from google.colab import files

import os

import numpy as np

from sklearn.metrics import classification\_report, accuracy\_score, f1\_score

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras import layers, models

from tensorflow.keras.utils import load\_img, img\_to\_array

files.upload()

os.makedirs('/root/.kaggle', exist\_ok=True)

!mv kaggle.json /root/.kaggle/

!chmod 600 /root/.kaggle/kaggle.json

!pip install -q kaggle

!kaggle datasets download -d akshitgupta146/plant-disease-dataset-classification

!unzip -q -o plant-disease-dataset-classification.zip -d plant\_disease\_dataset

base\_path = 'plant\_disease\_dataset/PlantVillage'

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

train\_generator = datagen.flow\_from\_directory(

base\_path,

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical',

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

base\_path,

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical',

subset='validation'

)

model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

layers.MaxPooling2D(2, 2),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D(2, 2),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(train\_generator.num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.fit(train\_generator, epochs=5, validation\_data=validation\_generator)

validation\_generator.reset()

y\_pred\_probs = model.predict(validation\_generator, verbose=1)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

y\_true = validation\_generator.classes

class\_names = list(validation\_generator.class\_indices.keys())

acc = accuracy\_score(y\_true, y\_pred)

f1 = f1\_score(y\_true, y\_pred, average='macro')

print(f"\nAccuracy: {acc:.4f}")

print(f"F1 Score (macro): {f1:.4f}")

print("\nClassification Report:")

print(classification\_report(y\_true, y\_pred, target\_names=class\_names))

uploaded = files.upload()

img\_path = list(uploaded.keys())[0]

img = load\_img(img\_path, target\_size=(224, 224))

img\_array = img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class\_index = np.argmax(predictions, axis=1)[0]

predicted\_label = class\_names[predicted\_class\_index]

print(f"\nPredicted Disease: {predicted\_label}")