

**FINAL REPORT**

**By**

**CAPSTONE-CV1 Proj-Group2**

Table of Contents

[Background: 3](#_Toc132552311)

[Data 3](#_Toc132552312)

[Summary of Pre-processing, EDA and Findings 4](#_Toc132552313)

[Findings: 4](#_Toc132552314)

[CSV Files 4](#_Toc132552315)

[DATA SET - Training 5](#_Toc132552316)

[DATA SET - Testing 5](#_Toc132552317)

[Pre-processing 6](#_Toc132552318)

[Merging Dataframe and remove duplicates 6](#_Toc132552319)

[Print random images from merged dataframe along with its class 6](#_Toc132552320)

[Data balancing 7](#_Toc132552321)

[Encoding 7](#_Toc132552322)

[EDA 8](#_Toc132552323)

[Understand Target and Class 8](#_Toc132552324)

[Insights on coordinates 9](#_Toc132552325)

[Correlations 9](#_Toc132552326)

[Visualization of images with coordinates 9](#_Toc132552327)

[Analysis Summary and Next Step 10](#_Toc132552328)

[Milestone – 1 Basic CNN 10](#_Toc132552329)

[Train Test and Validation split 10](#_Toc132552330)

[Convert to tensor 11](#_Toc132552331)

[Basic CNN Model 11](#_Toc132552332)

[Architecture 11](#_Toc132552333)

[Trainable Parameters 11](#_Toc132552334)

[Accuracy and Loss 12](#_Toc132552335)

[Accuracy and Recall using dedicated 13](#_Toc132552336)

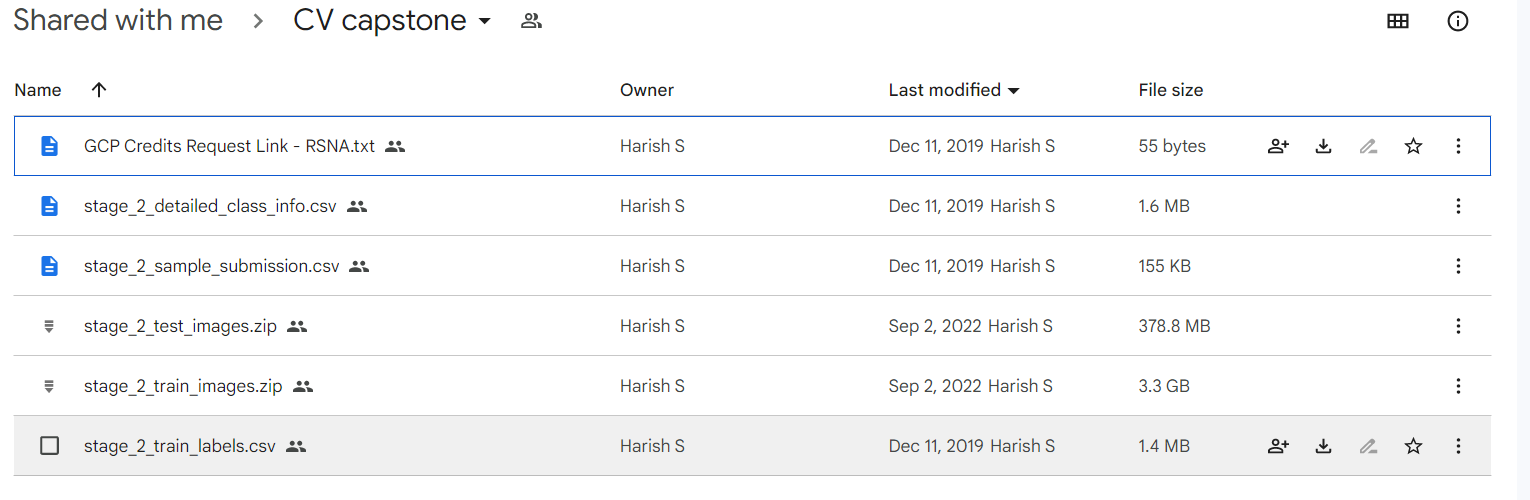
[Summary and Next step: 13](#_Toc132552337)

# 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,

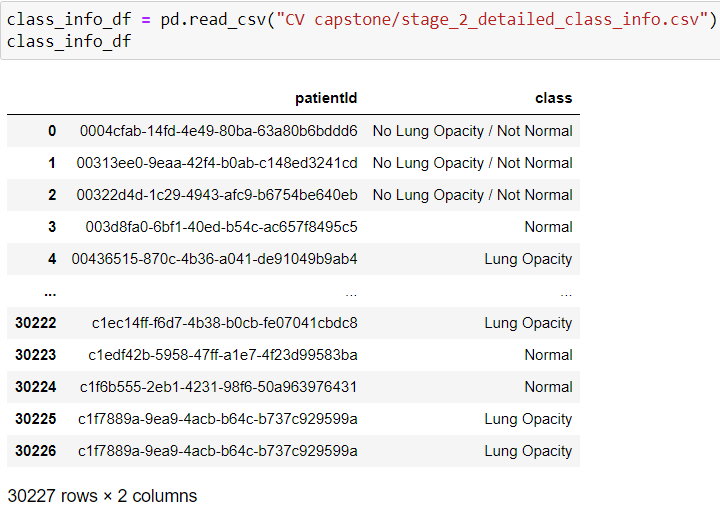


1. GCP Credits Request Link - RSNA.txt: The credit file which we don’t need to process in the project. It is to give credit to the author of this data.
2. stage\_2\_detailed\_class\_info.csv: CSV file having patientid and corresponding class of the disease.
3. stage\_2\_sample\_submission.csv: CSV file 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.

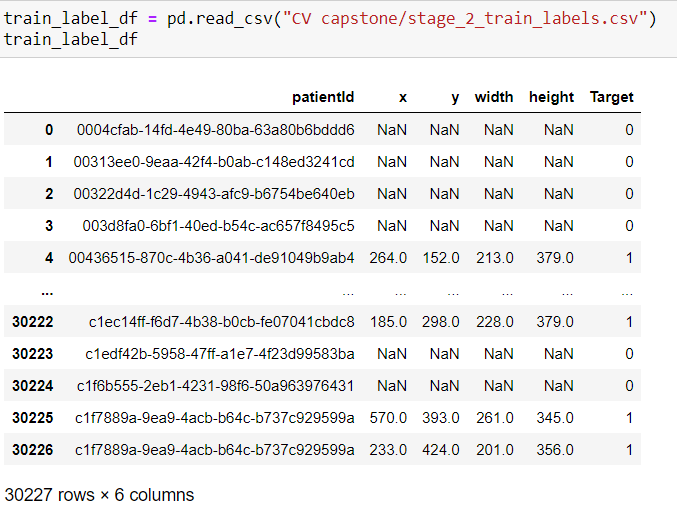
# Summary of Pre-processing, EDA and Findings

# Findings:

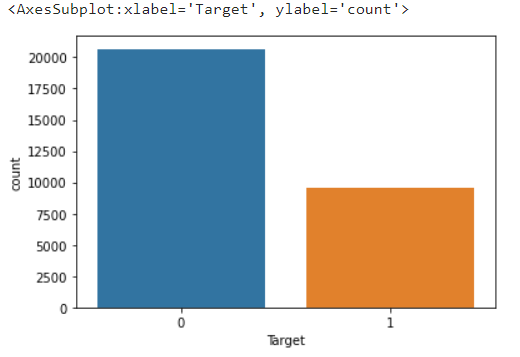
# CSV Files

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

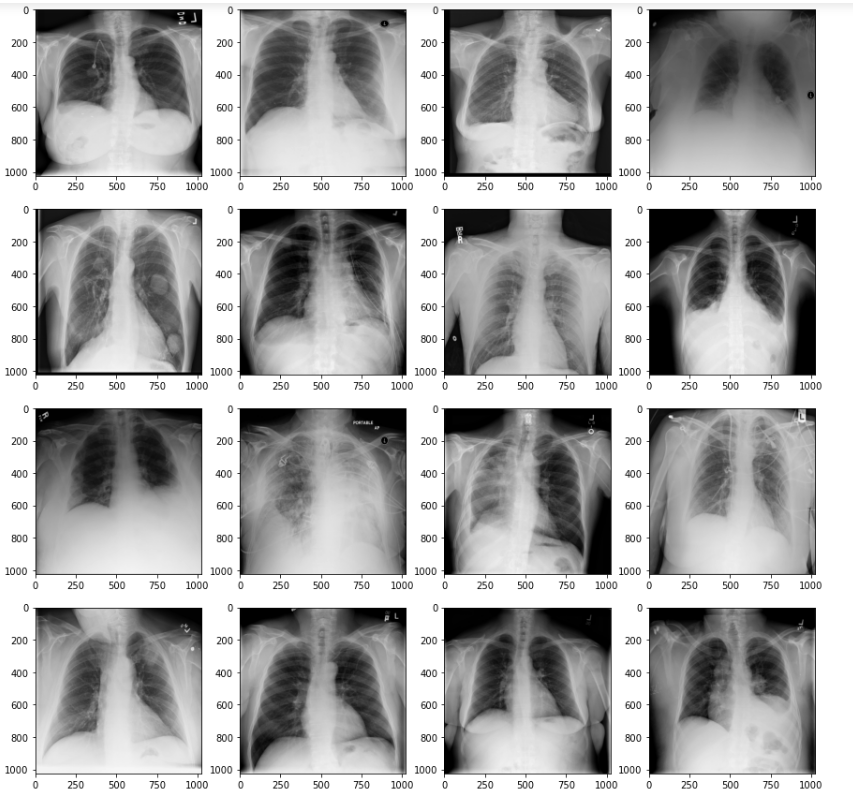
1. No Lung Opacity/Not Normal
2. Normal
3. Lung Opacity



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,



# 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](https://pydicom.github.io/) 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.

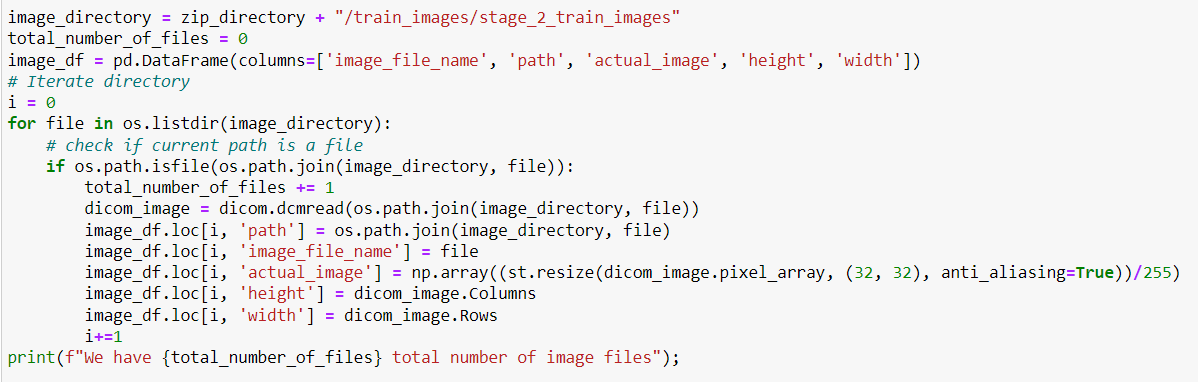
# DATA SET - Testing

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

# Pre-processing

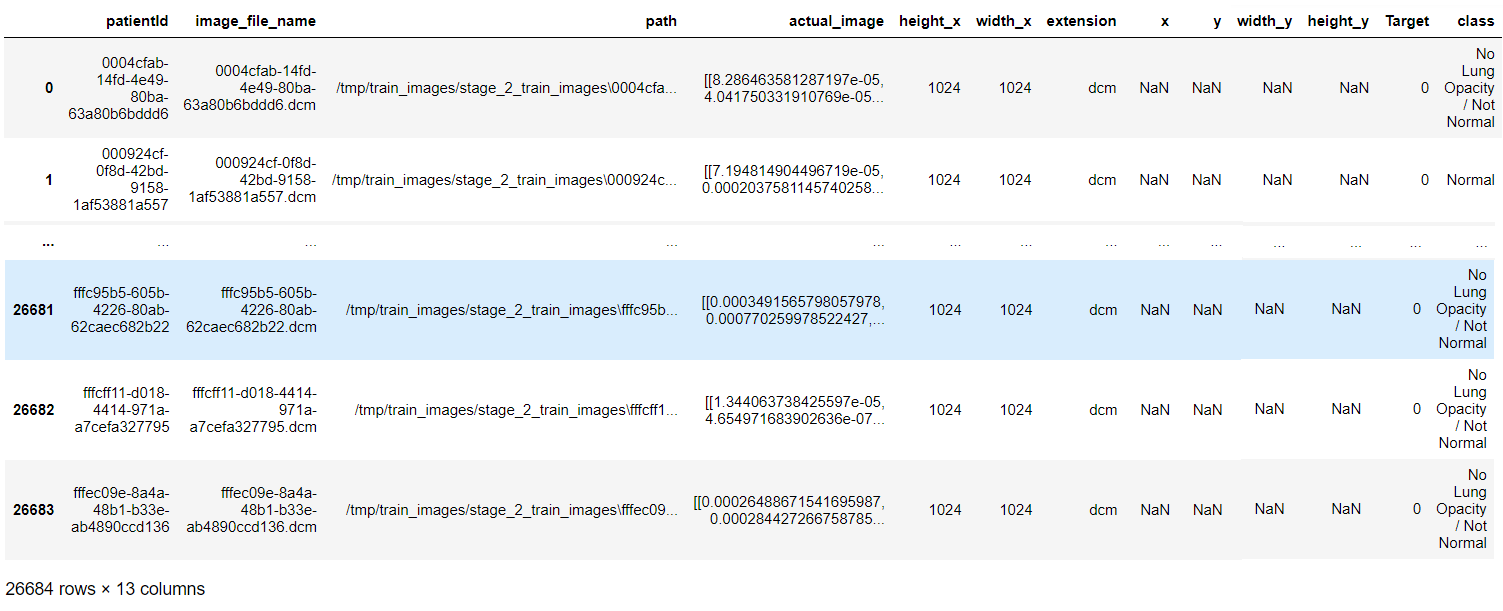
We can load images into pandas dataframe for further processing. While loading we can create attribute of the images such as patentienid, image width, height, filename, path and the actual content after resizing images to 32 \* 32.

Following line of code can help get the image dataframe.

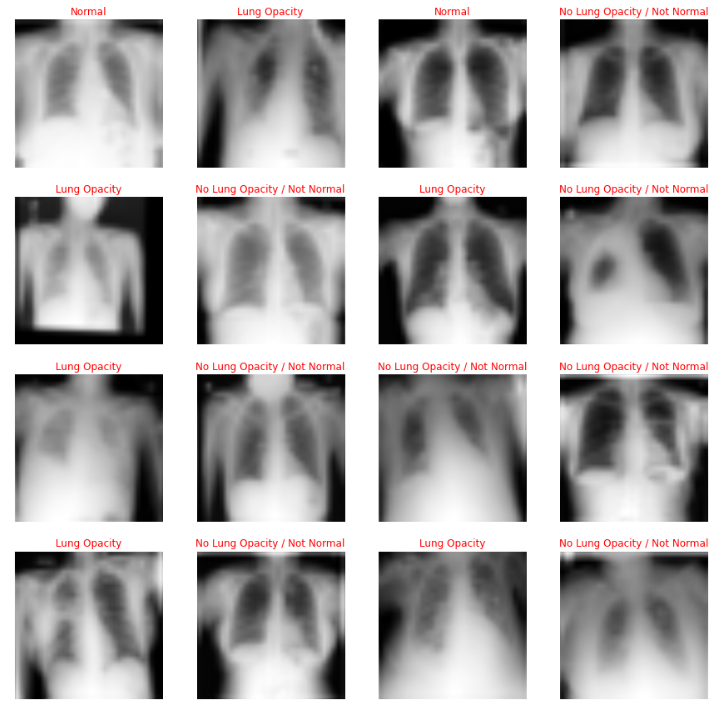


# Merging Dataframe and remove duplicates

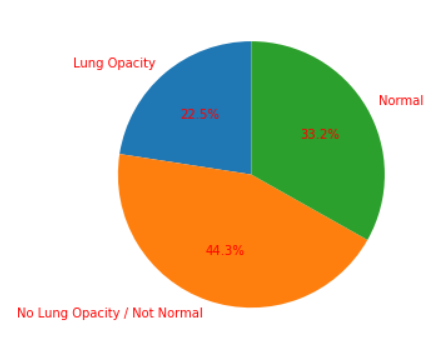
We can merge image, class and label dataframe to remove duplicate using patientid as common column. After successfully merged three dataframe we are getting 26684 records of image data which will something like below,



# Print random images from merged dataframe along with its class

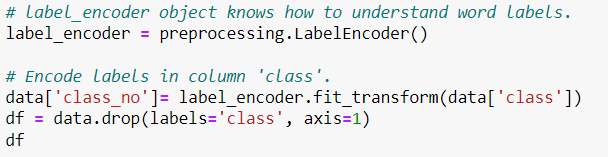


# Data balancing

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

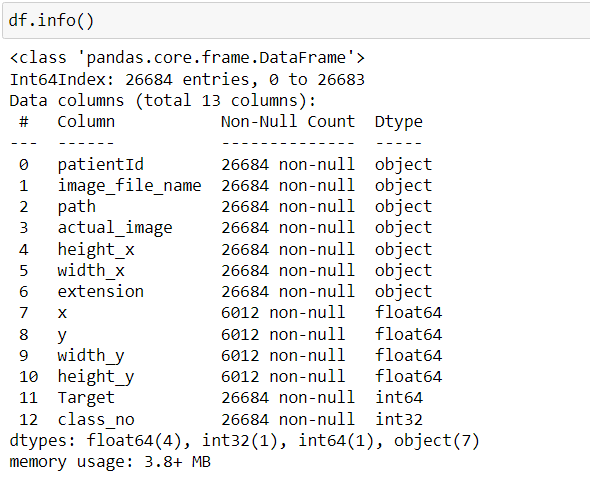
# Encoding

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,



# EDA

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

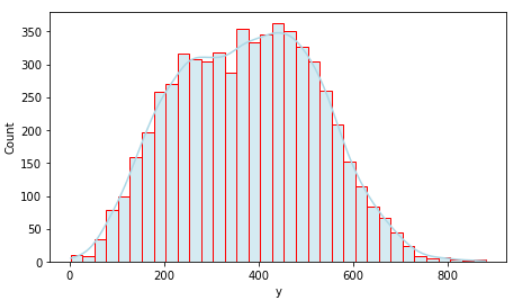
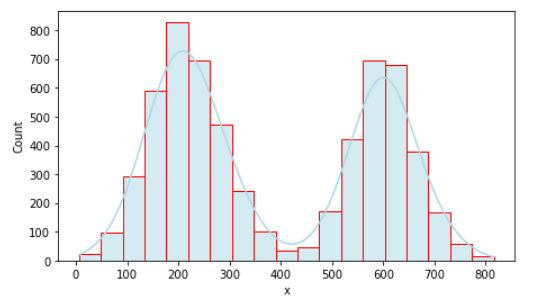


# Understand Target and Class



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

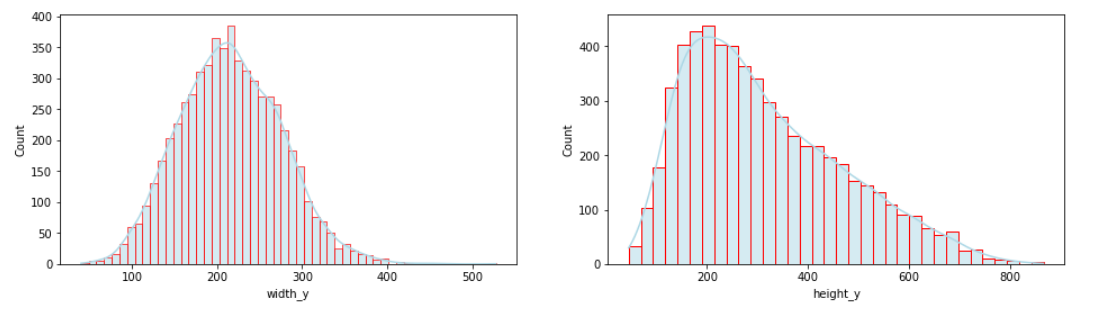


Y is range from 0 to 800 with some extremes. It has lot of records value range from 5o to 450

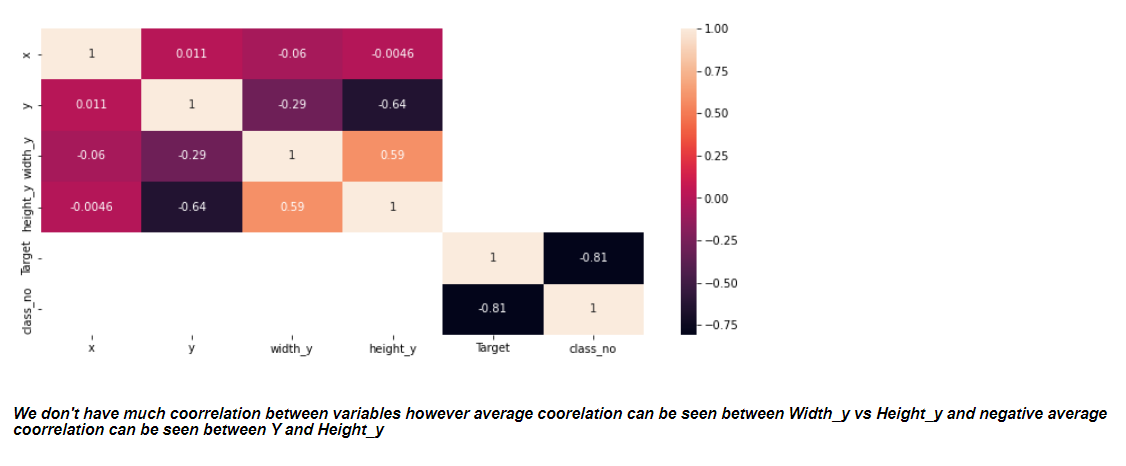
X is range from 0 to 800. There are lot of values are range from 150 to 300 and 550 to 700. We have less count on other ranges

Height\_y is right skewed where we have lot of records having value greater than 200

Width\_y range from 50 to 400 with few extremes on both side.



# Correlations



# Visualization of images with coordinates



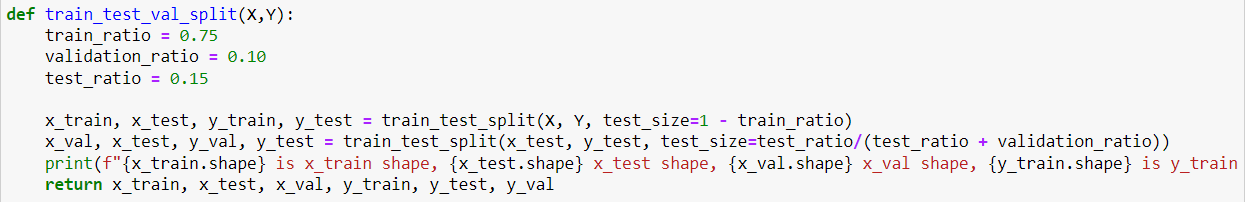
# Analysis Summary and Next Step

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

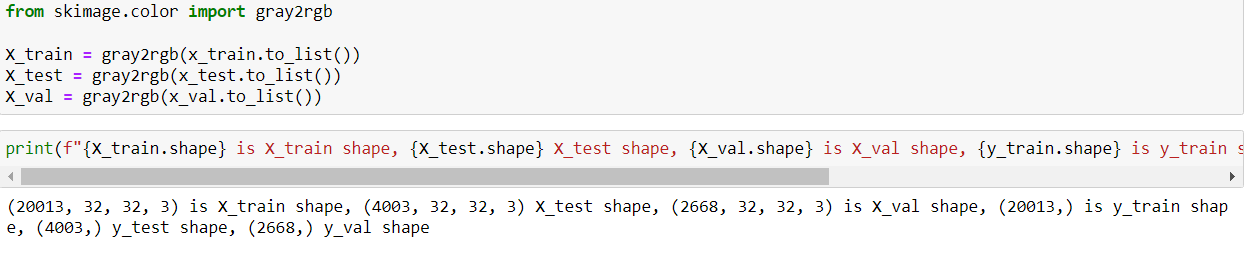
In order for us to evaluate the model more accurately we will have train, test and validation split. This practise is more nuance. In order to do this split we use following function,



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

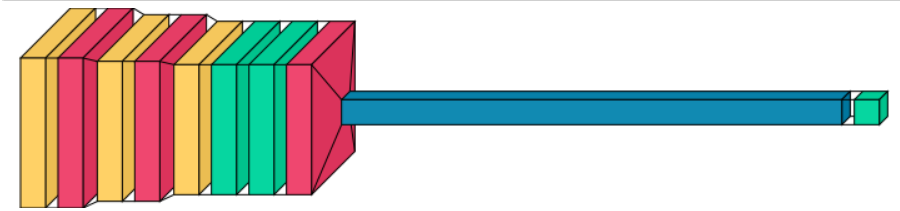
The CNN models will take image of size (N,width,height,RGB). We need to convert our actual\_image array to tensor and following function can help you achieve that.



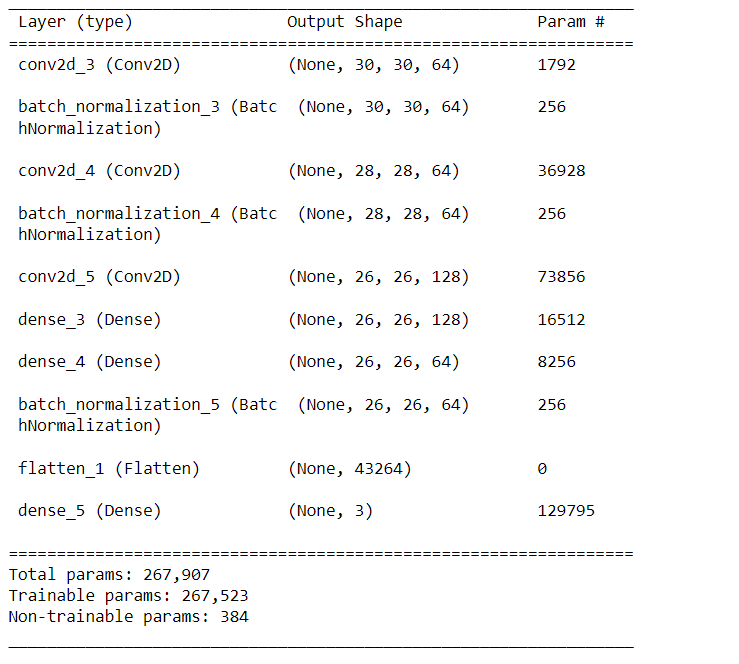
# Basic CNN Model

# Architecture

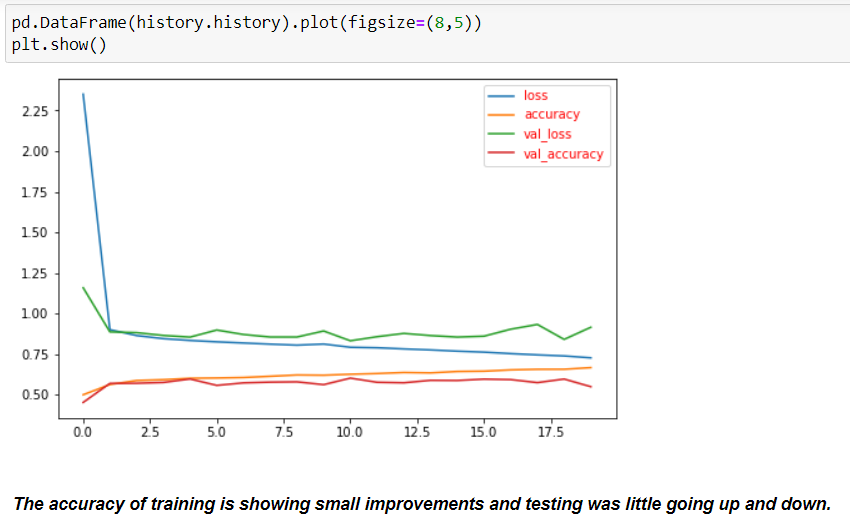
3 Conv Layer, 3 Batch Normalization, 2 Dense, 1 Flatten and Output layer with softmax.



# Trainable Parameters

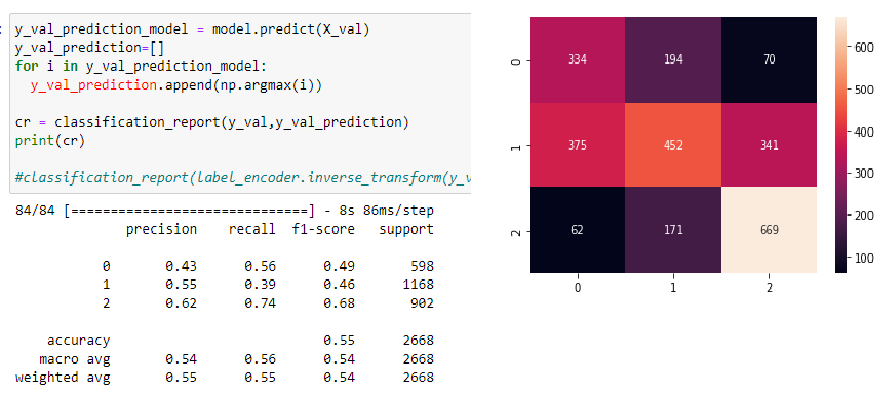


# Accuracy and Loss

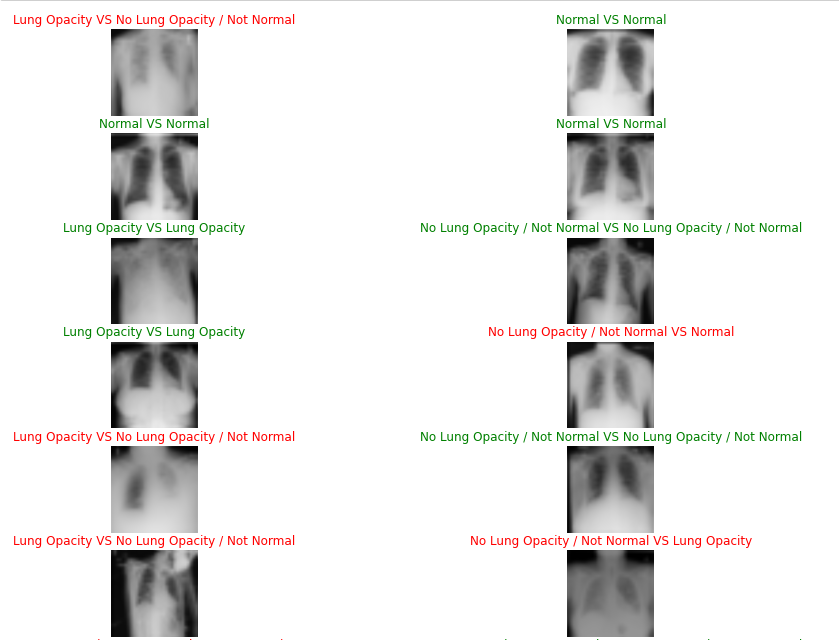


# Accuracy and Recall using dedicated

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,



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



# 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.

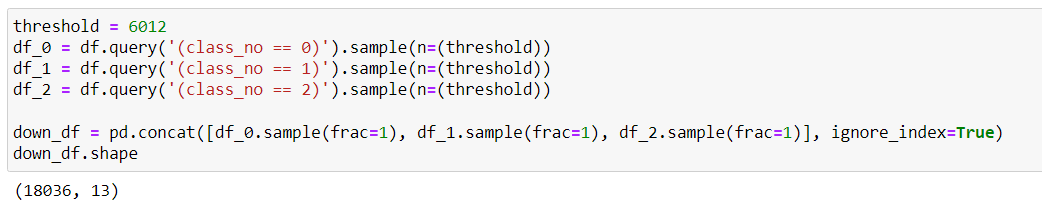
# Fine Tuning Classification Model:

Following are the approach we are going to try to fine tune the model we had to get more accuracy,

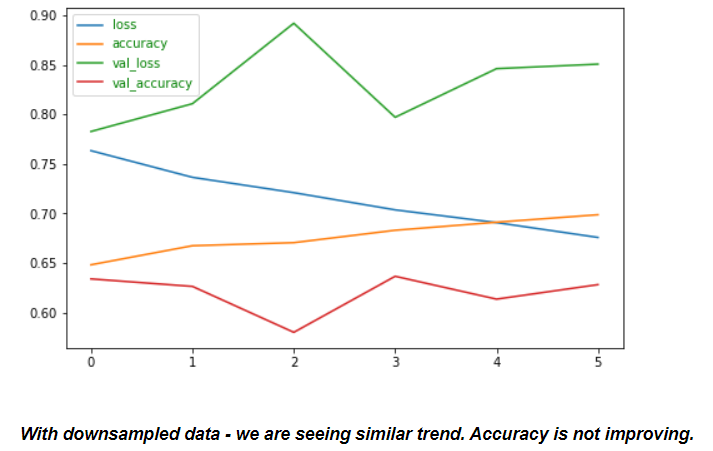
1. Down size the data and apply the CNN model
2. Hyper parameter tunning
3. Apply Following Transfer Learning
   1. VGG16
   2. ResNet50
   3. DenseNet169
   4. MobileNet

# DownSize data:

In order to downsample the data, apply lower denominator threshold like below in each class and use the resulting sample for training and validation

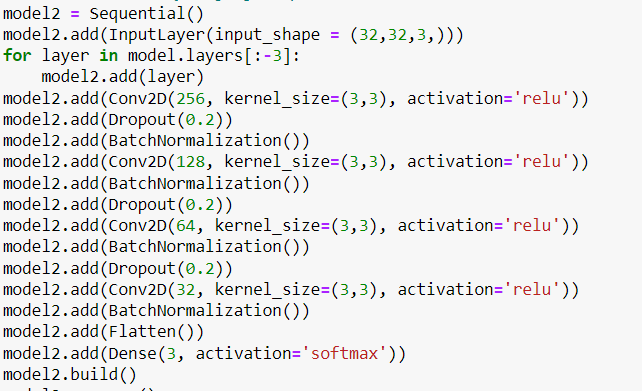


Following is the result of the training using the previous model,

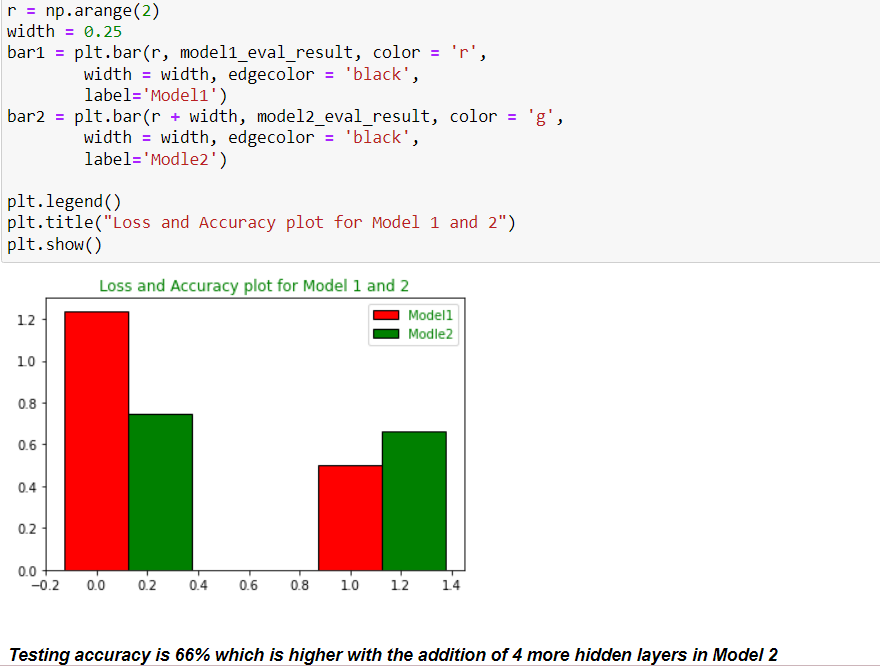


# Increase Hidden Layer

We then added more hidden layer using following code on the previous model,

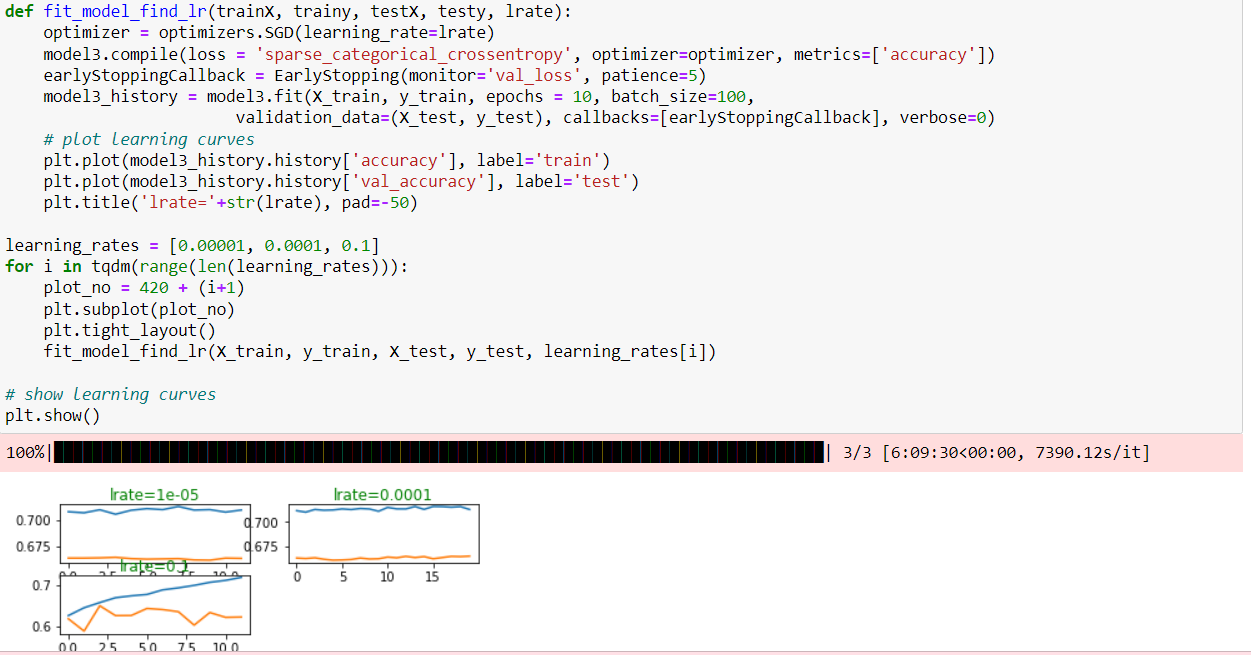


With the additional layers in the existing model when we performed test on the original data we get following result which shows clear improvement of accuracy and reduction of loss from previous model,

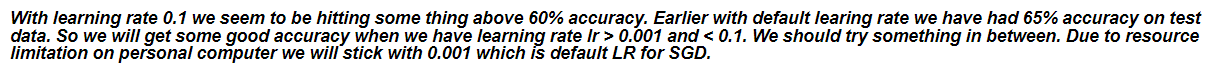


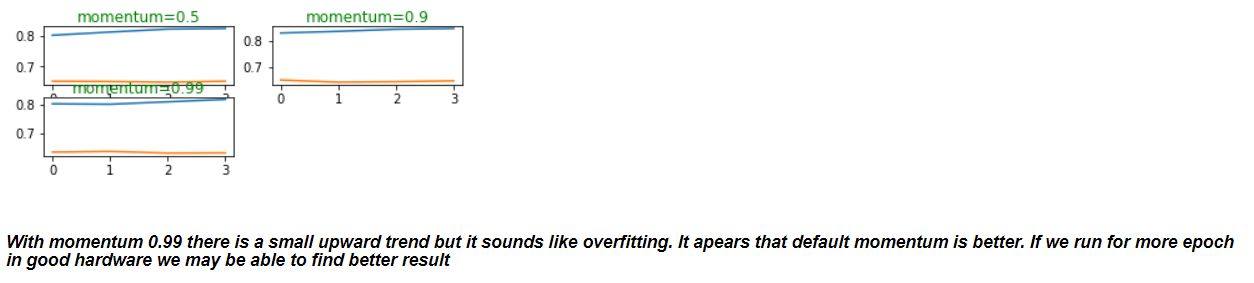
# Hyper Parameter tunning

We then tried to find the better learning rate, optimum and used Adam optimizer using following code but due to hardware/time limitation – we could not run for all possible combinations. With the standard values suggested over internet we did not see improvements.



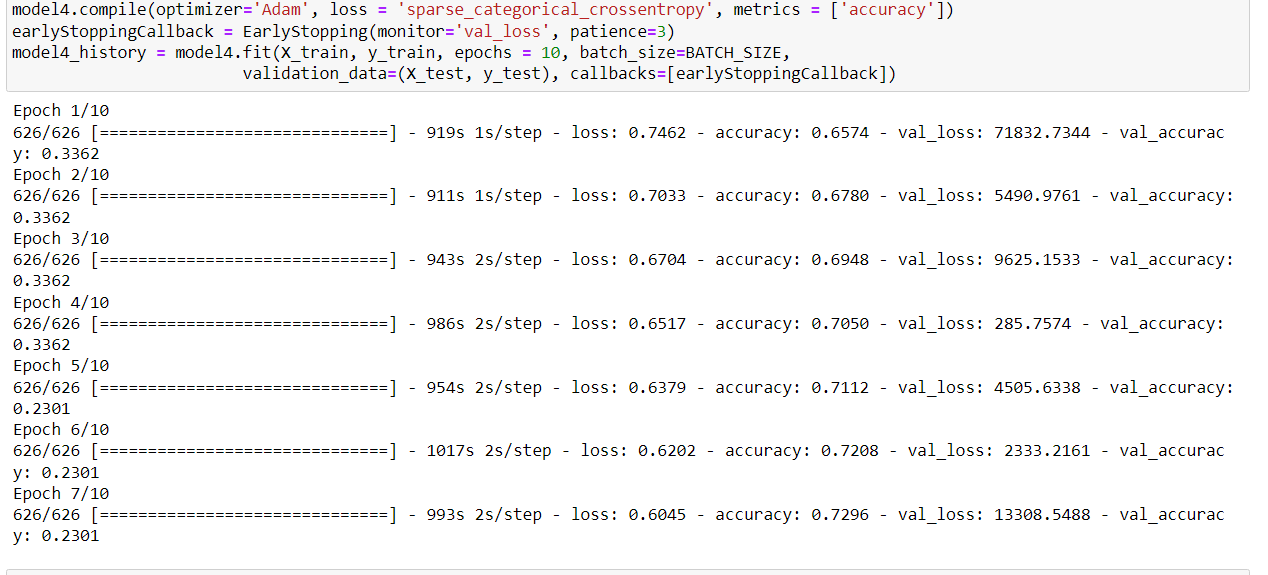
Description: Code to find best LR





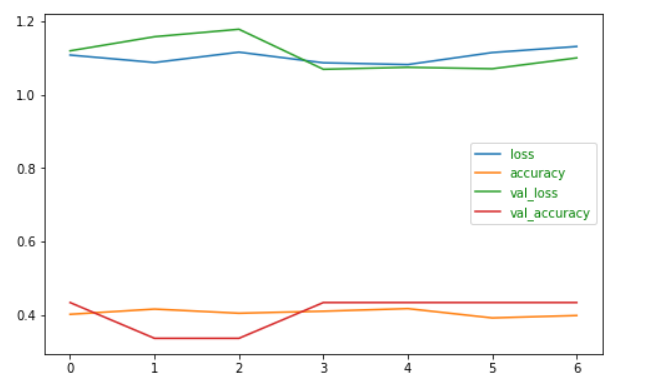
Description: Testing Momentum

We tried Adam Optimizer which resulted in overfitting,

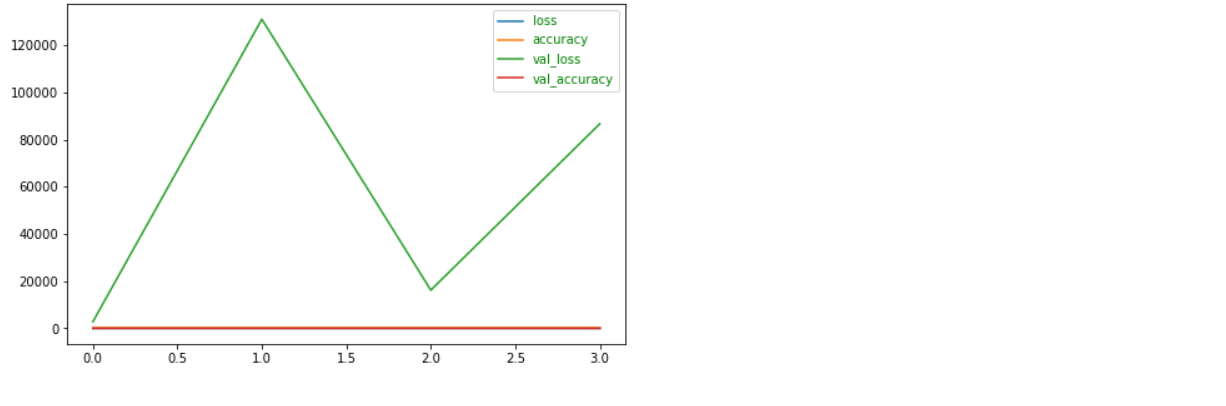


# Transfer Learning

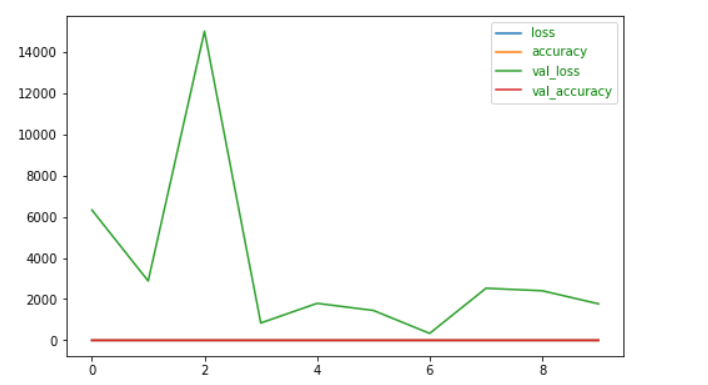
# VGG16 : Accuracy is very poor



# ResNet50 and DenseNet: It was showing very high loss and a clear indication for overfitting

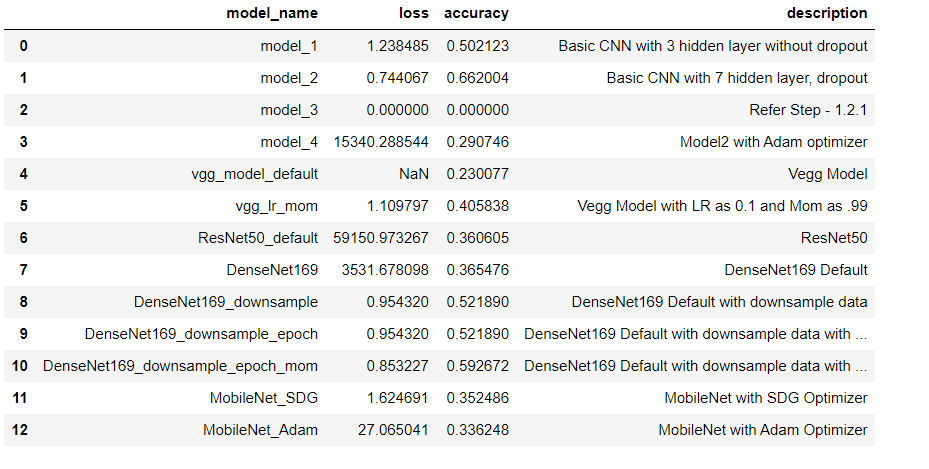


Description RestNet Result



Description: DesnseNet Result

# Classification Summary



The above table clearly indicates that our model with 7 hidden layer is producing better accuracy however this is something not acceptable. In order to try with more hidden layer and to try more optimization parameters we can use packages like KerasTuner. This also means that we need more dedicated system available 24/7 (kind of commodity server). With the PC’s we could not run anything more further.

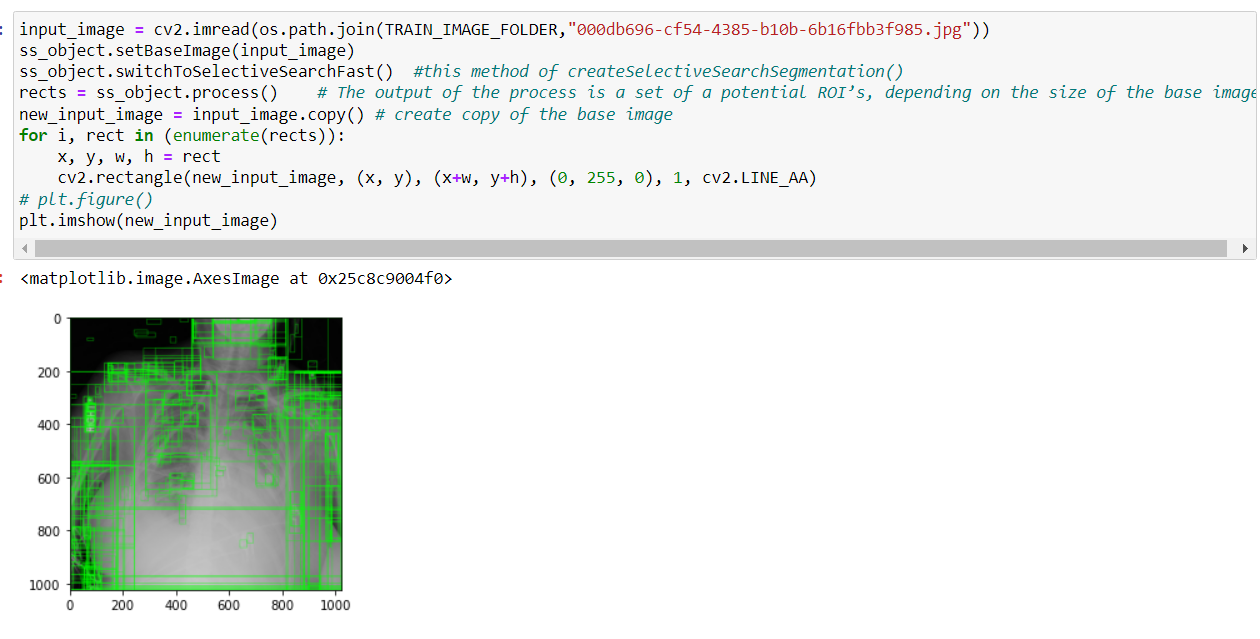
# Object Detection

This is a main puzzle which we want to solve however due to project time limit we could not complete this. Following is what we would like to try,

RCNN:

1. It works based on Selective Search

Following results were obtained on one image to see the clustering,



Mask RCNN:

<TODO>

# SUMMARY

We got zip consists of image and CSV files which was clearly preprocessed and peformed EDA on the dataset. Later we tried basic CNN and did classification and then improved performance accuracy for classification using additional hidden layers. We also tried transfer learning for improvements however we could not better results with transfer learning. It is also possible that we can still do more hyper parameter tunning on transfer learning, etc. However due to time, hardware and resource limitations we could not try anything more. Finally the object detection is something is not tried to the full extend due to time limit.