**PROBLEM STATEMENT:**

In agriculture, plant diseases significantly impact crop yield, farmer income, and overall food security. Traditional disease detection methods rely heavily on manual inspection by agricultural experts, which is time-consuming, costly, and often unavailable in rural or under-resourced areas. This leads to delayed treatment, overuse of chemical pesticides, and large-scale crop losses.

To address these challenges, this project proposes a **Plant Disease Detection System using Machine Learning**, specifically leveraging **Convolutional Neural Networks (CNNs)** to automatically identify plant diseases from leaf images. The system aims to provide a fast, accessible, and accurate solution that can assist farmers and agricultural stakeholders in early disease detection, reduce dependence on expert intervention, and promote sustainable farming practices.

**PIPELINE:**

1. Concept & Project Overview

* Project is based on Convolutional Neural Networks (CNN).
* Mimics how humans see and recognize patterns (Computer Vision).
* Input: Leaf image → Output: Disease prediction (or Healthy).

2. Data Collection & Preparation

* Collect dataset of plant leaf images.
* Dataset split into:
  + Training set
  + Validation set
  + Testing set

3. Google Colab Setup

* Use Google Colab for coding and model training.
* Steps:
  + Sign in → New Notebook
  + Change runtime to GPU
  + Save and start coding

4. Dataset Handling

* Prepare dataset as a ZIP folder
  + Upload to Google Drive
  + Mount drive in Colab
  + Use Python code to unzip and access dataset

5. Image Preprocessing

* All images must have same dimensions (e.g., 100x100 or 224x224).
* Resize and normalize image data
* Convert images to numerical format for model input

6. Image Augmentation

* Purpose: Increase data diversity and avoid overfitting
* Techniques:
  + Rotation
  + Flipping
  + Zoom
  + Shifting
* Performed using Python code (e.g., Keras ImageDataGenerator)

7. Model Building (CNN)

* Build or use pretrained CNN models (like VGG16, Mobile Net)
* Layers:
  + Convolution → ReLU → Pooling → Fully Connected → Output

8. Model Training

* Use training data to train the CNN model
* Monitor accuracy and loss using validation data

9. Model Testing & Evaluation

* Evaluate model performance on unseen test data
* Check if it can correctly identify:
  + Diseased vs Healthy
  + Specific disease class
* Metrics: Accuracy, Precision, Recall, F1-Score