Smart Internz

Pneumonia Detection using X-rays using Watson Studio

Index

1. Introduction:

1.1. Overview

Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill the pus causing cough with pus, fever, chills and difficulty in breathing. Patients with pneumonia may need to be hospitalized or admitted to the ICU which is more than a million and over 50,000 deaths each year. It is the world's leading cause of death for children under the age of 5. In US, it is the most common cause of hospitalization for children and adults.

1.2. Purpose

Chest x-rays have been considered as the best tool to detect any form of Pneumonia. Studies have shown that even experienced radiologists have a hard time to correctly identify whether something on the x-rays is an infiltrate because the pus is denser than air. The lack of doctors and treatment of Pneumonia may also become more difficult. So, an accurate and fast diagnosis of Pneumonia is urgently needed, especially for people in poverty. So, the main purpose is to build a model to automatically identify whether a patient is suffering from Pneumonia or not by checking x-ray images.

2. Literature Survey:

2.1. Existing Problem

Lack of doctors or inaccurate predictions of Pneumonia may make the situations worse and may cause deaths especially for poor and uneducated people as they don't know what to do and what doctors say when they are attacked with Pneumonia. Especially, in these tough situations like COVID-19, doctors have no time to check x-rays and detect the Pneumonia as they are busy in serving COVID-19 patients. In some cases, even though a doctor checks an x-ray of a patient who is mildly attacked by the Pneumonia but considered it as non-infected, his situation may become serious in the coming days.

2.2. Proposed Solution

Keeping the problems and difficulties of Pneumonia patients in mind, a new solution is proposed. An app has been built so that the users just need to upload x-rays and click on predict. Then the prediction will be displayed after clicking on predict button. Users can make use of it in these tough situations or when they are unable to consult a doctor or when they want accurate prediction. It is easy to use and free of cost. Based on that prediction, they can move on to further medications or they may consult doctors virtually as checking x-ray physically is not required.

3. <u>Theoretical Analysis</u>:

3.1. Block Diagram
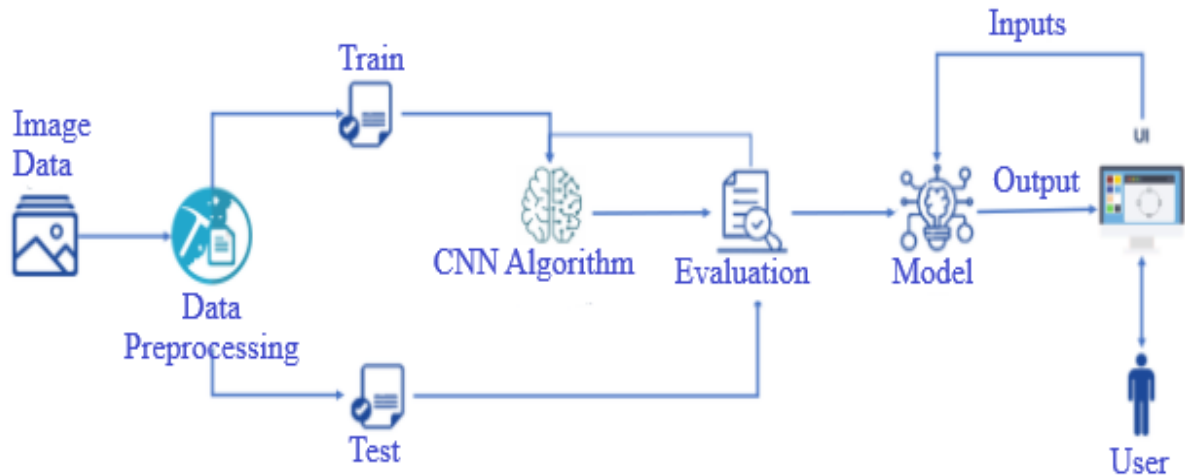
3.2. Hardware / Software Designing

Software Requirements:

★ OS - Windows XP 7/8/10
★ Anaconda
★ Jupyter notebook/Pycharm/Spyder
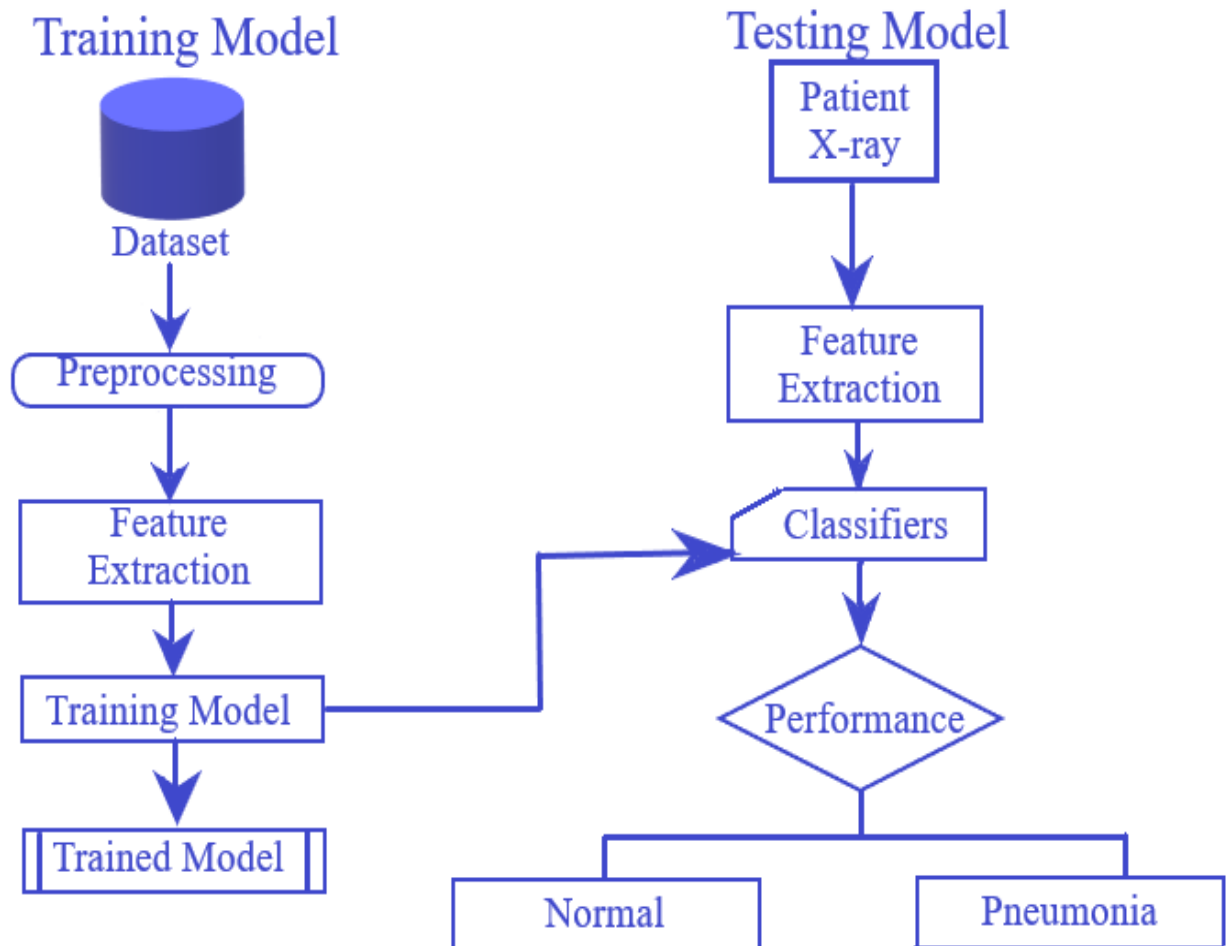★ Python
★ Cloud storage space in IBM

Hardware Requirements:

★ I3 processor
★ RAM – 1GB

4. Experimental Investigations:

In this project, the main aim is to predict whether a person is affected with pneumonia or not. So, firstly, image preprocessing is done in which the images are classified into 2 classes – NORMAL, PNEUMONIA. After preprocessing, model is initialized and CNN layers are added to that model and it is fitted and saved. At last, we test the model. After testing the model, an app is built for user interface. Users can access that app and they can upload their x-rays and they themselves can predict whether they are infected or they are alright. So simply and easily, we can predict Pneumonia with an x-ray at our own place and at any time at free of cost.

5. Flowchart:

Training Model

Dataset

↓

Preprocessing

↓

Feature Extraction

↓

Training Model

↓

Trained Model

Testing Model

Patient X-ray

↓

Feature Extraction

↓

Classifiers

↓

Performance

Normal   Pneumonia

6. <u>Result:</u>

→ The user will be uploading an x-ray image through the UI interface which is created by flask application.

→ The input image is loaded into the model.

→ This model performs the preprocessing and classifies the class name of the uploaded image (0-Normal, 1-Pneumonia) and predicts whether the x-ray is of affected person or not.

→ The output will be sent to the flask app using an http request and when the user clicks on predict, it displays the prediction.

7. Advantages and Disadvantages:

Advantages:

☺ Very simple and easy as the user just need to upload the x-ray image.

☺ Free of cost. The user can predict pneumonia without consulting doctor.

☺ Prediction is 90% accurate.

☺ Early prediction of Pneumonia before the situation becomes worse as early prediction may just need medication rather than hospitalization.

☺ It is mostly useful for people who is living in remote areas where there may be no doctors.

Disadvantages:

☹ The user should have x-ray to predict Pneumonia. Without x-ray, there is no use with the app.

☹ For uneducated people and for old people, it may take some time to get used to it.

☹ Users must have smart phones and internet to capture the x-ray images and to upload them into the app for prediction.

☹ Poor people who do not have at least a smart phone may have no use with this app.

8. <u>Applications</u>:

- ➢ In hospitals there is no need for doctors to predict Pneumonia. The compounders who have basic knowledge may predict and may suggest the patients to consult the doctor for medication.
- ➢ It can be used at residentials houses where people with basic knowledge of mobiles can use it.
- ➢ Today's youth have basic skills about smart phones. So, they can help poor and old people or children who are living in their surroundings to predict the disease.
- ➢ It can be used anywhere and anybody who is having smartphones and internet.

9. <u>Conclusion</u>:

- ✓ Our CNN model can effectively detect pneumonia disease with extremely high accuracy.
- ✓ Radiologists could combine this model and their expertise to make a highly accurate diagnosis of Pneumonia.
- ✓ Mortality rate can be reduced by early prediction of Pneumonia.

10. <u>Future Scope</u>:

In the future, it would be interesting to see approaches in which the weights corresponding to different models can be estimated more efficiently and a model that takes into account the patient's history while making predictions.

11. <u>Bibliography</u>:

- o https://towardsdatascience.com/pneumonia-detection-with-keras-and-fastapi-6c10dab657e0
- o Diagnostics | Free Full-Text | Efficient Pneumonia Detection in Chest Xray Images Using Deep Transfer Learning | HTML (mdpi.com)
- o Chest X-Rays Pneumonia Detection Using Convolutional Neural Network | by Max | Analytics Vidhya | May, 2021 | Medium
- o https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7345724/

12. Appendix:

12.1. Source Code

```python
In [1]: from keras.preprocessing.image import ImageDataGenerator, load_img
        from keras.models import Sequential
        from keras.layers import Conv2D, MaxPooling2D
        from keras.layers import Activation, Dropout, Flatten, Dense
        from keras import backend as K
        import os
        import numpy as np
        import pandas as np
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```python
In [2]: train = 'E:/project/dataset/chest_xray/train'
        test = 'E:/project/dataset/chest_xray/test'
        val = 'E:/project/dataset/chest_xray/val'
```

```python
In [3]: img_width,img_height= 150,150
        input_shape = (img_width,img_height,3)
```

```python
In [4]: model = Sequential()
        # The number of filters are 32 and the kernal_size is (3,3)
        model.add(Conv2D(32, (3, 3), input_shape=input_shape))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(32, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(128, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Flatten())
        model.add(Dropout(0.5))
        model.add(Dense(128))
        model.add(Activation('relu'))
        model.add(Dropout(0.5))
        model.add(Dense(64))
        model.add(Activation('relu'))
        model.add(Dropout(0.5))
        model.add(Dense(1))
        model.add(Activation('sigmoid'))
```

```
In [5]: model.compile(loss='binary_crossentropy',
                optimizer='rmsprop',
                metrics=['accuracy'])
```

```
In [6]: batch_size = 16
        train_datagen = ImageDataGenerator(rescale=1. / 255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        test_datagen = ImageDataGenerator(rescale=1. / 255)
```

```
In [7]: train_generator = train_datagen.flow_from_directory(
            train,
            target_size=(img_width, img_height),
            batch_size=16,
            class_mode='binary')

        test_generator = test_datagen.flow_from_directory(
            test,
            target_size=(img_width, img_height),
            batch_size=16,
            class_mode='binary')

        validation_generator = test_datagen.flow_from_directory(
            val,
            target_size=(img_width, img_height),
            batch_size=16,
            class_mode='binary')
```

```
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

```
In [8]: model.fit_generator(
            train_generator,
            steps_per_epoch=5217 // 16,
            epochs = 15,
            validation_data=validation_generator,
            validation_steps=17 // 16)
```

C:\Users\Home\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

```
Epoch 1/15
326/326 [==============================] - 1068s 3s/step - loss: 0.6060 - accuracy: 0.7155 - val_loss: 0.6731 - val_accuracy: 0.5625
Epoch 2/15
326/326 [==============================] - 812s 2s/step - loss: 0.3610 - accuracy: 0.8449 - val_loss: 0.7464 - val_accuracy: 0.8125
Epoch 3/15
326/326 [==============================] - 734s 2s/step - loss: 0.2939 - accuracy: 0.8877 - val_loss: 0.5322 - val_accuracy: 0.8750
Epoch 4/15
326/326 [==============================] - 745s 2s/step - loss: 0.2723 - accuracy: 0.9073 - val_loss: 1.1103 - val_accuracy: 0.7500
Epoch 5/15
326/326 [==============================] - 574s 2s/step - loss: 0.2771 - accuracy: 0.9066 - val_loss: 1.2444 - val_accuracy: 0.6250
Epoch 6/15
326/326 [==============================] - 525s 2s/step - loss: 0.2633 - accuracy: 0.9061 - val_loss: 0.8852 - val_accuracy: 0.7500
Epoch 7/15
326/326 [==============================] - 499s 2s/step - loss: 0.2618 - accuracy: 0.8993 - val_loss: 1.3730 - val_accuracy: 0.6250
Epoch 8/15
326/326 [==============================] - 481s 1s/step - loss: 0.2404 - accuracy: 0.9148 - val_loss: 1.2890 - val_accuracy: 0.6250
Epoch 9/15
326/326 [==============================] - 474s 1s/step - loss: 0.3186 - accuracy: 0.9194 - val_loss: 1.2960 - val_accuracy: 0.6250
Epoch 10/15
326/326 [==============================] - 496s 2s/step - loss: 0.2397 - accuracy: 0.9133 - val_loss: 0.6559 - val_accuracy: 0.7500
Epoch 11/15
```

```
In [9]:  test_acc = model.evaluate_generator(test_generator,624/16)
         print("\nAccuracy:"+" %.2f%%" % ( test_acc[1]*100))
```

C:\Users\Home\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1877: UserWarning: `Model.evaluate_generator` is depr
ecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '

Accuracy: 90.38%

```
In [10]:  model.save('mymodel.h5')
```

```
In [11]:  img_n = load_img('E:/project/dataset/chest_xray/train/NORMAL/IM-0234-0001.jpeg')
          plt.imshow(img_n)
          plt.title("Normal")
          plt.show()

          img_p = load_img('E:/project/dataset/chest_xray/train/PNEUMONIA/person7_bacteria_24.jpeg')
          plt.imshow(img_p)
          plt.title("Pneumonia")
          plt.show()
```



```
img_p = load_img('E:/project/dataset/chest_xray/train/PNEUMONIA/person7_bacteria_24.jpeg')
plt.imshow(img_p)
plt.title("Pneumonia")
plt.show()
```
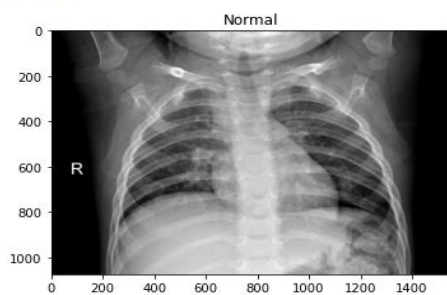
```
In [17]: from tensorflow.python.keras.models import load_model
         from keras.preprocessing import image
         import numpy as np
         import cv2
         model = load_model('mymodel.h5')
         model.compile(loss='categorical_crossentropy',
                       optimizer='adam',
                       metrics=['accuracy'])
         from skimage.transform import resize
         def detect(frame):
             #print(type(frame))
           #try:
             img = resize(frame,(150,150))
             img = np.expand_dims(img,axis=0)
             #if(np.max(img)>1):
              #   img = img/255.0
             prediction = model.predict(img)
             print(prediction)
             prediction = model.predict_classes(img)
             print(prediction)
           #except AttributeError:
            # print("shape not found")
```

```
In [21]: frame=cv2.imread("E:/project/dataset/chest_xray/test/PNEUMONIA/person109_bacteria_512.jpeg")
         #print(frame)
         data = detect(frame)
```

```
[[0.9969543]]
[[1]]
```

```
In [22]: frame=cv2.imread("E:/project/dataset/chest_xray/test/NORMAL/NORMAL2-IM-0249-0001.jpeg")
         #print(frame)
         data = detect(frame)
```
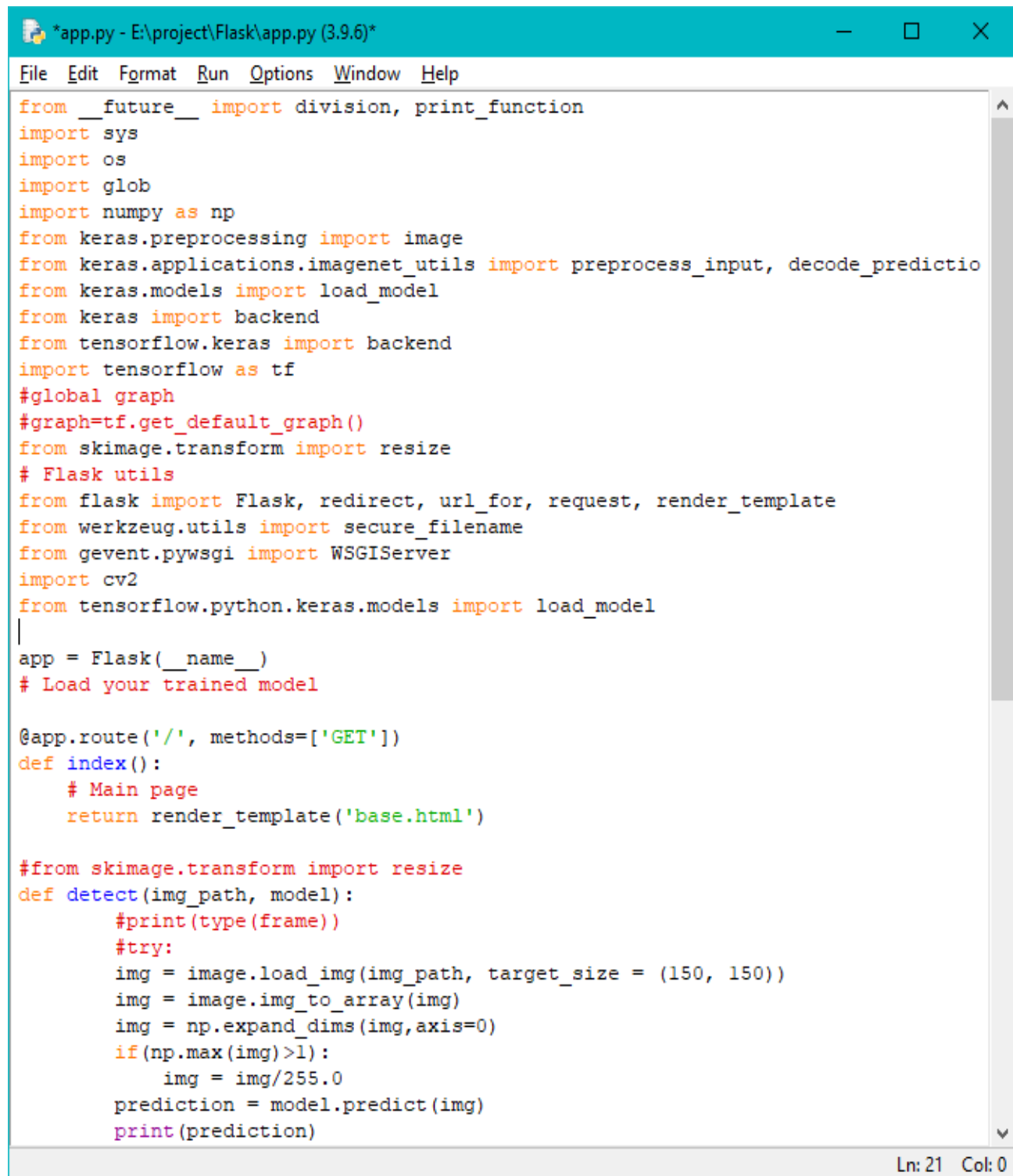
```
[[0.04447159]]
[[0]]
```

```
In [23]: frame=cv2.imread("E:/project/dataset/chest_xray/test/NORMAL/NORMAL2-IM-0290-0001.jpeg")
         #print(frame)
         data = detect(frame)
```

```
[[0.01325998]]
[[0]]
```

```
In [24]: frame=cv2.imread("E:/project/dataset/chest_xray/test/PNEUMONIA/person137_bacteria_655.jpeg")
         #print(frame)
         data = detect(frame)
```

```
[[0.9398306]]
[[1]]
```

Python (Flask):

```
from __future__ import division, print_function
import sys
import os
import glob
import numpy as np
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_predictio
from keras.models import load_model
from keras import backend
from tensorflow.keras import backend
import tensorflow as tf
#global graph
#graph=tf.get_default_graph()
from skimage.transform import resize
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
import cv2
from tensorflow.python.keras.models import load_model

app = Flask(__name__)
# Load your trained model

@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('base.html')

#from skimage.transform import resize
def detect(img_path, model):
        #print(type(frame))
        #try:
        img = image.load_img(img_path, target_size = (150, 150))
        img = image.img_to_array(img)
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img = img/255.0
        prediction = model.predict(img)
        print(prediction)
```

Ln: 21   Col: 0

File Edit Format Run Options Window Help

```python
def detect(img_path, model):
        #print(type(frame))
        #try:
        img = image.load_img(img_path, target_size = (150, 150))
        img = image.img_to_array(img)
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img = img/255.0
        prediction = model.predict(img)
        print(prediction)
        prediction = model.predict_classes(img)
        print(prediction)
        return prediction
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        model = load_model("mymodel.h5")
        print('Model loaded. Check http://127.0.0.1:5000/')
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics
        #img = image.load_img(file_path, target_size=(150, 150))
        preds = detect(file_path, model)
        #print(float(preds))
        if preds == 0:
            text = "You are perfectly fine"
        else:
            text = "You are infected! Please Consult Doctor"
        text = text
            # ImageNet Decode
        return text


if __name__ == '__main__':
    app.run(debug=True,threaded = False)
```

Ln: 21   Col: 0

12.2. UI Output Screenshot