Pneumonia detection from x-ray images

Internship title: RSIP Career Basic AI 077
Project ID: SPS_PRO_182

1. INTRODUCTION:

OVERVIEW:

In general, a patient suffering from Pneumonia goes to the hospital to take an X-ray image waits for the doctor and then the doctor will check the X-ray then he decides whether the person has pneumonia or not. The results are not only concluded based on just seeing the X-ray images but furthermore, tests were conducted on the patient to verify the results of the doctor. The process is time-consuming and if the patient has severe pneumonia or not he has to wait several days to get the test results. But in recent developments of the artificial intelligence and the computational powers of the computers have increased it helps in predicting pneumonia by just passing the X-ray image as an input to our model. pneumonia takes place when an infection makes the air sacs (alveoli) in the lungs fill **with** fluid or pus that might affect either one or both lungs. If your doctor thinks you might have **pneumonia**, a chest **X-ray** will be performed to find the infection in the patient's lungs and how far it's spread.the process of testing and verifying by doctor is veru time consuming and if patient has severe pneumonia or if he has to wait for long for the results.DeepLearning is very efficient approach` for predection.

Purpose:

The main objective of this project is to help the doctors to predict the pneumonia disease more accurately using a deep learning model. The objective is not only to help the doctors but also to the patients to verify whether they have pneumonia or not. By using this model we can precisely predict pneumonia.

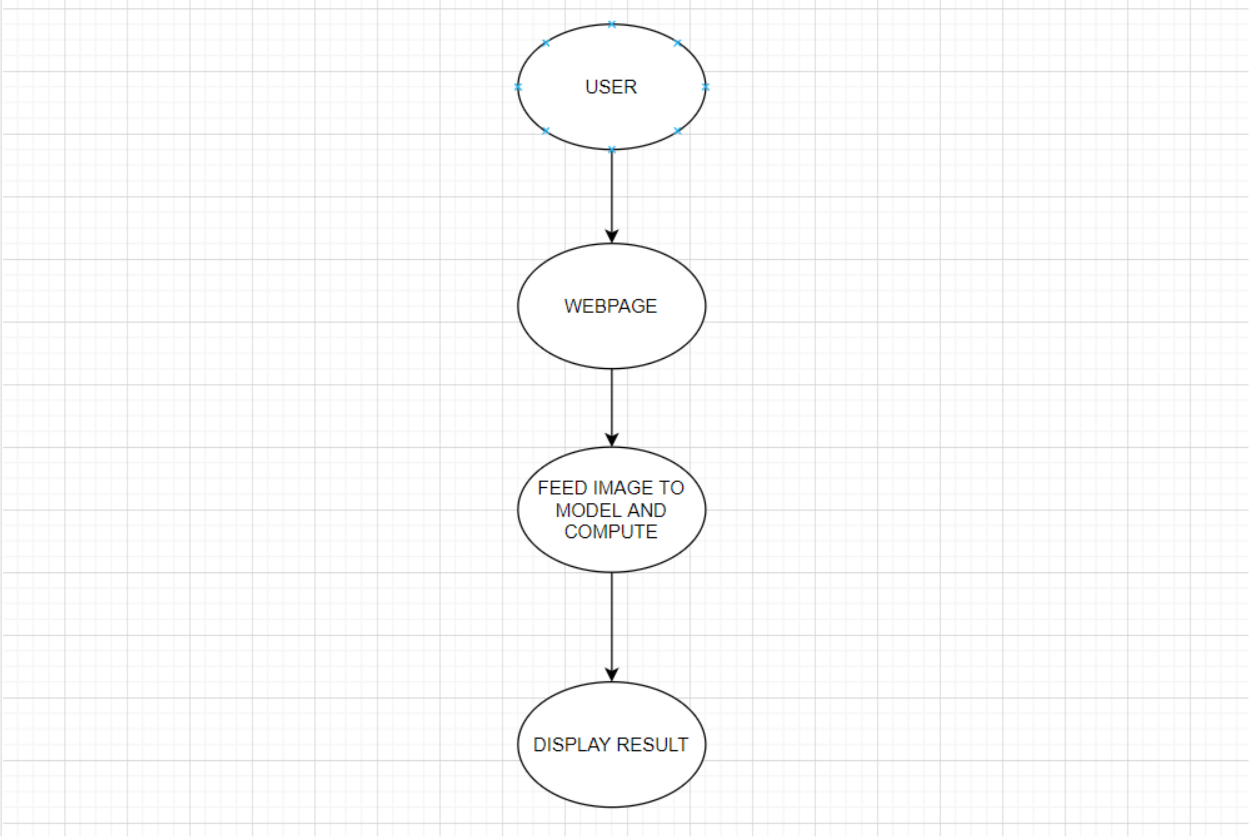
2.Literature Survey:

Exisiting problem:this is time consuming and even cost consuming

Proposed Solution:A convolutional neural network model is built from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with pneumonia. a web is built where the user can upload the x -ray image and the result is shown on the UI

3.Theoritical Analysis:

block diagram



Hardware / Software designing

Model Building

Importing the required libraries

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.layers import
Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization,add
from tensorflow.keras.models  import Model,Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
import os
```

->Transfer Learning:

• Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem. In this particular case, we have used Inceptionv3 and trained the latter layers to create CNN that can help us detect pneumonia from these chest x-ray images.

• In transfer learning, a base network is trained on a base dataset and task, and then it is used to "repurpose the learned features or transfer them" to a "second target network" to be trained on a target dataset and task.This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.Tensorflow supports transfer learning with a variety of models with pre-trained weight

->Tensor supports the transfer learning with various models like **VGG16**

VGG16:

The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

Loading the VGG16 mode

In [2]:

```
# Initializing VGG16 model
vgg = VGG16(weights= 'imagenet', include_top = False, input_shape = (224,224,3))
# include_top = False (loads full model without LAST fully connected layers)
```

```
for layer in vgg.layers:
  layer.trainable = False #making all the layers non-trainable


# Flattening out the last layer
x = Flatten()(vgg.output)
# Adding a dense layer
# To determine if the person is Healthy of Pneumonitic
predictions = Dense(2,activation='softmax')(x)
model = Model(inputs=vgg.input, outputs=predictions)
model.summary()
Model: "model"
_________________________________________________________________
Layer (type)             Output Shape             Param #
=================================================================
input_1 (InputLayer)       [(None, 224, 224, 3)]    0
_________________________________________________________________
block1_conv1 (Conv2D)       (None, 224, 224, 64)     1792
_________________________________________________________________
block1_conv2 (Conv2D)       (None, 224, 224, 64)     36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)     0
_________________________________________________________________
block2_conv1 (Conv2D)       (None, 112, 112, 128)    73856
_________________________________________________________________
block2_conv2 (Conv2D)       (None, 112, 112, 128)    147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)      0
_________________________________________________________________
block3_conv1 (Conv2D)       (None, 56, 56, 256)      295168
_________________________________________________________________
block3_conv2 (Conv2D)       (None, 56, 56, 256)      590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, 56, 56, 256)      590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)      0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, 28, 28, 512)      1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, 28, 28, 512)      2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, 28, 28, 512)      2359808
_________________________________________________________________
```

```
block4_pool (MaxPooling2D)   (None, 14, 14, 512)      0

block5_conv1 (Conv2D)       (None, 14, 14, 512)      2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)      2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)      2359808

block5_pool (MaxPooling2D)   (None, 7, 7, 512)        0

flatten (Flatten)           (None, 25088)            0

dense (Dense)               (None, 2)                50178
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
```

Data Generator Initialization for trainning the model

In [3]:

```
# Getting current directory
base_dir = os.getcwd()
# Defining the input shape
target_shape = (224,224)
train_dir = base_dir+"\\chest_xray\\train" #
val_dir = base_dir+"\\chest_xray\\val"    # -- Directories for data
test_dir = base_dir+"\\chest_xray\\test"   #
```

Loading the VGG16 model with Imagenet weights without the Fully Connected layers

In [4]:

```
vgg = VGG16(weights= 'imagenet', include_top = False, input_shape = (224,224,3))
for layer in vgg.layers:
    layer.trainable = False # Making all the layers non-trainable
```

In [5]:

```
# Flattening out the last layer
x = Flatten()(vgg.output)
predictions = Dense(2,activation='softmax')(x) #Dense layer to predict wether their is pneumonia or not
model = Model(inputs=vgg.input, outputs=predictions)
model.summary()
```

```
Model: "model_1"
```

```
_________________________________________________________________
Layer (type)              Output Shape           Param #
=================================================================
input_2 (InputLayer)      [(None, 224, 224, 3)]    0
_________________________________________________________________
block1_conv1 (Conv2D)      (None, 224, 224, 64)     1792
_________________________________________________________________
block1_conv2 (Conv2D)      (None, 224, 224, 64)     36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)     0
_________________________________________________________________
block2_conv1 (Conv2D)      (None, 112, 112, 128)    73856
_________________________________________________________________
block2_conv2 (Conv2D)      (None, 112, 112, 128)    147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)      0
_________________________________________________________________
block3_conv1 (Conv2D)      (None, 56, 56, 256)     295168
_________________________________________________________________
block3_conv2 (Conv2D)      (None, 56, 56, 256)     590080
_________________________________________________________________
block3_conv3 (Conv2D)      (None, 56, 56, 256)     590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)      0
_________________________________________________________________
block4_conv1 (Conv2D)      (None, 28, 28, 512)     1180160
_________________________________________________________________
block4_conv2 (Conv2D)      (None, 28, 28, 512)     2359808
_________________________________________________________________
block4_conv3 (Conv2D)      (None, 28, 28, 512)     2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)      0
_________________________________________________________________
block5_conv1 (Conv2D)      (None, 14, 14, 512)     2359808
_________________________________________________________________
block5_conv2 (Conv2D)      (None, 14, 14, 512)     2359808
_________________________________________________________________
block5_conv3 (Conv2D)      (None, 14, 14, 512)     2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)       0
_________________________________________________________________
flatten_1 (Flatten)        (None, 25088)           0
_________________________________________________________________
```

```
dense_1 (Dense)          (None, 2)            50178
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [6]:

```
# Making the data loader for training data
train_gen = ImageDataGenerator(rescale=1/255.0,
                    horizontal_flip=True,
                    zoom_range=0.2,
                    shear_range=0.2)
# Making the data loader for validation data
test_gen = ImageDataGenerator(rescale=1/255.0)
# Function to make iterable object for training
train_data_gen = train_gen.flow_from_directory(train_dir,
                              target_shape,
                              batch_size=16,
                              class_mode='categorical')
# Function to make iterable object for training
test_data_gen = train_gen.flow_from_directory(test_dir,
                              target_shape,
                              batch_size=16,
                              class_mode='categorical')
```

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

Compiling and Training the Model

In [13]:

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
hist = model.fit_generator(train_data_gen,
     steps_per_epoch=30,
     epochs=30,
     validation_data=test_data_gen,
     validation_steps=20)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
```

```
Train for 30 steps, validate for 20 steps
Epoch 1/30
30/30 [==============================] - 233s 8s/step - loss: 0.1462 - accuracy: 0.9479 - val_loss: 0.6283 -
val_accuracy: 0.8594
Epoch 2/30
30/30 [==============================] - 232s 8s/step - loss: 0.1685 - accuracy: 0.9521 - val_loss: 0.9052 -
val_accuracy: 0.7781
Epoch 3/30
30/30 [==============================] - 236s 8s/step - loss: 0.1647 - accuracy: 0.9542 - val_loss: 0.3593 -
val_accuracy: 0.8969
Epoch 4/30
30/30 [==============================] - 233s 8s/step - loss: 0.2274 - accuracy: 0.9250 - val_loss: 0.3991 -
val_accuracy: 0.8813
Epoch 5/30
30/30 [==============================] - 233s 8s/step - loss: 0.1178 - accuracy: 0.9604 - val_loss: 0.6033 -
val_accuracy: 0.8813
Epoch 6/30
30/30 [==============================] - 233s 8s/step - loss: 0.1679 - accuracy: 0.9521 - val_loss: 1.8353 -
val_accuracy: 0.7188
Epoch 7/30
30/30 [==============================] - 233s 8s/step - loss: 0.1584 - accuracy: 0.9438 - val_loss: 1.0335 -
val_accuracy: 0.8219
Epoch 8/30
30/30 [==============================] - 233s 8s/step - loss: 0.1987 - accuracy: 0.9458 - val_loss: 0.6965 -
val_accuracy: 0.8625
Epoch 9/30
30/30 [==============================] - 233s 8s/step - loss: 0.2276 - accuracy: 0.9396 - val_loss: 0.7445 -
val_accuracy: 0.8625
Epoch 10/30
30/30 [==============================] - 238s 8s/step - loss: 0.1540 - accuracy: 0.9479 - val_loss: 0.3813 -
val_accuracy: 0.8969
Epoch 11/30
30/30 [==============================] - 244s 8s/step - loss: 0.0942 - accuracy: 0.9583 - val_loss: 0.4050 -
val_accuracy: 0.8906
Epoch 12/30
30/30 [==============================] - 246s 8s/step - loss: 0.0882 - accuracy: 0.9792 - val_loss: 1.0757 -
val_accuracy: 0.7969
Epoch 13/30
30/30 [==============================] - 239s 8s/step - loss: 0.1860 - accuracy: 0.9521 - val_loss: 0.6986 -
val_accuracy: 0.8750
Epoch 14/30
30/30 [==============================] - 240s 8s/step - loss: 0.1717 - accuracy: 0.9438 - val_loss: 0.3003 -
val_accuracy: 0.9094
```

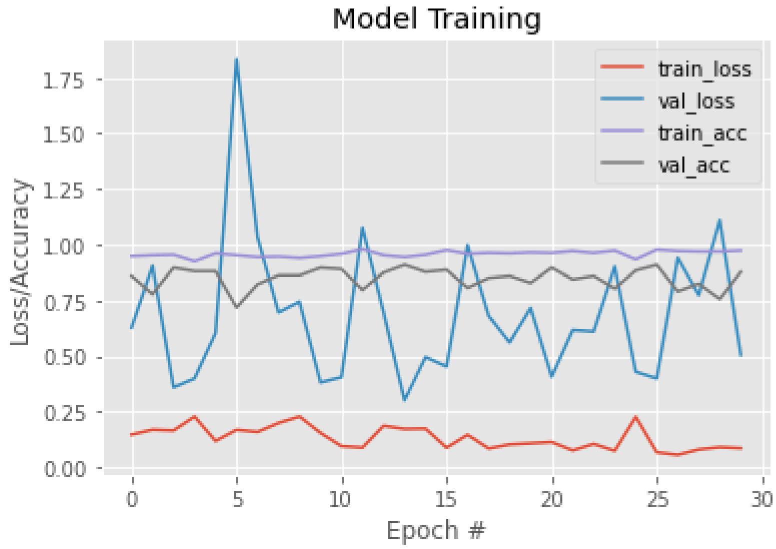
```
Epoch 15/30
30/30 [==============================] - 257s 9s/step - loss: 0.1732 - accuracy: 0.9542 - val_loss: 0.4950 -
val_accuracy: 0.8781
Epoch 16/30
30/30 [==============================] - 252s 8s/step - loss: 0.0875 - accuracy: 0.9750 - val_loss: 0.4528 -
val_accuracy: 0.8875
Epoch 17/30
30/30 [==============================] - 238s 8s/step - loss: 0.1455 - accuracy: 0.9583 - val_loss: 0.9970 -
val_accuracy: 0.8062
Epoch 18/30
30/30 [==============================] - 237s 8s/step - loss: 0.0839 - accuracy: 0.9625 - val_loss: 0.6814 -
val_accuracy: 0.8500
Epoch 19/30
30/30 [==============================] - 253s 8s/step - loss: 0.1020 - accuracy: 0.9604 - val_loss: 0.5613 -
val_accuracy: 0.8594
Epoch 20/30
30/30 [==============================] - 255s 9s/step - loss: 0.1072 - accuracy: 0.9646 - val_loss: 0.7163 -
val_accuracy: 0.8281
Epoch 21/30
30/30 [==============================] - 266s 9s/step - loss: 0.1126 - accuracy: 0.9625 - val_loss: 0.4070 -
val_accuracy: 0.8969
Epoch 22/30
30/30 [==============================] - 245s 8s/step - loss: 0.0752 - accuracy: 0.9708 - val_loss: 0.6170 -
val_accuracy: 0.8438
Epoch 23/30
30/30 [==============================] - 250s 8s/step - loss: 0.1043 - accuracy: 0.9625 - val_loss: 0.6107 -
val_accuracy: 0.8594
Epoch 24/30
30/30 [==============================] - 238s 8s/step - loss: 0.0734 - accuracy: 0.9729 - val_loss: 0.9017 -
val_accuracy: 0.8031
Epoch 25/30
30/30 [==============================] - 249s 8s/step - loss: 0.2263 - accuracy: 0.9333 - val_loss: 0.4293 -
val_accuracy: 0.8844
Epoch 26/30
30/30 [==============================] - 273s 9s/step - loss: 0.0665 - accuracy: 0.9771 - val_loss: 0.3992 -
val_accuracy: 0.9094
Epoch 27/30
30/30 [==============================] - 274s 9s/step - loss: 0.0551 - accuracy: 0.9708 - val_loss: 0.9402 -
val_accuracy: 0.7906
Epoch 28/30
30/30 [==============================] - 266s 9s/step - loss: 0.0794 - accuracy: 0.9688 - val_loss: 0.7739 -
val_accuracy: 0.8250
Epoch 29/30
```

```
30/30 [==============================] - 293s 10s/step - loss: 0.0897 - accuracy: 0.9688 - val_loss: 1.1101 -
val_accuracy: 0.7563
Epoch 30/30
30/30 [==============================] - 287s 10s/step - loss: 0.0852 - accuracy: 0.9729 - val_loss: 0.5054 -
val_accuracy: 0.8781
```

Plotting the Accuracy and Loss curves

In [14]:

```
plt.style.use("ggplot")
plt.figure()
plt.plot(hist.history["loss"], label="train_loss")
plt.plot(hist.history["val_loss"], label="val_loss")
plt.plot(hist.history["accuracy"], label="train_acc")
plt.plot(hist.history["val_accuracy"], label="val_acc")
plt.title("Model Training")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig("epochs.png")
```



Saving the Model

In [15]:

```
model.save('model.h5')
```

In [ ]:

flask files:

app.py:

```
# Importing the libraries
from flask import Flask,render_template,redirect,request,send_from_directory
from tensorflow.keras.models import load_model
import os
from PIL import Image
import numpy as np

# Loading the model before starting the app
model_file = "model.h5"
model = load_model(model_file)

# Initializing the flask app
# We start the web app and add path to "upload folder" for our uploaded images
app = Flask(__name__,template_folder='templates')
UPLOAD_FOLDER = 'static'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Creating the makePredictions() function
def makePredictions(path):

  '''
  Method to predict if the image uploaed is healthy or pneumonic
  '''


  # we open the image
  img = Image.open(path)

  # we resize the image for the model
  img_d = img.resize((224,224))
  rgbimg=None
  #We check if image is RGB or not
  if len(np.array(img_d).shape)<3:
    rgbimg = Image.new("RGB", img_d.size)
    rgbimg.paste(img_d)
  else:
      rgbimg = img_d



  # we convert the image to NumPy array and reshape it to (1,224,224,3)
  rgbimg = np.array(rgbimg,dtype=np.float64)
  rgbimg = rgbimg.reshape((1,224,224,3))


   # we make predictions from the model, convert them to our labels
(healthy/pneumonic)
  predictions = model.predict(rgbimg)
  a = int(np.argmax(predictions))
  if a==1:
    a = "Result : Pneumonic"
  else:
    a = "Result : Healthy"
  return a


# Defining a route for the HOMEPAGE
@app.route('/',methods=['GET','POST'])
def home():
    if request.method=='POST':
        if 'img' not in request.files:
            return
render_template('home.html',filename="illuminators.png",message="Please upload
an file")
        f = request.files['img']
        if f.filename=="":
            return
render_template('home.html',filename="illuminators.png",message="No file
selected")
        if not ('jpeg' in f.filename or 'png' in f.filename or 'jpg' in f.filename):
            return
render_template('home.html',filename="illuminators.png",message="Please upload
an image with .png or .jpg/.jpeg extension")
        files = os.listdir(app.config['UPLOAD_FOLDER'])
        if len(files)==1:
            f.save(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))
        else:
            files.remove("illuminators.png")
            file_ = files[0]
            os.remove(app.config['UPLOAD_FOLDER']+'/'+file_)
            f.save(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))

        # At the home method, we take this label and send this to our template
        predictions =
makePredictions(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))
        return
render_template('home.html',filename=f.filename,message=predictions,show=True
)
    return render_template('home.html',filename='illuminators.png')

if __name__=="__main__":
    app.run(debug=True)
```

train.py:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import
Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization,add
from tensorflow.keras.models  import Model,Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
import os

# Loading training and validation data at the time of training

# Getting current directory
base_dir = os.getcwd()

#defining the input shape
target_shape = (224,224)

train_dir = base_dir+"\\chest_xray\\train" #
val_dir = base_dir+"\\chest_xray\\val"     # -- Directories for data
test_dir = base_dir+"\\chest_xray\\test"   #

# loading the VGG16 model with imagenet weights without the FC layers
vgg = VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
for layer in vgg.layers:
    layer.trainable = False #making all the layers non-trainable
```

```
x = Flatten()(vgg.output) #flattening out the last layer
predictions = Dense(2,activation='softmax')(x) #Dense layer to predict wether their is pneumonia or not
model = Model(inputs=vgg.input, outputs=predictions)
model.summary()


train_gen = ImageDataGenerator(rescale=1/255.0,
                               horizontal_flip=True,
                               zoom_range=0.2,
                               shear_range=0.2) # making the data loader for training data
test_gen = ImageDataGenerator(rescale=1/255.0) # making the data loader for validation data

train_data_gen = train_gen.flow_from_directory(train_dir,
                                               target_shape,
                                               batch_size=16,
                                               class_mode='categorical') # function to make iterable object for training
test_data_gen = train_gen.flow_from_directory(test_dir,
                                              target_shape,
                                              batch_size=16,
                                              class_mode='categorical') # function to make iterable object for training

# plot_model(model, to_file='model.png')

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
hist = model.fit_generator(train_data_gen,
    steps_per_epoch=20,
        epochs=20,
        validation_data=test_data_gen,
        validation_steps=10)

plt.style.use("ggplot")
plt.figure()
plt.plot(hist.history["loss"], label="train_loss")
plt.plot(hist.history["val_loss"], label="val_loss")
plt.plot(hist.history["accuracy"], label="train_acc")
plt.plot(hist.history["val_accuracy"], label="val_acc")
plt.title("Model Training")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig("epochs.png")

model.save('model.h5')
```

FLASK FILES:

app.py:

```
# Importing the libraries
from flask import Flask,render_template,redirect,request,send_from_directory
from tensorflow.keras.models import load_model
import os
from PIL import Image
import numpy as np
# Loading the model before starting the app
model_file = "model.h5"
model = load_model(model_file)
```

```
# Initializing the flask app
# We start the web app and add path to "upload folder" for our uploaded images
app = Flask(__name__,template_folder='templates')
UPLOAD_FOLDER = 'static'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Creating the makePredictions() function
def makePredictions(path):
  '''
  Method to predict if the image uploaed is healthy or pneumonic
  '''
  # we open the image
  img = Image.open(path)

  # we resize the image for the model
  img_d = img.resize((224,224))

  rgbimg=None
  #We check if image is RGB or not
  if len(np.array(img_d).shape)<3:
    rgbimg = Image.new("RGB", img_d.size)
    rgbimg.paste(img_d)
  else:
      rgbimg = img_d
  # we convert the image to NumPy array and reshape it to (1,224,224,3)
  rgbimg = np.array(rgbimg,dtype=np.float64)
  rgbimg = rgbimg.reshape((1,224,224,3))

   #we make predictions from the model, convert them to our labels
(healthy/pneumonic)
  predictions = model.predict(rgbimg)
  a = int(np.argmax(predictions))
```

```
 if a==1:
    a = "Result : Pneumonic"
  else:
    a = "Result : Healthy"
  return a

# Defining a route for the HOMEPAGE
@app.route('/',methods=['GET','POST'])
def home():
    if request.method=='POST':
        if 'img' not in request.files:
            return render_template('home.html',filename="illuminators.png",message="Please upload an file")
        f = request.files['img']
        if f.filename==":
            return render_template('home.html',filename="illuminators.png",message="No file selected")
        if not ('jpeg' in f.filename or 'png' in f.filename or 'jpg' in f.filename):
            return render_template('home.html',filename="illuminators.png",message="Please upload an image with .png or .jpg/.jpeg extension")
        files = os.listdir(app.config['UPLOAD_FOLDER'])
        if len(files)==1:
            f.save(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))
        else:
            files.remove("illuminators.png")
            file_ = files[0]
            os.remove(app.config['UPLOAD_FOLDER']+'/'+file_)
            f.save(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))
        # At the home method, we take this label and send this to our template
        predictions =
makePredictions(os.path.join(app.config['UPLOAD_FOLDER'],f.filename))
        return
render_template('home.html',filename=f.filename,message=predictions,show=True
)
    return render_template('home.html',filename='illuminators.png')

if __name__=="__main__":
    app.run(debug=True)
```

train.py:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import
Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalizati
on,add
from tensorflow.keras.models  import Model,Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
import os
# Loading training and validation data at the time of training

# Getting current directory
base_dir = os.getcwd()

#defining the input shape
target_shape = (224,224)
```

```
train_dir = base_dir+"\\chest_xray\\train" #
val_dir = base_dir+"\\chest_xray\\val"     # -- Directories for data
test_dir = base_dir+"\\chest_xray\\test"   #

# loading the VGG16 model with imagenet weights without the FC layers
vgg = VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
for layer in vgg.layers:
    layer.trainable = False #making all the layers non-trainable

x = Flatten()(vgg.output) #flattening out the last layer
predictions = Dense(2,activation='softmax')(x) #Dense layer to predict wether their is pneumonia or not
model = Model(inputs=vgg.input, outputs=predictions)
model.summary()

train_gen = ImageDataGenerator(rescale=1/255.0,
                  horizontal_flip=True,
                  zoom_range=0.2,
                  shear_range=0.2) # making the data loader for training data
test_gen = ImageDataGenerator(rescale=1/255.0) # making the data loader for validation data

train_data_gen = train_gen.flow_from_directory(train_dir,
                           target_shape,
                           batch_size=16,
                           class_mode='categorical') # function to make iterable object for training
test_data_gen = train_gen.flow_from_directory(test_dir,
                           target_shape,
                           batch_size=16,
                           class_mode='categorical') # function to make iterable object for training
```

```
# plot_model(model, to_file='model.png')


model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accurac
y'])
hist = model.fit_generator(train_data_gen,
        steps_per_epoch=20,
        epochs=20,
        validation_data=test_data_gen,
        validation_steps=10)

plt.style.use("ggplot")
plt.figure()
plt.plot(hist.history["loss"], label="train_loss")
plt.plot(hist.history["val_loss"], label="val_loss")
plt.plot(hist.history["accuracy"], label="train_acc")
plt.plot(hist.history["val_accuracy"], label="val_acc")
plt.title("Model Training")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig("epochs.png")

model.save('model.h5')
```

HTML FILES

base.html:

```
<html lang="en">
```

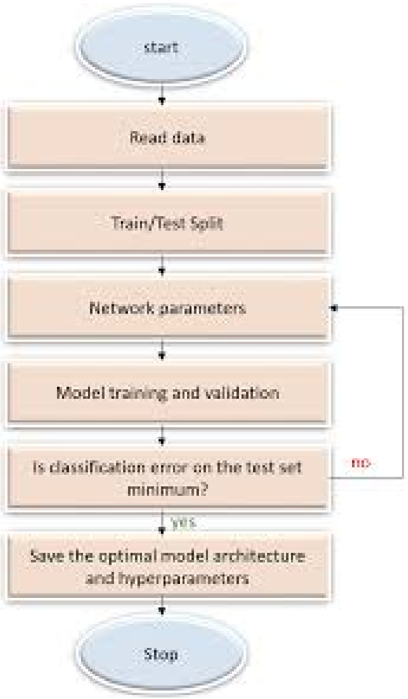
```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Pnuemonia Detection System</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script
src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
          <style>
          .bg-dark {
                    background-color: #42678c!important;
          }
          #result {
                    color: "green";
          }
          </style>

</head>

<body style="background-color:rgb(255,255,125)">
    <nav class="navbar navbar-dark bg-light">
        <div class="container">
            <a class="navbar-brand" href="#"><font size="10"
color="red">Pneumonia Detection</font></a>
        </div>
    </nav>
    <div class="container">
        <div id="content" style="margin-top:2em">
                    <div class="container">
                      <div class="row">


                                        <div>
                                                  <h4>Kindly upload <b><font size="6"><u>Chest
X-Ray Image</u></font></b></h4>
                              <form action = "http://localhost:5000/" id="upload-file"
method="post" enctype="multipart/form-data">
                                        <label for="imageUpload" class="upload-label">
                                                  Select...
                                        </label>
                                        <input type="file" name="image" id="imageUpload"
accept=".png, .jpg, .jpeg">
                              </form>



                              <div class="image-section" style="display:none;">
                                        <div class="img-preview">
                                                  <div id="imagePreview">
                                                  </div>
                                        </div>
                                        <div>
                                                  <input  type="button" class="btn btn-info btn-lg "
id="btn-predict">CHECK!
                                        </div>
                              </div>

                              <div class="loader" style="display:none;"></div>
                              <h3>
                                        <span id="result"> </span>
                              </h3>
                    </div>
                              </div>
                              <div class="col-sm-6 bd" >
                                <h3><font size="6"><u><i>Pneumonia
Detection</i></u></font></h3>
                                <br>
                                <p align="justify"><font size="4"><i>the results purely
depends on the provided x-ray reports.
                       </i></font></p>
            <img src="pneu.png" width= "522" height="174">
                              </div>
                      </div>
                    </div>
                    </div>
    </div>

</body>

<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}"
type="text/javascript"></script>
</footer>

</html>
```

4.EXPERIMENTAL INVESTIGATION

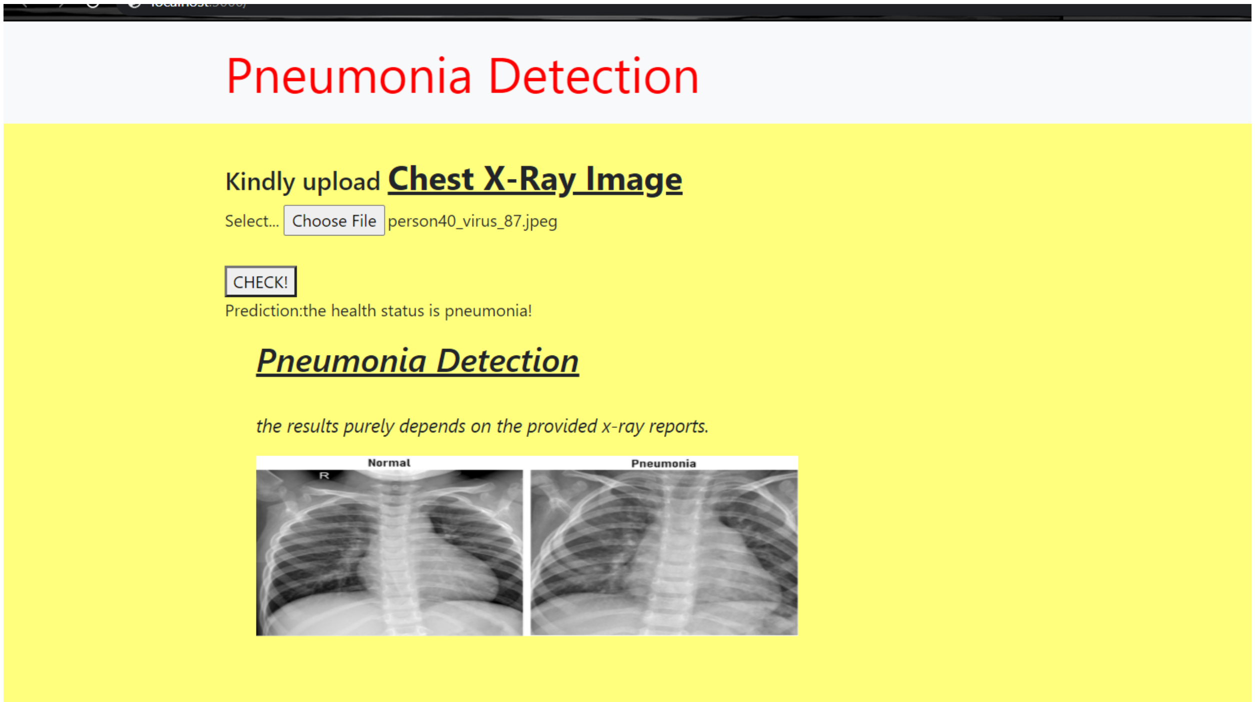
1.tensorflow 2.0 version

2.chest x-ray sample images

5.FLOWCHART:


start
Read data
Train/Test Split
Network parameters
Model training and validation
Is classification error on the test set minimum?
no
yes
Save the optimal model architecture and hyperparameters
Stop

6.Result:



Pneumonia Detection

Kindly upload **Chest X-Ray Image**

Select... Choose File person40_virus_87.jpeg

Prediction:the health status is pneumonia!

Pneumonia Detection

the results purely depends on the provided x-ray reports.

9.Conclusions:

this process is cost friendly and less time consuming

10.Future Scope:

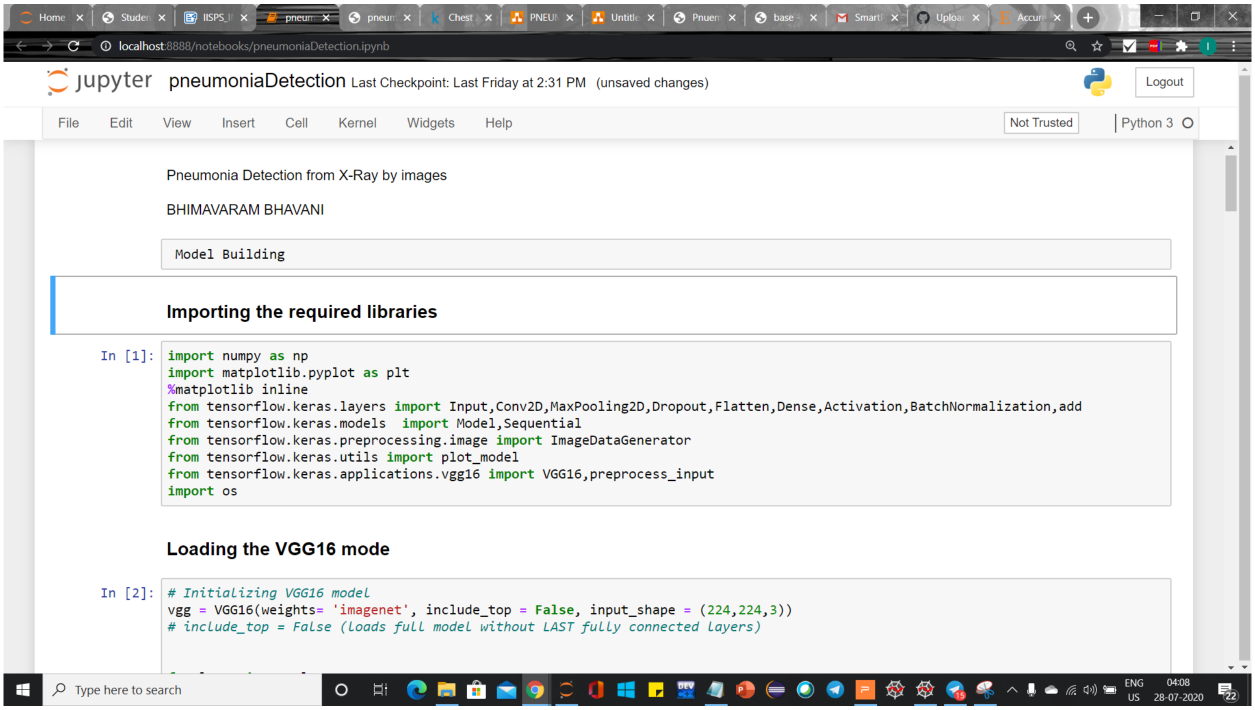
diagonsis is to be done sing charts,images (like ECG) and so on.

11.Bibilography:

1.dataset=kaggle.com

2.HTML=w3schools

12.screenshots



localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3

Pneumonia Detection from X-Ray by images

BHIMAVARAM BHAVANI

```
Model Building
```

Importing the required libraries

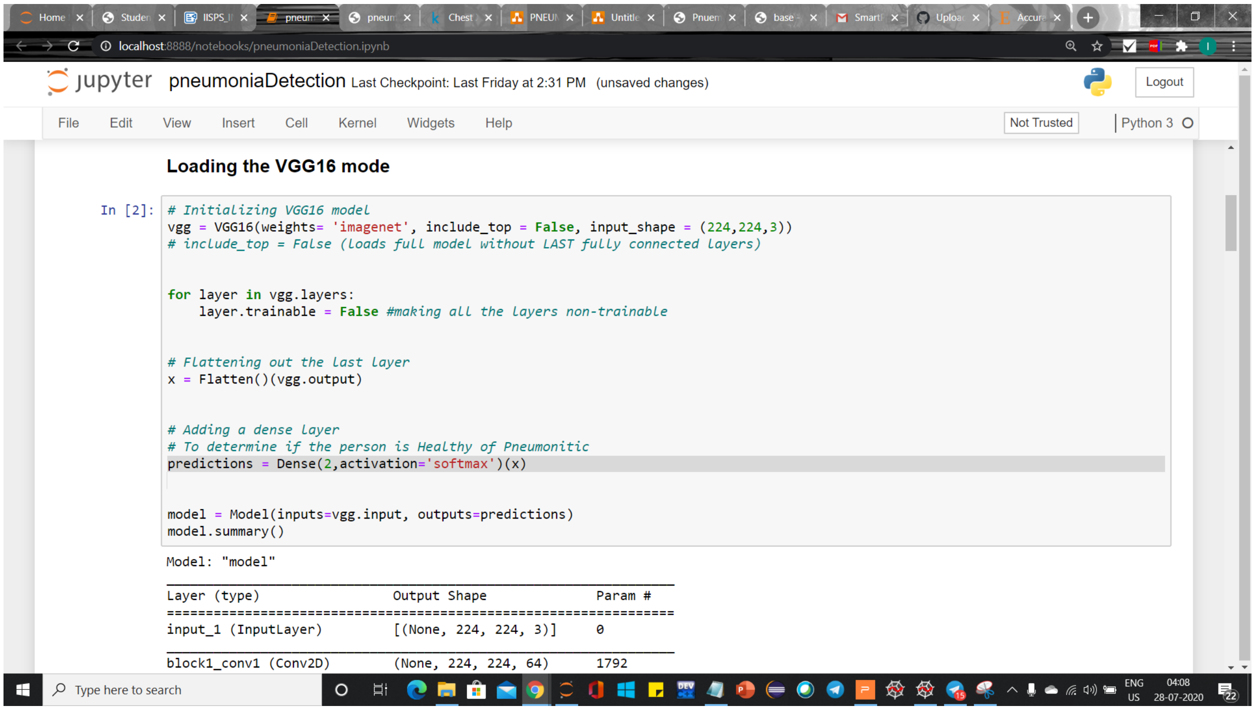
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.layers import Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization,add
from tensorflow.keras.models  import Model,Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
import os
```

Loading the VGG16 mode

In [2]:

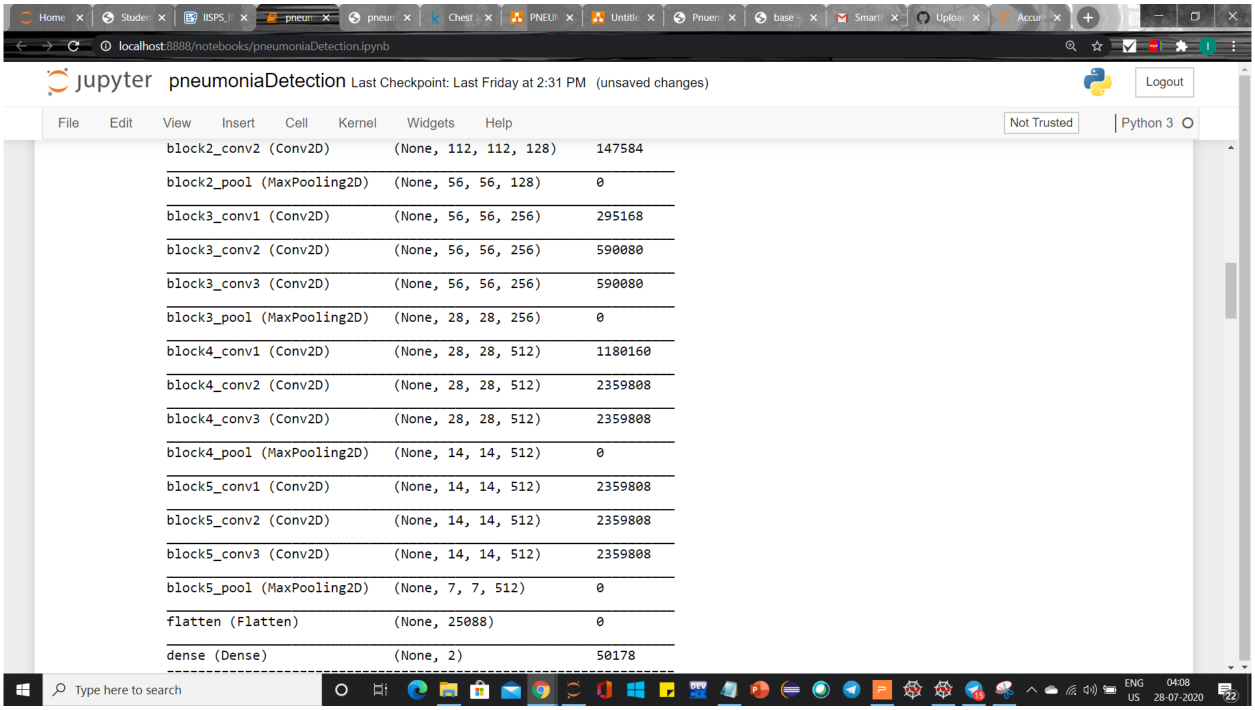
```
# Initializing VGG16 model
vgg = VGG16(weights= 'imagenet', include_top = False, input_shape = (224,224,3))
# include_top = False (loads full model without LAST fully connected layers)
```


localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Type here to search
04:08
28-07-2020

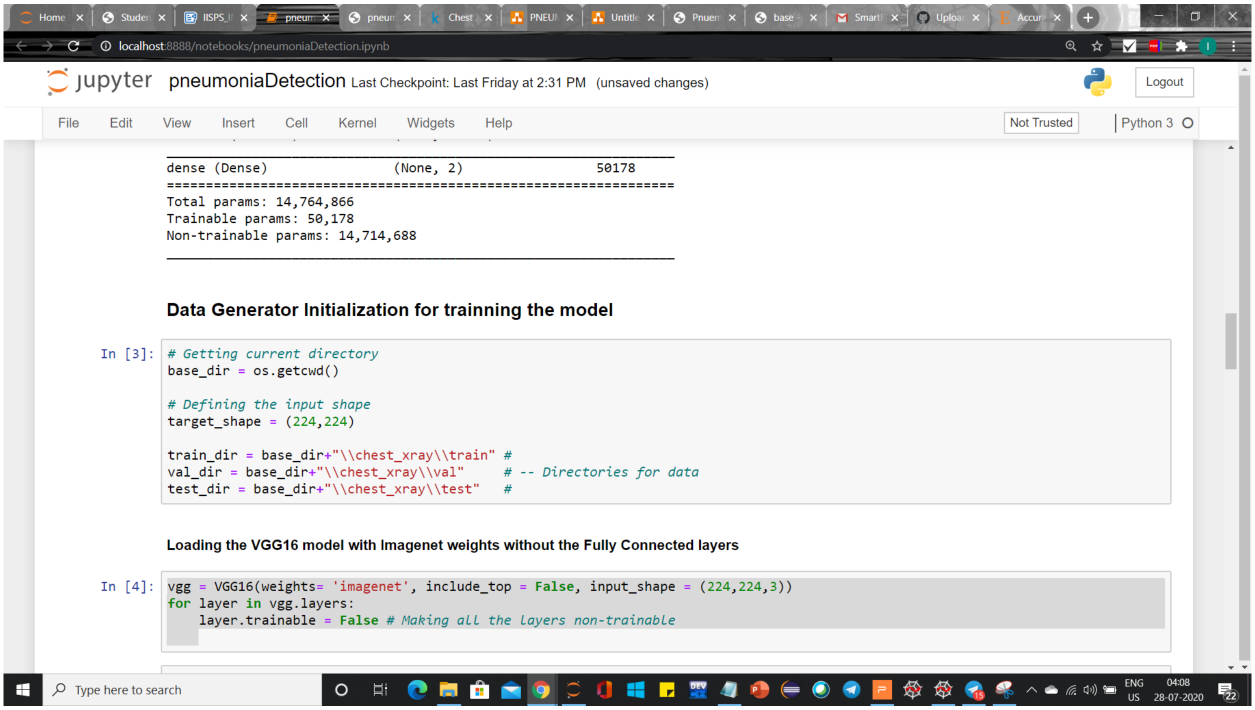
Loading the VGG16 mode

```
In [2]: # Initializing VGG16 model
        vgg = VGG16(weights= 'imagenet', include_top = False, input_shape = (224,224,3))
        # include_top = False (loads full model without LAST fully connected layers)


        for layer in vgg.layers:
            layer.trainable = False #making all the layers non-trainable


        # Flattening out the last layer
        x = Flatten()(vgg.output)


        # Adding a dense layer
        # To determine if the person is Healthy of Pneumonitic
        predictions = Dense(2,activation='softmax')(x)


        model = Model(inputs=vgg.input, outputs=predictions)
        model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
```


localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Type here to search
04:08
28-07-2020

```
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
```


localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Type here to search
ENG US 04:08 28-07-2020

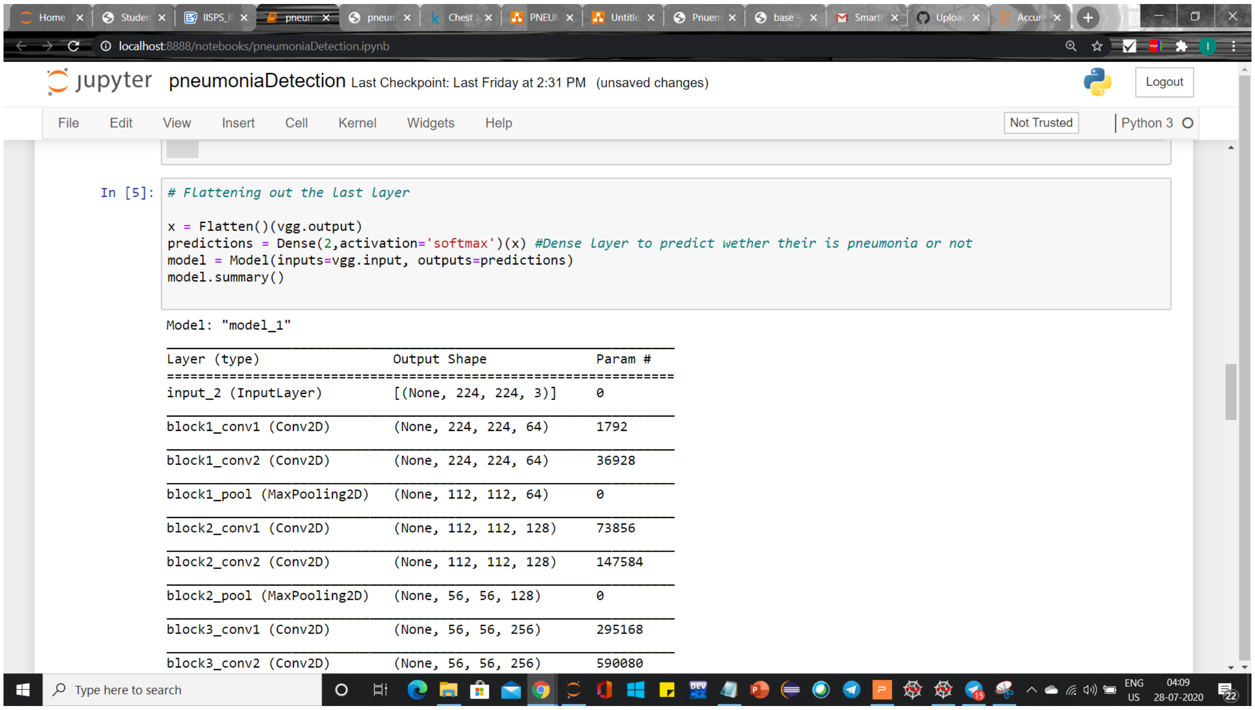
```
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

Data Generator Initialization for trainning the model

```
In [3]: # Getting current directory
        base_dir = os.getcwd()

        # Defining the input shape
        target_shape = (224,224)

        train_dir = base_dir+"\\chest_xray\\train" #
        val_dir = base_dir+"\\chest_xray\\val"     # -- Directories for data
        test_dir = base_dir+"\\chest_xray\\test"   #
```

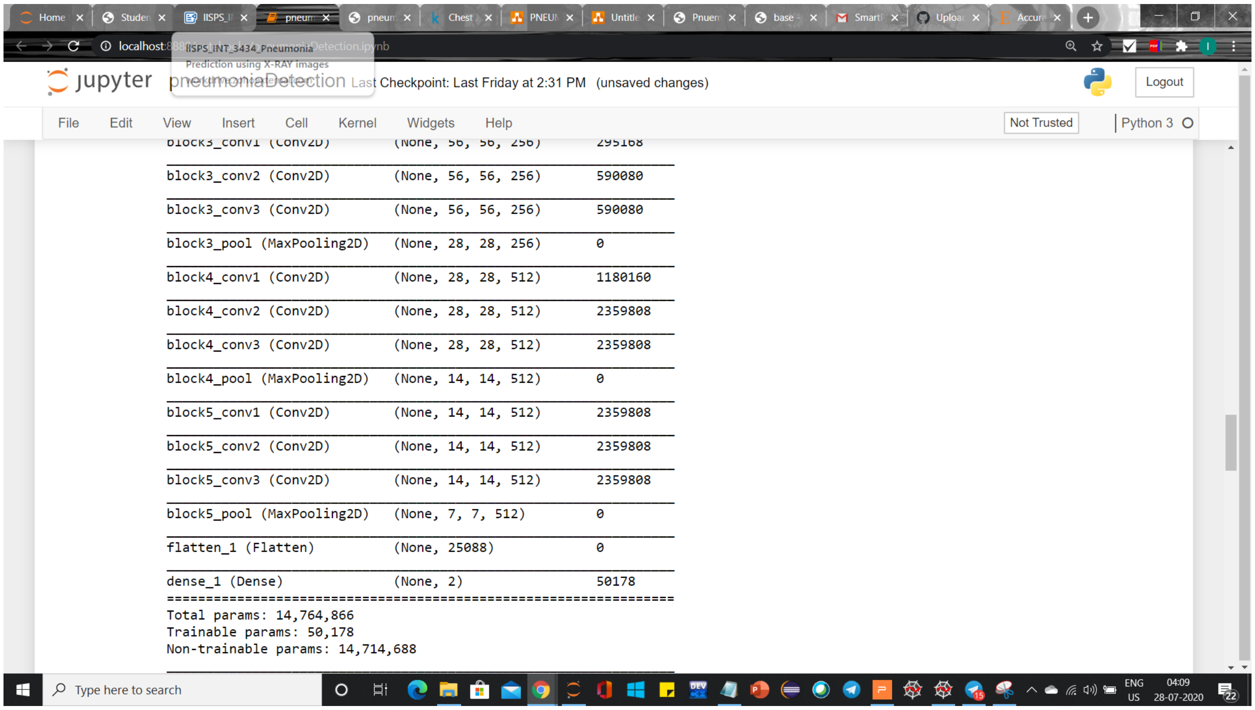
Loading the VGG16 model with Imagenet weights without the Fully Connected layers

```
In [4]: vgg = VGG16(weights= 'imagenet', include_top = False, input_shape = (224,224,3))
        for layer in vgg.layers:
            layer.trainable = False # Making all the layers non-trainable
```


localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Type here to search
ENG US 04:09 28-07-2020

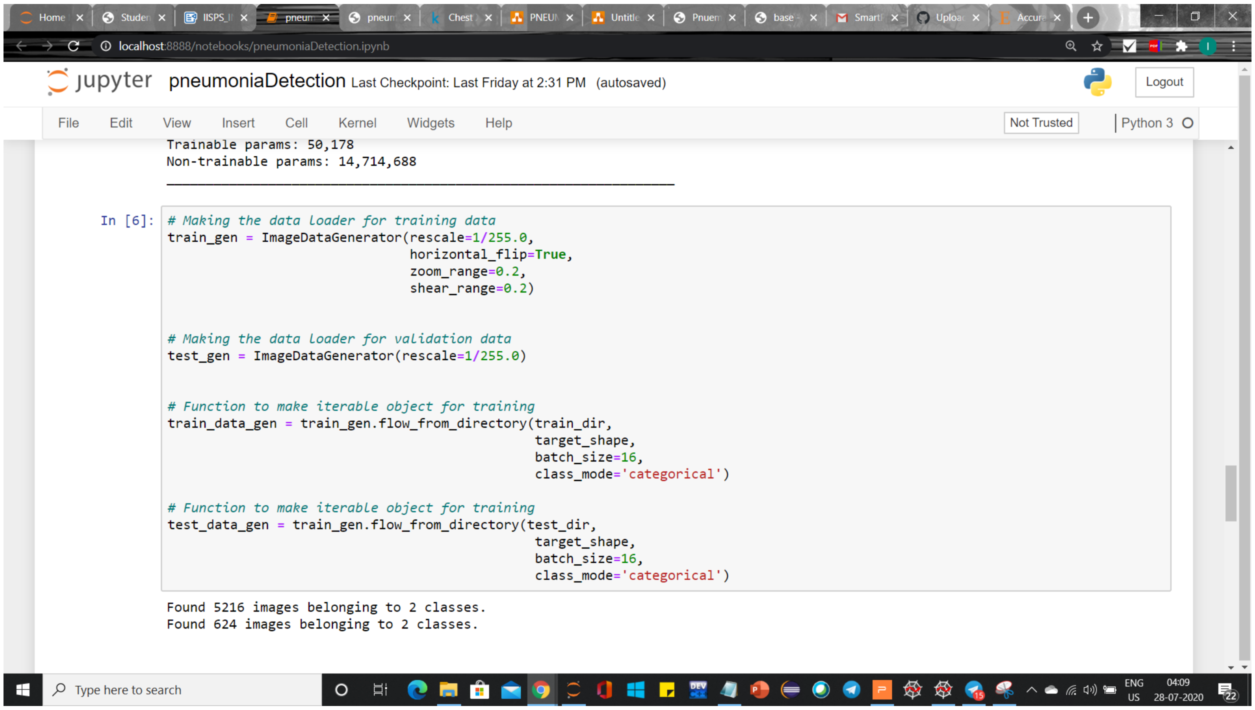
```
In [5]: # Flattening out the last layer

        x = Flatten()(vgg.output)
        predictions = Dense(2,activation='softmax')(x) #Dense Layer to predict wether their is pneumonia or not
        model = Model(inputs=vgg.input, outputs=predictions)
        model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
```



pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (unsaved changes)

```
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 50178     
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```



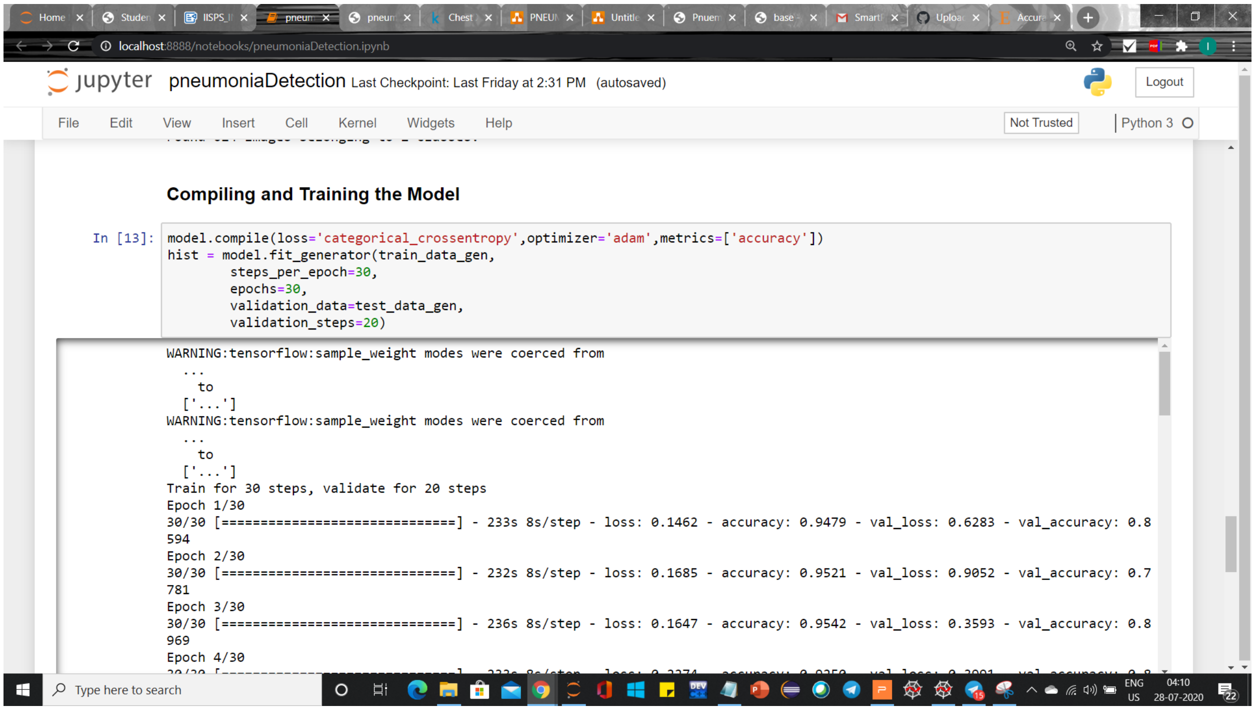
pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (autosaved)

```
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [6]:

```
# Making the data loader for training data
train_gen = ImageDataGenerator(rescale=1/255.0,
                               horizontal_flip=True,
                               zoom_range=0.2,
                               shear_range=0.2)


# Making the data loader for validation data
test_gen = ImageDataGenerator(rescale=1/255.0)


# Function to make iterable object for training
train_data_gen = train_gen.flow_from_directory(train_dir,
                                               target_shape,
                                               batch_size=16,
                                               class_mode='categorical')

# Function to make iterable object for training
test_data_gen = train_gen.flow_from_directory(test_dir,
                                               target_shape,
                                               batch_size=16,
                                               class_mode='categorical')
```

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

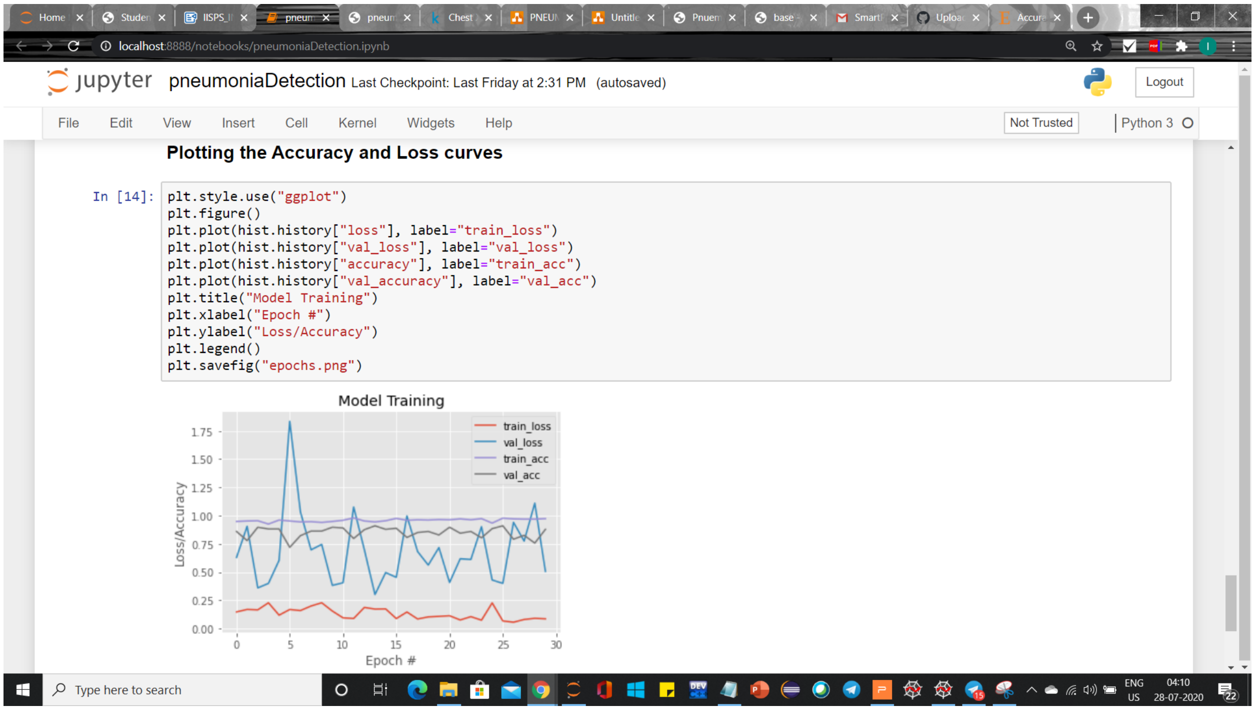

localhost:8888/notebooks/pneumoniaDetection.ipynb
jupyter pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Type here to search
ENG US 04:10 28-07-2020

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

Compiling and Training the Model

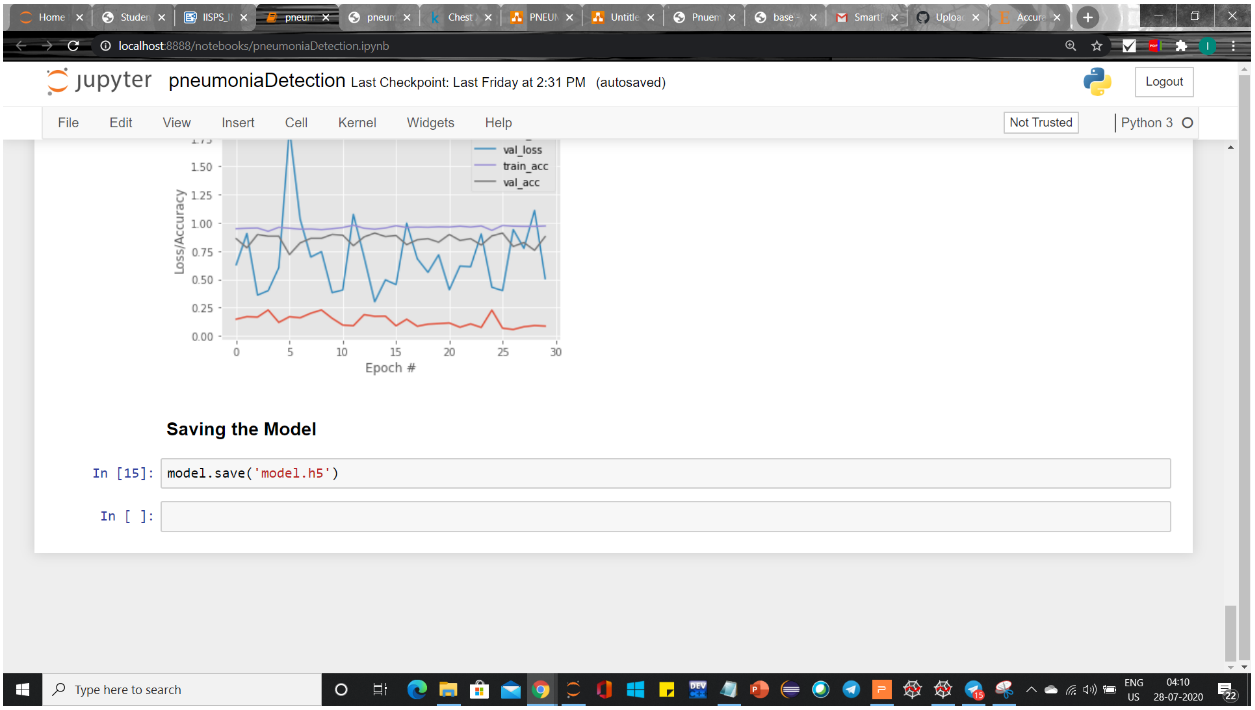
```
In [13]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
         hist = model.fit_generator(train_data_gen,
                 steps_per_epoch=30,
                 epochs=30,
                 validation_data=test_data_gen,
                 validation_steps=20)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 30 steps, validate for 20 steps
Epoch 1/30
30/30 [==============================] - 233s 8s/step - loss: 0.1462 - accuracy: 0.9479 - val_loss: 0.6283 - val_accuracy: 0.8
594
Epoch 2/30
30/30 [==============================] - 232s 8s/step - loss: 0.1685 - accuracy: 0.9521 - val_loss: 0.9052 - val_accuracy: 0.7
781
Epoch 3/30
30/30 [==============================] - 236s 8s/step - loss: 0.1647 - accuracy: 0.9542 - val_loss: 0.3593 - val_accuracy: 0.8
969
Epoch 4/30
30/30 [==============================] - 233s 8s/step - loss: 0.2274 - accuracy: 0.9250 - val_loss: 0.3901 - val_accuracy: 0.8
```



Plotting the Accuracy and Loss curves

```
In [14]: plt.style.use("ggplot")
         plt.figure()
         plt.plot(hist.history["loss"], label="train_loss")
         plt.plot(hist.history["val_loss"], label="val_loss")
         plt.plot(hist.history["accuracy"], label="train_acc")
         plt.plot(hist.history["val_accuracy"], label="val_acc")
         plt.title("Model Training")
         plt.xlabel("Epoch #")
         plt.ylabel("Loss/Accuracy")
         plt.legend()
         plt.savefig("epochs.png")
```

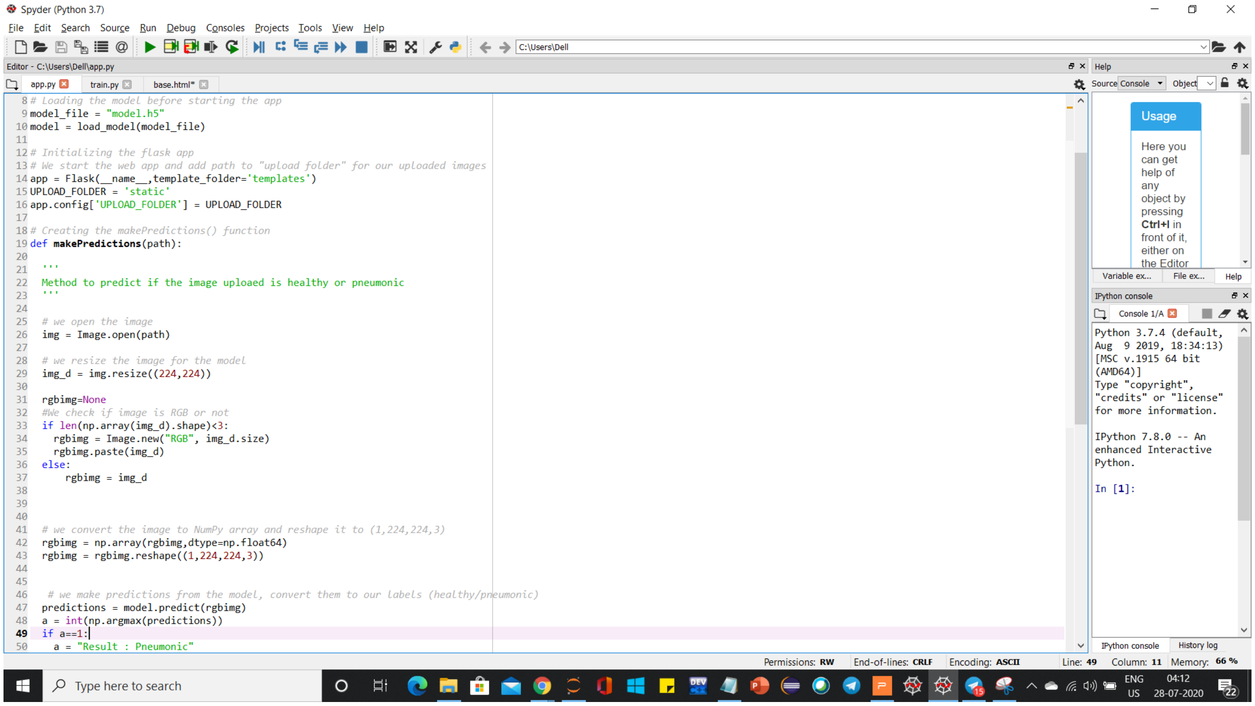


localhost:8888/notebooks/pneumoniaDetection.ipynb

pneumoniaDetection Last Checkpoint: Last Friday at 2:31 PM (autosaved)

Saving the Model

```
In [15]: model.save('model.h5')

In [ ]:
```

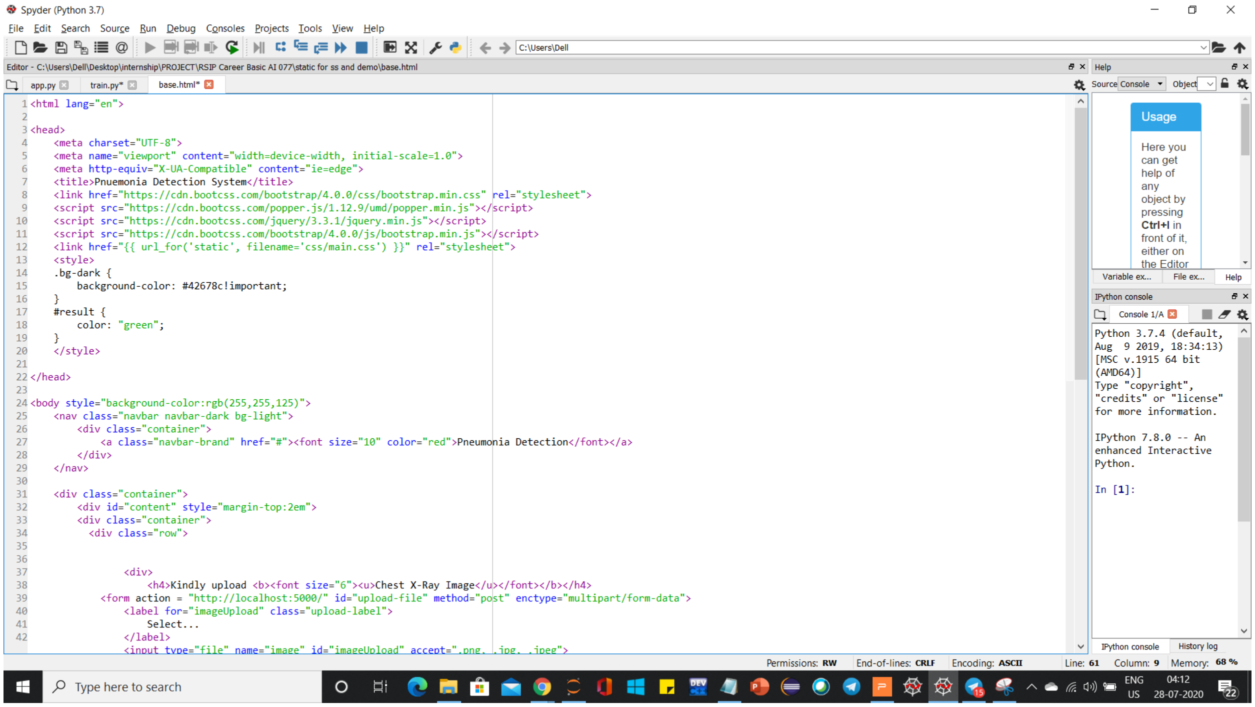


Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor

```
# Loading the model before starting the app
model_file = "model.h5"
model = load_model(model_file)

# Initializing the flask app
# We start the web app and add path to "upload folder" for our uploaded images
app = Flask(__name__,template_folder='templates')
UPLOAD_FOLDER = 'static'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Creating the makePredictions() function
def makePredictions(path):

  '''
  Method to predict if the image uploaed is healthy or pneumonic
  '''

  # we open the image
  img = Image.open(path)

  # we resize the image for the model
  img_d = img.resize((224,224))

  rgbimg=None
  #We check if image is RGB or not
  if len(np.array(img_d).shape)<3:
    rgbimg = Image.new("RGB", img_d.size)
    rgbimg.paste(img_d)
  else:
      rgbimg = img_d



  # we convert the image to NumPy array and reshape it to (1,224,224,3)
  rgbimg = np.array(rgbimg,dtype=np.float64)
  rgbimg = rgbimg.reshape((1,224,224,3))


   # we make predictions from the model, convert them to our labels (healthy/pneumonic)
  predictions = model.predict(rgbimg)
  a = int(np.argmax(predictions))
  if a==1:
    a = "Result : Pneumonic"
```

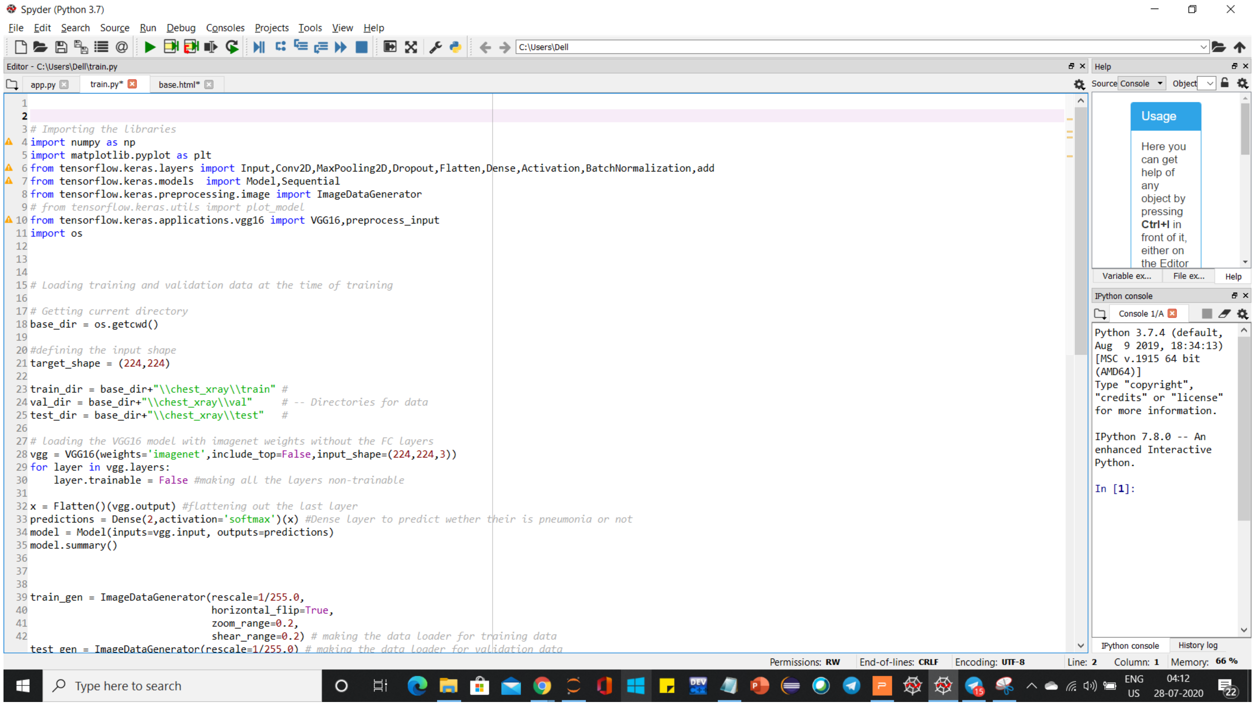
```
Python 3.7.4 (default,
Aug  9 2019, 18:34:13)
[MSC v.1915 64 bit
(AMD64)]
Type "copyright",
"credits" or "license"
for more information.

IPython 7.8.0 -- An
enhanced Interactive
Python.

In [1]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 49 Column: 11 Memory: 66 %


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Dell
Editor - C:\Users\Dell\Desktop\internship\PROJECT\RSIP Career Basic AI 077\static for ss and demo\base.html
app.py
train.py*
base.html*
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor
Variable ex... File ex... Help
IPython console
Console 1/A
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 61 Column: 9 Memory: 68 %
Type here to search
ENG US 04:12 28-07-2020

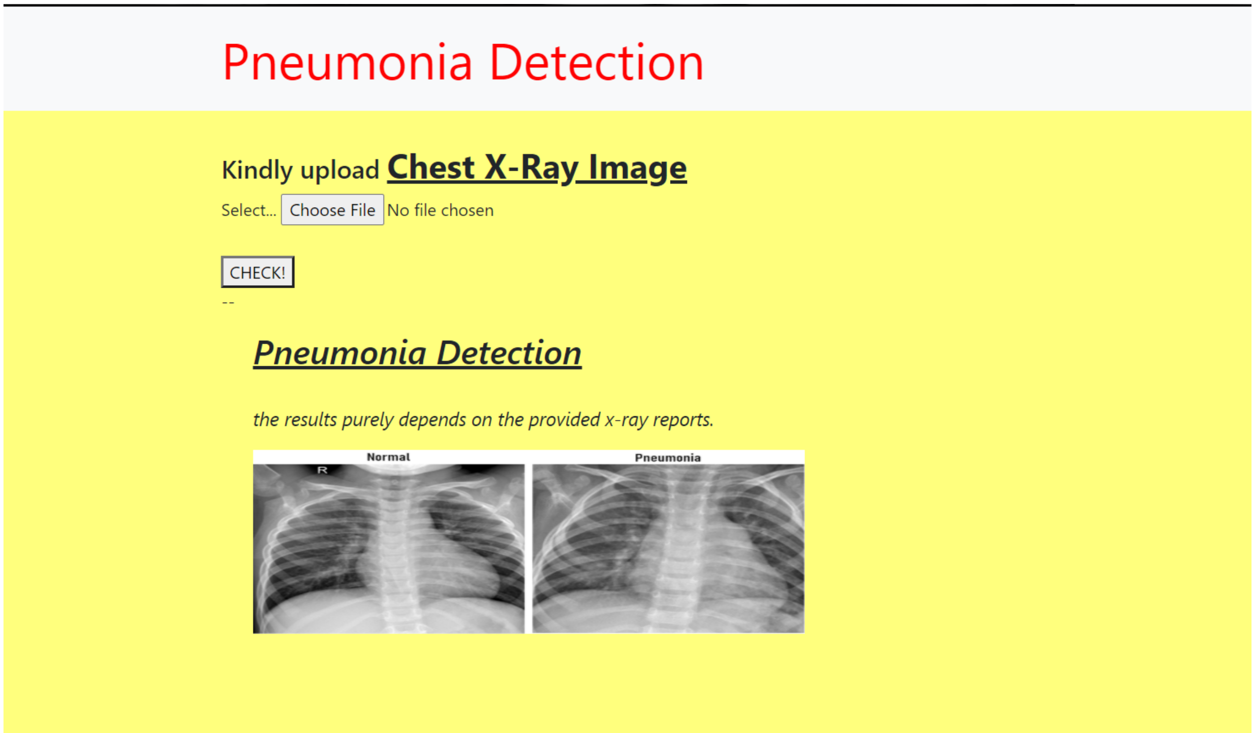
```html
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Pnuemonia Detection System</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
    <style>
    .bg-dark {
        background-color: #42678c!important;
    }
    #result {
        color: "green";
    }
    </style>

</head>

<body style="background-color:rgb(255,255,125)">
    <nav class="navbar navbar-dark bg-light">
        <div class="container">
            <a class="navbar-brand" href="#"><font size="10" color="red">Pneumonia Detection</font></a>
        </div>
    </nav>

    <div class="container">
        <div id="content" style="margin-top:2em">
        <div class="container">
          <div class="row">


                <div>
                    <h4>Kindly upload <b><font size="6"><u>Chest X-Ray Image</u></font></b></h4>
            <form action = "http://localhost:5000/" id="upload-file" method="post" enctype="multipart/form-data">
                <label for="imageUpload" class="upload-label">
                    Select...
                </label>
                <input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg">
```

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]:
```


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Dell
Editor - C:\Users\Dell\train.py
app.py
train.py*
base.html*
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor
Variable ex... File ex... Help
IPython console
Console 1/A
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 2 Column: 1 Memory: 66 %
Type here to search
ENG US 04:12 28-07-2020

```python


# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization,add
from tensorflow.keras.models  import Model,Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
import os



# Loading training and validation data at the time of training

# Getting current directory
base_dir = os.getcwd()

#defining the input shape
target_shape = (224,224)

train_dir = base_dir+"\\chest_xray\\train" #
val_dir = base_dir+"\\chest_xray\\val"     # -- Directories for data
test_dir = base_dir+"\\chest_xray\\test"   #

# loading the VGG16 model with imagenet weights without the FC layers
vgg = VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
for layer in vgg.layers:
    layer.trainable = False #making all the layers non-trainable

x = Flatten()(vgg.output) #flattening out the last layer
predictions = Dense(2,activation='softmax')(x) #Dense layer to predict wether their is pneumonia or not
model = Model(inputs=vgg.input, outputs=predictions)
model.summary()



train_gen = ImageDataGenerator(rescale=1/255.0,
                               horizontal_flip=True,
                               zoom_range=0.2,
                               shear_range=0.2) # making the data loader for training data
test_gen = ImageDataGenerator(rescale=1/255.0) # making the data loader for validation data
```

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]:
```



Pneumonia Detection

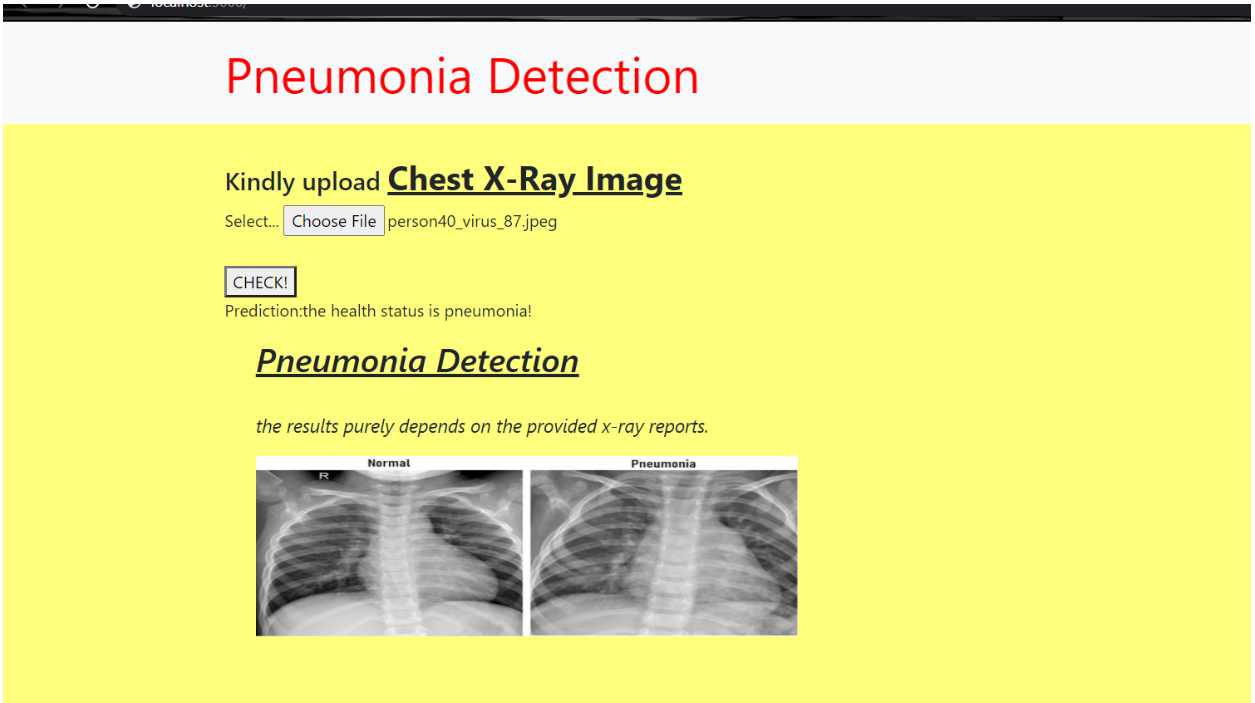
Kindly upload **Chest X-Ray Image**

Select... No file chosen

--

Pneumonia Detection

the results purely depends on the provided x-ray reports.



Pneumonia Detection

Kindly upload **Chest X-Ray Image**

Select... person40_virus_87.jpeg

Prediction:the health status is pneumonia!

Pneumonia Detection

the results purely depends on the provided x-ray reports.