Great Learning

AIML

CAPSTONE PROJECT

COMPUTER VISION

PNEUMONIA DETECTION CHALLENGE

INTERIM REPORT

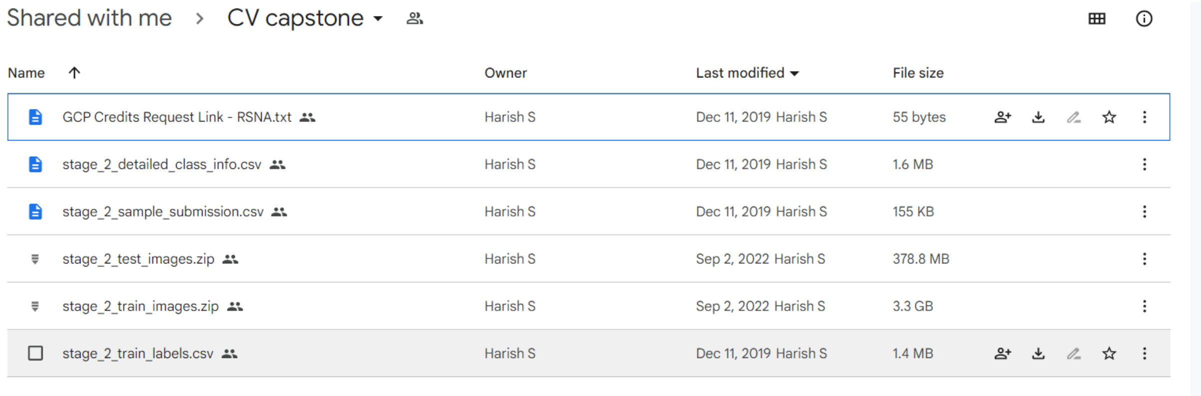
Table of Contents

Background:

Pneumonia is a Health Condition which is caused by Infection that inflames air sacs in one or both **lungs**, which may fill with fluid. With pneumonia, the air sacs may fill with fluid or pus. The infection can be life-threatening to anyone, but particularly to infants, children and people over 65. This project is aims to create a Model using Computer Vision algorithms to detect a visual signal for pneumonia from medical images given as input. The algorithm should provide marker for Lung opacities on the Xray images. The infection in lungs can be in more than one location and algorithm should detect and provide marker for all the inflammation.

Data

Following files are shared for this project,



Shared with me › CV capstone

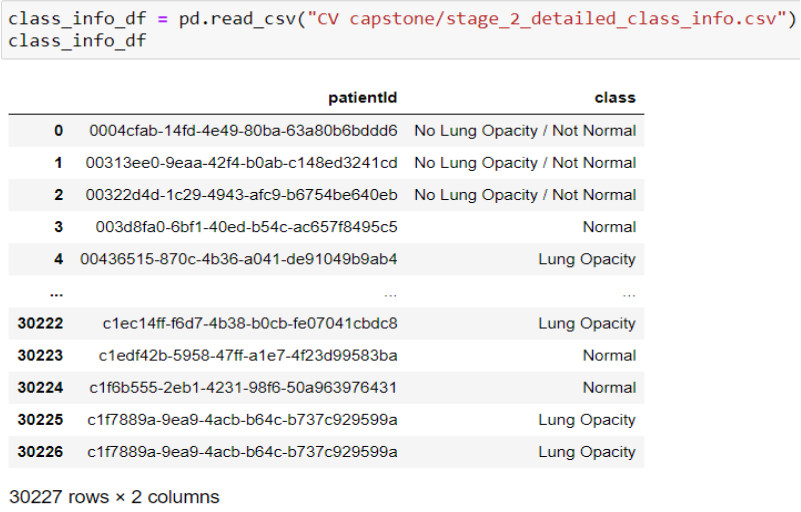
Name	Owner	Last modified	File size
GCP Credits Request Link - RSNA.txt	Harish S	Dec 11, 2019 Harish S	55 bytes
stage_2_detailed_class_info.csv	Harish S	Dec 11, 2019 Harish S	1.6 MB
stage_2_sample_submission.csv	Harish S	Dec 11, 2019 Harish S	155 KB
stage_2_test_images.zip	Harish S	Sep 2, 2022 Harish S	378.8 MB
stage_2_train_images.zip	Harish S	Sep 2, 2022 Harish S	3.3 GB
stage_2_train_labels.csv	Harish S	Dec 11, 2019 Harish S	1.4 MB

1. GCP Credits Request Link - RSNA.txt: The credit file which we don't need to process in the project. It is to give credit to the author of this data.
2. stage_2_detailed_class_info.csv: CSV file having patientid and corresponding class of the disease.
3. stage_2_sample_submission.csv: CSV file which has patientid and predictionstring which is a constant value shown for example. This file may not be required any processing.
4. stage_2_test_images.zip: Zip file containing test images of type DICOM.
5. stage_2_train_images.zip: Zip file containing list of DICOM images which we can use for model training
6. stage_2_train_labels.csv: The CSV File having patientid, coordinates(x, y, width, height) and Target. The target is 0 if there are no coordinates. The Target is 1 if there is a coordinates available.

Summary of Pre-processing, EDA and Findings

Findings:

CSV Files



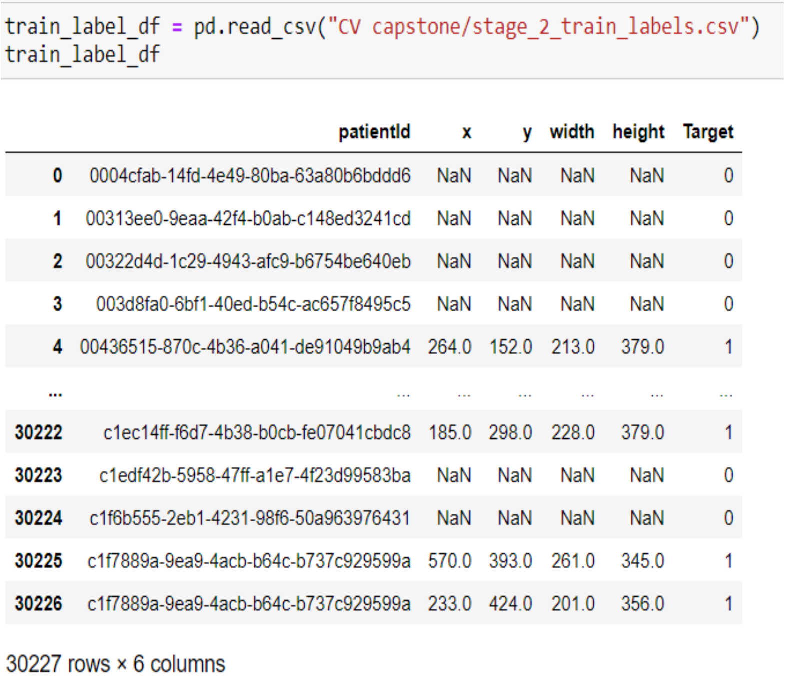
```
class_info_df = pd.read_csv("CV capstone/stage_2_detailed_class_info.csv")
class_info_df
```

	patientId	class
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	No Lung Opacity / Not Normal
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	No Lung Opacity / Not Normal
2	00322d4d-1c29-4943-afc9-b6754be640eb	No Lung Opacity / Not Normal
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	Normal
4	00436515-870c-4b36-a041-de91049b9ab4	Lung Opacity
...	...	...
30222	c1ec14ff-f6d7-4b38-b0cb-fe07041cbdc8	Lung Opacity
30223	c1edf42b-5958-47ff-a1e7-4f23d99583ba	Normal
30224	c1f6b555-2eb1-4231-98f6-50a963976431	Normal
30225	c1f7889a-9ea9-4acb-b64c-b737c929599a	Lung Opacity
30226	c1f7889a-9ea9-4acb-b64c-b737c929599a	Lung Opacity

30227 rows × 2 columns

The class info CSV has 30227 records with two column such as patented and class. There are three classes. They are,

1. No Lung Opacity/Not Normal
2. Normal
3. Lung Opacity

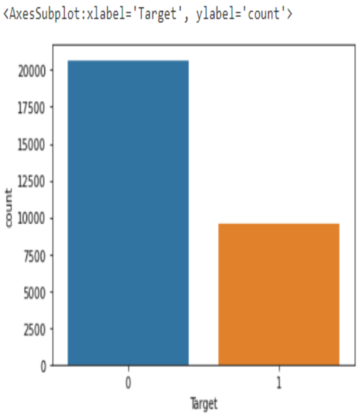


```
train_label_df = pd.read_csv("CV capstone/stage_2_train_labels.csv")
train_label_df
```

	patientId	x	y	width	height	Target
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1
...	...	...	...	...	...	...
30222	c1ec14ff-f6d7-4b38-b0cb-fe07041cbdc8	185.0	298.0	228.0	379.0	1
30223	c1edf42b-5958-47ff-a1e7-4f23d99583ba	NaN	NaN	NaN	NaN	0
30224	c1f6b555-2eb1-4231-98f6-50a963976431	NaN	NaN	NaN	NaN	0
30225	c1f7889a-9ea9-4acb-b64c-b737c929599a	570.0	393.0	261.0	345.0	1
30226	c1f7889a-9ea9-4acb-b64c-b737c929599a	233.0	424.0	201.0	356.0	1

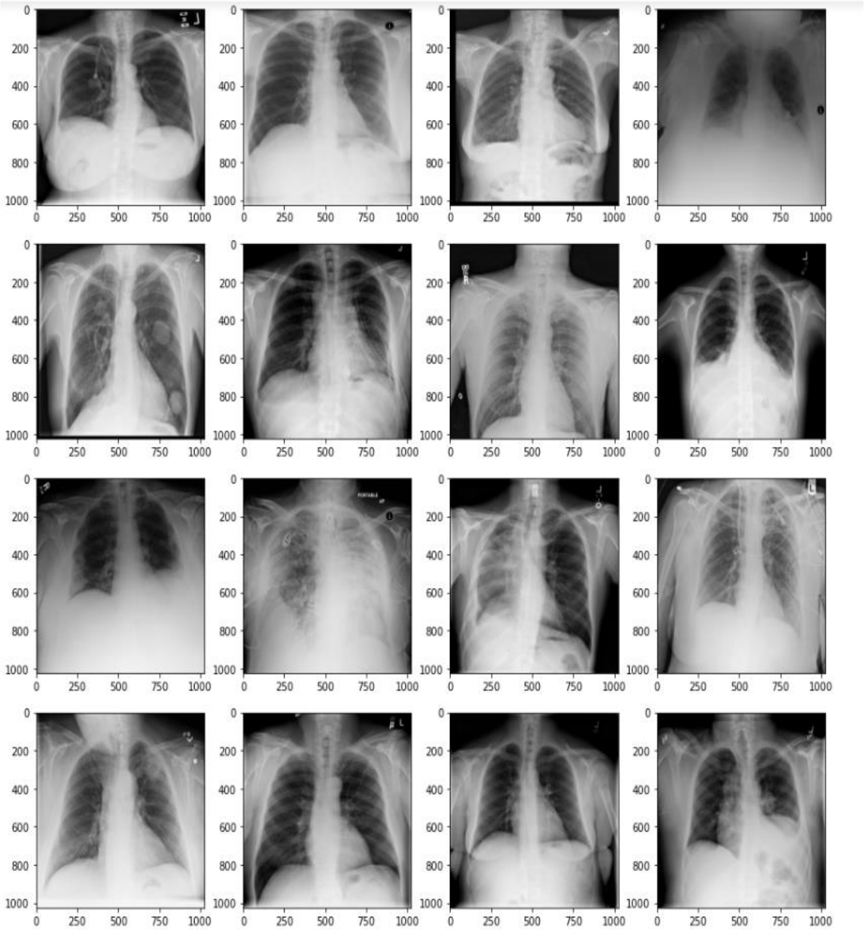
30227 rows × 6 columns

The train label CSV has 30227 records with four coordinate column such as (x, y, width and height) and we have Target Feature and its distribution is,



```
<AxesSubplot:xlabel='Target', ylabel='count'>
```

DATA SET- Training



Randomly picked images zipped inside stage_2_train_images.zip file. These are DCM images which needs special library such as pydicom to process. We should install them as it won't come by default. The images can be read and displayed like below,

```
img = dicom.dcmread(img.dcm)

plt.imshow(img)
```

NOTE: We have 26684 image and individual files are named as patiendid.<dcm>. We will need to pre-process these files as we have more label and class info from CSV, hence remove duplicate if any.

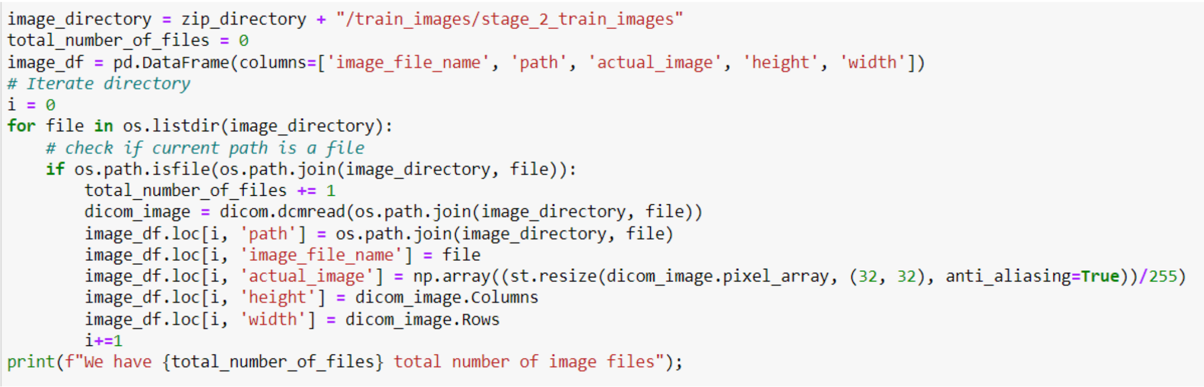
DATA SET- Testing

- We have got 3K images and there is no label and class information details available as they are pure test image set.

Pre-processing

We can load images into pandas dataframe for further processing. While loading we can create attribute of the images such as patentienid, image width, height, filename, path and the actual content after resizing images to 32 * 32.

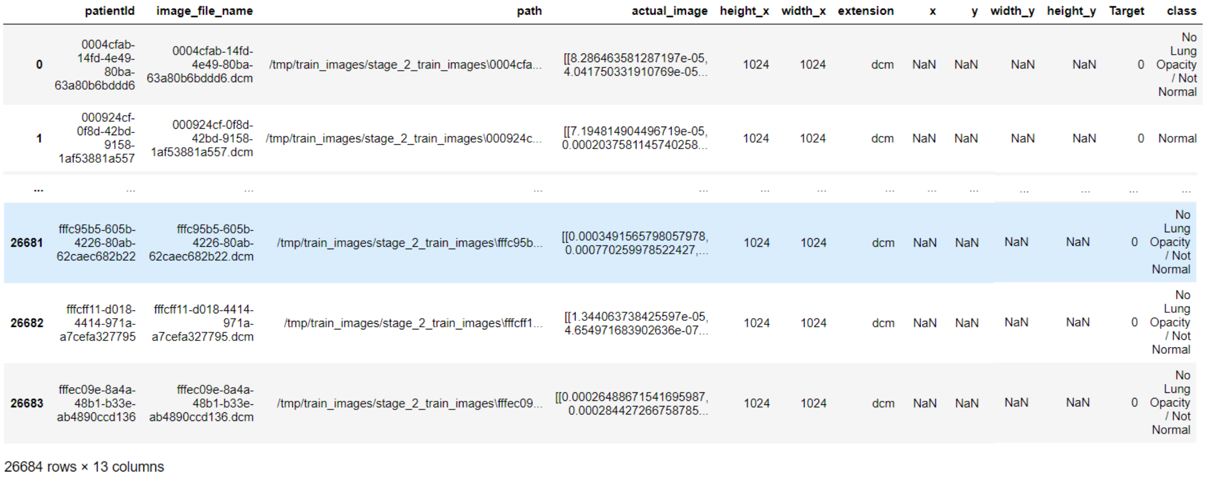
Following line of code can help get the image dataframe.



```
image_directory = zip_directory + "/train_images/stage_2_train_images"
total_number_of_files = 0
image_df = pd.DataFrame(columns=['image_file_name', 'path', 'actual_image', 'height', 'width'])
# Iterate directory
i = 0
for file in os.listdir(image_directory):
    # check if current path is a file
    if os.path.isfile(os.path.join(image_directory, file)):
        total_number_of_files += 1
        dicom_image = dicom.dcmread(os.path.join(image_directory, file))
        image_df.loc[i, 'path'] = os.path.join(image_directory, file)
        image_df.loc[i, 'image_file_name'] = file
        image_df.loc[i, 'actual_image'] = np.array((st.resize(dicom_image.pixel_array, (32, 32), anti_aliasing=True))/255)
        image_df.loc[i, 'height'] = dicom_image.Columns
        image_df.loc[i, 'width'] = dicom_image.Rows
        i+=1
print(f"We have {total_number_of_files} total number of image files");
```

Merging Dataframe and remove duplicates

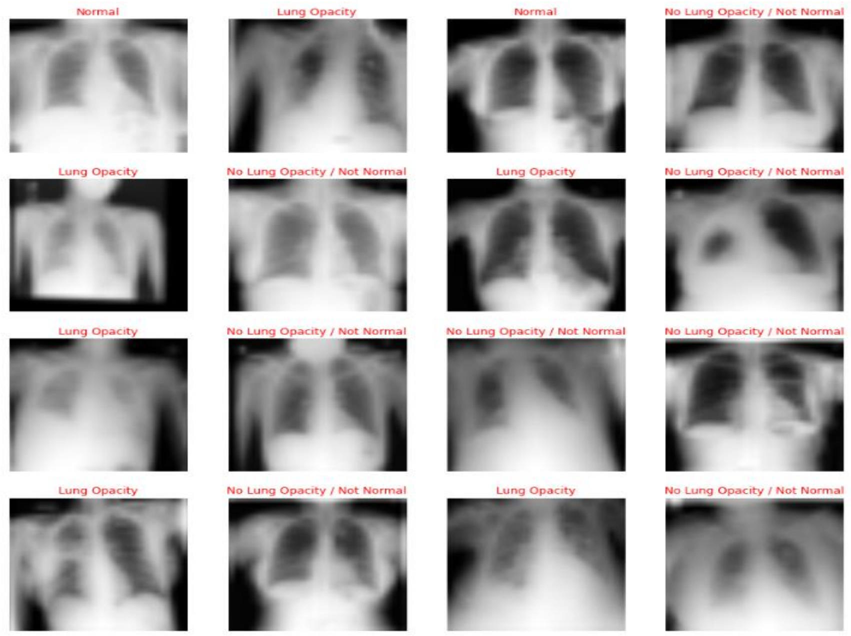
We can merge image, class and label dataframe to remove duplicate using patientid as common column. After successfully merged three dataframe we are getting 26684 records of image data which will something like below,



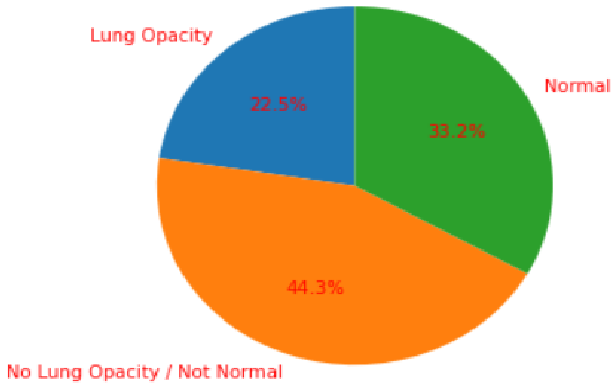
	patientId	image_file_name	path	actual_image	height_x	width_x	extension	x	y	width_y	height_y	Target	class
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	0004cfab-14fd-4e49-80ba-63a80b6bddd6.dcm	/tmp/train_images/stage_2_train_images\0004cfa...	[[8.286463581287197e-05, 4.041750331910769e-05...	1024	1024	dcm	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
1	000924cf-0f8d-42bd-9158-1af53881a557	000924cf-0f8d-42bd-9158-1af53881a557.dcm	/tmp/train_images/stage_2_train_images\000924c...	[[7.194814904496719e-05, 0.0002037581145740258...	1024	1024	dcm	NaN	NaN	NaN	NaN	0	Normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...
26681	fffc95b5-605b-4226-80ab-62caec682b22	fffc95b5-605b-4226-80ab-62caec682b22.dcm	/tmp/train_images/stage_2_train_images\fffc95b...	[[0.0003491565798057978, 0.00077025997852427,...	1024	1024	dcm	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
26682	fffcff11-d018-4414-971a-a7cefa327795	fffcff11-d018-4414-971a-a7cefa327795.dcm	/tmp/train_images/stage_2_train_images\fffcff1...	[[1.344063738425597e-05, 4.654971683902636e-07...	1024	1024	dcm	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
26683	fffec09e-8a4a-48b1-b33e-ab4890ccd136	fffec09e-8a4a-48b1-b33e-ab4890ccd136.dcm	/tmp/train_images/stage_2_train_images\fffec09...	[[0.00026488671541695987, 0.000284427266758785...	1024	1024	dcm	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal

26684 rows × 13 columns

Print random images from merged dataframe along with its class


Normal
Lung Opacity
Normal
No Lung Opacity / Not Normal
Lung Opacity
No Lung Opacity / Not Normal
Lung Opacity
No Lung Opacity / Not Normal
Lung Opacity
No Lung Opacity / Not Normal
No Lung Opacity / Not Normal
No Lung Opacity / Not Normal
Lung Opacity
No Lung Opacity / Not Normal
Lung Opacity
No Lung Opacity / Not Normal

Data balancing



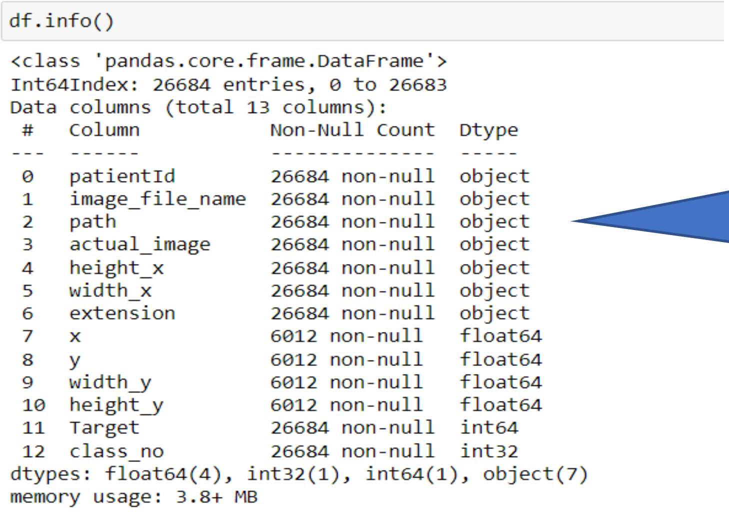
The class is not perfectly balanced there is a slight imbalance. We will address this data unbalancing in the second part of this project

Encoding

The machine learning or deep learning algorithm requires numbers hence we should convert the class into numbers. We can use LabelEncoder from sklearn preprocessing library like below,

```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'class'.
data['class_no']= label_encoder.fit_transform(data['class'])
df = data.drop(labels='class', axis=1)
df
```

EDA

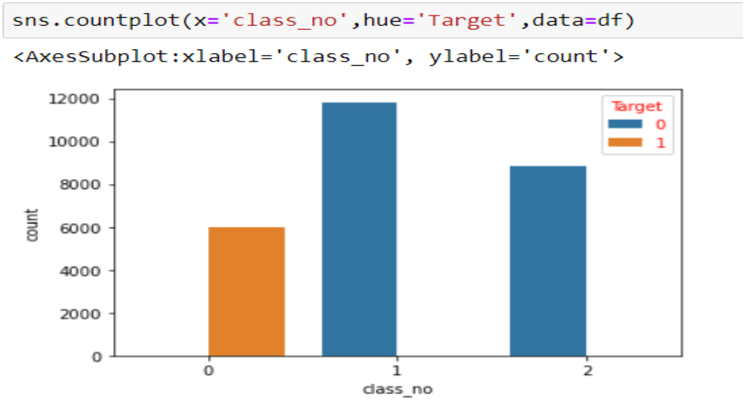


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26684 entries, 0 to 26683
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   patientId        26684 non-null  object 
 1   image_file_name  26684 non-null  object 
 2   path             26684 non-null  object 
 3   actual_image     26684 non-null  object 
 4   height_x         26684 non-null  object 
 5   width_x          26684 non-null  object 
 6   extension        26684 non-null  object 
 7   x                6012 non-null   float64
 8   y                6012 non-null   float64
 9   width_y          6012 non-null   float64
 10  height_y         6012 non-null   float64
 11  Target           26684 non-null  int64  
 12  class_no         26684 non-null  int32  
dtypes: float64(4), int32(1), int64(1), object(7)
memory usage: 3.8+ MB
```

The dataframe that we have after preprocessing and ready to do EDA. The Target and class_no is a category type

Understand Target and Class

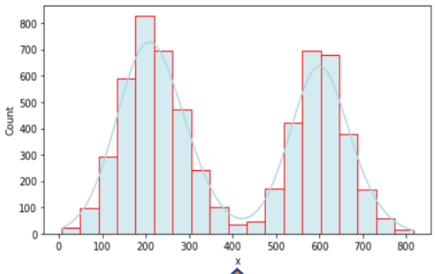


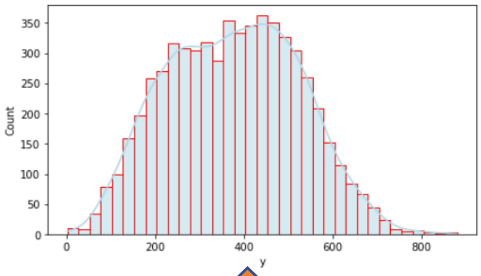
```
sns.countplot(x='class_no',hue='Target',data=df)
```

```
<AxesSubplot:xlabel='class_no', ylabel='count'>
```

There are 6k records having target as 1 remaining records are set to zero. The zero is nothing but “Lung Opacity” class.

Insights on coordinates



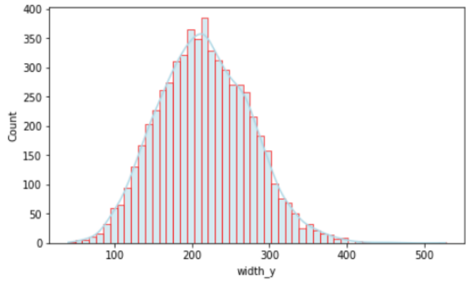


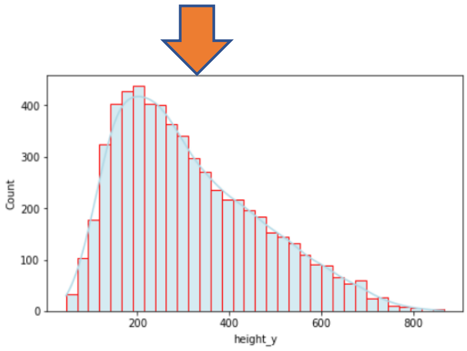
X is range from 0 to 800. There are lot of values are range from 150 to 300 and 550 to 700. We have less count on other ranges

Y is range from 0 to 800 with some extremes. It has lot of records value range from 5o to 450

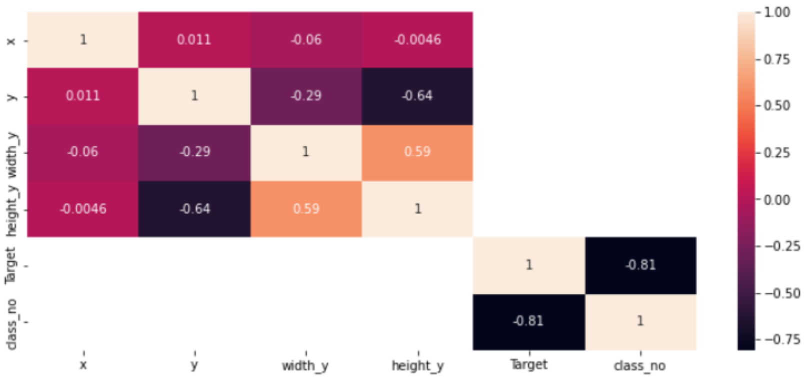
Width_y range from 50 to 400 with few extremes on both side.

Height_y is right skewed where we have lot of records having value greater than 200



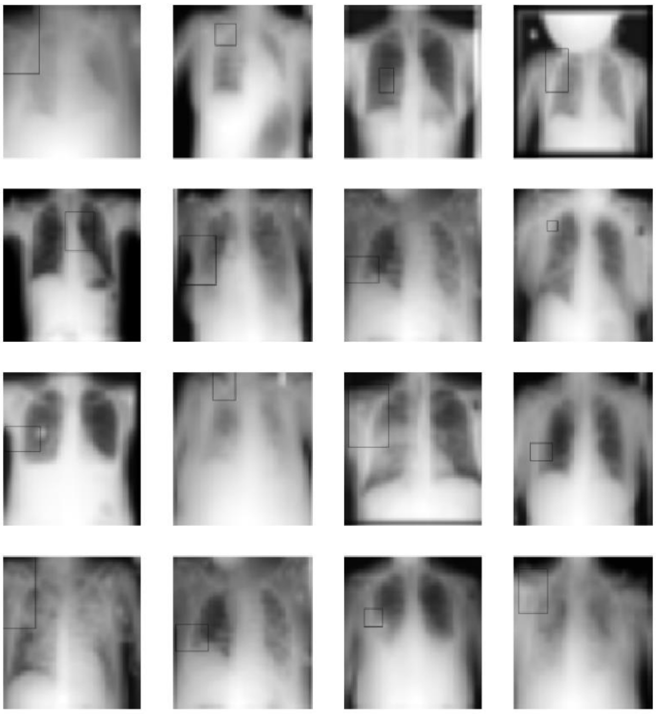


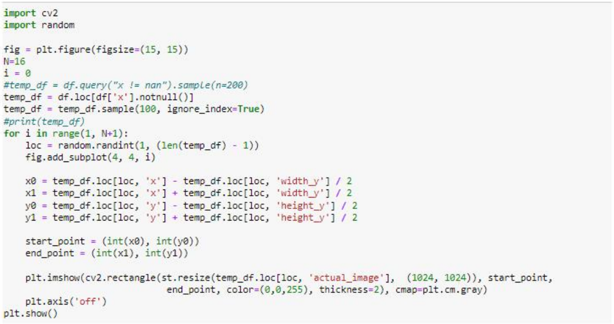
Correlations



We don't have much coorrelation between variables however average coorelation can be seen between Width_y vs Height_y and negative average coorrelation can be seen between Y and Height_y

Visualization of images with coordinates





```
import cv2
import random

fig = plt.figure(figsize=(15, 15))
N=16
i = 0
#temp_df = df.query("x != nan").sample(n=200)
temp_df = df.loc[df['x'].notnull()]
temp_df = temp_df.sample(100, ignore_index=True)
#print(temp_df)
for i in range(1, N+1):
    loc = random.randint(1, (len(temp_df) - 1))
    fig.add_subplot(4, 4, i)

    x0 = temp_df.loc[loc, 'x'] - temp_df.loc[loc, 'width_y'] / 2
    x1 = temp_df.loc[loc, 'x'] + temp_df.loc[loc, 'width_y'] / 2
    y0 = temp_df.loc[loc, 'y'] - temp_df.loc[loc, 'height_y'] / 2
    y1 = temp_df.loc[loc, 'y'] + temp_df.loc[loc, 'height_y'] / 2

    start_point = (int(x0), int(y0))
    end_point = (int(x1), int(y1))

    plt.imshow(cv2.rectangle(st.resize(temp_df.loc[loc, 'actual_image'],  (1024, 1024)), start_point,
                             end_point, color=(0,0,255), thickness=2), cmap=plt.cm.gray)
    plt.axis('off')
plt.show()
```

The above code iterate over the dataframe, reads image array and perform coordinate marking. From the random image, many coordinates comes under right side of the chest xray with very less number of coordinates comes under patient left side.

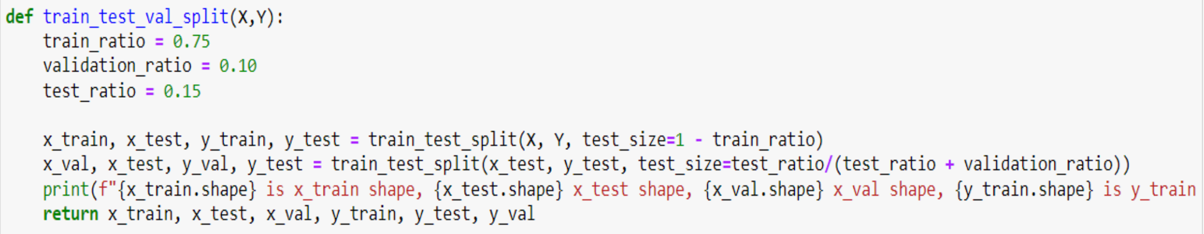
Summary

We have merged training images with class and training label CSV file. The Analysis suggest that we have 26684 images with 6K records having coordinates and remaining with out coordinates. Since we have images, we should CNN algorithm. We have three classes hence we should use Softmax in the output layer. We will build basic classification model in the first milestone, perform testing and then we will apply down sampling, image augmentation to adjust class balancing, additionally we will use transfer learning technique, Faster RCNN and Mask RCNN in the next phase.

Milestone – 1 Basic CNN

Train Test and Validation split

In order for us to evaluate the model more accurately we will have train, test and validation split. This practise is more nuance. In order to do this split we use following function,

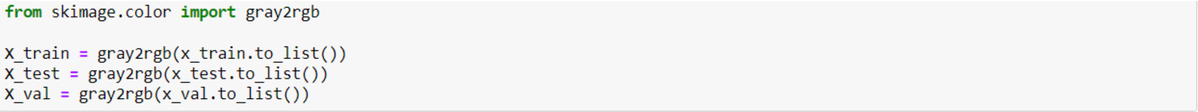


```
def train_test_val_split(X,Y):
    train_ratio = 0.75
    validation_ratio = 0.10
    test_ratio = 0.15

    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=1 - train_ratio)
    x_val, x_test, y_val, y_test = train_test_split(x_test, y_test, test_size=test_ratio/(test_ratio + validation_ratio))
    print(f"{x_train.shape} is x_train shape, {x_test.shape} x_test shape, {x_val.shape} x_val shape, {y_train.shape} is y_train
    return x_train, x_test, x_val, y_train, y_test, y_val
```

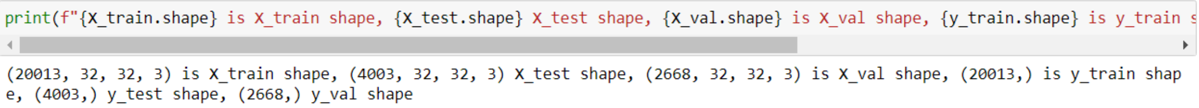
We have our X stored in Dataframe as ‘actual_image’ and Y as “class_no”. Take them out appropriately and split them up.

Convert to tensor

The CNN models will take image of size (N,width,height,RGB). We need to convert our actual_image array to tensor and following function can help you achieve that.



```
from skimage.color import gray2rgb

X_train = gray2rgb(x_train.to_list())
X_test = gray2rgb(x_test.to_list())
X_val = gray2rgb(x_val.to_list())
```



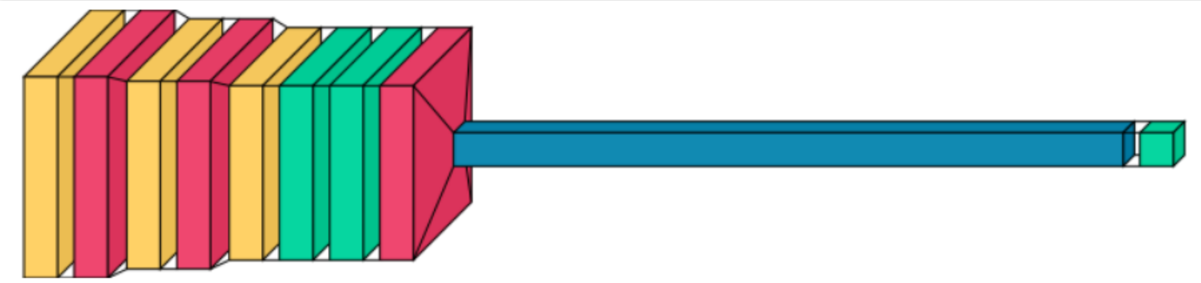
```
print(f"{X_train.shape} is X_train shape, {X_test.shape} X_test shape, {X_val.shape} is X_val shape, {y_train.shape} is y_train s
```

```
(20013, 32, 32, 3) is X_train shape, (4003, 32, 32, 3) X_test shape, (2668, 32, 32, 3) is X_val shape, (20013,) is y_train shap
e, (4003,) y_test shape, (2668,) y_val shape
```

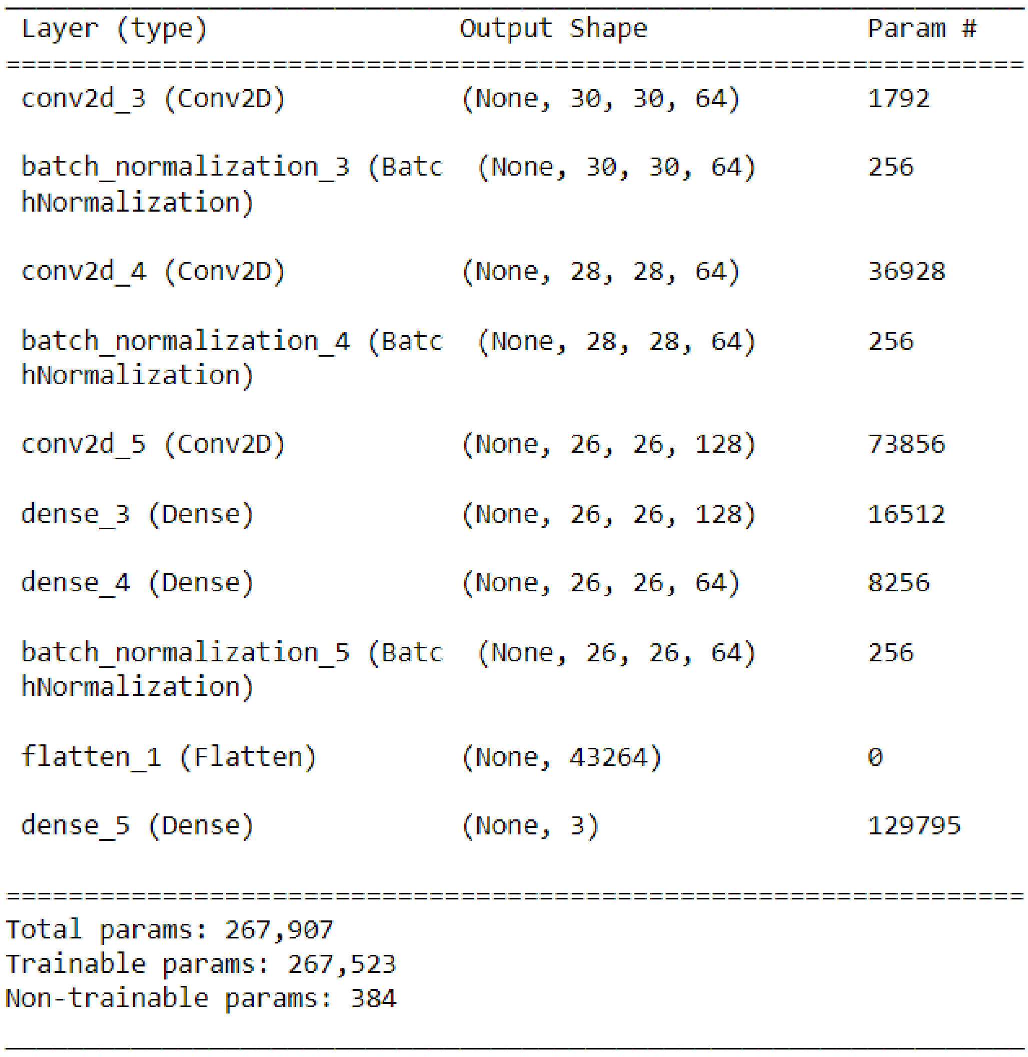
Basic CNN Model

Architecture

3 Conv Layer, 3 Batch Normalization, 2 Dense, 1 Flatten and Output layer with softmax.

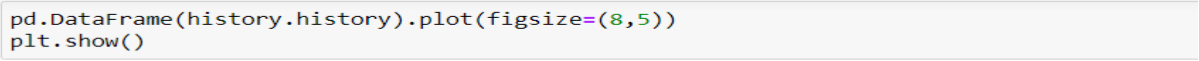


Trainable Parameters

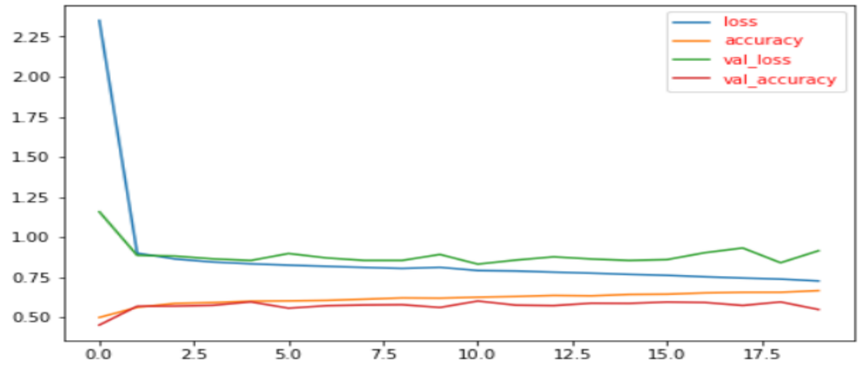


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 64)        1792

 batch_normalization_3 (Batc  (None, 30, 30, 64)       256
 hNormalization)

 conv2d_4 (Conv2D)           (None, 28, 28, 64)        36928

 batch_normalization_4 (Batc  (None, 28, 28, 64)       256
 hNormalization)

 conv2d_5 (Conv2D)           (None, 26, 26, 128)       73856

 dense_3 (Dense)             (None, 26, 26, 128)       16512

 dense_4 (Dense)             (None, 26, 26, 64)        8256

 batch_normalization_5 (Batc  (None, 26, 26, 64)       256
 hNormalization)

 flatten_1 (Flatten)         (None, 43264)             0

 dense_5 (Dense)             (None, 3)                 129795

=================================================================
Total params: 267,907
Trainable params: 267,523
Non-trainable params: 384
_________________________________________________________________
```

Accuracy and Loss



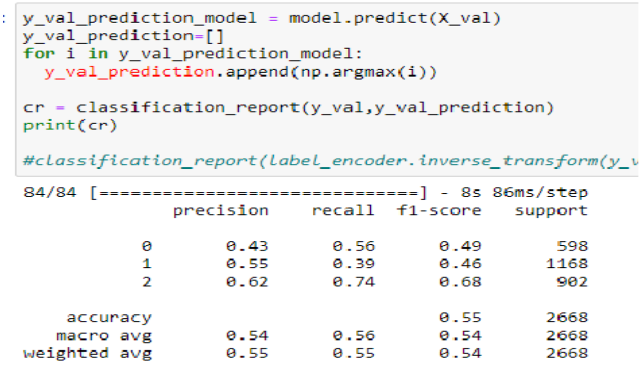
```
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.show()
```



The accuracy of training is showing small improvements and testing was little going up and down.

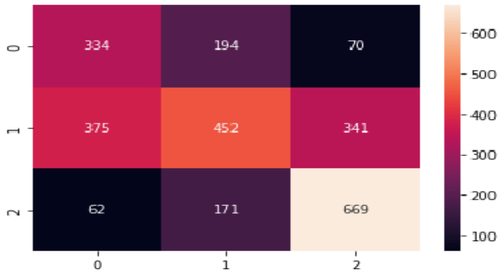
Accuracy and Recall using dedicated

We have a validation data available with use which we can use to run against the model to accurately understand the accuracy, recall and precision. We can do this using following code,

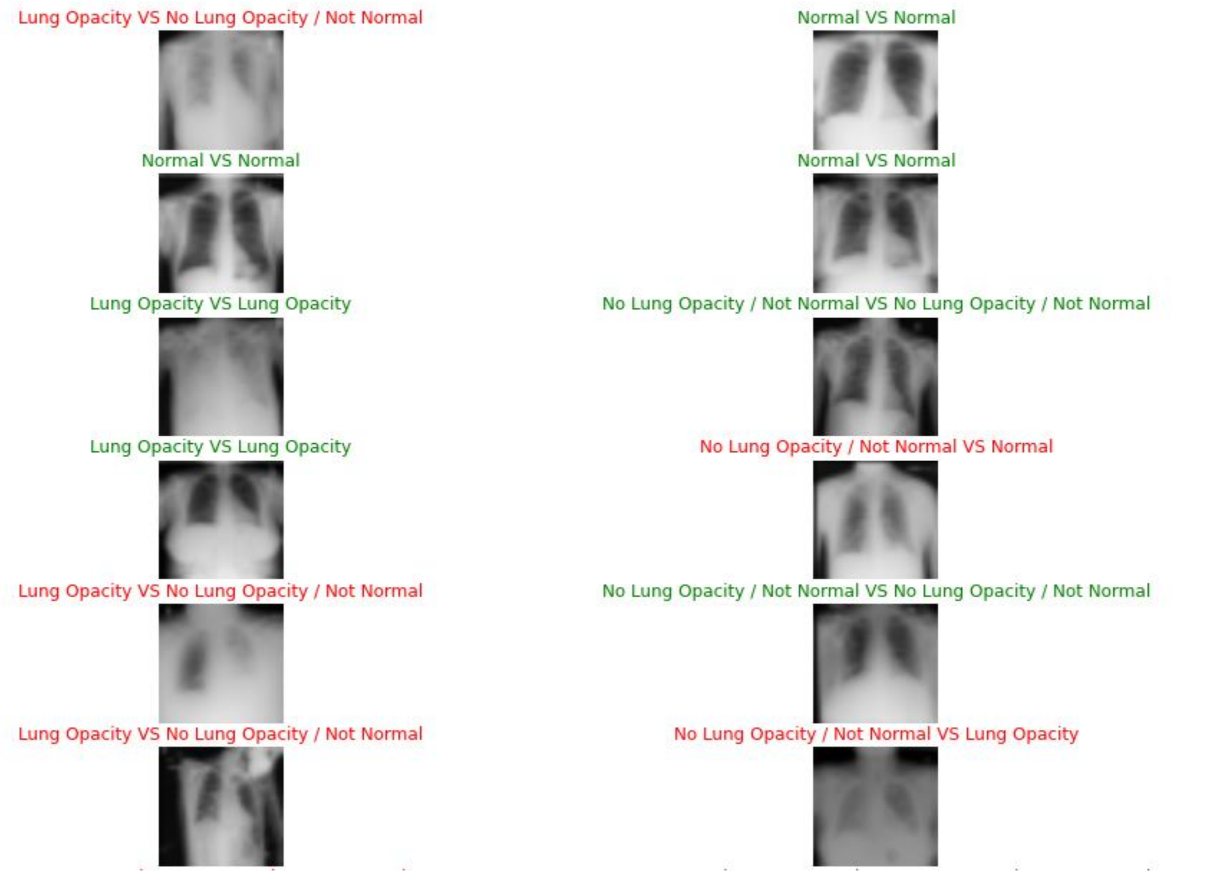


```
y_val_prediction_model = model.predict(X_val)
y_val_prediction=[]
for i in y_val_prediction_model:
  y_val_prediction.append(np.argmax(i))

cr = classification_report(y_val,y_val_prediction)
print(cr)

#classification_report(label_encoder.inverse_transform(y_v
```

```
84/84 [==============================] - 8s 86ms/step
              precision    recall  f1-score   support

           0       0.43      0.56      0.49       598
           1       0.55      0.39      0.46      1168
           2       0.62      0.74      0.68       902

    accuracy                           0.55      2668
   macro avg       0.54      0.56      0.54      2668
weighted avg       0.55      0.55      0.54      2668
```



We can also get actual classification vs predicted classification like below which shows lot of images are misclassified to different class (Red color)


Lung Opacity VS No Lung Opacity / Not Normal
Normal VS Normal
Normal VS Normal
Normal VS Normal
Lung Opacity VS Lung Opacity
No Lung Opacity / Not Normal VS No Lung Opacity / Not Normal
Lung Opacity VS Lung Opacity
No Lung Opacity / Not Normal VS Normal
Lung Opacity VS No Lung Opacity / Not Normal
No Lung Opacity / Not Normal VS No Lung Opacity / Not Normal
Lung Opacity VS No Lung Opacity / Not Normal
No Lung Opacity / Not Normal VS Lung Opacity

Summary and Next step:

The basic model we have got to do classification is giving low recall on predicting lung opacity and for normal class as low as well. We will be doing further optimization for this model like Image Augmentation, Parameter tunning such updating learning rate, trying with Adam Optimizer, transfer learning optimization and trying with more convolution layer, etc in the second phase. Later we will implement Object detection using Faster RCNN and Mask RCNN, etc.