```
from zipfile import ZipFile

filename = "plant_images.zip"

with ZipFile(filename, 'r') as zip:
    zip.extractall()

print("done")
```

```
#prerequisites

from zipfile import ZipFile
import os
import matplotlib.pyplot as plt
import random
import numpy as np

IMAGE_WIDTH=200
IMAGE_HEIGHT=200
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3 # RGB color
batch_size=1
```

```
#prerequisites

from zipfile import ZipFile
import os
import matplotlib.pyplot as plt
import random
import numpy as np

IMAGE_WIDTH=200
IMAGE_HEIGHT=200
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3 # RGB color
batch_size=1


# Model
model = Sequential()
```

```
model.add(Conv2D(32, (3, 3), input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT,
IMAGE_CHANNELS), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
            optimizer=RMSprop(lr=0.0001),
            metrics=['accuracy'])

print('model created')
```

```
# training the model

history = model.fit_generator(
    train,
    steps_per_epoch  = train.samples // batch_size,
    epochs = 6,
    verbose = 1,
    validation_data = validation,
    validation_steps = validation.samples // batch_size,
)
print('model trained')

#saving the model in file
from keras.models import load_model
model.save('model.h')

import matplotlib.image as mpimg
from keras.preprocessing import image

# predicting about random image
test_data_dir = os.listdir(dir)
TrueLabel = random.choice(test_data_dir)
sample = random.choice(os.listdir(dir+"/"+TrueLabel))
path = dir+"/"+TrueLabel+"/"+sample
imag = image.load_img(path,target_size = IMAGE_SIZE)
imag = image.img_to_array(imag)
imag = np.expand_dims(imag,axis=0)
imag = imag/255
prob = model.predict(imag)

if prob > 0.5:
    plt.title("predicted: "+"%.2f" % (prob[0]*100) + "% uninfected " +
"        actual:" + TrueLabel)
else:
    plt.title("predicted: "+"%.2f" % ((1-prob[0])*100) + "% infected "+
"    actual:" + TrueLabel)



plt.imshow(image.load_img(path,target_size=(112,112)))
```