```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
import matplotlib.pyplot as plt
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import seaborn as sns
from sklearn.metrics import confusion_matrix

import cv2
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.models import model_from_json
import pickle
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

main = tkinter.Tk()
main.title("Crop Disease Detection & Classification ")
main.geometry("1000x650")

global ann_model
global filename
global X, Y
global X_train, X_test, y_train, y_test, testImage, pca
low_green = np.array([25, 52, 72])
high_green = np.array([102, 255, 255])
leaf_labels=['Apple___Apple_scab:drug1', 'Apple___Black_rot', 'Apple___Cedar_apple_rust',
'Apple___healthy',
        'Cherry_(including_sour)___healthy','Cherry_(including_sour)___Powdery_mildew',
        'Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot','Corn_(maize)___Common_rust_','Corn_(maize)___healthy','Corn_(maize)___
Northern_Leaf_Blight',

'Grape___Black_rot','Grape___Esca_(Black_Measles)','Grape___healthy','Grape___Leaf_blight
_(Isariopsis_Leaf_Spot)',

'Peach___Bacterial_spot','Peach___healthy','Pepper,_bell___Bacterial_spot','Pepper,_bell___h
```

```
ealthy',
        'Potato___Early_blight','Potato___healthy','Potato___Late_blight',
        'Rice___Brownspot','Rice___Healthy','Rice___RiceLeafblast','Rice___RiceLeafblight',
        'Strawberry___healthy','Strawberry___Leaf_scorch',
        'Sugarcane___healthy','Sugarcane___RedRot','Sugarcane___RedRust',

'Tomato___Bacterial_spot','Tomato___Early_blight','Tomato___healthy','Tomato___Late_blight',
        'Tomato___Leaf_Mold','Tomato___Septoria_leaf_spot','Tomato___Spider_mites
Two-spotted_spider_mite','Tomato___Target_Spot',
        'Tomato___Tomato_mosaic_virus','Tomato___Tomato_Yellow_Leaf_Curl_Virus']

pest_labels=['CAPTAN', 'CAPTAN', 'MANOCOZEB', 'NO PESTICIDE',
        'NO PESTICIDE','Pottasium K',
        'AZOXYSTOBIN','CHOLOROTHALONIN','NO PESTICIDE','CHOLOROTHALONIN',
        'CAPTAN','FOSTEY-ALUMINIUM','NO PESTICIDE','MANOCOZEB',
        'COPPER HYDEROXIDE','NO PESTICIDE','COPPER HYDEROXIDE','NO
PESTICIDE',
        'MANOCOZEB','NO PESTICIDE','CLOROTHYNOINE',
        'PROPICONAZOLE','NO PESTICIDE','PROPICONAZOLE','PROPICONAZOLE',
        'NO PESTICIDE','CAPTAN',
        'NO PESTICIDE','PROPICONAZOLE','TRIAZOLES',
        'COPPER HYDEROXIDE','MANOCOZEN','NO PESTICIDE','MEFENOZAN',
        'MAFENOZAN','CHLOROTHYNOINE','ABAMECTIN','MANOCOZEB',
        'AZOXYSTORBIN','NEEM OIL']

def loadDataset():
    global filename, dataset
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir=".")
    text.insert(END,str(filename)+" loaded\n\n")


def preprocessDataset():
    global X, Y, testImage
    text.delete('1.0', END)
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
    X = X.astype('float32')
    X = X/255
    testImage = X[0].reshape(64,64,3)
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    Y = to_categorical(Y)
    text.insert(END,"Image Processing Completed\n\n")
    text.insert(END,"Total images found in dataset: "+str(X.shape[0]))
```

```
def segmentation():
    text.delete('1.0', END)
    global X, Y, testImage, pca
    text.insert(END,"Total features available in image before applying Features Extraction
Algorithm: "+str(X.shape[1])+"\n")
    if os.path.exists('model/pca.pkl'):
        with open('model/pca.pkl', 'rb') as file:
            pca = pickle.load(file)
            X = pca.fit_transform(X)
        file.close()
    else:
        pca = PCA(n_components = 1200)
        X = pca.fit_transform(X)
        with open('model/pca.pkl', 'wb') as file:
            pickle.dump(pca, file)
        file.close()
    text.insert(END,"Total features available in image after applying Features Extraction
Algorithm: "+str(X.shape[1])+"\n")
    text.update_idletasks()
    cv2.imshow("Segmented Image",cv2.resize(testImage,(300,300)))
    cv2.waitKey(0)

def trainANN():
    text.delete('1.0', END)
    global X, Y
    global ann_model
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    text.insert(END,"Dataset Train & Test Split for CNN training\n")
    text.insert(END,"80% dataset will be used for training and 20% for testing\n\n")
    text.insert(END,"Training Size: "+str(X_train.shape[0])+"\n")
    text.insert(END,"Testing Size: "+str(X_test.shape[0])+"\n")
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            ann_model = model_from_json(loaded_model_json)
        json_file.close()
        ann_model.load_weights("model/model_weights.h5")
        ann_model._make_predict_function()
    else:
        #creating ann object
        ann_model = Sequential()
        #defining layers of ANN
        ann_model.add(Dense(512, input_shape=(X_train.shape[1],)))
        ann_model.add(Activation('relu'))
        ann_model.add(Dropout(0.3))
        ann_model.add(Dense(512))
        ann_model.add(Activation('relu'))
```

```
        ann_model.add(Dropout(0.3))
        ann_model.add(Dense(y_train.shape[1]))
        ann_model.add(Activation('softmax'))
        #compiling the model
        ann_model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
        #start training model
        hist = ann_model.fit(X_train, y_train, batch_size=16,epochs=15, validation_data=(X_test,
y_test))
        ann_model.save_weights('model/model_weights.h5')
        model_json = ann_model.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
    print(ann_model.summary())
    #perfrom prediction on test data
    predict = ann_model.predict(X_test)
    predict = np.argmax(predict, axis=1)
    y_test = np.argmax(y_test, axis=1)
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    a = accuracy_score(y_test,predict)*100    #calculate accuracy score
    text.insert(END,'CNN Accuracy  : '+str(a)+"\n")
    text.insert(END,'CNN Precision : '+str(p)+"\n")
    text.insert(END,'CNN Recall    : '+str(r)+"\n")
    text.insert(END,'CNN FScore    : '+str(f)+"\n\n")
    text.update_idletasks()
    LABELS = leaf_labels
    conf_matrix = confusion_matrix(y_test, predict) #calculate confusion matrix
    plt.figure(figsize =(16, 6))
    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS, annot = True,
cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,len(LABELS)])
    plt.title("CNN Leaf Disease Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()

def classification():
    global ann_model, pca
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename) #read input image
    Z = np.float32(img.reshape((-1,3))) #create Z value from image
```

```
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    K = 4
    _,labels,centers = cv2.kmeans(Z, K, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
#create labels and centroids
    labels = labels.reshape((img.shape[:-1]))
    reduced = np.uint8(centers)[labels] #segment image based on labels
    img = cv2.resize(img, (64,64), interpolation=cv2.INTER_CUBIC) #resize image
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(imgHSV, low_green, high_green) #remove out green part from the
image so we have only infected part
    mask = 255-mask #masking done here
    res = cv2.bitwise_and(img, img, mask=mask) #extract infected part
    segmented = res #get segmented image
    res = res.ravel()
    test = []
    test.append(res)
    test = np.asarray(test)
    test = pca.transform(test) #extract features using PCA
    print(test.shape)
    test = test.astype('float32') #normalized the pixel values
    test = test/255
    preds = ann_model.predict(test)#predict the disease using ann model
    predict = np.argmax(preds)
    print(predict)
    img = cv2.imread(filename)
    img = cv2.resize(img, (800,400))
    text.insert(END,'Crop Disease Detected & Classified as : '+leaf_labels[predict]+"\n")
    text.update_idletasks()
    cv2.putText(img, 'Crop Disease Detected & Classified as : '+leaf_labels[predict], (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
    cv2.putText(img, 'Pesticide Required is : '+pest_labels[predict], (10, 45),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)

    cv2.imshow('Crop Disease Detected & Classified as : '+leaf_labels[predict], img)
    cv2.imshow("Segmented Image",cv2.resize(segmented,(200,200)))
    cv2.waitKey(0)

font = ('times', 15, 'bold')
title = Label(main, text='Crop Disease Detection & Classification', justify=LEFT)
title.config(bg='lavender blush', fg='DarkOrchid1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 12, 'bold')
loadButton = Button(main, text="Upload crop Disease Dataset", command=loadDataset)
loadButton.place(x=10,y=100)
```

```
loadButton.config(font=font1)

preprocessButton = Button(main, text="Image Preprocessing", command=preprocessDataset)
preprocessButton.place(x=300,y=100)
preprocessButton.config(font=font1)

segButton = Button(main, text="Segmentation & Features Extraction", command=segmentation)
segButton.place(x=530,y=100)
segButton.config(font=font1)

annButton = Button(main, text="Train CNN Algorithm", command=trainANN)
annButton.place(x=10,y=150)
annButton.config(font=font1)

clsButton = Button(main, text="Disease Classification", command=classification)
clsButton.place(x=300,y=150)
clsButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=160)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)

main.config(bg='light coral')
main.mainloop()
```