and few images	Number of images vary between classes or some classes have only few images.	Reusing instance from minority class is one strategy by instance-level operation, Simple Copy-Paste [39]. Most studies attempt to generate images for the minority class: ImbCGAN [46], DAGAN [48], MFC-GAN [49], EmoGAN [56], δ-encoder [57], GAN-MBD [60], SCIT [2].
Domain shift	Training and testing datasets represent different domains, commonly referring to styles.	Changing styles for existing images is a main strategy, including S+U Learning [51], StyleAug [54], Shape bias [55], Debiased NN [58], StyleMix [59].
Data remembering	Larger models with many learnable parameters tend to remember specific data points, which may result in overfitting.	The mechanism is increasing dataset size within or between vicinity distributions. Within version assumes label-preserving while between version changes labels, such as Mixup [28], AugMix [31], Co-Mixup [33].

Model-free approach:

- Single-Image Augmentation:
  - Geometric transformation:
    - tries to modify the spatial relationship between pixels, including affine transformation and elastic deformation
    - Rotation, Flipping, Translation, Scaling, Elastic distortion.

- Color-Image Augmentation:
  - color image processing aims to vary the color of an input image.
  - the use of robust features for contrast learning via color image processing, which represents a case of task-agnostic learning.
- Intensity Transformation:
  - The intensity transformation is advocated to change parts of the images and has recently received more attention.
  - entail changes at the pixel or patch levels.
  - The underlying concept is that the changes push the model to learn robust features by avoiding trivial solutions.
  - Gaussian noise, cutout (cost is high), hide-and-seek (simple & fast, obscuring significant part by directly blocking a part),
-

Reference:

- A Comprehensive Survey of Image Augmentation Techniques for Deep Learning - ScienceDirect

12. Which one to use for data preprocessing - OpenCV or Pillow?

Feature/Aspect	OpenCV	PIL/Pillow
Primary Use Case	Advanced image processing, computer vision tasks like surveillance systems, robotics, & AR	Basic image manipulation and enhancements such as web development, digital image archiving, and simple photo editing
Library Scope	Comprehensive, includes tools for image/video processing, machine learning integration	Focused on basic image operations
Performance	High performance, optimized for real-time applications	Lightweight, not optimized for real-time tasks
Cross-Platform Support	Yes (Windows, Linux, macOS, Android, iOS)	Yes (Windows, Linux, macOS)
Supported File Formats	Wide range of image and video formats	Wide range of image formats
Ease of Use	Moderate, with a steep learning curve	High, with an intuitive and simple API
Advanced Filters	Yes (e.g., GaussianBlur, MedianBlur)	Basic filters (e.g.,ImageFilter.GaussianBlur)
Edge Detection	Yes (e.g., cv2.Canny)	No built-in edge detection
Object Detection	Yes (e.g., Haar Cascades)	No built-in object detection
Integration with ML	Yes (supports TensorFlow, PyTorch, Caffe)	No direct integration with ML frameworks

Reference:

- Image Processing — OpenCV Vs PIL - GeeksforGeeks

13. **Feature Analysis and Feature Extraction.**
14. **What are the metrics used for calculating the performance of the model?**

Evaluating the performance of plant disease detection involves:

- **Accuracy** the ratio of correctly classified samples to the total number of samples in the dataset.

$$Accuracy = \frac{TN + TP}{TN + TP + FP + FN}$$

- **Precision** indicates the proportion of the predicted positive instances that are actually positive, signifying that higher precision results in fewer false positives.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** measures the classifier's ability to identify all positive instances, with higher recall implying fewer false negatives. Conversely, lower recall suggests a higher number of false negatives, so improving recall may lead to a decrease in precision.

$$Recall = \frac{TP}{TP + FN}$$

- **Map** when the Combination over Affiliation (IOU) is more noticeable than or comparable to the edge, Guide is the precision. It is the extent of a legitimate and expected locale of interests' intersection guide district toward the relationship of the same. At different edges, area execution is assessed in constraints of Guide.
- **Review** A model's review is characterized as the model's capacity to distinguish Genuine Up-sides accurately. It is characterized as the proportion of accurately ordered positive results to accurately arranged yields.
- **Exactness** is portrayed as the ability to perceive simply pertinent articles. It is portrayed as the extent of precisely requested positive outcomes to amount to positive outcomes.
- **F1-score** The f1 score is likewise acquainted with survey the model's precision. The f1-score thinks about both the model's accuracy and review.

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precisio}$$