Name: Mark Abrenio

Data Set: NIH XRAYs (https://www.kaggle.com/nih-chest-xrays/data)

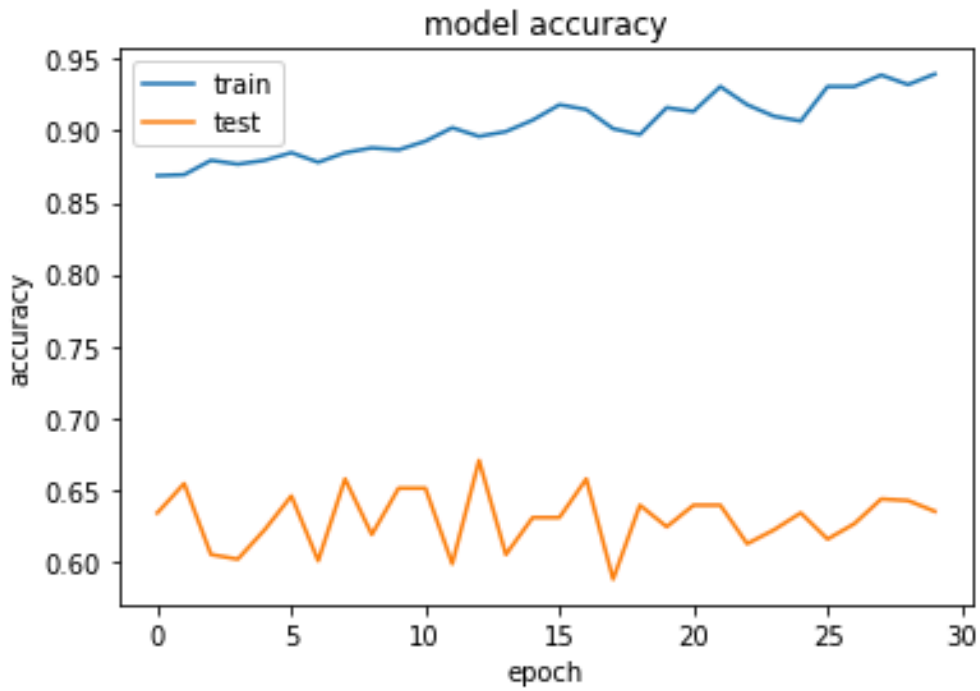
In an attempt to train an algorithm to recognize Pneumonia in images, I went with a VGG16 Neural Network model. I mostly picked this model because its big and robust, and I have used it before for image recognition and it learned images very well. I broke out the data in the CSV, to where every patient had a different column for each disease. And the patient was assigned a 1 if they had that disease, or a 0 if they did not.

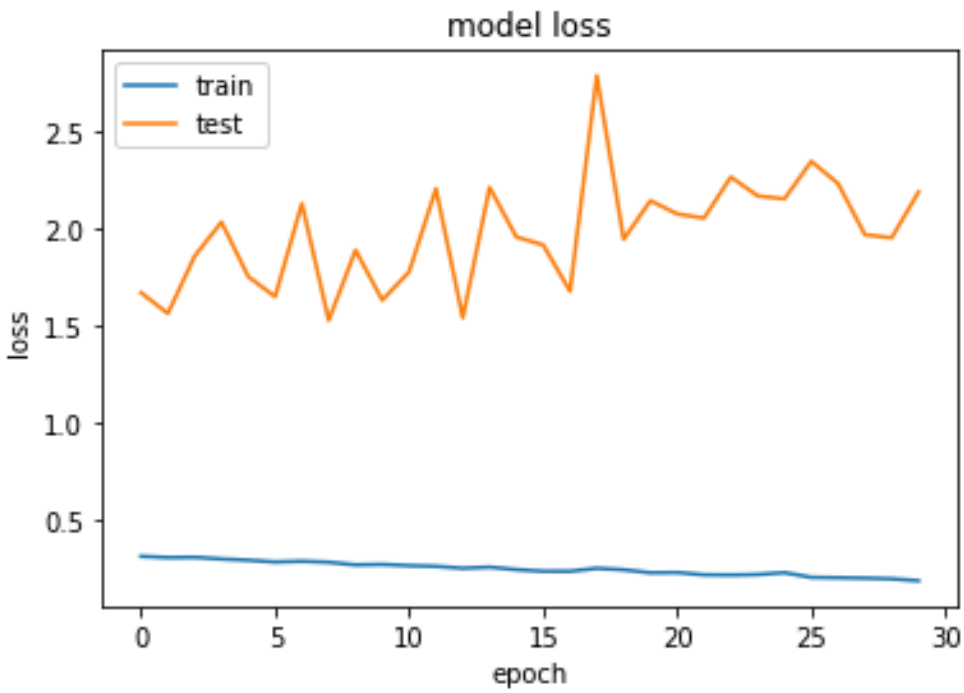
In the example below, the 58 year old patient had Cardiomegaly but none of the other diseases. Formatting the data this way made it easier to determine which patients had Pneumonia.

```
,Image Index,Finding Labels,Follow-up #,Patient ID,Patient Age,Patient Gender,View Position,OriginalImage[Width,Height],OriginalImagePixelSpacing[
x,y],Unnamed: 11,path,Cardiomegaly,Emphysema,Effusion,No
Finding,Hernia,Infiltration,Mass,Nodule,Atelectasis,Pneumothorax,Pleural_Thickening,Pneumonia,Fibrosis,Edema,Consolidation
0,00000001_000.png,Cardiomegaly,0,1,58,M,PA,2682,2749,0.14300000000000002,0.14300000000000002,target_images/images_001/images/
00000001_000.png,1,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
Finding,Hernia,Infiltration,Mass,Nodule,Atelectasis,Pneumothorax,Pleural_Thickening,Pneumonia,Fibrosis,Edema,Consolidation
0,00000001_000.png,Cardiomegaly,0,1,58,M,PA,2682,2749,0.14300000000000002,0.14300000000000002,target_images/images_001/images/
00000001_000.png,1,0,0,0,0,0,0,0,0,0,0,0,0,0
```

First I had the the algorithm just try to predict images that were only positive for Pneumonia, and no other disease. This was not very effective. The model trying to learn that data is plotted below. As is typical when I was trying to get the network to learn this data, the model learned the actual XRAY images it sees very well. It can almost learn them perfectly if it is given enough time. However, the model is not able to actually generalize on these Pneumonia images very well at all.





I finally switched over and started looking for images that were positive for both Pneumonia and lung infiltration, and I grouped these as positive for Pneumonia. I thought this would be a good idea since Pneumonia and lung infiltration are similar diseases, and may look similar in an XRAY. Using this

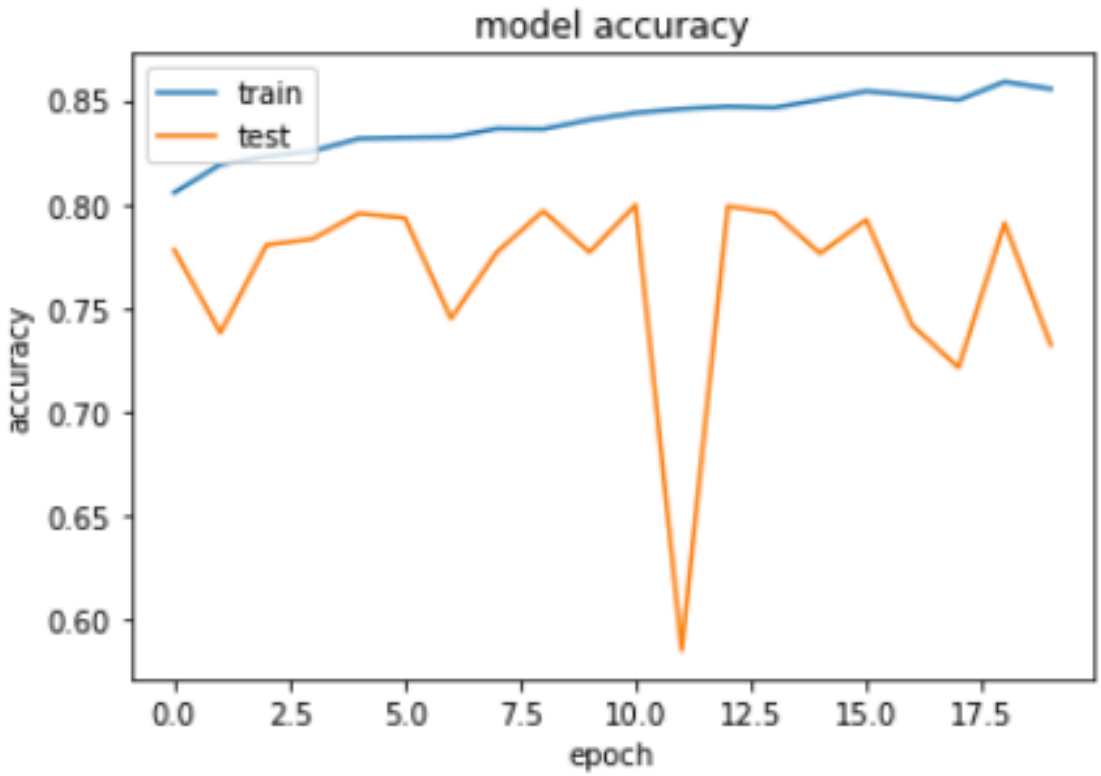
technique produced the following training log.

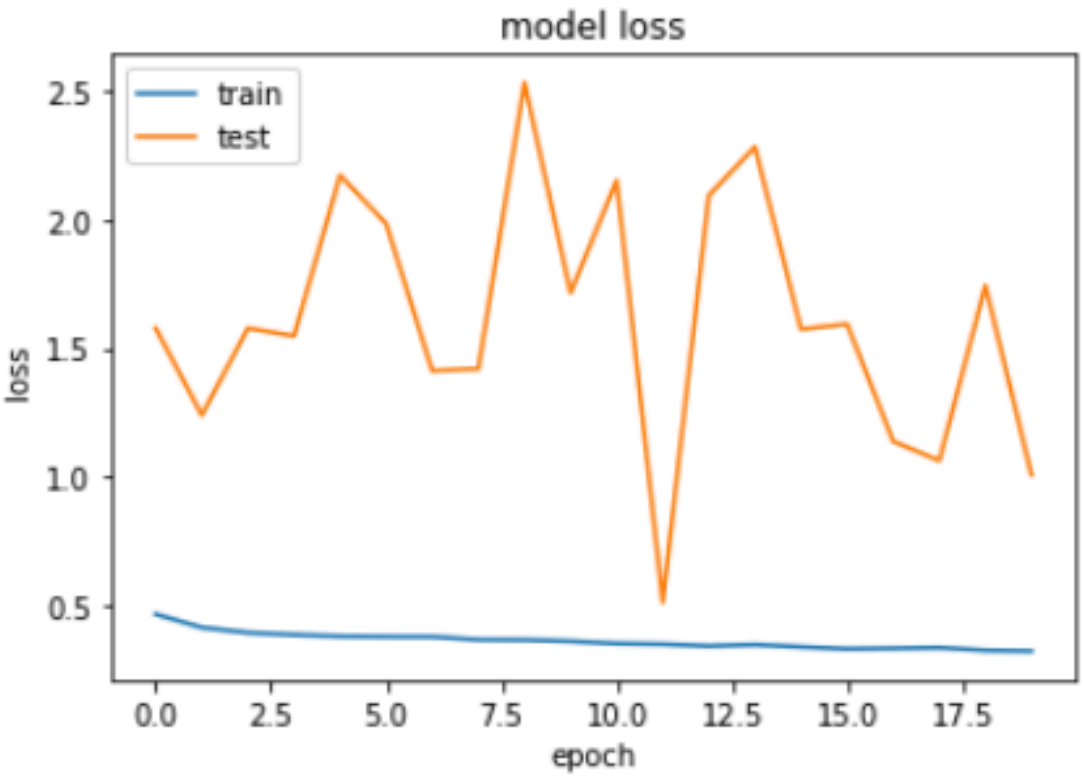
```
SUCCESS: Returning built model!
SUCCESS: LOADED MODEL FROM DISK
Loaded H5 Model Success
SUCCESS: LOADED MODEL
Begin Training
Number of Training Batches: 196
Number of Validation Batches: 196
Epoch 1/20
196/196 [==============================] - 566s 3s/step - loss:
0.4681 - binary_accuracy: 0.8055 - val_loss: 1.5773 - val_binary_accuracy:
0.7780
Epoch 2/20
196/196 [==============================] - 527s 3s/step - loss:
0.4146 - binary_accuracy: 0.8190 - val_loss: 1.2396 - val_binary_accuracy:
0.7381
Epoch 3/20
196/196 [==============================] - 499s 3s/step - loss:
0.3953 - binary_accuracy: 0.8230 - val_loss: 1.5775 - val_binary_accuracy:
0.7804
Epoch 4/20
196/196 [==============================] - 496s 3s/step - loss:
0.3866 - binary_accuracy: 0.8255 - val_loss: 1.5471 - val_binary_accuracy:
0.7833
Epoch 5/20
196/196 [==============================] - 499s 3s/step - loss:
0.3806 - binary_accuracy: 0.8314 - val_loss: 2.1715 - val_binary_accuracy:
0.7955
Epoch 6/20
196/196 [==============================] - 492s 3s/step - loss:
0.3792 - binary_accuracy: 0.8319 - val_loss: 1.9833 - val_binary_accuracy:
0.7933
Epoch 7/20
196/196 [==============================] - 493s 3s/step - loss:
0.3790 - binary_accuracy: 0.8322 - val_loss: 1.4118 - val_binary_accuracy:
0.7450
Epoch 8/20
```

```
196/196 [==============================] - 488s 2s/step - loss:
0.3670 - binary_accuracy: 0.8363 - val_loss: 1.4200 - val_binary_accuracy:
0.7770
Epoch 9/20
196/196 [==============================] - 491s 3s/step - loss:
0.3657 - binary_accuracy: 0.8361 - val_loss: 2.5312 - val_binary_accuracy:
0.7967
Epoch 10/20
196/196 [==============================] - 488s 2s/step - loss:
0.3614 - binary_accuracy: 0.8406 - val_loss: 1.7151 - val_binary_accuracy:
0.7769
Epoch 11/20
196/196 [==============================] - 488s 2s/step - loss:
0.3524 - binary_accuracy: 0.8438 - val_loss: 2.1515 - val_binary_accuracy:
0.7996
Epoch 12/20
196/196 [==============================] - 495s 3s/step - loss:
0.3503 - binary_accuracy: 0.8458 - val_loss: 0.5126 - val_binary_accuracy:
0.5855
Epoch 13/20
196/196 [==============================] - 491s 3s/step - loss:
0.3427 - binary_accuracy: 0.8469 - val_loss: 2.0932 - val_binary_accuracy:
0.7990
Epoch 14/20
196/196 [==============================] - 491s 3s/step - loss:
0.3480 - binary_accuracy: 0.8463 - val_loss: 2.2816 - val_binary_accuracy:
0.7957
Epoch 15/20
196/196 [==============================] - 490s 2s/step - loss:
0.3390 - binary_accuracy: 0.8502 - val_loss: 1.5738 - val_binary_accuracy:
0.7763
Epoch 16/20
196/196 [==============================] - 493s 3s/step - loss:
0.3315 - binary_accuracy: 0.8543 - val_loss: 1.5933 - val_binary_accuracy:
0.7925
Epoch 17/20
196/196 [==============================] - 494s 3s/step - loss:
0.3339 - binary_accuracy: 0.8524 - val_loss: 1.1383 - val_binary_accuracy:
```

```
0.7417
Epoch 18/20
196/196 [==============================] - 494s 3s/step - loss:
0.3370 - binary_accuracy: 0.8501 - val_loss: 1.0629 - val_binary_accuracy:
0.7214
Epoch 19/20
196/196 [==============================] - 491s 3s/step - loss:
0.3258 - binary_accuracy: 0.8588 - val_loss: 1.7431 - val_binary_accuracy:
0.7909
Epoch 20/20
196/196 [==============================] - 488s 2s/step - loss:
0.3245 - binary_accuracy: 0.8554 - val_loss: 1.0083 - val_binary_accuracy:
0.7323
Training completed in time:  2:45:54.786849
Done Training
```

In [14]:


model accuracy
train
test
accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch


model loss
train
test
loss
2.5
2.0
1.5
1.0
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch

As illustrated, the model did learn better when it was looking for images that had both Pneumonia and Lung infiltration. However, it still is not good enough to submit to the FDA for approval. The model scored over an 85% accuracy rate on images it was training on. And it looks like when the training stopped, this training data accuracy was still increasing. This is great. However, the validation data was still under 80% accuracy the entire time. It is not learning to generalize on a lot of these images. The question is why?

My training data is split up in a simple fashion. First for the training set, there are 10000 Pneumonia (or Lung Infiltration) positive XRAY images and 10000 negative XRAY images. And for the validation set, there are 2500 Pneumonia (or Lung Infiltration) positive XRAY images and 2500 Pneumonia negative XRAYs.

```
df_sample_pos = patched_df[patched_df["Pneumonia_or_Infiltration"] == "1"] #*these will have pr
print("Length of Specific Pneumonia: " + str(len(df_sample_pos)))
#print(df_sample_pos.head(5))
print(df_sample_pos.head(1))

pne_positive_train = df_sample_pos[0 : 10000] #**use these for training #*train with one thousa
print(len(pne_positive_train))
pne_positive_val = df_sample_pos[10001 : 20000] #**use these for validation. These have pneumon
print(len(pne_positive_val))

df_sample_neg = patched_df[patched_df["Pneumonia_or_Infiltration"] == "0"] #*these wont have pr
print(len(df_sample_neg))
non_pne_train = df_sample_neg[0 : 2500]      #*validate with 500 positive
print(len(non_pne_train))
non_pne_val = df_sample_neg[2501 : 5001]     #*validate with 500 negative
print(len(non_pne_val))

#sys.exit(1)

temp_combine = [pne_positive_train, non_pne_train]
patched_df_train = pd.concat(temp_combine)

temp_combine = [pne_positive_val, non_pne_val]
patched_df_test = pd.concat(temp_combine)
```

The model utilized for training is illustrated below. It is simply a VGG16 model with the input layer modified to accept images sized 224 x 224 x 3. The final layer is modified to allow an output of only 1. This is simply to determine whether the image looks like it has Pneumonia or not. The printed model is illustrated below.

Model: "model_1"

_____

```
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_1 (Flatten)          (None, 25088)             0
_____________________________________________________________________
_____
dense_1 (Dense)              (None, 1)                 25089
=================================================================
========
Total params: 14,739,777
Trainable params: 25,089
Non-trainable params: 14,714,688
```

Importantly, the activation model for the network is set to SIGMOID so that we have probabilities to work with during the prediction phase. Also, the loss rate I settled on was 0.0007.

```
IMAGE_SIZE = [224, 224] #**use a little bigger images, to get more xray data!
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
# here [3] denotes for RGB images(3 channels)

# don't train existing weights GPU TOO SLOW
for layer in vgg.layers:
    layer.trainable = False

x = Flatten()(vgg.output)
prediction = Dense(number_output, activation='sigmoid')(x)
model = Model(inputs=vgg.input, outputs=prediction)
#model.compile(loss='categorical_crossentropy',
#              optimizer=optimizers.Adam(),
#              metrics=['accuracy'])

#model.compile(optimizers.rmsprop(lr=0.0005, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
model.compile(optimizer = Adam(lr=0.0007), loss='binary_crossentropy', metrics=['binary_accuracy'])
model.summary()

from datetime import datetime
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-6)

print("SUCCESS: Returning built model!")
```

For our training set we used the image augmentations illustrated below. It seemed that the augmentations that caused the model to generalize the best was rotating them a little bit, zooming in and out a little bit, and varying the brightness of each XRAY image.

```
datagen = ImageDataGenerator(rescale = 1./255,
                             #horizontal_flip = True,
                             #height_shift_range = 0.1,
                             #width_shift_range = 0.1,
                             rotation_range = 15,
                             #shear_range = 0.1,
                             zoom_range = 0.25,
                             brightness_range=[0.35, 1.1],
                             samplewise_center = True,
                             samplewise_std_normalization = True
                             #validation_split=0.20
                            )
```

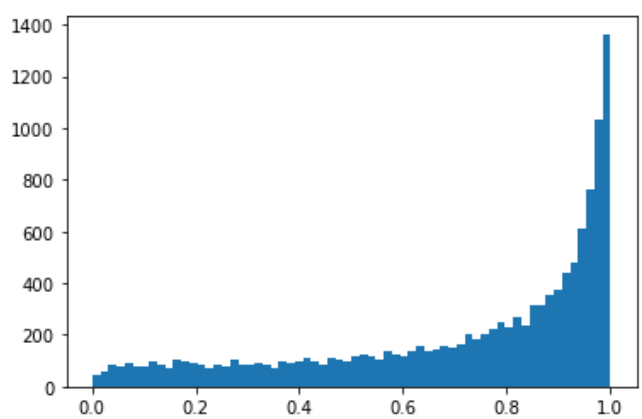
Illustrated below are the distribution of probabilities detected by this model from the validation set, mentioned above. Given the validation set was essentially set to be equal between Pneumonia and non Pneumonia XRAYs, this distribution does not seem correct. Its obvious the model leaned more towards predicting false positives since the predictions are skewed to be higher than 0.5.

In [17]:
```
print("Prediction Data from Validation Data.")

print(pred_Y_val.shape)
print(pred_Y_val)
print(valY_val)
# Look at the distribution of the prediction probabilities
plt.hist(pred_Y_val, bins=64)
```

```
Prediction Data from Validation Data.
(12500,)
[0.8867315  0.9964683  0.36683184 ... 0.6141787  0.8668419  0.90602577]
[1. 1. 1. ... 1. 1. 1.]
```

Out[17]:
```
(array([  44.,   58.,   85.,   75.,   91.,   81.,   78.,   96.,   85.,
          74.,  101.,   96.,   88.,   85.,   72.,   83.,   78.,  101.,
          86.,   87.,   89.,   86.,   73.,   95.,   93.,   98.,  110.,
          99.,   87.,  113.,  101.,   99.,  117.,  126.,  120.,  102.,
         140.,  124.,  120.,  135.,  156.,  136.,  141.,  154.,  149.,
         164.,  200.,  181.,  203.,  221.,  246.,  230.,  268.,  234.,
         314.,  314.,  355.,  376.,  437.,  480.,  609.,  766., 1031.,
        1364.]),
 array([0.00118367, 0.01678932, 0.03239496, 0.0480006 , 0.06360625,
        0.0792119 , 0.09481754, 0.11042318, 0.12602884, 0.14163448,
        0.15724012, 0.17284577, 0.18845141, 0.20405705, 0.2196627 ,
        0.23526834, 0.25087398, 0.26647964, 0.28208527, 0.29769093,
        0.31329656, 0.3289022 , 0.34450784, 0.3601135 , 0.37571913,
        0.3913248 , 0.40693042, 0.42253608, 0.43814173, 0.45374736,
        0.46935302, 0.48495865, 0.5005643 , 0.51616997, 0.5317756 ,
        0.5473812 , 0.56298685, 0.57859254, 0.59419817, 0.6098038 ,
        0.6254095 , 0.6410151 , 0.65662074, 0.67222637, 0.68783206,
        0.7034377 , 0.7190433 , 0.73464894, 0.75025463, 0.76586026,
        0.7814659 , 0.7970715 , 0.8126772 , 0.82828283, 0.84388846,
        0.85949415, 0.8750998 , 0.8907054 , 0.90631104, 0.9219167 ,
        0.93752235, 0.953128  , 0.9687336 , 0.9843393 , 0.9999449 ],
       dtype=float32),
 <a list of 64 Patch objects>)
```



The ROC curve generated from plotting out the predictions on the validation problem illustrates the same problem. As our theshold rises our false positive rates shoots up to an unacceptable level. This obviously

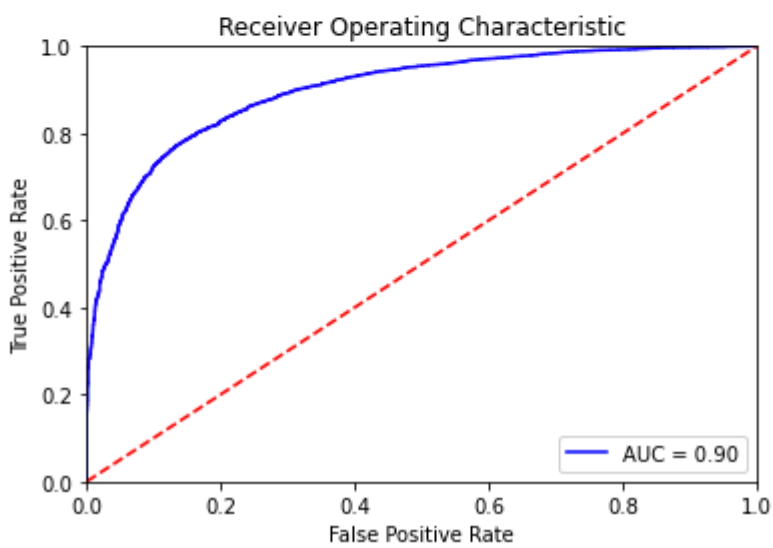
indicates the model is not generalizing on identifying Pneumonia XRAYs very well at all.

In [19]:
```
print("---ROC CURVE FROM VALIDATION DATA---")

fpr, tpr, threshold = metrics.roc_curve(valY_val, pred_Y_val)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
---ROC CURVE FROM VALIDATION DATA---
```



We can also clearly see this problem simply by evaluating the model itself with the training and validation data generators. By default Keras uses a 0.5 threshold for this validation but I do not believe it would improve even if we adjusted the threshold. I simply think the problem is that the model itself is not learning to generalize Pneumonia XRAYs very well from this specific dataset. It should be noted however that the model has over an 85

percent accuracy rate on the training data, and the accuracy rate was still climbing when training stopped. This indicates the VGG is actually capable of learning the XRAY images themselves, but there is a problem with generalizing them correctly. I suspected it might be a problem with the dataset itself. I explore that suspicion a little further below.

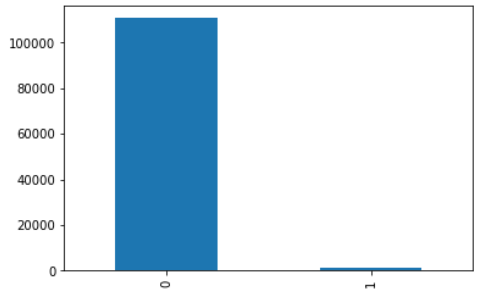
```
196/196 [==============================] - 329s 2s/step
Test Loss: 0.3579466938972473
Test accuracy: 0.8533599972724915
---EVALUATE EVAL GENERATOR--
```

```
196/196 [==============================] - 193s 983ms/step
Test Loss: 1.0083415508270264
Test accuracy: 0.7322986125946045
```

I made an effort to understand why the model was not generalizing the data very well, because this needs to be understood if we are ever able to get this code FDA approved to identify Pneumonia positive XRAYs. The FDA simply is not going to accept a model unless the accuracy of the model on validation data is well over 90 percent on a very large XRAY dataset the model has not seen before, and the false negative rate is very low — less than 1 percent.

A quick look at the model finds some immediate weaknesses. First of all, there just are not many images that are marked as having just Pneumonia. We can see in our plot below, there are only 1431 Pneumonia positive images out of over 110,000 XRAY images.

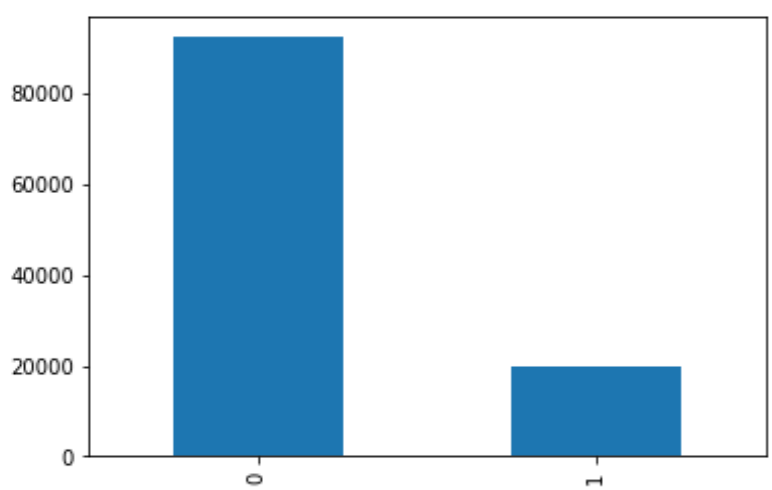
```
DISTRUBUTION of Pneumonia Positive XRAYS in dataset
Pneumonia
0    110689
1      1431
Name: Pneumonia, dtype: int64
Problem: This data set is not really effective at helping the model learn to generalize on Pneumonia. There are over
100k xrays but only a fraction of them are Pneumonia positive.
```



If we look at the distribution of just lung infiltration, we get some more XRAYs — which gives us a bit more training data. That is why I decided to have the model look for images that were positive for both pneumonia and lung infiltration. Just training on only pneumonia positive images simply did not give us enough data.

```
DISTRIBUTION of Lung Infiltration in data set
Infiltration
0    92226
1    19894
Name: Infiltration, dtype: int64

Out[99]: <matplotlib.axes._subplots.AxesSubplot at 0x7fea0a36f190>
```

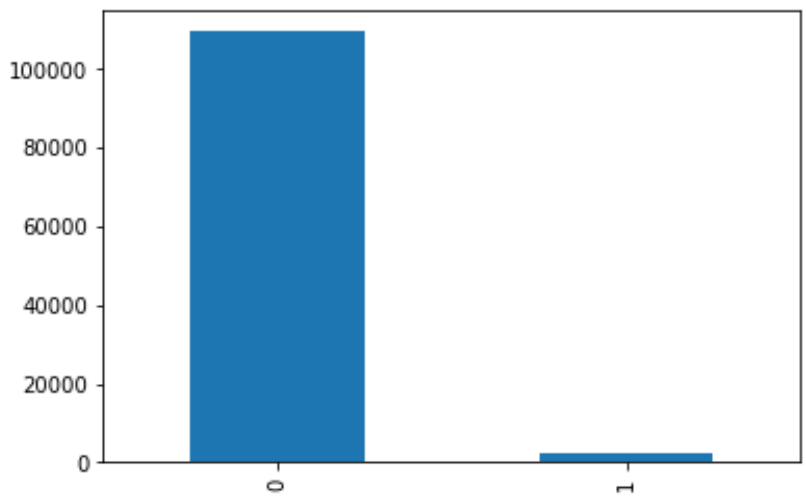


I was going to try to group emphysema in, but there just aren't really

many emphysema positive XRAYs either so I wasn't sure it was even worth the effort.

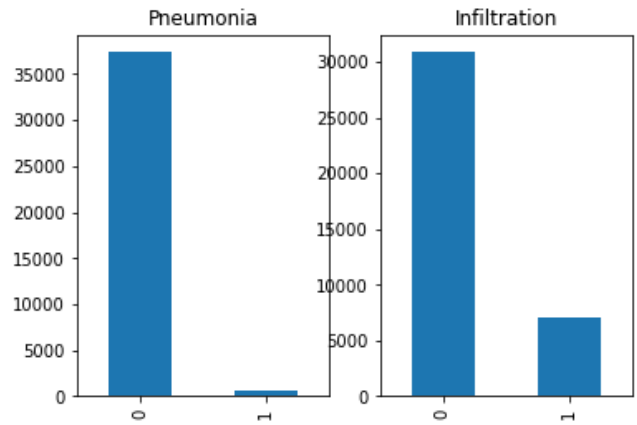
```
DISTRIBUTION of Emphysema in data set
Emphysema
0    109604
1      2516
Name: Emphysema, dtype: int64

Out[100]: <matplotlib.axes._subplots.AxesSubplot at 0x7fea0a396510>
```



Displayed below is a plot of both the pneumonia and lung infiltration positive images in the distribution.

```
DISTRIBUTION of lung infiltration and pneumonia positive XRAYS in people under 40
Pneumonia
0    110689
1      1431
Name: Pneumonia, dtype: int64
Infiltration
0    92226
1    19894
Name: Infiltration, dtype: int64
```



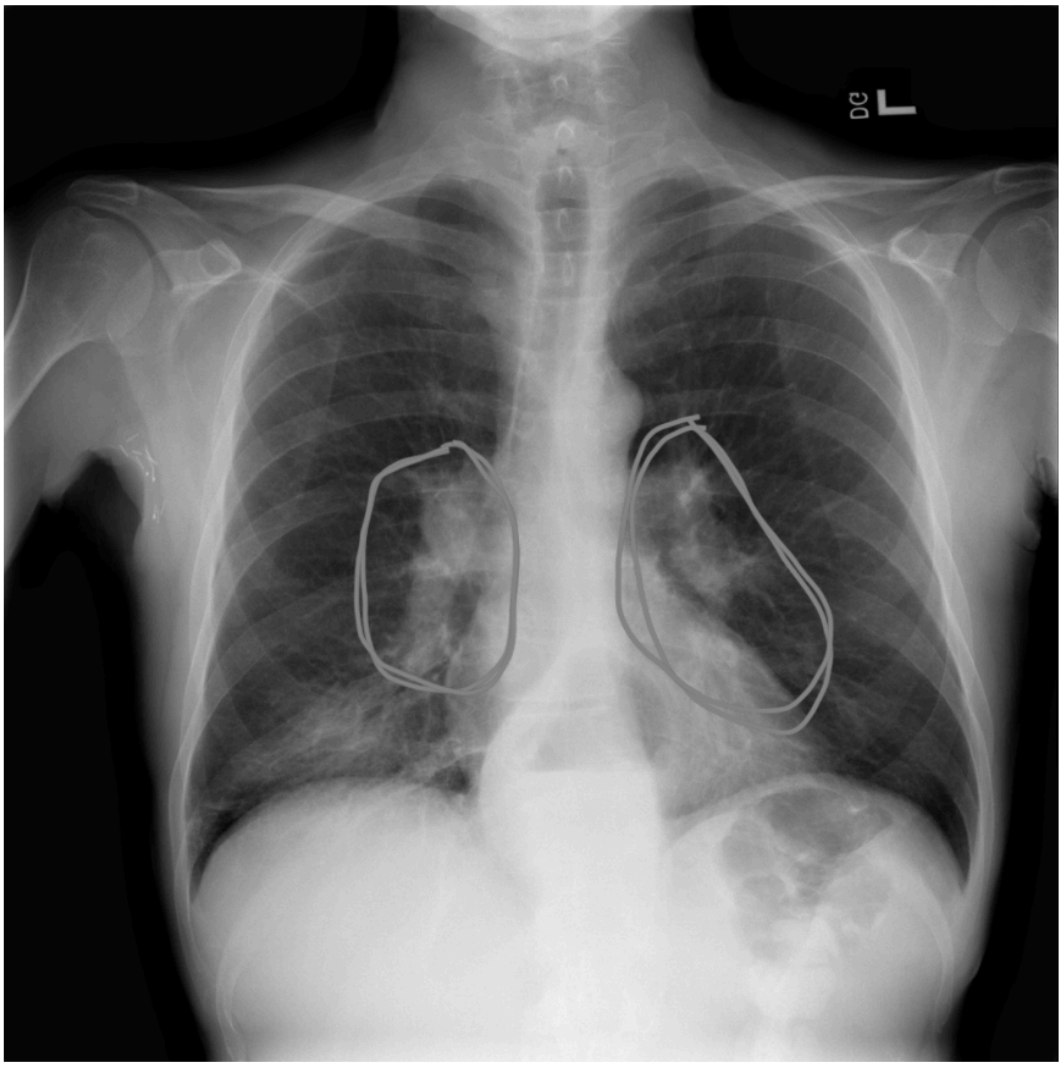
From my inspection I also realized that the problem wasn't just the lack of Pneumonia data, it was also the quality of the XRAY images themselves. Displayed below is an image that was marked as Pneumonia positive by the dataset. Its obvious to the naked eye, that this XRAY does appear to show Pneumonia or some other type of lung inflammation.

```python
from IPython.display import Image
display(Image(filename='MARKEDpositive_00009507_002.png'))

print("This XRAY was marked Pneumonia positive by the dataset. The XRAY Does obviously have lur
```

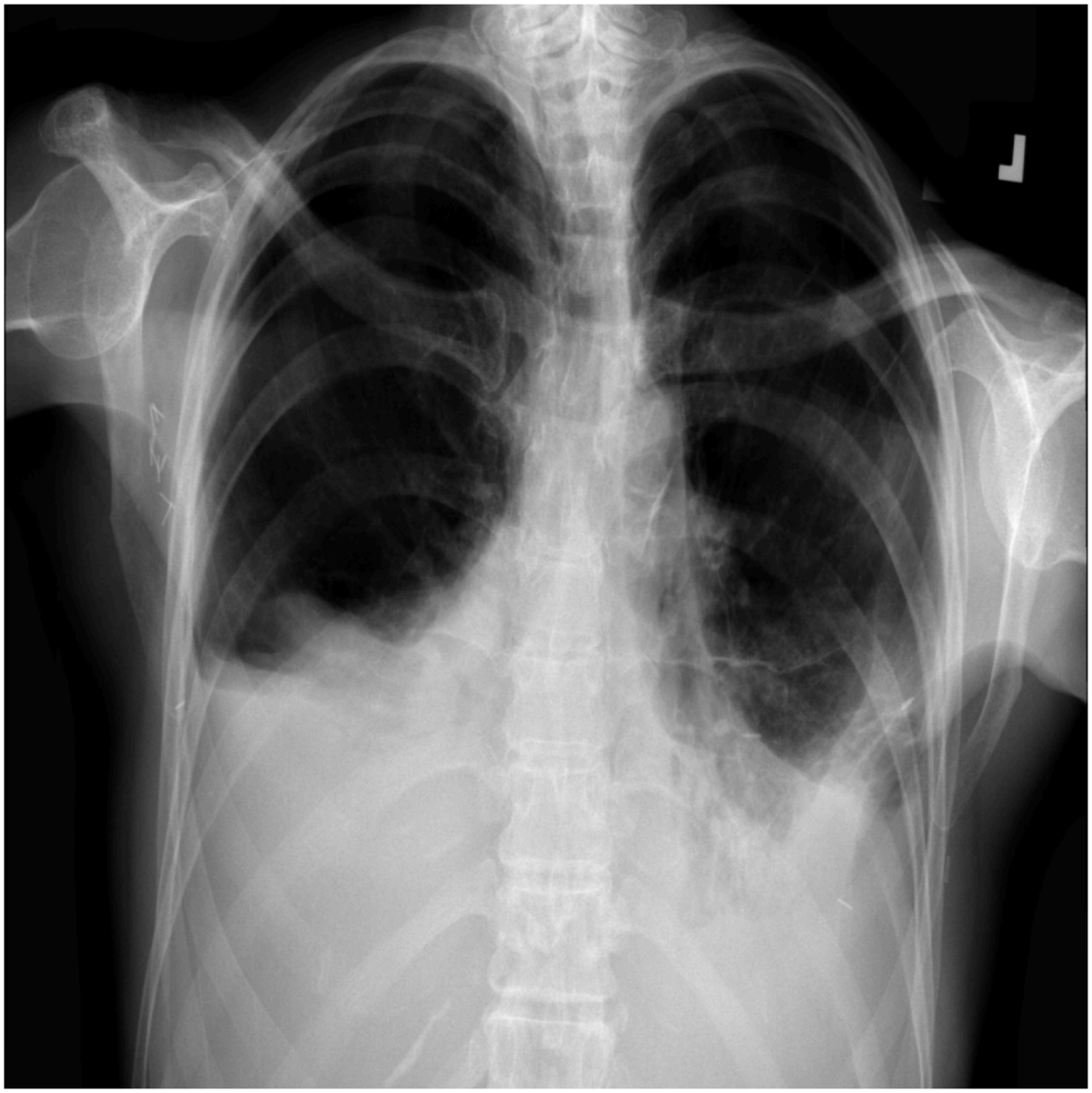
Its the same situation with the XRAY below. We can obviously see some type of lung inflammation in the XRAY image.

```
In [106]: display(Image(filename='MARKEDpositive_00009507_004.png'))
          print("This XRAY was marked positive by the dataset. The XRAY Does obviously have lung infiltra
```



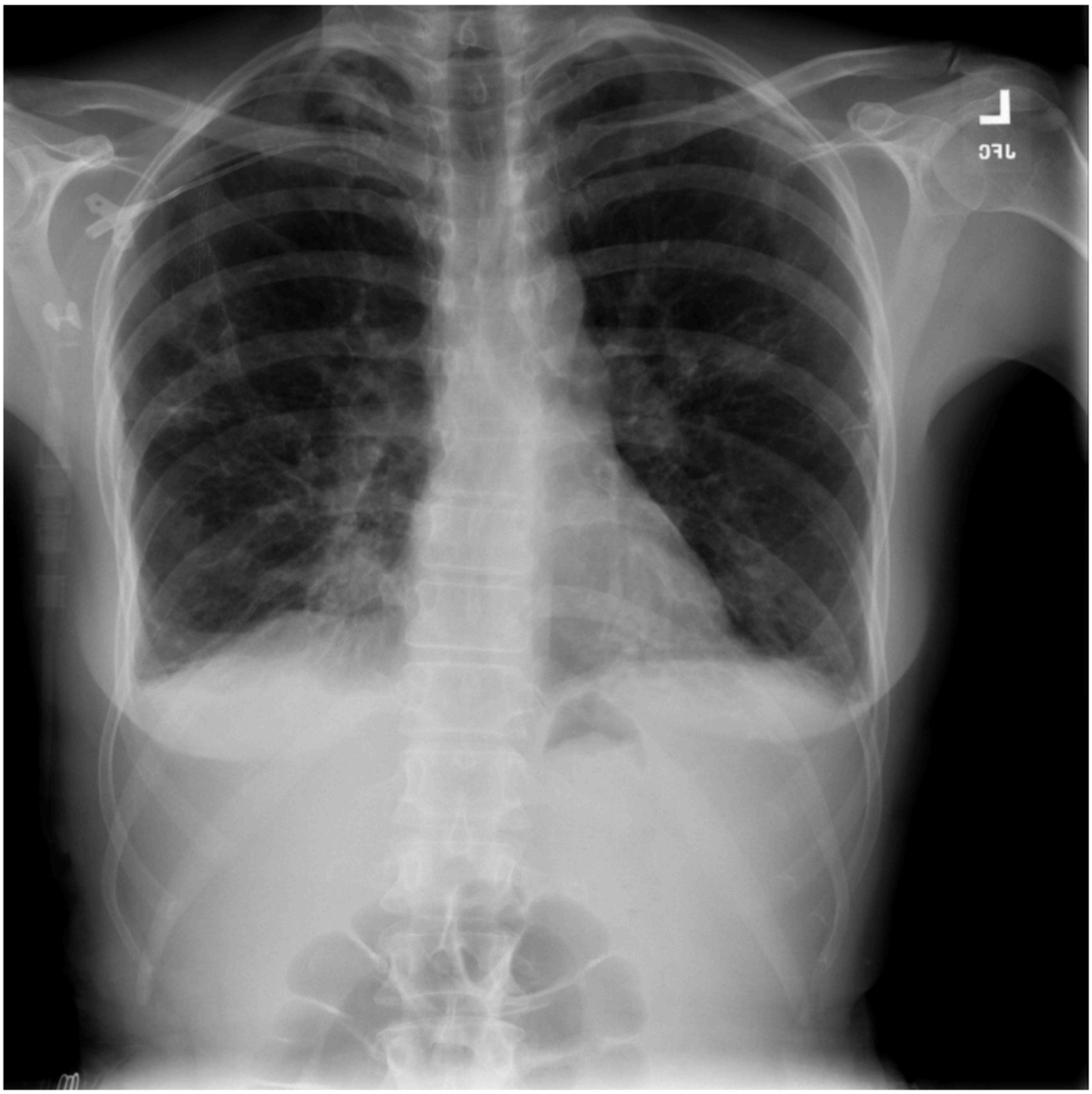
The dataset has some really good Pneumonia Negative images as well, such as the following 2 XRAY images. From my naked eye these XRAYs to not appear to indicate any type of inner lung inflammation.

```
print("This XRAY is marked negative via the dataset. It does appear to have NO serious lung inf
```



```
This XRAY is marked negative via the dataset. It does appear to have NO serious lung infiltra
tion that I can see with my naked eye
```

```
print("This XRAY is marked negative via the dataset. It does appear to have NO serious lung inf
```

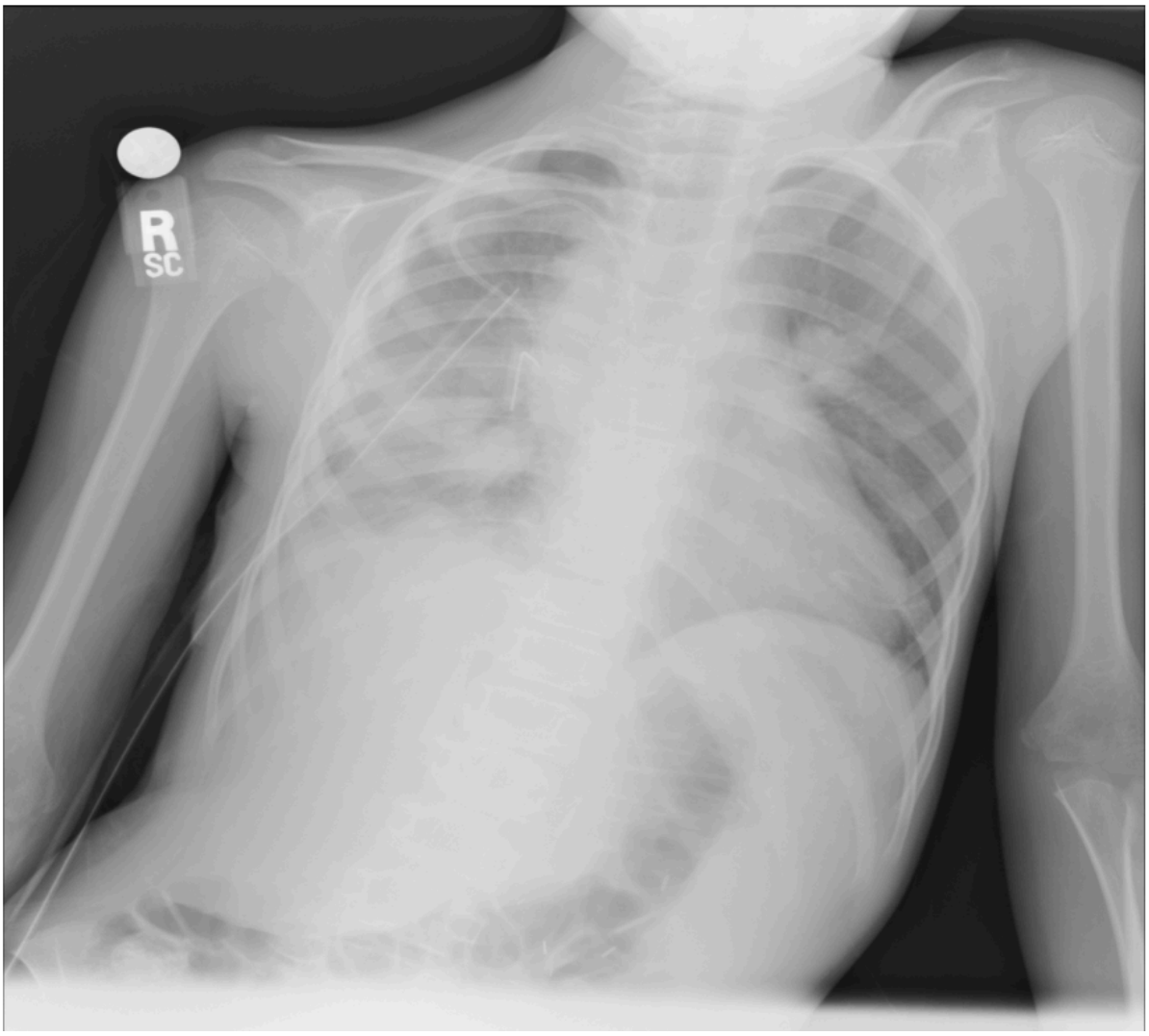


```
This XRAY is marked negative via the dataset. It does appear to have NO serious lung infiltra
tion that I can see with my naked eye
```

But the following four images are problematic. They are all marked Pneumonia negative, but they appear to either actively show inner lung inflammation or the image is of such poor quality you can't tell one way or another. Obviously if I can't tell them apart with my eye, an algorithm might get confused on these images as well!

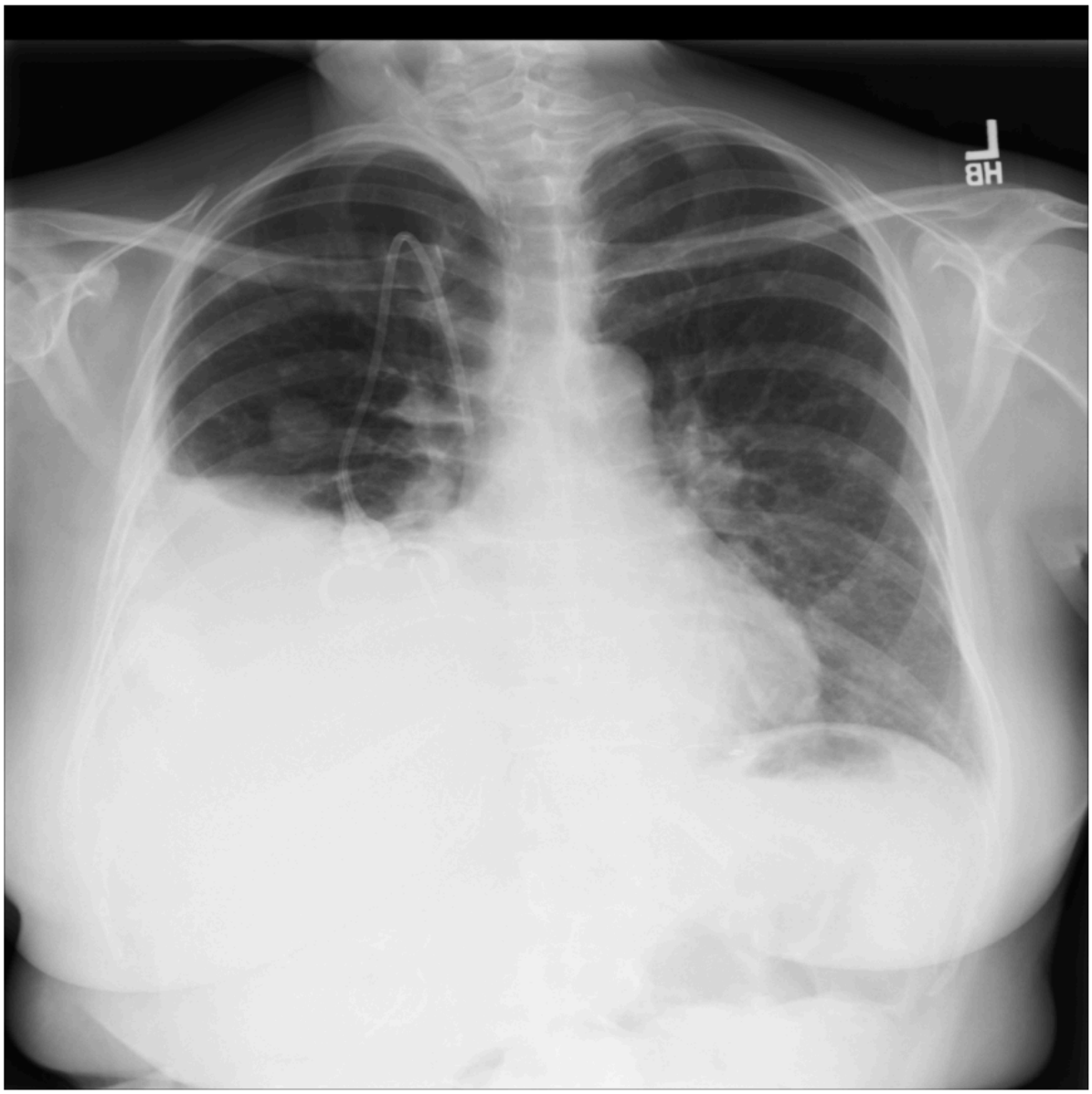
```
display(Image(filename='FALSE_NEGATIVES00007830_005.png'))

print("This XRAY was marked Pneumonia negative by the dataset. However it is obviously poor qua
```



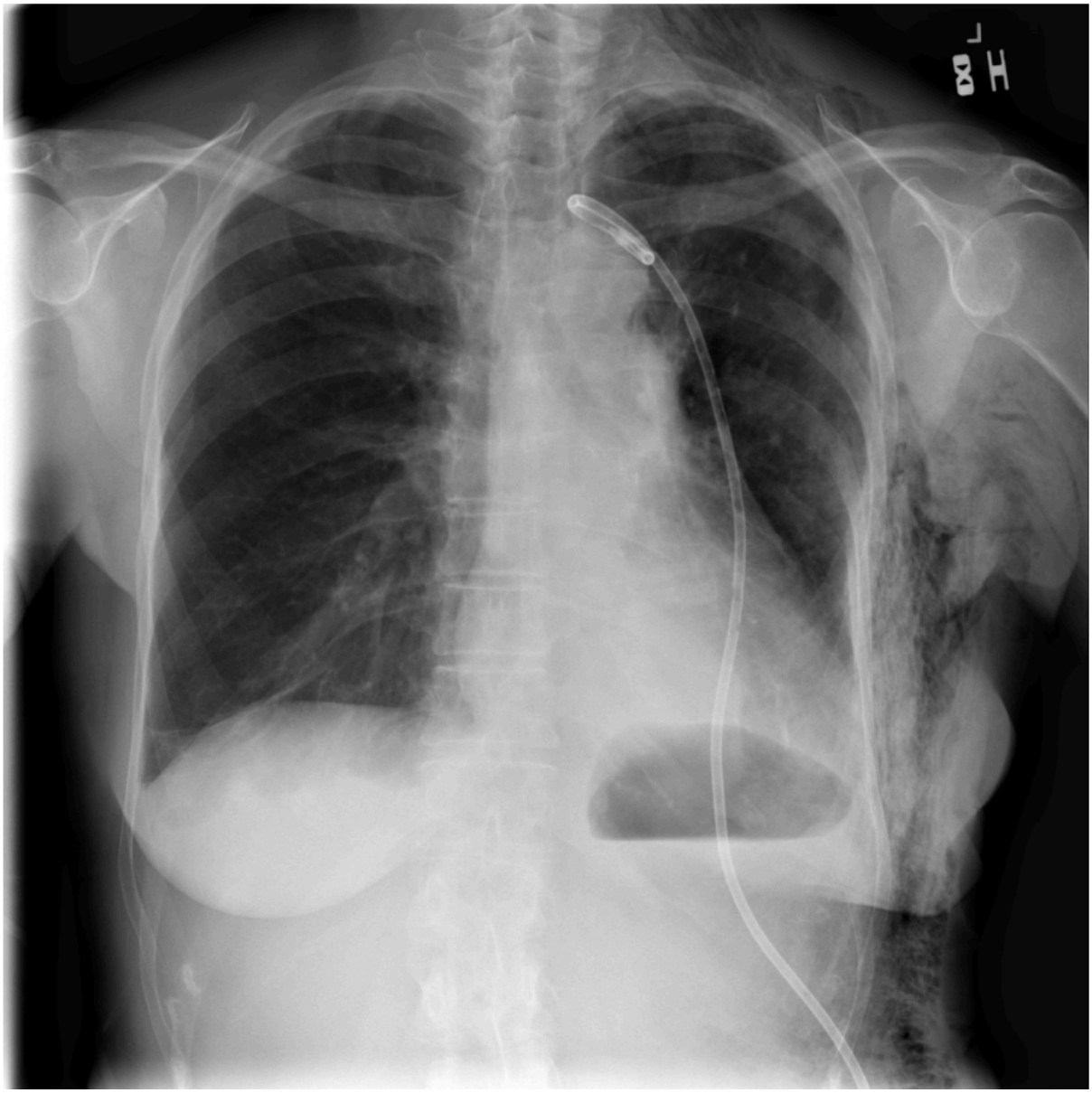
```
display(Image(filename='FALSE_NEGATIVES00007848_004.png'))

print("This XRAY was marked Pneumonia negative by the dataset. However it is obviously poor qua
```



```
display(Image(filename='FALSE_NEGATIVES00007865_003.png'))

print("This XRAY was marked Pneumonia negative by the dataset. However it is obviously poor qua
```



```
print("This XRAY was marked Pneumonia negative by the dataset. However it is obviously poor qua
```

I think the best strategy if we had to use this specific dataset would be to manually go through and discard the poor quality images. However, that doesn't fix the problem that this dataset is simply not balanced enough. It does not have enough Pneumonia positive XRAY images, and many of the Pneumonia images it has are not specific for Pneumonia and contain other accompanying diseases.

In summary, I do not think it is possible from this specific NIH dataset

to develop a Pneumonia detecting algorithm that could meet FDA approval. For the FDA to accept it, it would have to have over a 90 percent accuracy rate of detecting Pneumonia — with a false negative rate of under 1 percent. It would be dangerous for such an algorithm to deploy in a clinical setting with a high false negative rate — because it might cause patients lung inflammation / Pneumonia disease to be missed by the care giver!