# Plant Disease Detection System for Sustainable Agriculture

# Problem Statement

Develop a CNN-based model capable of detecting and classifying plant diseases from images of leaves of various crops such as apple, cherry, grape, and corn. The model should accurately identify both healthy and diseased leaves while predicting the specific type of disease. This system will aid in precision agriculture by enabling early detection and effective disease management.

# Pipeline

## 1. Data Collection

* The dataset was provided by the instructor and contains labeled images of healthy and diseased plant leaves from various crops such as apple, cherry, grape, and corn.
* Each image is associated with a class label indicating either the disease type or a "healthy" condition.
* The dataset is organized into folders or a CSV file, making it suitable for direct use in model training.
* Since the data is curated, it reduces the need for external collection and ensures consistency across training and evaluation.

## 2. Data Preprocessing

* Resizing: Standardize image dimensions (128x128 pixels).
* Normalization: Scale pixel values to [0, 1] range for faster convergence.
* Augmentation: Apply transformations such as rotation, flip, zoom, and brightness/contrast adjustments to increase dataset diversity and model generalization.
* Label Encoding: Convert categorical disease labels into numerical form using one-hot encoding or label encoding.

## 3. Model Design (CNN Architecture)

* Architecture: Build a CNN model with convolutional layers, max-pooling, dropout (to avoid overfitting), and dense layers.
* Activation Functions: Use ReLU for intermediate layers and Softmax for the output layer.
* Pretrained Models: Optionally apply transfer learning with VGG16, ResNet50, or MobileNet.

## 4. Model Training

* Splitting: Divide the dataset into Train (70%), Validation (15%), and Test (15%) sets.
* Loss Function: Use Categorical Cross-Entropy for multi-class classification.
* Optimizer: Adam or SGD with a suitable learning rate (e.g., 0.001).
* Metrics: Track Accuracy, Precision, Recall, and F1-Score.
* Epochs & Batch Size: Tune based on hardware (e.g., 30–50 epochs, batch size 32 or 64).

## 5. Model Evaluation

* Confusion Matrix: Visualize classification performance.
* Training vs. Validation Curves: Check for overfitting or underfitting.
* Test Accuracy: Evaluate model on unseen test data.

## 6. Model Deployment (Optional)

* Interface: Create a web or mobile app using Flask, Django, or Streamlit.
* Functionality: Upload an image and display disease prediction.
* Export Model: Save in formats like .h5, .pkl, or ONNX.