**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.

Explanation: In the field of agriculture, plants often get affected by different types of diseases. Farmers usually find it difficult to detect these diseases at an early stage, especially just by looking at the leaves. If a disease is not identified on time, it can spread and damage the whole crop. This leads to a big loss in farming and affects the farmer’s income.

To solve this issue, we are going to build a system that can automatically detect whether a plant leaf is healthy or infected. This system will use images of plant leaves and apply deep learning techniques to identify the disease. We will work mainly on crops like apple, cherry, grape, corn, and others. The system will help in early detection so that farmers can take action quickly and save their crops.

**Aim**

The main aim of this project is to create a deep learning-based system using CNN (Convolutional Neural Network) which can detect and classify diseases in plant leaves. This model will look at images of leaves and decide whether the plant is healthy or infected. If it is infected, it should also try to identify what disease it has. The system is designed to support smart and sustainable farming by making it easier for farmers to monitor their crops and take quick action.

**Project Pipeline**

1. Data Collection and Data Loading:

We already have a dataset in a ZIP file. It contains folders named train, test and validation. Inside each folder, there are more folders, and each one has images of leaves showing a particular disease or healthy condition (like 'Apple\_Black\_rot' or 'Apple\_healthy'). We use special tools called data loaders or data generators (from libraries like Keras, TensorFlow, or PyTorch) to load these images. While loading, we resize all images to the same size, divide them into small groups (batches), and shuffle them to make training better and faster.

2. Image Preprocessing and Image Augmentation:

Before giving the images to the model, we do some cleaning and preparation. First, we resize every image (for example, to 224x224 pixels) so all images look similar in size. We also scale the pixel values so the model can learn better. Then we apply image augmentation, which means we make small changes like rotating the image a little, zooming in, flipping it sideways, or changing the brightness. This helps the model learn from more variety and avoid overfitting (memorizing instead of understanding). These changes are only done during training, not while testing or validating.

3. CNN Model:

We use a Convolutional Neural Network (CNN) to train our model. It works in layers. First are convolution layers that find patterns in the image. Then come activation layers like ReLU that help the model understand complex things. Pooling layers reduce the size of data but keep important parts. After that, we use dense (fully connected) layers to make the final decision. We also use dropout layers to avoid overfitting. Finally, we compile the model using optimizers (like Adam) and loss functions (like categorical cross entropy) depending on how many classes we are predicting.

4. Test and Evaluate:

After training the model, we test it on the test dataset, which the model hasn’t seen before. We check how well it performs using accuracy, precision, recall, and a confusion matrix (a table that shows where the model is right or wrong). We also use the validation data while training to adjust the model and improve it. If results are not good, we can change the model's structure, increase layers, change the learning rate, or try better image augmentation.