In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/dock
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list a

import os
for dirname, _, filenames in os.walk('chest-xray-covid/chest_xray'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
# You can also write temporary files to /kaggle/temp/, but they won't be saved ou
```

```
chest-xray-covid/chest_xray\chest_xray\.DS_Store
chest-xray-covid/chest_xray\chest_xray\test\.DS_Store
chest-xray-covid/chest_xray\chest_xray\test\COVID\cavitating-pneumonia-4-day0
-PA.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\cavitating-pneumonia-4-day2
8-L.png
chest-xray-covid/chest_xray\chest_xray\test\COVID\cavitating-pneumonia-4-day2
8-PA.png
chest-xray-covid/chest_xray\chest_xray\test\COVID\cb60786c.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\cb706009.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\CD50BA96-6982-4C80-AE7B-5F6
7ACDBFA56.jpeg
chest-xray-covid/chest_xray\chest_xray\test\COVID\cdf5605e45874c28262c81b7ab8
0b3_jumbo.jpeg
chest-xray-covid/chest_xray\chest_xray\test\COVID\ce09cfab.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\ce68550a.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\cfcdf8d9.jpg
chest-xray-covid/chest_xray\chest_xray\test\COVID\chlamydia-pneumonia-L.png
chest-xray-covid/chest_xray\chest_xray\test\COVID\chlamydia-pneumonia-PA.png
```

In [2]:
```
import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report , confusion_matrix

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

import keras
from keras.models import Sequential
from keras.layers import Conv2D , BatchNormalization , MaxPool2D ,Dense , Flatter
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau

import cv2
import os
```

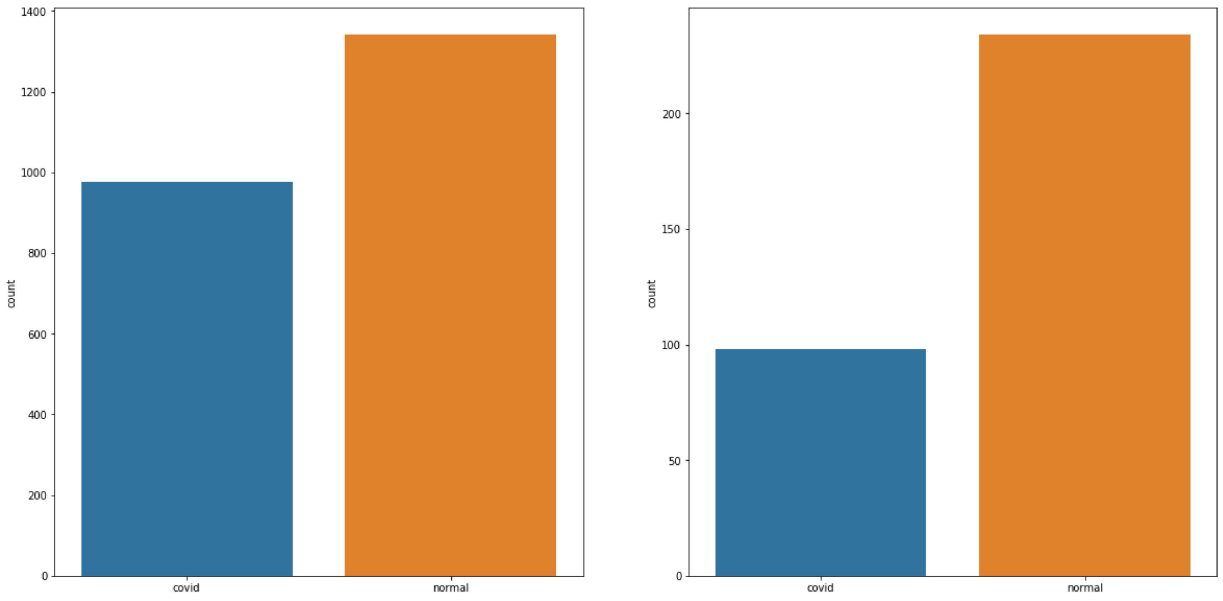
```
Using TensorFlow backend.
```

In [3]:
```
output_labels=['COVID','NORMAL']
img_size=200
def get_data(path_given):
    data=[]
    for labels in output_labels:
        path=os.path.join(path_given,labels)
        ind=output_labels.index(labels)
        for images in os.listdir(path):
            try:
                img=cv2.imread(os.path.join(path,images),cv2.IMREAD_GRAYSCALE)
                resizing=cv2.resize(img,(img_size,img_size))

                data.append([resizing,ind])
            except Exception as e:
                print(e)

    return(np.array(data))
```

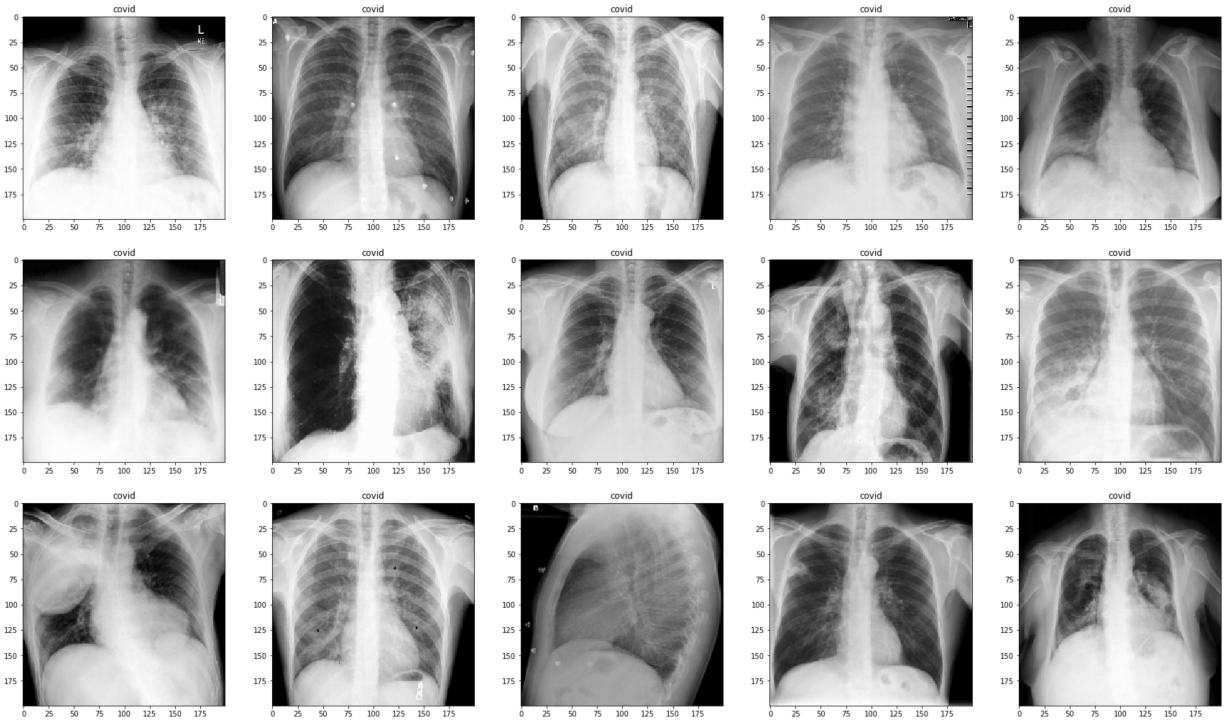
In [4]:
```
#getting data
training_data=get_data("chest-xray-covid/chest_xray/train")
test_data=get_data("chest-xray-covid/chest_xray/test")
valid_data=get_data("chest-xray-covid/chest_xray/val")
```

In [5]:
```
l=[]
k=[]
plt.figure(figsize=(20,10))
for i in training_data:
    if(i[1]==0):
        l.append("covid")
    else:
        l.append("normal")
#print(l)
plt.subplot(1,2,1)
sns.countplot(l)

for j in test_data:
    if(j[1]==0):
        k.append('covid')
    else:
        k.append('normal')
plt.subplot(1,2,2)
sns.countplot(k)
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x2412bc0d748>



In [6]:
```
plt.figure(figsize=(30,30))
for i in range(15):
    plt.subplot(5,5,i+1)
    plt.imshow(training_data[i][0],cmap='gray')
    if(training_data[i][1]==0):

        plt.title('covid')
    else:

        plt.title('Normal')
plt.show()
```



In [7]:
```
x_train,y_train=[],[]

x_test , y_test =[],[]

x_val , y_val =[],[]

for i in training_data:
    x_train.append(i[0])
    y_train.append(i[1])

for j in test_data:
    x_test.append(j[0])
    y_test.append(j[1])

for k in valid_data:
    x_val.append(k[0])
    y_val.append(k[1])


x_train=np.array(x_train)/255
y_train=np.array(y_train)

x_test=np.array(x_test)/255
y_test=np.array(y_test)

x_val=np.array(x_val)/255
y_val=np.array(y_val)


x_train=x_train.reshape(-1,200,200,1)
x_test=x_test.reshape(-1,200,200,1)
x_val=x_val.reshape(-1,200,200,1)
```

In [8]:
```
augmentation= ImageDataGenerator(featurewise_center=False,
                                 # set input mean to 0 over the dataset
        samplewise_center=False,
                                 # set each sample mean to 0
        featurewise_std_normalization=False,
                                 # divide inputs by std of the dataset
        samplewise_std_normalization=False,
                                 # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=30,
        zoom_range=0.2,
        horizontal_flip=True,
        vertical_flip=True,
        width_shift_range=0.1,
        height_shift_range=0.1)
augmentation.fit(x_train)
```

In [9]:

```
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'rel
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'rel
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Flatten())

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

model.add(Dense(units = 1 , activation = 'sigmoid'))

model.compile(optimizer = "Adam" , loss = 'binary_crossentropy' , metrics = ['acc

model.summary()
```

```
WARNING:tensorflow:From C:\Users\admin\Anaconda3\lib\site-packages\keras\backen
d\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use
tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\admin\Anaconda3\lib\site-packages\tensorflow\p
ython\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from tensorflo
w.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 32)      320
_________________________________________________________________
```

```
batch_normalization_1 (Batch (None, 200, 200, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 100, 100, 64)      18496
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 100, 64)      0
_________________________________________________________________
batch_normalization_2 (Batch (None, 100, 100, 64)      256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 50, 50, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 50, 50, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 50, 50, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 25, 25, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 25, 25, 128)       73856
_________________________________________________________________
dropout_2 (Dropout)          (None, 25, 25, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 25, 25, 128)       512
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 13, 13, 256)       295168
_________________________________________________________________
dropout_3 (Dropout)          (None, 13, 13, 256)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 13, 13, 256)       1024
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1605760
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 2,032,833
Trainable params: 2,031,745
Non-trainable params: 1,088
_________________________________________________________________
```

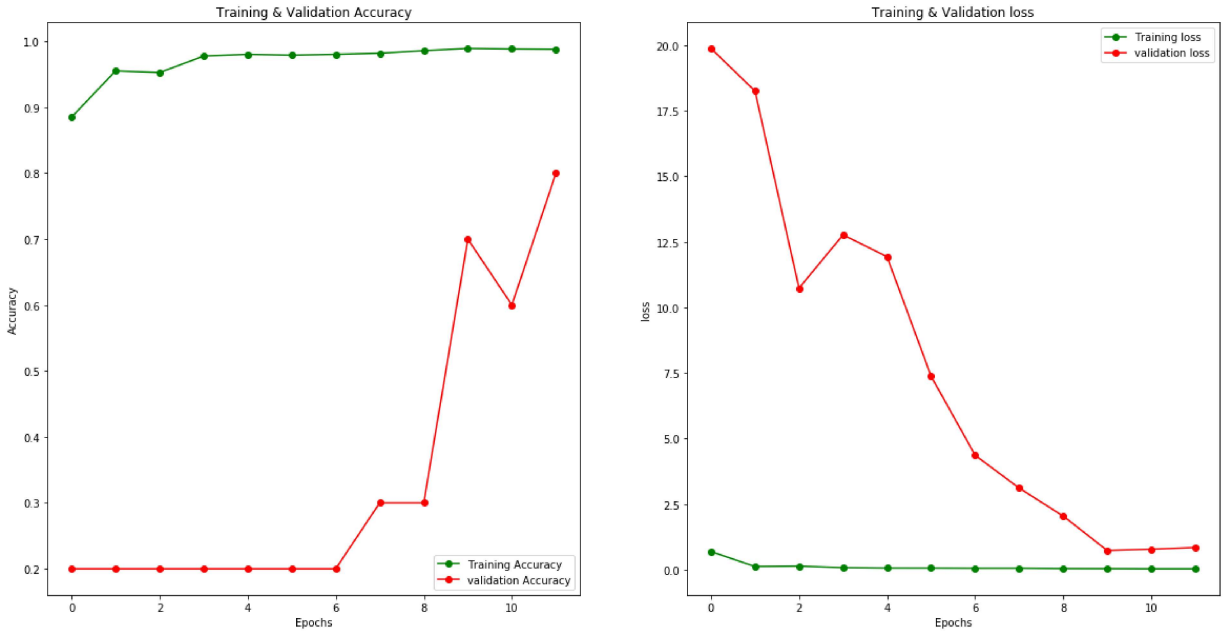
In [10]:
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 2,

history=model.fit(augmentation.flow(x_train,y_train,batch_size=32),epochs=12 ,
                  validation_data=augmentation.flow(x_val,y_val),callbacks=[learn
```

```
WARNING:tensorflow:From C:\Users\admin\Anaconda3\lib\site-packages\keras\backen
d\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please
use tf.compat.v1.global_variables instead.

Epoch 1/12
73/73 [==============================] - 215s 3s/step - loss: 0.6863 - accurac
y: 0.8848 - val_loss: 19.8763 - val_accuracy: 0.2000
Epoch 2/12
73/73 [==============================] - 212s 3s/step - loss: 0.1181 - accurac
y: 0.9551 - val_loss: 18.2516 - val_accuracy: 0.2000
Epoch 3/12
73/73 [==============================] - 213s 3s/step - loss: 0.1323 - accurac
y: 0.9525 - val_loss: 10.7320 - val_accuracy: 0.2000

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 4/12
73/73 [==============================] - 217s 3s/step - loss: 0.0748 - accurac
y: 0.9776 - val_loss: 12.7662 - val_accuracy: 0.2000
Epoch 5/12
73/73 [==============================] - 212s 3s/step - loss: 0.0582 - accurac
y: 0.9801 - val_loss: 11.9317 - val_accuracy: 0.2000

Epoch 00005: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 6/12
73/73 [==============================] - 212s 3s/step - loss: 0.0586 - accurac
y: 0.9789 - val_loss: 7.3736 - val_accuracy: 0.2000
Epoch 7/12
73/73 [==============================] - 212s 3s/step - loss: 0.0549 - accurac
y: 0.9801 - val_loss: 4.3600 - val_accuracy: 0.2000

Epoch 00007: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
Epoch 8/12
73/73 [==============================] - 222s 3s/step - loss: 0.0572 - accurac
y: 0.9819 - val_loss: 3.1090 - val_accuracy: 0.3000
Epoch 9/12
73/73 [==============================] - 225s 3s/step - loss: 0.0425 - accurac
y: 0.9858 - val_loss: 2.0424 - val_accuracy: 0.3000
Epoch 10/12
73/73 [==============================] - 213s 3s/step - loss: 0.0393 - accurac
y: 0.9892 - val_loss: 0.7291 - val_accuracy: 0.7000
Epoch 11/12
73/73 [==============================] - 213s 3s/step - loss: 0.0340 - accurac
y: 0.9883 - val_loss: 0.7759 - val_accuracy: 0.6000
Epoch 12/12
73/73 [==============================] - 218s 3s/step - loss: 0.0332 - accurac
y: 0.9879 - val_loss: 0.8426 - val_accuracy: 0.8000
```

In [11]:
```
print("Loss of the model is:",model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is:",model.evaluate(x_test,y_test)[1]*100, "%")
```

```
332/332 [==============================] - 14s 42ms/step
Loss of the model is: 0.20984923357076674
332/332 [==============================] - 13s 40ms/step
Accuracy of the model is: 92.46987700462341 %
```

In [12]:
```
epochs=[i for i in range(12)]
fig,ax=plt.subplots(1,2)
train_acc=history.history['accuracy']
train_loss=history.history['loss']
val_acc=history.history['val_accuracy']
val_loss=history.history['val_loss']

fig.set_size_inches(20,10)
ax[0].plot(epochs,train_acc,'go-',label='Training Accuracy')
ax[0].plot(epochs,val_acc,'ro-',label='validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")


ax[1].plot(epochs,train_loss,'go-',label='Training loss')
ax[1].plot(epochs,val_loss,'ro-',label='validation loss')
ax[1].set_title('Training & Validation loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("loss")
plt.show()
```

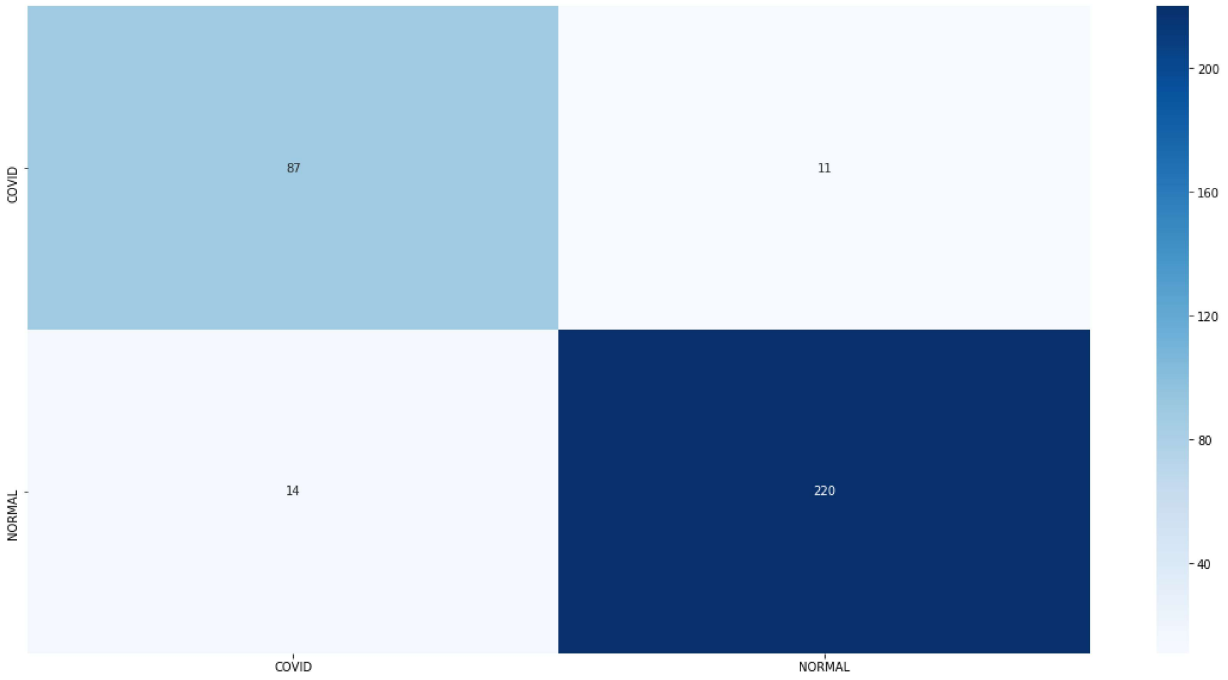


In [13]:
```
prediction=model.predict_classes(x_test)
prediction=prediction.reshape(1,-1)[0]

V=classification_report(y_test,prediction,target_names=['covid 0','Normal 1'])
print(V)
print(prediction)
```

```
              precision    recall  f1-score   support

    covid 0       0.86      0.89      0.87        98
   Normal 1       0.95      0.94      0.95       234

avg / total       0.93      0.92      0.93       332

[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 0 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1
 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [14]:
```
cm = confusion_matrix(y_test,prediction)
print(cm)
cm=pd.DataFrame(cm,index=['0','1'],columns=['0','1'])
plt.figure(figsize=(20,10))
sns.heatmap(cm,cmap='Blues',annot=True,fmt='',xticklabels=output_labels,yticklabe
plt.show()
```

```
[[ 87  11]
 [ 14 220]]
```



In [15]:
```
img_size=200
def get_data_new(path_given):
    data1=[]

    #path=os.path.join(path_given,labels)
    #ind=output_labels.index(labels)
    for images in os.listdir(path_given):
        try:
            img=cv2.imread(os.path.join(path_given,images),cv2.IMREAD_GRAYSCALE)
            resizing=cv2.resize(img,(img_size,img_size))

            data1.append(resizing)
        except Exception as e:
            print(e)

    return(np.array(data1))


try_test=get_data_new("try1")
try_test=np.array(try_test)/255
try_test=try_test.reshape(-1,200,200,1)
prediction=model.predict_classes(try_test)
prediction=prediction.reshape(1,-1)[0]
print(prediction)
```

```
[0]
```

In [16]:
```
model.save('covid_model.h5')
```

In [17]:
```
from keras.models import load_model
model_tuberculosis = load_model('covid_model.h5')
```

In [18]:
```
img_size=200
def get_data_new(path_given):
    data1=[]

    #path=os.path.join(path_given,labels)
    #ind=output_labels.index(labels)
    for images in os.listdir(path_given):
        try:
            img=cv2.imread(os.path.join(path_given,images),cv2.IMREAD_GRAYSCALE)
            resizing=cv2.resize(img,(img_size,img_size))

            data1.append(resizing)
        except Exception as e:
            print(e)

    return(np.array(data1))


try_test=get_data_new("try1")
try_test=np.array(try_test)/255
try_test=try_test.reshape(-1,200,200,1)
prediction=model_tuberculosis.predict_classes(try_test)
prediction=prediction.reshape(1,-1)[0]
print(prediction)
```

```
[0]
```

In [ ]:

In [ ]: