PLANT DISEASE DETECTION & RECOMMENDATION SYSTEM

(4. Choose Model & 5. Model Training)

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

1. List the Machine Learning and Deep Learning models available.

ML Models: -

Supervised Learning:

Methods	Models
Linear Models	Linear Regression, Logistic Regression, Ridge Regression, Lasso Regression
Support Vector Machines (SVM)	Linear SVM, Kernel SVM (e.g., RBF, Polynomial)
Decision Trees and Ensemble Methods	Decision Tree, Random Forest, Gradient Boosting Machines (GBM), XGBoost, LightGBM, CatBoost, AdaBoost
Bayesian Methods	Naive Bayes (Gaussian, Multinomial, Bernoulli), Bayesian Networks
k-Nearest Neighbors (k-NN)	Classification, Regression
Neural Networks (Shallow Networks)	Multilayer Perceptron (MLP)

Unsupervised Learning:

Methods	Models
Clustering Algorithms	k-Means, Hierarchical Clustering, DBSCAN (Density-Based Spatial Clustering), Mean Shift
Dimensionality Reduction	Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-SNE (t-Distributed Stochastic Neighbor Embedding), UMAP (Uniform Manifold Approximation and Projection)
Association Rule Learning	Apriori, Eclat

Reinforcement Learning:

1. Q-Learning
2. Deep Q-Learning
3. SARSA (State-Action-Reward-State-Action)
4. Policy Gradient Methods

Deep Learning Algorithms

Algorithms	Models
Artificial Neural Networks (ANN)	Feedforward Neural Networks (FNN)
Convolutional Neural Networks (CNN) - Used for image data	AlexNet, VGGNet, EfficientNet, GoogLeNet/Inception, ResNet, DenseNet
Recurrent Neural Networks (RNN) - Used for sequential data	Vanilla RNN, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) Bidirectional RNNs
Transformers - Revolutionized NLP tasks	BERT (Bidirectional Encoder Representations from Transformers) GPT (Generative Pre-trained Transformer) T5 (Text-to-Text Transfer Transformer) ViT (Vision Transformer)
Generative Models	Variational Autoencoders (VAE), Generative Adversarial Networks (GAN) – [DCGAN, StyleGAN, CycleGAN]
Deep Reinforcement Learning	Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), A3C (Asynchronous Advantage Actor-Critic)
Specialized Networks	1. Autoencoders: Denoising Autoencoders, Sparse Autoencoders 2. Graph Neural Networks (GNN): GCN (Graph Convolutional Networks), GAT (Graph Attention Networks)
Self-Supervised Learning	SimCLR, BYOL (Bootstrap Your Own Latent), MoCo (Momentum Contrast)
Other Architectures	Capsule Networks, Attention Mechanisms (used in multiple deep learning domains)

2. **Learn about the models used specifically for plant disease and list the most appropriate models to use and why?**

1. Traditional Machine Learning Algorithms

Feature Extraction Methods:

- Histogram of Oriented Gradients (HOG)
- Gray-Level Co-Occurrence Matrix (GLCM)
- Local Binary Patterns (LBP)

Machine Learning Algorithms:

1. Support Vector Machine (SVM)
2. k-Nearest Neighbors (k-NN)
3. Random Forest

---

2. Deep Learning Algorithms

1. Convolutional Neural Networks (CNNs):
    - AlexNet: A simple and fast architecture for basic tasks.
    - VGGNet: Good for general-purpose image classification.
    - ResNet: Excellent for deeper networks, avoids vanishing gradients.
    - InceptionNet: Combines multiple convolution sizes to capture features at different scales.
    - DenseNet: Ensures efficient feature reuse by connecting every layer to every other layer.
2. Transfer Learning:
    - Using pre-trained CNN models and fine-tuning them on your dataset saves time and computational power.
    - Pre-trained models on ImageNet:
        - MobileNet: Lightweight, ideal for mobile or edge devices.
        - EfficientNet: State-of-the-art for efficient training and inference.
        - ResNet50 or InceptionV3: Widely used for plant disease detection.
3. Vision Transformers (ViT):
    - Emerging deep learning architecture for image classification tasks.
    - Useful for large datasets and complex patterns.
4. Generative Models for Augmentation:
    - Use GANs (Generative Adversarial Networks) or VAEs (Variational Autoencoders) to generate synthetic plant disease images to increase your dataset size and variety.

---

3. Hybrid and Ensemble Methods: Combining traditional ML and DL or using ensembles can improve performance:

1. Hybrid Feature Extraction: Use deep learning (e.g., ResNet) to extract features and feed them into a traditional ML algorithm like SVM for classification.
2. Ensemble DL Models: Combine predictions from multiple deep learning models (e.g., ResNet + InceptionV3) for better accuracy.

Reference**:**

- Paper19709.pdf
- Machine Learning and Deep Learning for Crop Disease Diagnosis: Performance Analysis and Review

- [(PDF) An advanced deep learning models-based plant disease detection: A review of recent research]

3. **What are the issues of overfitting and underfitting? How to resolve it?**
4. **CNN Hog**
5. **Build CNN and what is happening in augmentation.**
6. **How to find severity.**
7. **CNN -> CNN HOG -> SVM -> LDM (Linear Discriment Model**
8. **Results:**

MODEL CONFIG	MODEL RESULTS
Dataset: pre-augmented Epoch: 30 Batch size: 32	Accuracy: 95% Loss: 25%
Dataset: only original Epoch: 30, Batch size: 32 Layers: 3, Split: 80	Precision, Recall, F1-Score, Accuracy: 81% Loss: 82%
Dataset: only original Epoch: 50, Batch Size: 50 Layers: 3, Split: 70	Precision, Recall, F1-Score, Accuracy: 83.78% Loss: 116%
Dataset: only original Epoch: 20, Batch Size: 32 Layers: 3, Split: 70 Image Augmentation: rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode="nearest"	Precision, Recall, F1-Score, Accuracy: 63.17% Loss: 87%
Dataset: only original Epoch: 50, Batch Size: 50 Layers: 3, Split: 70 Image Augmentation: same	Precision, Recall, F1-Score, Accuracy: 77% Loss: 58% Prediction confidence: 1/3, 82%
Dataset: only original Epoch: 20, Batch Size: 50 Layers: 3, Split: 70 Image Augmentation: rotation_range=20, others same	Precision, Recall, F1-Score, Accuracy: 62% Loss: 93%
Dataset: only original Preprocessing: resize, crop & normalize Epoch: 50, Batch Size: 32 Layers: 3, Split: 70/30 Image Augmentation: rotation_range=20, others same	Precision, Recall, F1-Score, Accuracy: 78% Loss: 53% Prediction confidence: 58%, 2/3

Dataset: only original Preprocessing: only resize & normalise Epoch: 50, Batch Size: 32 Layers: 3, Split: 70/30 Image Augmentation: rotation_range=20, others same	Precision, Recall, F1-Score, Accuracy: 78.68% Loss: 49% Prediction confidence: 69%, ¾ Model name: 79acc_model
Dataset: only original Preprocessing: only resize & normalize Epoch: 50, Batch Size: 32 Layers: 3, Split: 80/20 Image Augmentation: rotation_range=20, others same	Precision, Recall, F1-Score, Accuracy: 86.05% Loss: 42% Prediction confidence: ¾, 63, 93, 100% Model name: 86acc_model
Dataset: only original Preprocessing: only resize & normalize Epoch: 50, Batch Size: 32 Layers: 3, Split: 80/20 Image Augmentation: rotation_range=30, others same	Precision, Recall, F1-Score, Accuracy: 91% Loss: 28% Prediction confidence: 4/4, 70-99% Model name: 91acc_model
Dataset: only original Preprocessing: only resize & normalize Epoch: 50, Batch Size: 16 Layers: 3, Split: 80/20 Image Augmentation: rotation_range=30, others same	Precision, Recall, F1-Score, Accuracy: 93.60% Loss: 19% Prediction confidence: 4/4, 90-99% Model name: 93acc_model
Dataset: only original Preprocessing: only resize & normalize Epoch: 50, Batch Size: 64 Layers: 3, Split: 80/20 Image Augmentation: rotation_range=30, others same	Precision, Recall, F1-Score, Accuracy: 85.47% Loss: 45% Prediction confidence: 3/4, 60-95% Model name: 854acc_model
Dataset: only original Preprocessing: only resize & normalize Epoch: 75, Batch Size: 16 Layers: 3, Split: 80/20 Image Augmentation: rotation_range=30, others same	Precision, Recall, F1-Score, Accuracy: 91.28% Loss: 29% Prediction confidence: 4/4, 42-99% Model name: 912acc_model Time: 22 minutes

Results:

- Increase epoch, increases accuracy.
- Remove crop, reduce loss of 4%.
- Split at 80:20, reduce loss of 7% & increase accuracy of 8%.
- Rotation range = 30 is good for my model.
- Batch size = 16 with epoch = 50 gave the best model of 93.60% with loss of 19%, checked in a fresh runtime, but took a bit more time than previous ones.
- Reduce batch_size, increases accuracy & vice-versa.