**PROBLEM STATEMENT**

The goal of this project is to develop a Convolutional Neural Network (CNN)-based model capable of detecting and classifying plant diseases from leaf images of various crops such as apple, cherry, grape, and corn. The model must accurately differentiate between healthy and diseased leaves, and further identify the specific type of disease affecting each leaf. This system will support precision agriculture by enabling timely disease detection and promoting efficient crop management strategies, ultimately helping to minimize crop loss and improve yield quality.

**PROJECT PIPELINE**

The project follows a structured pipeline as discussed in the lecture, encompassing the following key stages:

1. **Data Collection & Data Loading**
   * A pre-organized dataset is utilized, divided into train, test, and validation directories.
   * Each directory contains subfolders corresponding to different plant health categories (e.g., *Apple Scab*, *Grape Black Rot*, *Corn Gray Leaf Spot*, *Healthy*).
   * The dataset is loaded using data generators or TensorFlow utilities to facilitate real-time image preprocessing during training.
2. **Data ZIP & Mounting in Google Colab**
   * The dataset is compressed (zipped) and uploaded to Google Drive.
   * Using Google Colab, the Drive is mounted via authentication, and the dataset is extracted into the Colab environment.
   * This setup ensures seamless access to files for training without re-uploading during each session.
3. **Image Preprocessing & Augmentation**
   * Images are resized to a consistent resolution (e.g., 128x128 pixels) to standardize input dimensions.
   * Augmentation techniques such as horizontal/vertical flipping, zooming, rotation, and brightness adjustments are applied.
   * This enhances model robustness by simulating real-world variability and reducing overfitting.
4. **CNN Model Development**
   * A custom CNN architecture is defined using frameworks such as TensorFlow/Keras.
   * The model includes convolutional layers, activation functions (ReLU), pooling layers, dropout for regularization, and fully connected layers.
   * The network is trained on the processed training dataset, with validation data used to monitor overfitting and performance.
5. **Testing & Evaluation**
   * The trained model is evaluated on the unseen test dataset.
   * Evaluation metrics include accuracy, confusion matrix, precision, recall, and F1-score to provide a comprehensive understanding of performance.
   * Visualization tools such as training/validation loss and accuracy curves are used to interpret model learning behavior.