**Ai for AgriTech Hackathon – Python Code Files on Project.**

CNN-Based Plant Disease Detection – Python Code

This example uses Keras + TensorFlow and works with any dataset structured like:

dataset/

├── train/

│ ├── healthy/

│ ├── bacterial\_spot/

│ └── early\_blight/

└── validation/

├── healthy/

├── bacterial\_spot/

└── early\_blight/

**Step 1: Import Required Libraries**

import os

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam

**Step 2: Set Parameters and Paths**

IMG\_HEIGHT = 128

IMG\_WIDTH = 128

BATCH\_SIZE = 32

EPOCHS = 10

train\_dir = "dataset/train"

val\_dir = "dataset/validation"

**Step 3: Data Preprocessing**

train\_datagen = ImageDataGenerator(rescale=1./255)

val\_datagen = ImageDataGenerator(rescale=1./255)

train\_data = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

val\_data = val\_datagen.flow\_from\_directory(

val\_dir,

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

**Step 4: CNN Model Architecture**

model = Sequential([

Conv2D(32, (3,3), activation='relu', input\_shape=(IMG\_HEIGHT, IMG\_WIDTH, 3)),

MaxPooling2D(2,2),

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

MaxPooling2D(2,2),

Conv2D(128, (3,3), activation='relu'),

MaxPooling2D(2,2),

Flatten(),

Dense(256, activation='relu'),

Dropout(0.5),

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

])

model.compile(

optimizer=Adam(learning\_rate=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy']

)

model.summary()

**Step 5: Model Training**

history = model.fit(

train\_data,

epochs=EPOCHS,

validation\_data=val\_data

)

**Step 6: Save the Model**

model.save("plant\_disease\_model.h5")

**Step 7: Plot Accuracy & Loss Graphs**

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.title('Accuracy')

plt.show()

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.legend()

plt.title('Loss')

plt.show()