**Capstone Project Final Report -Pneumonia Detection using Computer Vision**

**AIML - Post Graduate Program**

**14-Mar-2020**

**1.Definition**

Project Overview

**Executive Summary**

This project represents a culmination of the Ten modules of the AI and ML Specialization offered by Great Lakes Executive Learning and University of Texas at Austin via Great Learning. The Pneumonia Detection prediction model is built based on the basics of Computer Vision Technique techniques learned throughout the specialization. The project focuses on to build Prediction model on Pneumonia Detection

* Build a pneumonia detection model starting from basic CNN and then improving upon it.
* Train the model. To deal with large training time, save the weights so that you can use them when training the model for the second time without starting from scratch.
* Test the model and report as per evaluation metrics - IOU - Intersection over Union
* Build models on SSD, Mask R CNN, YoloV3 for the Pneumonia Detection
* Set different hyper parameters, by trying different optimizers, loss functions, epochs, learning rate, batch size, check pointing, early stopping etc..for these models to fine tune them
* Evaluate metrics for these models along with your observation on how changing different hyper parameters leads to change in the final evaluation metric.
* Deploy the Best Predicting Model on to Google Cloud Platform

**2.Analysis**

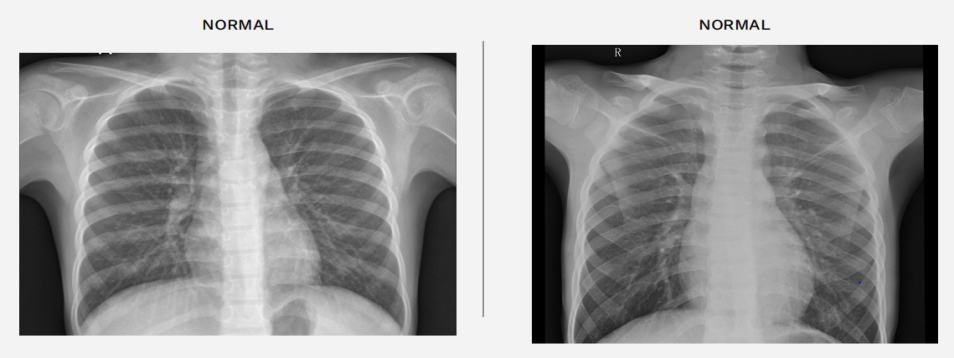
**Exploratory Data Analysis**

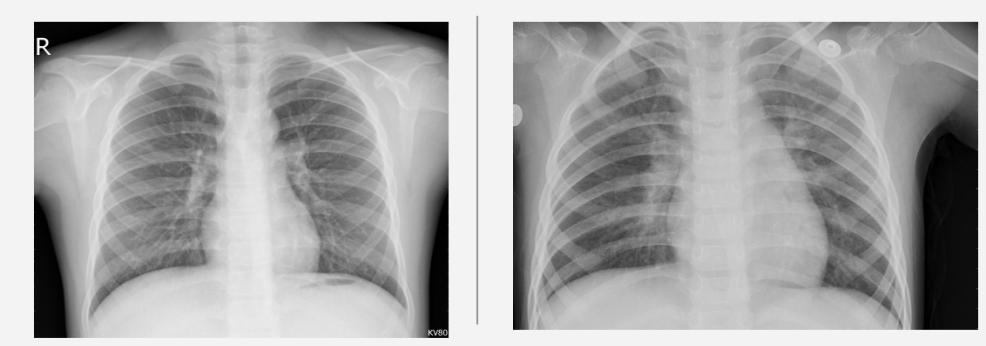
The corpora given comprises X-RAY of Lung images of very large datasets from Kaggle competition - of more than1000 and above DICOM images with file size of over 4 GB.

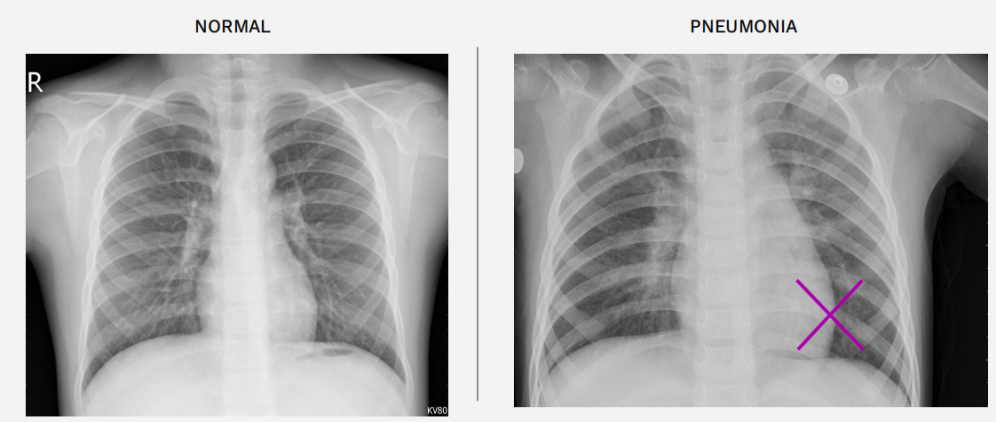
**Reference Jupyter notebook- EDA\_PneumoniaDetection.ipynb**

Kaggle dataset - quick look

* 5’863 X-Ray images
* pediatric patients 1-5 years old
* labeled by several specialists







In order to be accomodated within my system limitations (and in keeping with the approach recommended) a sample of the corpus was selected for study in order to build and train the prediction model.

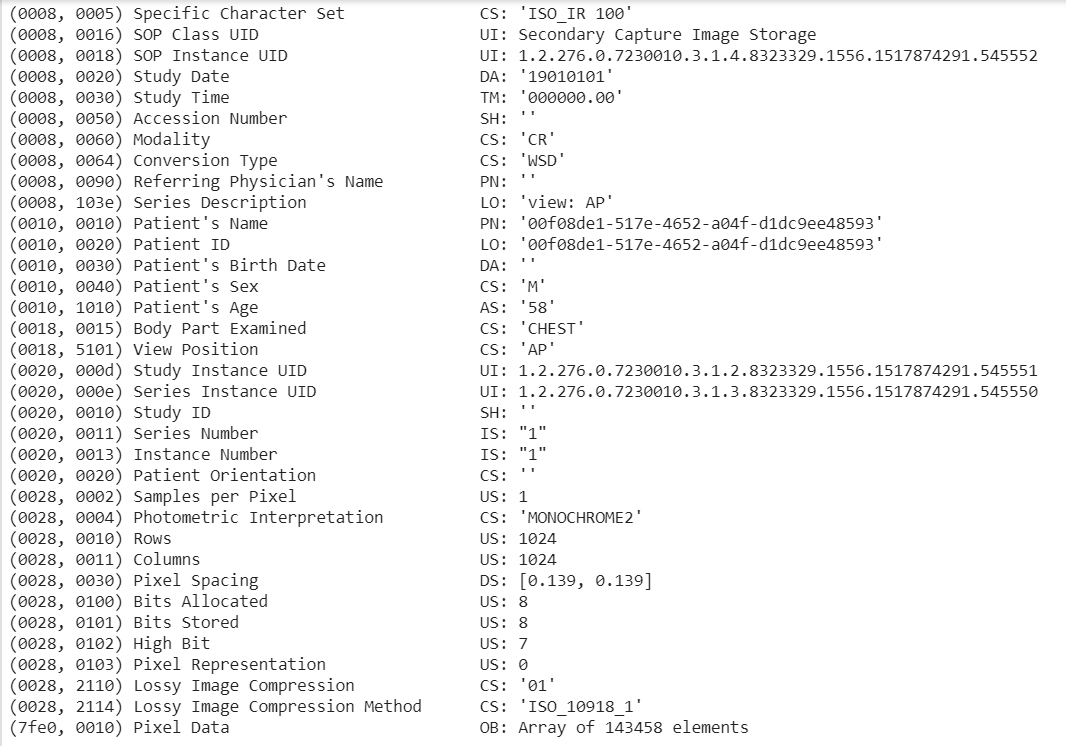
It is very important to understand the data in DICOM files before we work on Prediction Models.

import pydicom

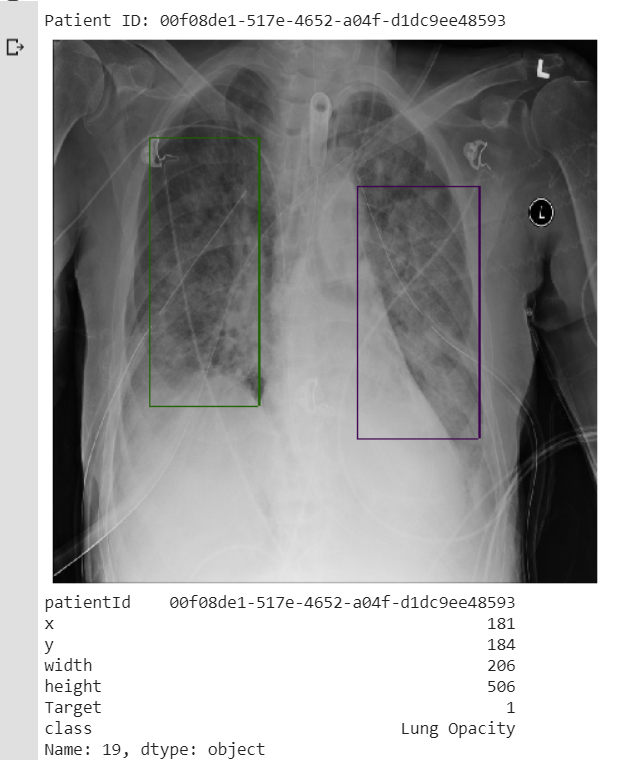
dcm\_file = '../content/RSNAdata/data/stage\_2\_train\_images/%s.dcm' % pat\_choose

dcm\_data = pydicom.read\_file(dcm\_file)

print(dcm\_data)

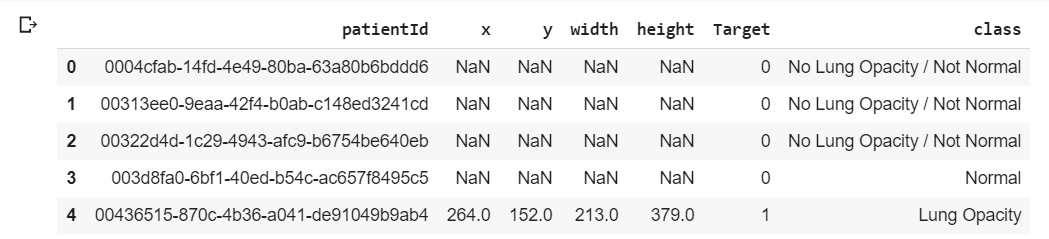


Understanding the data from the DICOM files is imperative to being able to ensure one's conceptualization of bounding boxes on the arrays from those files. We need to visualize those boxes in order to augment the knowledge regarding the visual aspects of pneumonia:

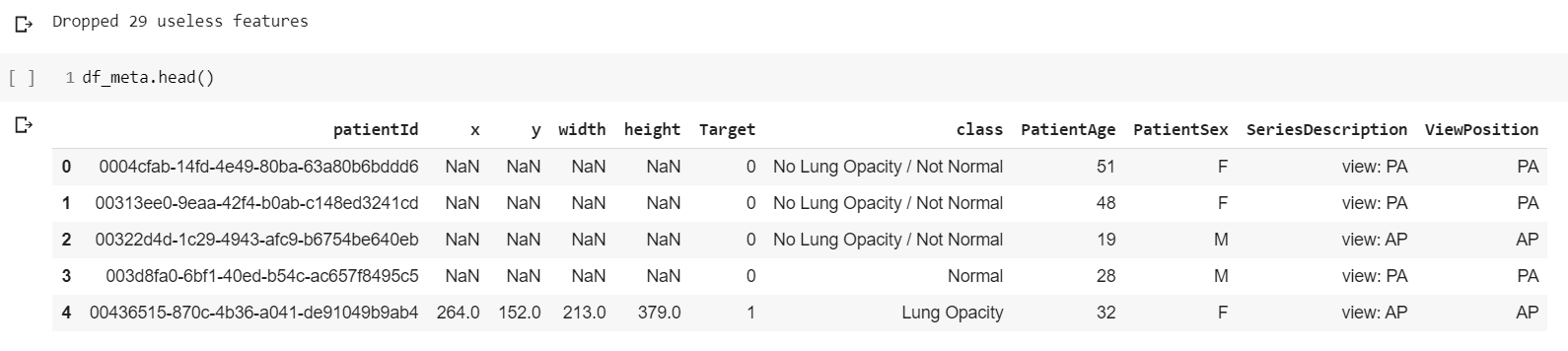


Now that we have visualized the bounding boxes and the XRAYs, we should take a look at the demographics of our features

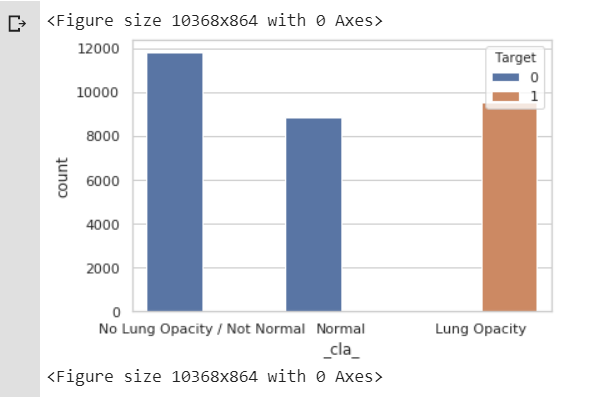
Next Perform EDA of the dataset Setup a dataframe - Gender, Viewing Position , Age etc.



After dropping 29 features



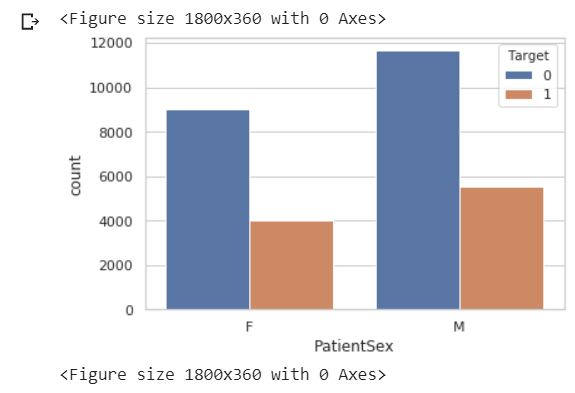
Frequency Chart of our detailed class ( Colored by Binary Class )



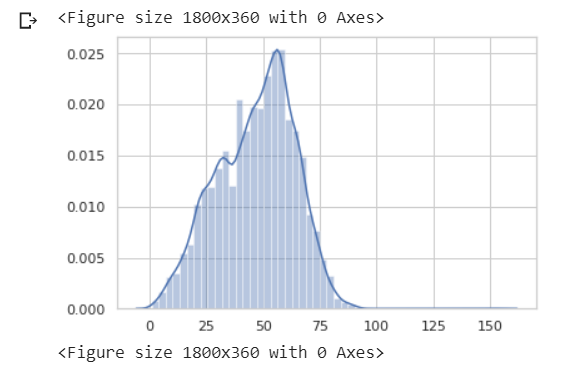
Frequency Chart of Binary Class ( Colored by Binary Class )



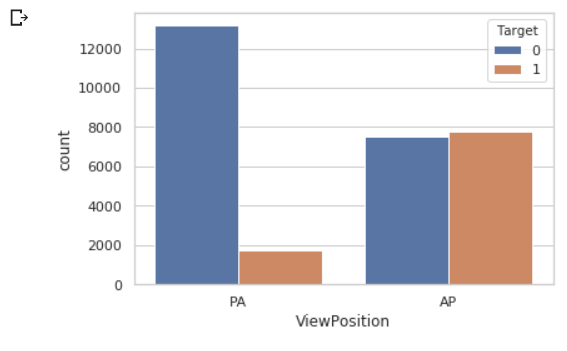
Frequency Chart of Sex ( Colored by Binary Class)



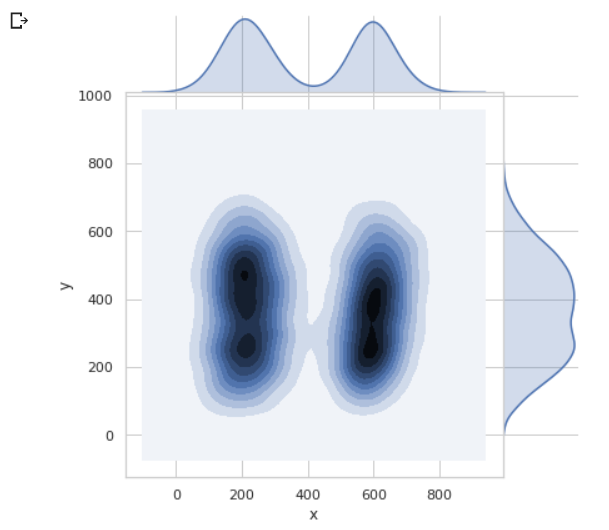
Distribution Plot of Patient Age



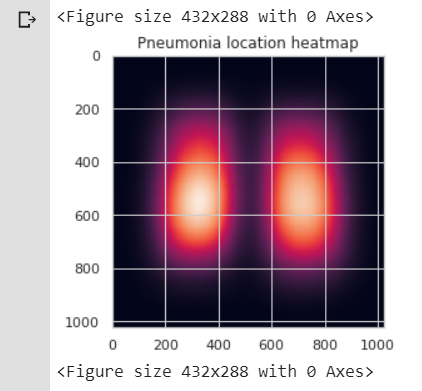
Clustered Column Chart based on viewing position



Heat map for x & y corners of each bounding box. But the below heatplot is imperfect.



So below will display Heat map of Pneumonia Presence in the sample image.



1. **Methodology**

**Model Building**

**Approach 1- Bounding Box Predictor with CNN**

Reference jupyter notebook- BoundingBoxPredictor(CNN).ipynb

Step 1: Load the trainingdataSample , BoundingdataSamples and testdatasample and parse through Patient ID

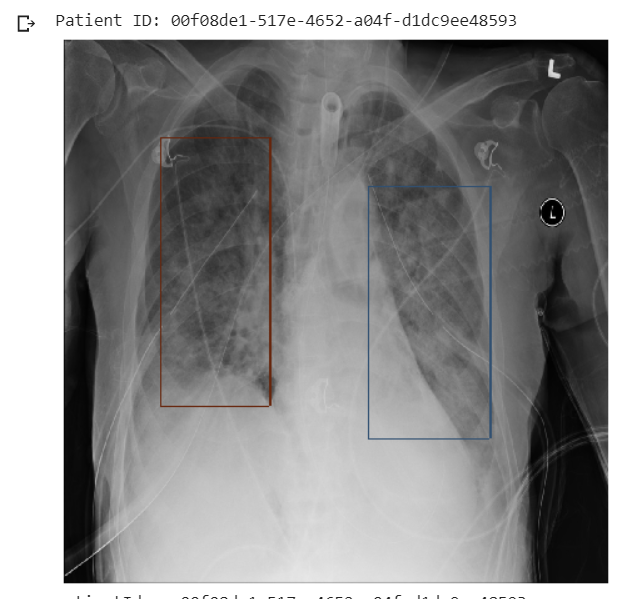
Step 2: Create Classification labels on “Pneumonia” and “LungOpacity” and load images with label “Pneumonia” and perform masking using CV2.

Step 3: Perform Image annotations over the Training images  
Step 4: Plot the masked images and set Detector configurations

Step 5: Load the pre-trained Mobilenet SSD models with all layers and batched dataset.

Step 6: Load the weights on the Mask RCNN model and get colors for Class ID =1

Step 7: Once the model is loaded, plot and visualize the model detection output



**Approach 2- Bounding Box Predictor using SSD**

Reference jupyter notebook- PneumoniaDetection\_SSD\_Nitesh.ipynb

Step 1: Load the trainingdataSample , BoundingdataSamples and testdatasample and parse through Patient ID

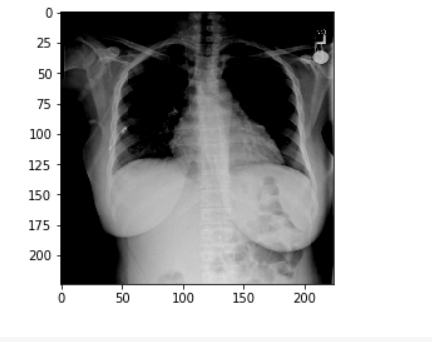
Step 2: Create Classification labels on “Pneumonia” and “LungOpacity” and load images with label “Pneumonia” and perform masking using CV2.

Step 3: Perform Image annotations over the Training images  
Step 4: Plot the masked images and set Detector configurations

Step 5: Load the pre-trained Mobilenet SSD models with all layers and batched dataset.

Step 6: Load the weights on the Mask RCNN model and get colors for Class ID =1

Step 7: Once the model is loaded, plot and visualize the model detection output



**Approach 3 - Bounding Box Predictor using Mask R CNN**

Reference jupyter notebook- PneumoniaDetection\_Maskrcnn\_Nitesh.ipynb

Step 1: Load the trainingdataSample , BoundingdataSamples and testdatasample and parse through Patient ID

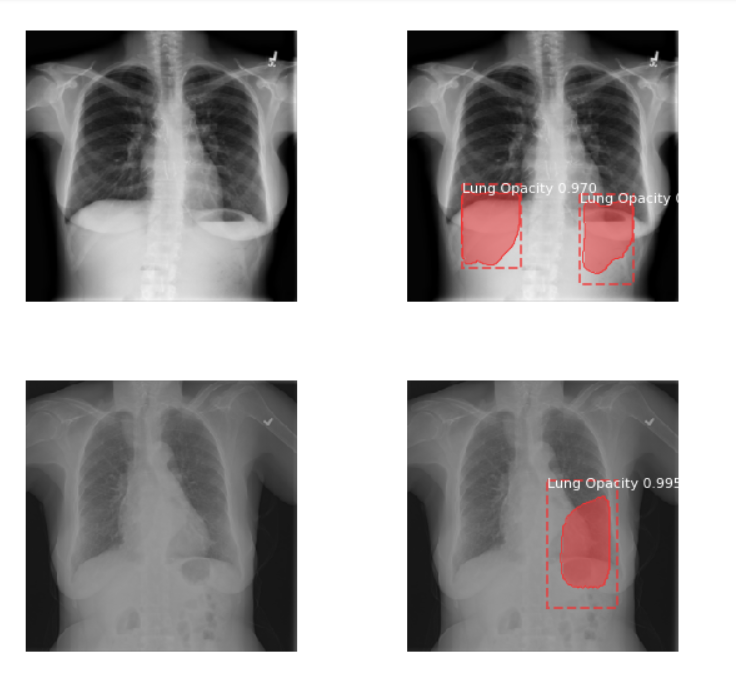
Step 2: Create Classification labels on “Pneumonia” and “LungOpacity” and load images with label “Pneumonia” and perform masking using CV2.

Step 3: Perform Image annotations over the Training images  
Step 4: Plot the masked images and set Detector configurations

Step 5: Load the pre-trained Mask RCNN models with all layers and batched dataset.

Step 6: Load the weights on the Mask RCNN model and get colors for Class ID =1

Step 7: Once the model is loaded, plot and visualize the model detection output



**Approch 4 - Bounding Box Predictor using Yolo V3**

Reference jupyter notebook- PneumoniaDetection\_YoloV3.ipynb

Step 1: Load the trainingdataSample , BoundingdataSamples and testdatasample and parse through Patient ID

Step 2: Create Classification labels on “Pneumonia” and “LungOpacity” and load images with label “Pneumonia” and perform masking using CV2.

Step 3: Perform Image annotations over the Training images  
Step 4: Plot the masked images and set Detector configurations

Step 5: Load the pre-trained Mask RCNN models with all layers and batched dataset.

Step 6: Load the weights on the Mask RCNN model and get colors for Class ID =1

Step 7: Once the model is loaded, plot and visualize the model detection output

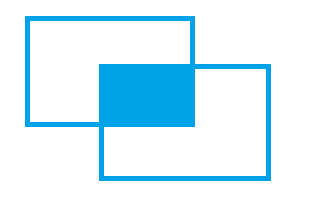
1. **Results**

**Model Evaluation and Validation**

**IOU -** Intersection over union has been set as a benchmark against all 4 Models ( CNN, SSD, Mask R CNN, Yolo V3 ). Objects in an Image/Frame are detected with a simple box plotted around them.This task of plotting a box around the Object can be called bounding boxes.The bounding box is nothing but (x-y ) coordinates of the object in the image. These co-ordinates uniquely defined objects in the Image.Now, the bounding box for an Object in Image is primarily hand labeled and can be called as Primary Boundary Box.The Deep Learning model predicts a bounding box around the Object which can be called Predicted Boundary Box.

IOU can be computed as Area of Intersection divided over Area of Union

Areas of Intersection



Areas of Union



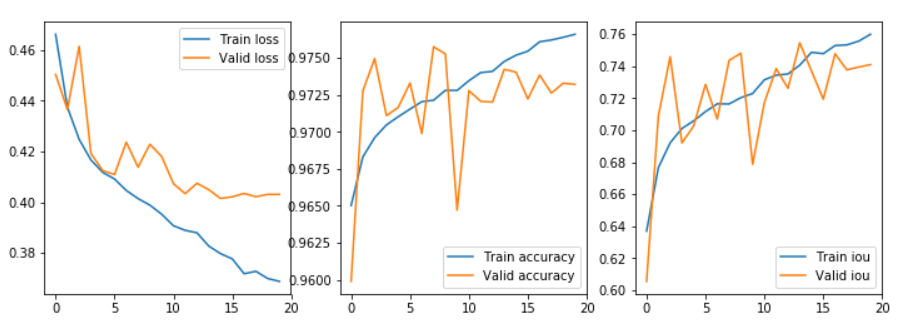
The model output for predicted bounding box is extremely unlikely to be as an exact primary bounding box in reality. Therefore to measure how accurate is the object identified in the Image/Frame we can make use of metric IOU.

This gives us an option to consider the object detected is complete or not. The IOU is a simple way of evaluation of our training model +bounding box with its performance on the testing set.

General Threshold for the IOU can be 0.5. This can vary from problem to problem. Normally **IOU>0.5 is considered a good prediction.**

 IOU is an important metric in deciding the object prediction of deep learning models.

Sample IOU plot is shown below



As per our Analysis, our findings on each models shows as below

|  |  |  |
| --- | --- | --- |
| ***Model*** | ***IOU Metric*** | ***Remarks*** |
| CNN | 0.65 | Bad |
| SSD | 0.75 | Good |
| Mask R CNN | 0.89 | Good |
| Yolo V3 | 0.70 | Good |

Since the Mask R CNN model scores more than other models, we chose this to be Best Model for Pneumonia Detection to deploy on to GCP. The Next Section details vividly about the GCP Deployment.

1. **Deployment**

**Run Models within GCP :**

Generally, we were running our models in Jupyter notebook on Google Colab’s hosted run time.

Hosted run time runs on a new machine instance in Google cloud and we don’t need to set up any hardware.

We face situations like run time getting disconnected due to notebook inactivity and we had to re-run the code from the beginning.

For loading training and test data, first we have to upload to Google Drive and have to import it in our note book for accessing the data.

We can avoid the above two problems, by running Note book on local run time.

Local Runtime runs on your machine. You need to install Python, Jupyter, and set-up some necessary libraries required to run models. It is useful if you have a lot of data to process locally, or if you have your own powerful GPU to use.

Google Cloud Platform provides an option to create local run time through Virtual Machine Instance. Developers can customize the VM instance by choosing the computing power (CPU’s and GPU’s) which is required to run the models.

Below are the steps to create Virtual machine instance in Google Cloud platform.

In Chrome, go to URL: <https://cloud.google.com/>

Select “Get Started for free”. GCP allows free trial for 12 months and 300$ is credited for free.

In Step 1, select Country for example: India and click Accept Terms and Conditions check box and select continue.

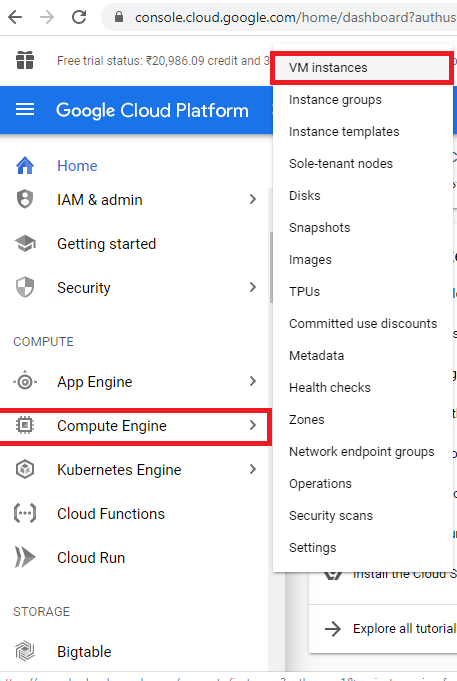
In Step 2, provide Name, Address and Credit Card details in payment. (Note: Provide valid credit card details only, as Rs.1/- will be deducted after OTP verification.)

Next, click on Start my free trial.

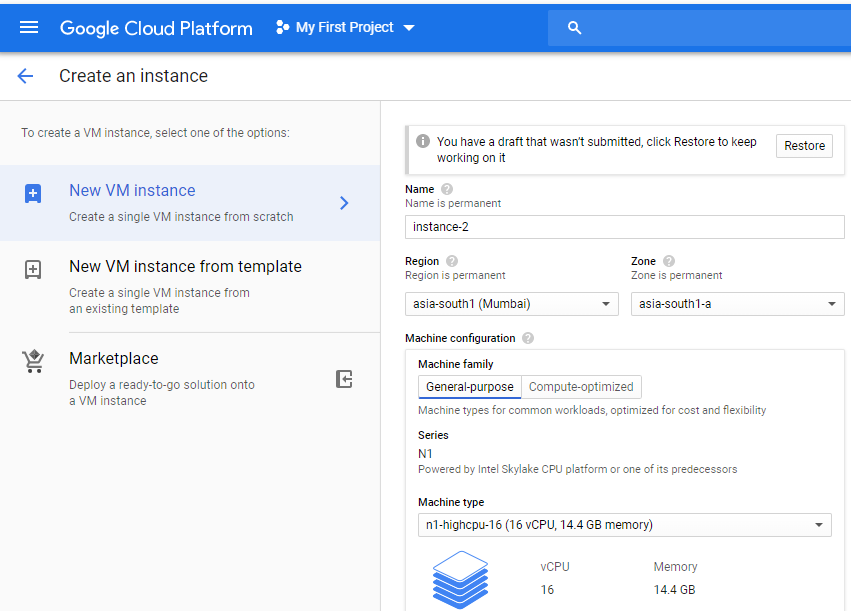
After Google Cloud Platform page is open, create a new project to access the all features of the GCP.

**Creating VM Instance:**

1. In the Navigation menu, go to Compute Engine > VM Instances as shown below.

****

1. In the VM instances page, click on “CREATE INSTANCE”. Below screen is displayed.



2.a Enter a name for the Instance.

2.b Change region to **asia-south1 Mumbai**, Select Zone as **asia-south1-a**

2.c Change the machine type to **n1-highCPU-8** or **n1-highCPU-16**

2.d GPU cannot be added under free trial.

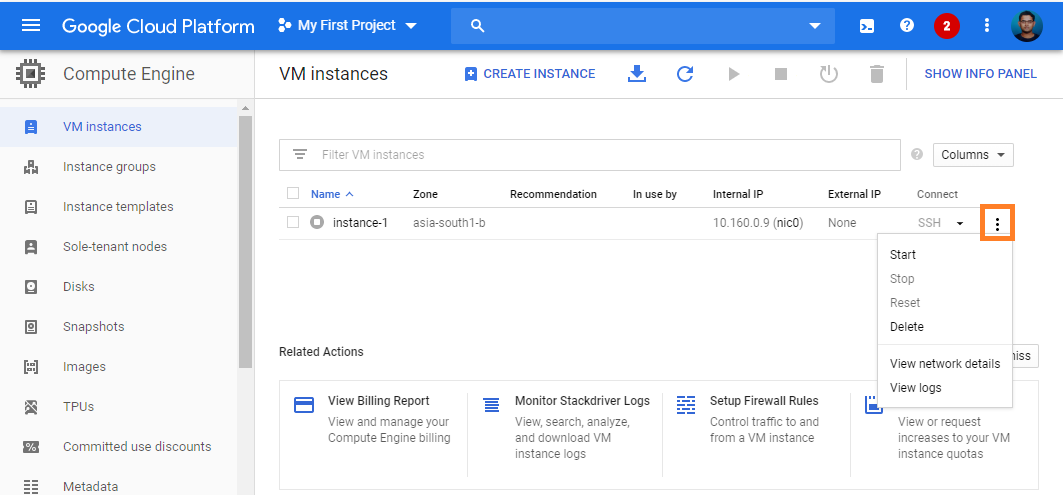
2.e Don’t change the boot disk, default Debian GNU/Linux is selected.

2.f Under fire wall, check the Allow http traffic and Allow https traffic checkboxes.

Click on Create.

1. Newly created VM instance is displayed as below.

Next, Click on Options menu to Start/Stop VM instance.



1. Once VM instance is started remotely in GCP, the Linux Operating System can be accessed from our local Window machine via SSH.

SSH provides a [secure channel](https://en.wikipedia.org/wiki/Secure_channel) over an unsecured network in a [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) architecture, connecting an [SSH client](https://en.wikipedia.org/wiki/SSH_client) application with an [SSH server](https://en.wikipedia.org/wiki/SSH_server).

1. For connecting to Linux operating via SSH. Google Cloud has provided a **Command Line** Tool which is available in Google Cloud SDK. Download the SDK installer from the below link and install Google Cloud SDK on your local machine.

Link: <https://cloud.google.com/sdk/docs/downloads-interactive>

1. After installation, open Google Cloud command-line tool from Windows local machine as below:



1. For more info about Google Cloud Command-line tool, please refer to <https://cloud.google.com/sdk/gcloud>

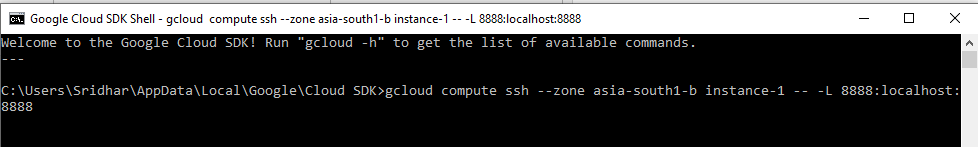
**What is the gcloud command-line tool?**

The gcloud command-line interface is a tool that provides the primary Command Line Interface to Google Cloud Platform. You can use this tool to perform many common platform tasks either from the command-line or in scripts and other automations.

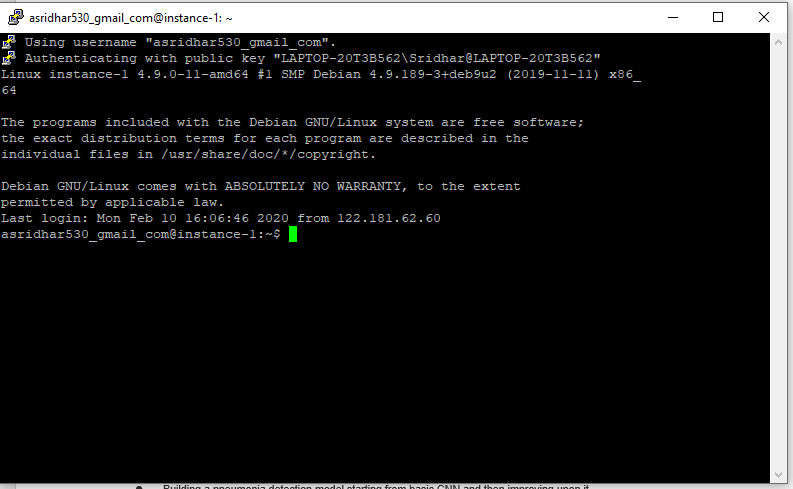
1. Connecting to Linux OS – Virtual Machine Instance.

In the command line interface, please run the below command.

**gcloud compute ssh --zone asia-south1-a instance-1 -- -L 8888:localhost:8888**

****

1. After running the above command, we can able to access the Linux OS running on Virtual Machine instance via SSH.



1. In this console, we can run the Unix commands to install Python, Jupyter and other libraries which are required to run our models.

10.a First we need to Install PIP command. The pip command is a tool for installing and managing Python packages.

Please refer to <https://linuxize.com/post/how-to-install-pip-on-debian-9/>

10.b Install Jupyter notebook. Command: **pip install notebook**

10.C Install Pandas, Numpy etc. using Command **pip install “package-name”**

1. After installing the necessary libraries, Install and enable the **jupyter\_http\_over\_ws**jupyter extension (one-time).

This Jupyter server extension allows running Jupyter notebooks that use a WebSocket to proxy HTTP traffic. Browsers do not allow cross-domain communication to localhost via HTTP, but do support cross-domain communication to localhost via WebSocket.

**pip install jupyter\_http\_over\_ws**

**export PATH=$PATH:~/.local/bin**

**jupyter serverextension enable --py jupyter\_http\_over\_ws**

1. Starting Jupyter Notebook server in Linux OS. Below command allows connection from

Google Colab.

1. First go to bin folder path:

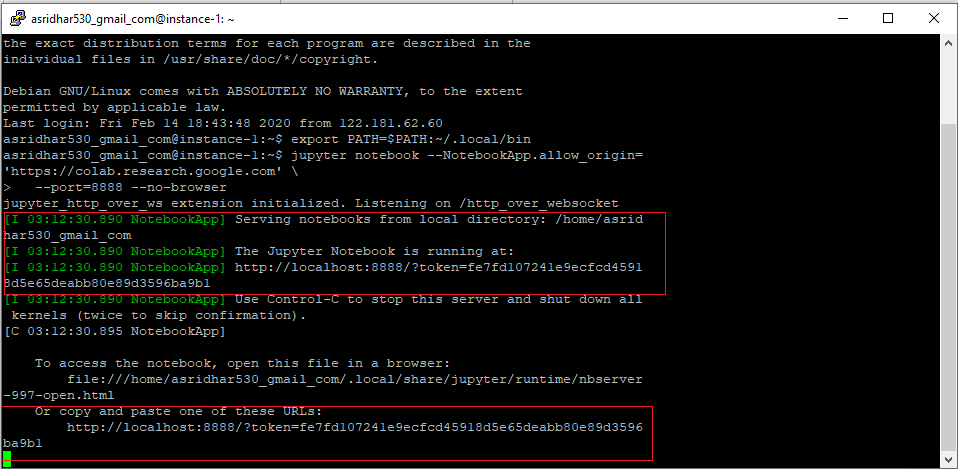
**export PATH=$PATH:~/.local/bin**

1. Run the command

**jupyter notebook --NotebookApp.allow\_origin='https://colab.research.google.com' \**

**--port=8888 --no-browser**

1. After running the above command, Jupyter Notebook will start running in VM instance as shown below:

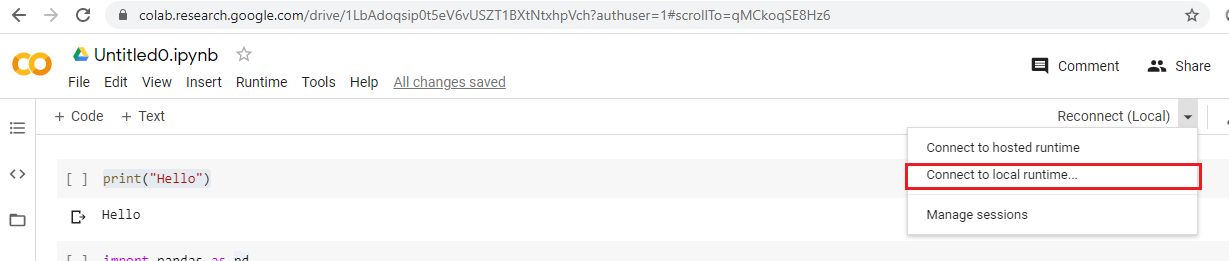


1. To access this Notebook in your Google Colab notebook, copy the URL displayed in the console.

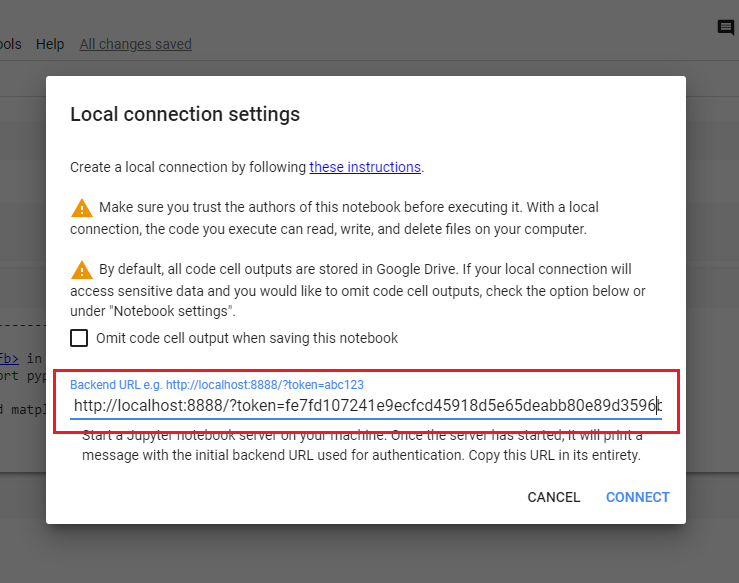
**Note: This URL will change every time when you start Notebook on the VM instance.**

**So, the above console should be running when you are running models in Google Colab.**

1. Open Google Colab in your Chrome browser, create a new Python note book. Click on **Connect to local runtime** as shown below.

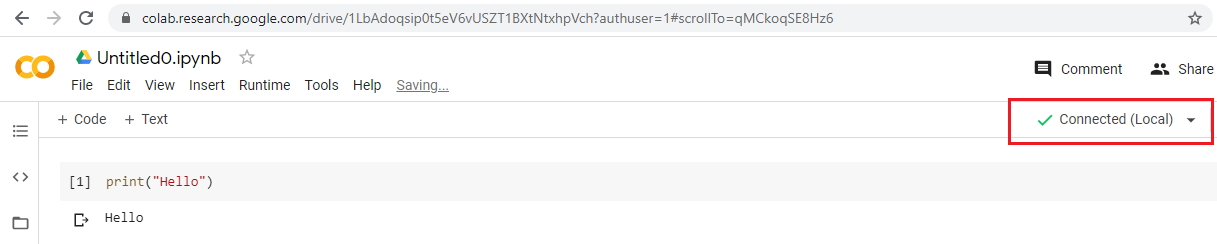


1. Copy the URL from console and paste it in the below marked as shown below and click on **Connect.**



1. After the connection is successful, the notebook should show as below:

Next, we can start running the code.



1. **Final Summary**

Summary and end to end description:

The problem that this capstone project addresses is detection and classification of X Ray Images of Lungs that infected with and without Pneumonia. Dataset of around 1500 images from Kaggle has been used for training, testing and validation for Bounding Box Prediction Models.

The first approach Bounding Box Predictor using CNN with feature extracted on Dicom images using Histograms of