Data Science Career Track

Capstone 2 -

Milestone 2 Report

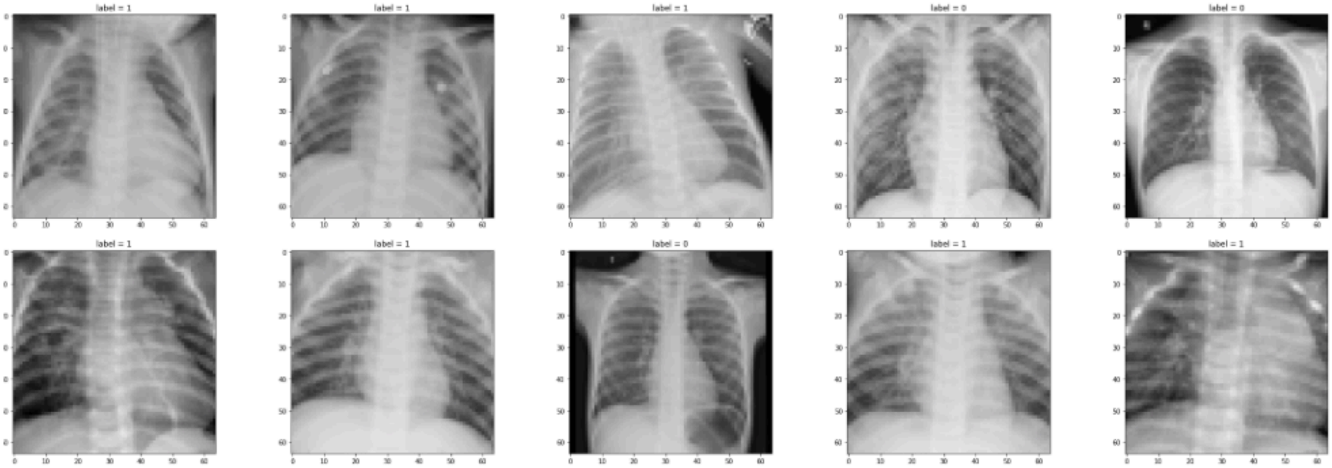
by Edward Franke
06/09/2019

Detect Pneumonia is chest x-rays Project – Milestone 1 Report

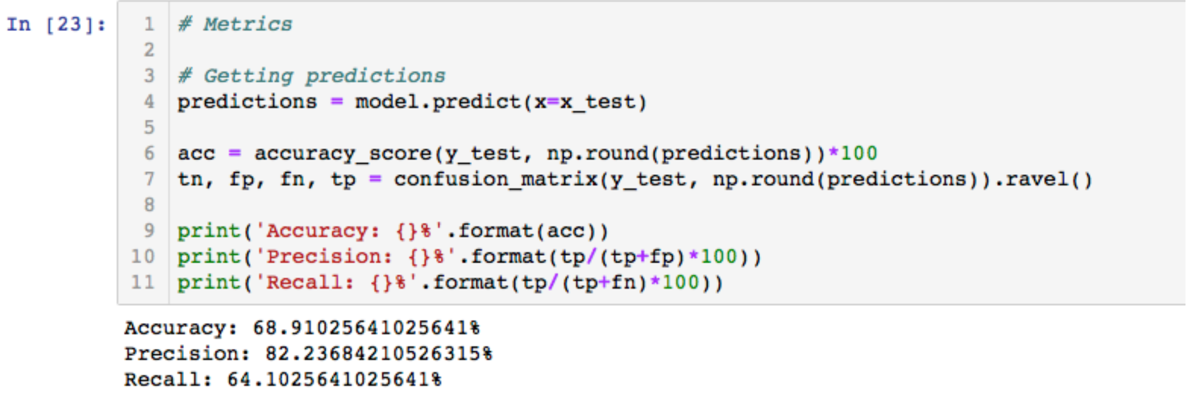
EXECUTIVE SUMMARY:

The purpose for this project is to find a correlation connected to chest x-rays containing pneumonia that can separate them in real time compared to normal healthy chest x-rays. The dataset is a cleaned dataset from Kaggle.com. Tensorflow and Keras has discovered some connections but further analysis is required. The second dataset has been a challenge and a solution yet to be discovered.

IDEA: A model to detect pneumonia is chest x-rays. (problem to solve)
CLIENT: Medical Industry
REASON: Pneumonia is a very serious condition that has the potential for death. I have personal knowledge of how serious it can be. The sooner it can be detected, the better the chance for survival and less damage to the lungs.
DATA: From Kaggle, 2 cleaned datasets with 5863 images and over 112,000 images.
SOLUTION: Create a model or analysis to discover what makes an x-ray image of the chest to have pneumonia and to capture this automatically.
DETAILS: See below.
DELIVERABLES: Code and a presentation outlining the discoveries.



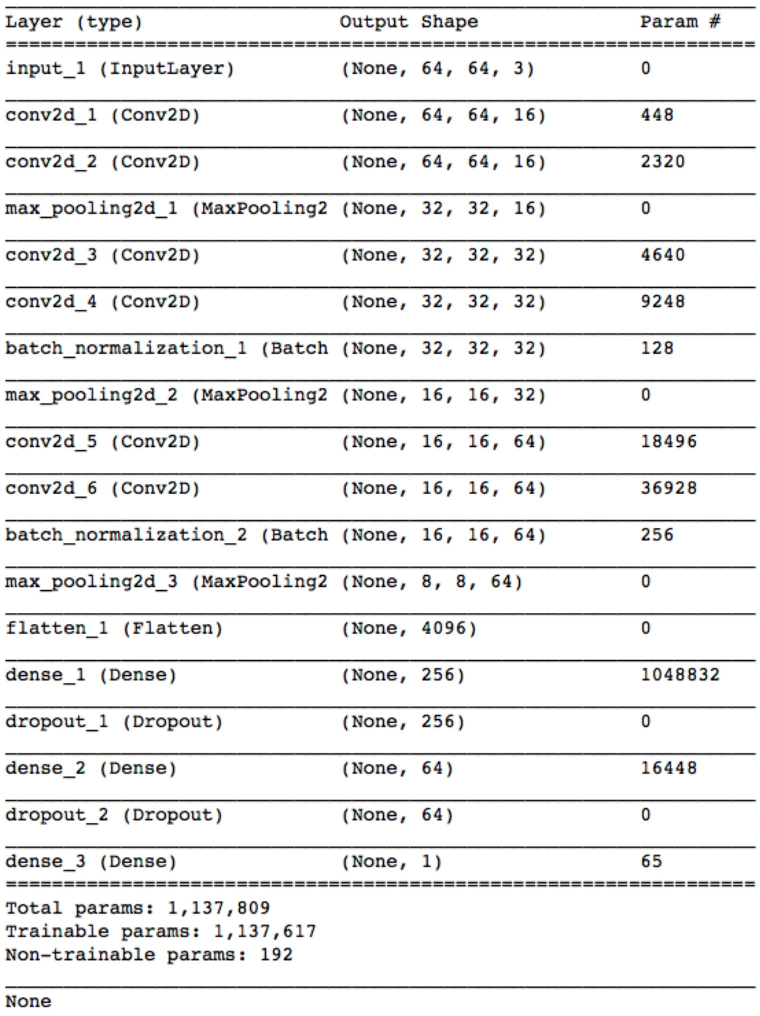
Initial findings from exploratory analysis



```
In [23]:

# Metrics

# Getting predictions
predictions = model.predict(x=x_test)

acc = accuracy_score(y_test, np.round(predictions))*100
tn, fp, fn, tp = confusion_matrix(y_test, np.round(predictions)).ravel()

print('Accuracy: {}%'.format(acc))
print('Precision: {}%'.format(tp/(tp+fp)*100))
print('Recall: {}%'.format(tp/(tp+fn)*100))
```

```
Accuracy: 68.91025641025641%
Precision: 82.23684210526315%
Recall: 64.1025641025641%
```

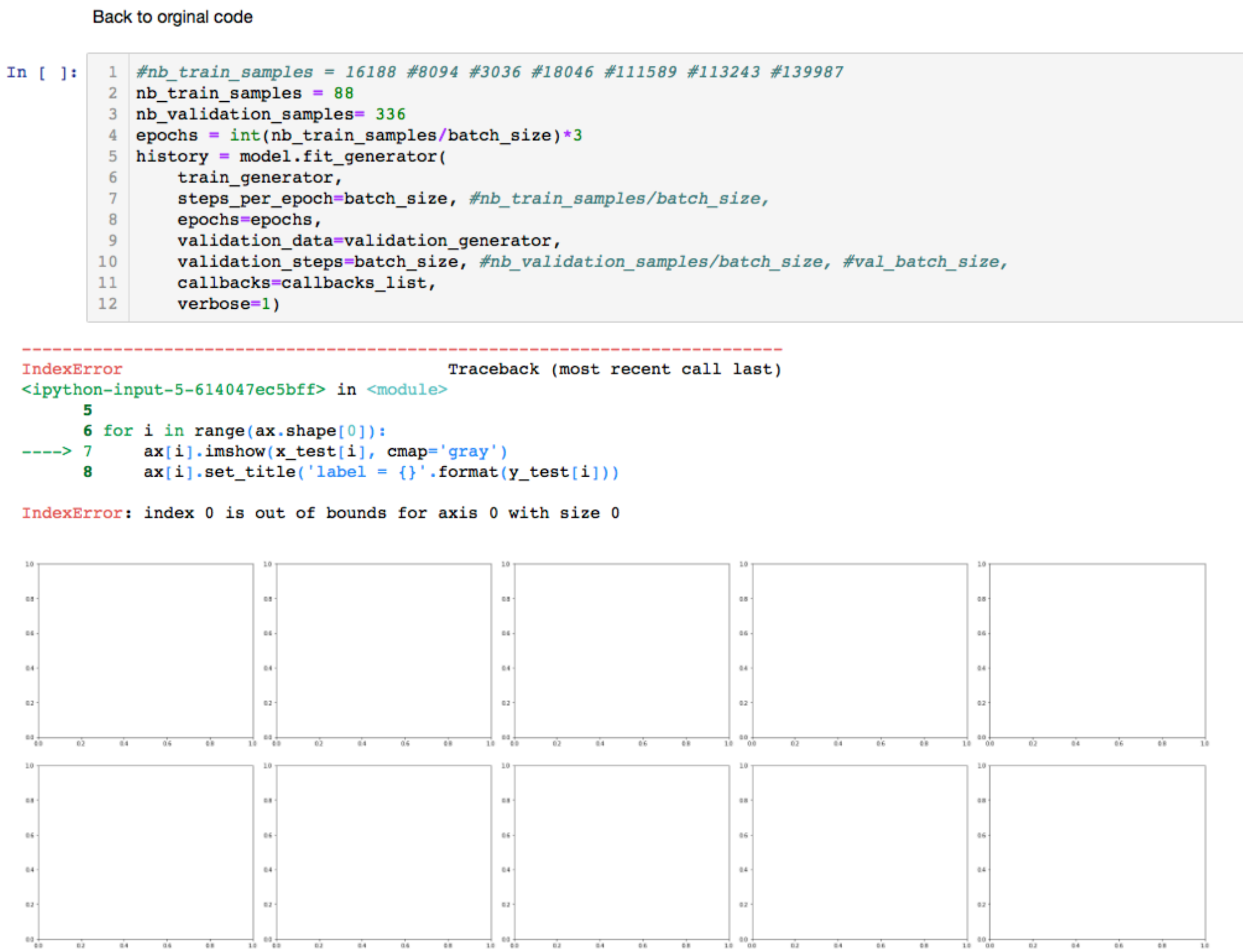
Tensorflow and Keras has discovered some connections but further analysis is required.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 16)        448       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 16)        2320      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 16)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 32)        4640      
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 32)        9248      
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 64)        18496     
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 64)        36928     
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               1048832   
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448     
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65        
=================================================================
Total params: 1,137,809
Trainable params: 1,137,617
Non-trainable params: 192
_________________________________________________________________
None
```

Update

All user code created for the second dataset have currently not worked in some degree. The most promising code has crashed the computer three times despite code changes. Using a merged code set from the first dataset to the second dataset has also been ineffective.



Back to orginal code

```
In [ ]:
#nb_train_samples = 16188 #8094 #3036 #18046 #111589 #113243 #139987
nb_train_samples = 88
nb_validation_samples= 336
epochs = int(nb_train_samples/batch_size)*3
history = model.fit_generator(
    train_generator,
    steps_per_epoch=batch_size, #nb_train_samples/batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=batch_size, #nb_validation_samples/batch_size, #val_batch_size,
    callbacks=callbacks_list,
    verbose=1)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-5-614047ec5bff> in <module>
      5
      6 for i in range(ax.shape[0]):
----> 7     ax[i].imshow(x_test[i], cmap='gray')
      8     ax[i].set_title('label = {}'.format(y_test[i]))

IndexError: index 0 is out of bounds for axis 0 with size 0
```

WHAT'S NEXT

Deep Dive

More code development and a solution found.

Further analysis to attempt greater accuracy and precision.

Final Report and Presentation