Assignment No-1

Title-To develop a machine learning model to build a recommendation system for crop disease detection and yield prediction in agriculture.

Objectives-

- To understand the fundamentals of machine learning and its application in agriculture, particularly for crop disease detection and yield prediction.
- To implement a machine learning model that can analyze crop data and make predictions about disease presence and expected yield.
- To evaluate the effectiveness of the model and interpret its predictions in the context of agricultural practices.

Outcomes-Students will be able to

- Grasp key concepts and applications in agriculture, including data handling and preprocessing skills.
- create and select effective features to improve model performance.
- Experience in training models, assessing performance, and analyzing outputs for actionable recommendations.
- Clear documentation of processes and enhanced problem-solving skills through iterative development

Theory-

To develop a machine learning model to build a recommendation system for crop disease detection and yield prediction in agriculture, follow these steps:

1. data collection

you need two types of data:

- **images of crop diseases**: images of healthy and diseased crops for disease detection.
- **agricultural/environmental data**: data on weather conditions, soil properties, water usage, fertilizers, etc., for yield prediction.

2. data preprocessing

- **image data**: prepare images for disease detection using techniques like resizing, normalization, and data augmentation.
- **tabular data**: clean and preprocess agricultural data (handling missing values, scaling, and feature engineering).

3. machine learning models

- **cnn (convolutional neural networks)**: used for disease detection.
- **regression models (random forest, xgboost, etc.)**: used for yield prediction based on environmental data.

4. building cnn for disease detection

- use cnn for image classification to detect diseases in crop images.
- the cnn architecture typically includes convolutional layers, max pooling, flattening, and fully connected layers.

5. building regression model for yield prediction

- use regression models to predict crop yield based on environmental factors such as rainfall, temperature, soil quality, and fertilizer usage.

6. training and evaluation

- train the cnn model for disease detection using labeled image data.
- train the regression model for yield prediction using the preprocessed tabular data.

7. recommendation system

- based on the disease detection and yield prediction results, provide recommendations.
- if a disease is detected, recommend treatments or preventive measures.
- if the yield is predicted to be low, suggest changes in farming practices (fertilizers, irrigation, etc.).

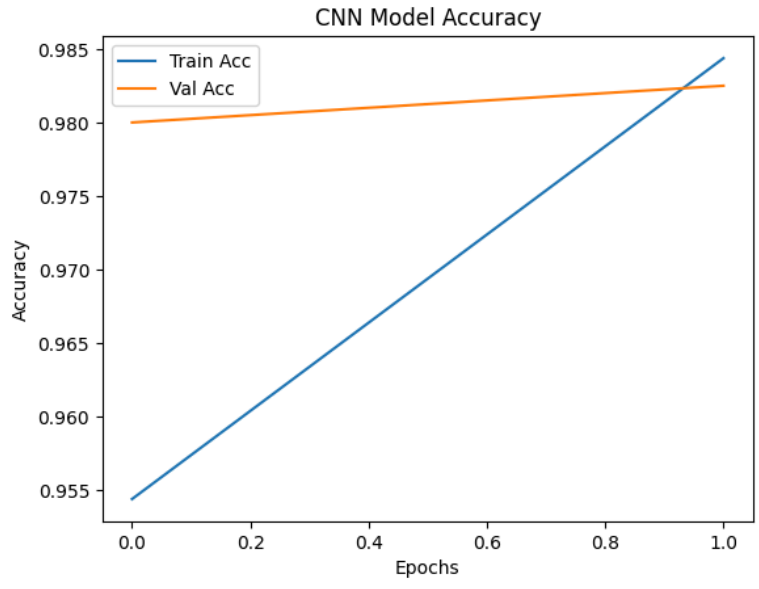
8. deployment

- deploy the trained models as part of a web or mobile application to allow farmers to upload images and input data for real-time recommendations.

```
# ================================================
# 🌾 Crop Disease & Yield Prediction System
# Using CNN + Random Forest + Recommendation Logic
# ================================================

# --- Step 0: Imports ---
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Changed import location
import matplotlib.pyplot as plt

# --- Step 1: CNN Model for Disease Detection ---
# ================================================

# Path to your dataset folder (change if needed)
dataset_path = '/content/drive/MyDrive/train'  # <-- upload your "train" folder to Colab root or Drive

# Data Preprocessing
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=(64, 64),
    batch_size=10,
    class_mode='binary',
    subset='training'
)

val_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=(64, 64),
    batch_size=10,
    class_mode='binary',
    subset='validation'
)

# Define CNN model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # binary output
])

# Compile
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train
history = cnn_model.fit(train_gen, validation_data=val_gen, epochs=2)

# Plot training accuracy
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('CNN Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# --- Step 2: Random Forest for Yield Prediction ---
# ================================================

# Create simple mock environmental dataset
np.random.seed(42)
X = np.random.rand(100, 3)  # rainfall, temperature, soil_quality
y = np.random.rand(100) * 100  # yield (in tons/hectare or any units)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest model
yield_model = RandomForestRegressor(n_estimators=100, random_state=42)
yield_model.fit(X_train, y_train)

# --- Step 3: Recommendation System ---
# ================================================
def recommend(disease_prediction, yield_prediction):
```

```
    if disease_prediction >= 0.5:
        return "⚠️ Disease detected! Recommended action: Apply pesticide."
    elif yield_prediction < 50:
        return "💧 Low yield predicted! Recommended action: Improve irrigation and soil quality."
    else:
        return "✅ Crop is healthy and yield prediction is optimal."

# --- Step 4: Testing the System ---
# ================================================

# Pick one random image from validation set
import random, os
from keras.preprocessing import image

# Path to a test image (you can replace with your own)
test_class = random.choice(['healthy', 'diseased'])
test_dir = os.path.join(dataset_path, test_class)
test_img_path = '/content/Gemini_Generated_Image_lxa0l3lxa0l3lxa0.png'

# Load and preprocess test image
img = image.load_img(test_img_path, target_size=(64, 64))
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)

# CNN prediction
disease_prediction = cnn_model.predict(img_array)[0][0]

# Simulated environmental data for yield model
test_env_data = np.array([[0.8, 0.6, 0.7]])  # rainfall, temp, soil quality
yield_prediction = yield_model.predict(test_env_data)[0]

# Generate recommendation
recommendation = recommend(disease_prediction, yield_prediction)
```

```
Found 1600 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
Epoch 1/2
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDatas
  self._warn_if_super_not_called()
160/160 ━━━━━━━━━━━━━━━━━━━━ 15s 48ms/step - accuracy: 0.9005 - loss: 0.2699 - val_accuracy: 0.9800 - val_loss: 0.0664
Epoch 2/2
160/160 ━━━━━━━━━━━━━━━━━━━━ 8s 49ms/step - accuracy: 0.9948 - loss: 0.0191 - val_accuracy: 0.9825 - val_loss: 0.0318
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 390ms/step
```

```
# --- Step 5: Display Output ---
# ================================================
print("🖼️ Test Image Path:", test_img_path)
print(f"🌿 Disease Prediction: {disease_prediction:.4f} (0: Healthy, 1: Diseased)")
print(f"🌾 Yield Prediction: {yield_prediction:.2f} units")
print(f"💡 Recommendation: {recommendation}")
```

```
🖼️ Test Image Path: /content/Gemini_Generated_Image_lxa0l3lxa0l3lxa0.png
🌿 Disease Prediction: 0.1881 (0: Healthy, 1: Diseased)
🌾 Yield Prediction: 46.35 units
💡 Recommendation: 💧 Low yield predicted! Recommended action: Improve irrigation and soil quality.
```