Pneumonia Detection: A comparative Overview

Background

Pneumonia is an infection that inflames the air sacs in one or both human lungs. Pneumonia accounts for over 15% of all deaths of children under 5 years old internationally. In 2015, 920,000 children under the age of 5 died from the disease. In the United States, pneumonia accounts for over 500,000 visits to emergency departments and over 50,000 deaths in 2015, keeping the ailment on the list of top 10 causes of death in the country.

There are multiple diagnosis methods possible to identify Pneumonia. Chest Radiograph (CXR) is the most commonly used diagnostic method performed by highly trained specialists. Pneumonia usually manifests itself as an area or areas of increased opacity in the lung area on CXR. Few other cases like bleeding, pulmonary edema (fluid overload), atelectasis (volume loss), Post radiation, lung cancer or surgical changes can also cause the areas of increased opacity on CXR. Another common reason for the areas of increased opacity on CXR is accumulation of fluid in the pleural space (outside the lungs). Highly trained specialists are required to correctly classify an increased opacity.

Objective

The objective of the project is to build efficient algorithms (for image segmentation and localization) to detect a visual signal for pneumonia by monitoring the lung opacity in chest radiographs. A faster, correct diagnosis of Pneumonia would result in reduced fatalities.

The medical relevance of the project

Though CXRs are the most commonly performed diagnostic method, it is complicated since the results are dependent on many factors such as positioning of the patient and depth of inspiration. With high number of infections every year, clinicians are faced with the challenge of reading high volumes of CXRs. This causes fatigue and introduces human error. In addition, the high number of cases lead to delay in diagnosing Pneumonia. This waiting time or lead time could be fatal to the patients.

Understanding the medical/social/business impact

By building powerful model, this project aims to design a Decision Support System (DSS), which highlights the infected area and provides hint to enable physicians to make efficient clinical decisions. As the model matures, the DSS could perform more automated decisions, thereby relieving the radiologists from the fatigue-driven visual inspection system. Powerful AI models can deliver very fast (on-the-spot) clinical decisions.

Summary of results

For training the CNN model, we used the chest radiographs from **this kaggle competition**. We used random sampling to retrieve the radiographs for train and test datasets that my machine can handle. Here are the steps that we took:

- There are 26,684 unique chest radiographs available from the competition. Of those radiographs, 31.6% are identified with Lung Opacity, 39.1% are identified as Not Normal and 29.3% as Normal. We used two classes for our classification task.

- Dataset consists of DICOM files of the chest X-Ray in 2 different sets (train and test datasets).
- We used data balancing to build and compare various classification models.
- We focussed on three classes for image classification task namely Lung Opacity, No Lung Opacity / Not Normal, and Normal.
- We focussed on two classes for semantic segmentation task namely Pneumonia and No Pneumonia.
- We used 2400 images for training, and 600 images for testing our classification models. For this analysis, we resized the original 1024x1024x3 images to 128x128x3.
- For object detection models, we used 2000 image for training and 2000 images for testing. For this analysis, we resized the original 1024x1024x3 images to 224x224x3.

Key Takeaways from the modelling:

- In image classification: With a simple Fully Connected NN, we achieved a decent accuracy: 0.6328, precision: 0.6537, recall: 0.6235, AUC: 0.7454, and F1 Score: 0.6283. Base model training time was nearabout 23s. The highest performance was recorded with EfficientNetV2B3 model: accuracy: 0.6934, precision: 0.6945, recall: 0.7022, AUC: 0.8696, and F1 Score: 0.6961. The model training time was nearabout 1min.
- In semantic segmentation: We were able to achieve the highest performance with UNet (MobileNet) model: precision: 0.2188, recall: 0.6026, AUC: 0.7675, F1 Score: 0.2832, Dice Coef: 0.2832, and MeanIoU: 0.5569.
- We used Adam optimizer for image classification and object detections models. Given the time and computational resources, we can also experiment with SGD and RMSprop.
- All image classification models were evaluated with epochs = 10 / 30, batch_size = 128, validation_split = 0.2. For semantic segmentation, batch_size of 6 / 8 was used.
- We also used a simple feature-wise cantering and normalization image pre-processing technique before feeding those train images into the model.
- Finally, we used a pre-trained ImageNet model weights for our classification and semantic segmentation tasks. The pre-trained models gave us the best performance in our case.
- Due to lack of time and limited computing resources, we have used limited data, smaller epochs and reduced image dimensions for modelling parts. The Google Colab notebook can be used as a template for more advanced models.

Data Understanding

Understanding how data was collected in terms of time, frequency and methodology:
Chest X-Ray Dataset is provided by US National Institutes of Health Clinical Centre in collaboration with Radiological Society of North America (RSNA®). This dataset is the largest publicly available chest x-ray dataset. Dataset is extracted from the clinical PACS database at National Institutes of Health Clinical Centre and consists of ~60% of all frontal chest x-rays in the hospital. Therefore, this dataset is significantly more representative to the real patient population distributions and realistic clinical diagnosis challenges. This dataset was collected in the year 2017.

Data consists of 2 different sets (train and test) of DICOM files. Two .csv files which provides the labels and the classifications information.

A DICOM file is an image saved in the Digital Imaging and Communications in Medicine (DICOM) format. It contains an image from a medical scan, such as an Radiography, CT scan, Ultrasound or MRI. But we are interested in only the Chest Xray (CXR).

Data Structure (Directory/Folder/Files):

1. stage_2_train_images --- directory which contains training DICOM image set.
2. stage_2_test_images --- directory which contains test DICOM image set.
3. stage_2_train_labels.csv --- File which contains the patientid, bounding box (x, y, width, height), target information
4. stage_2_detailed_class_info.csv - File contains the patientid and provides detailed information about the type of positive or negative class for each image.

Data fields from stage_2_train_labels.csv:

1. patientId - A patientId. Each patientId corresponds to a unique image.
2. x - the upper-left x coordinate of the bounding box.
3. y - the upper-left y coordinate of the bounding box.
4. width - the width of the bounding box.
5. height - the height of the bounding box.
6. Target - the binary Target, indicating whether this sample has evidence of pneumonia.

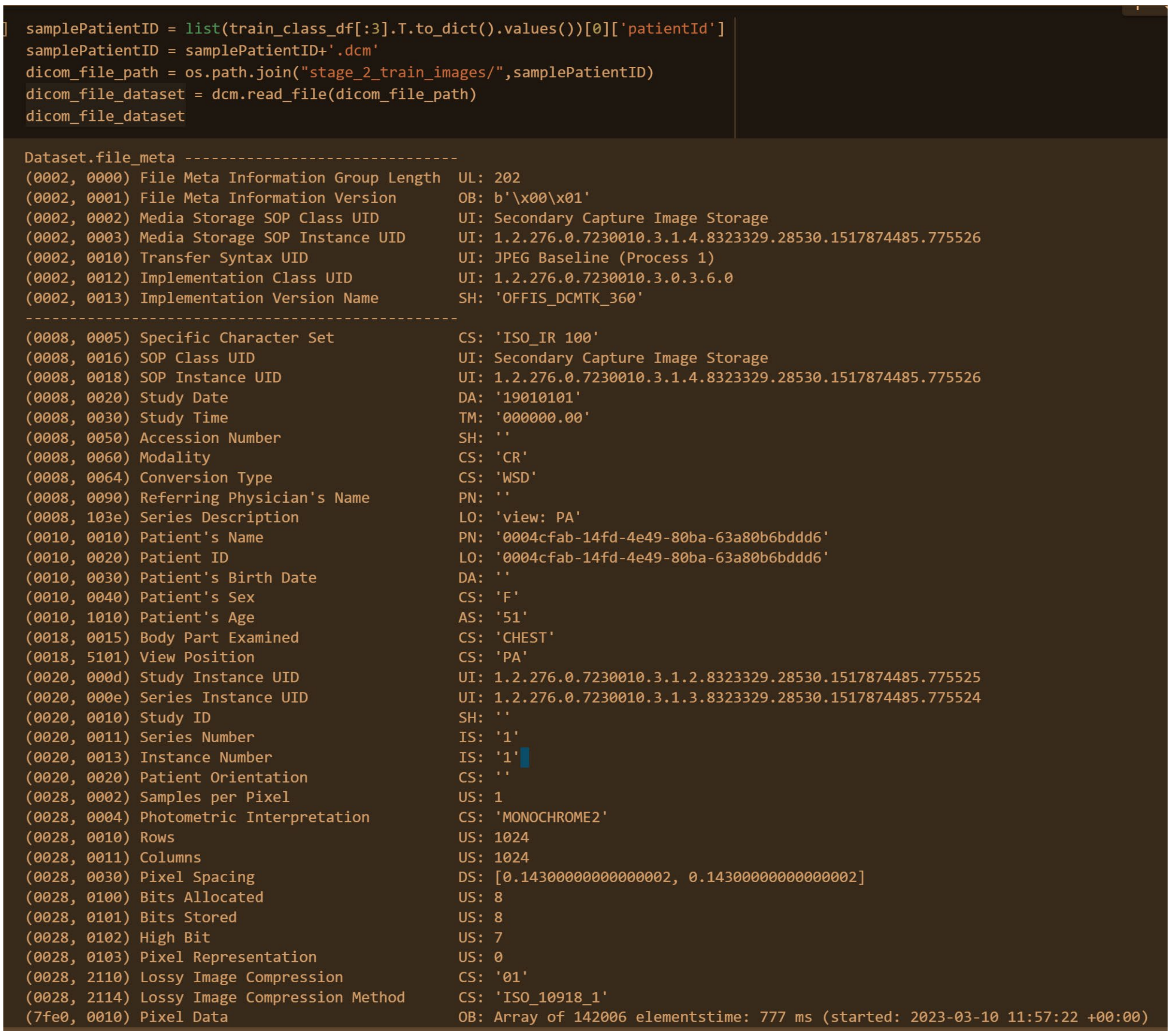
Data fields from stage_2_detailed_class_info.csv:

1. patientId - A patientId. Each patientId corresponds to a unique image.
2. class - information about the type of positive or negative class for each image

Information from the DICOM metadata of an image:

We can see that some attributes may have a higher predictive value:

- Patient sex;
- Patient age;
- Modality;
- Body part examined;
- View position;
- Rows & Columns;
- Pixel Spacing.
- Rows and columns
- ConversionType
- Patients Birth date Etc.



```
samplePatientID = list(train_class_df[:3].T.to_dict().values())[0]['patientId']
samplePatientID = samplePatientID+'.dcm'
dicom_file_path = os.path.join("stage_2_train_images/",samplePatientID)
dicom_file_dataset = dcm.read_file(dicom_file_path)
dicom_file_dataset
```

```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 202
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: Secondary Capture Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0002, 0010) Transfer Syntax UID                 UI: JPEG Baseline (Process 1)
(0002, 0012) Implementation Class UID            UI: 1.2.276.0.7230010.3.0.3.6.0
(0002, 0013) Implementation Version Name         SH: 'OFFIS_DCMTK_360'
-------------------------------------------------
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                       UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0008, 0020) Study Date                          DA: '19010101'
(0008, 0030) Study Time                          TM: '000000.00'
(0008, 0050) Accession Number                    SH: ''
(0008, 0060) Modality                            CS: 'CR'
(0008, 0064) Conversion Type                     CS: 'WSD'
(0008, 0090) Referring Physician's Name          PN: ''
(0008, 103e) Series Description                  LO: 'view: PA'
(0010, 0010) Patient's Name                      PN: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0020) Patient ID                          LO: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'F'
(0010, 1010) Patient's Age                       AS: '51'
(0018, 0015) Body Part Examined                  CS: 'CHEST'
(0018, 5101) View Position                       CS: 'PA'
(0020, 000d) Study Instance UID                  UI: 1.2.276.0.7230010.3.1.2.8323329.28530.1517874485.775525
(0020, 000e) Series Instance UID                 UI: 1.2.276.0.7230010.3.1.3.8323329.28530.1517874485.775524
(0020, 0010) Study ID                            SH: ''
(0020, 0011) Series Number                       IS: '1'
(0020, 0013) Instance Number                     IS: '1'
(0020, 0020) Patient Orientation                 CS: ''
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 1024
(0028, 0011) Columns                             US: 1024
(0028, 0030) Pixel Spacing                       DS: [0.14300000000000002, 0.14300000000000002]
(0028, 0100) Bits Allocated                      US: 8
(0028, 0101) Bits Stored                         US: 8
(0028, 0102) High Bit                            US: 7
(0028, 0103) Pixel Representation                US: 0
(0028, 2110) Lossy Image Compression             CS: '01'
(0028, 2114) Lossy Image Compression Method      CS: 'ISO_10918_1'
(7fe0, 0010) Pixel Data                          OB: Array of 142006 elementstime: 777 ms (started: 2023-03-10 11:57:22 +00:00)
```

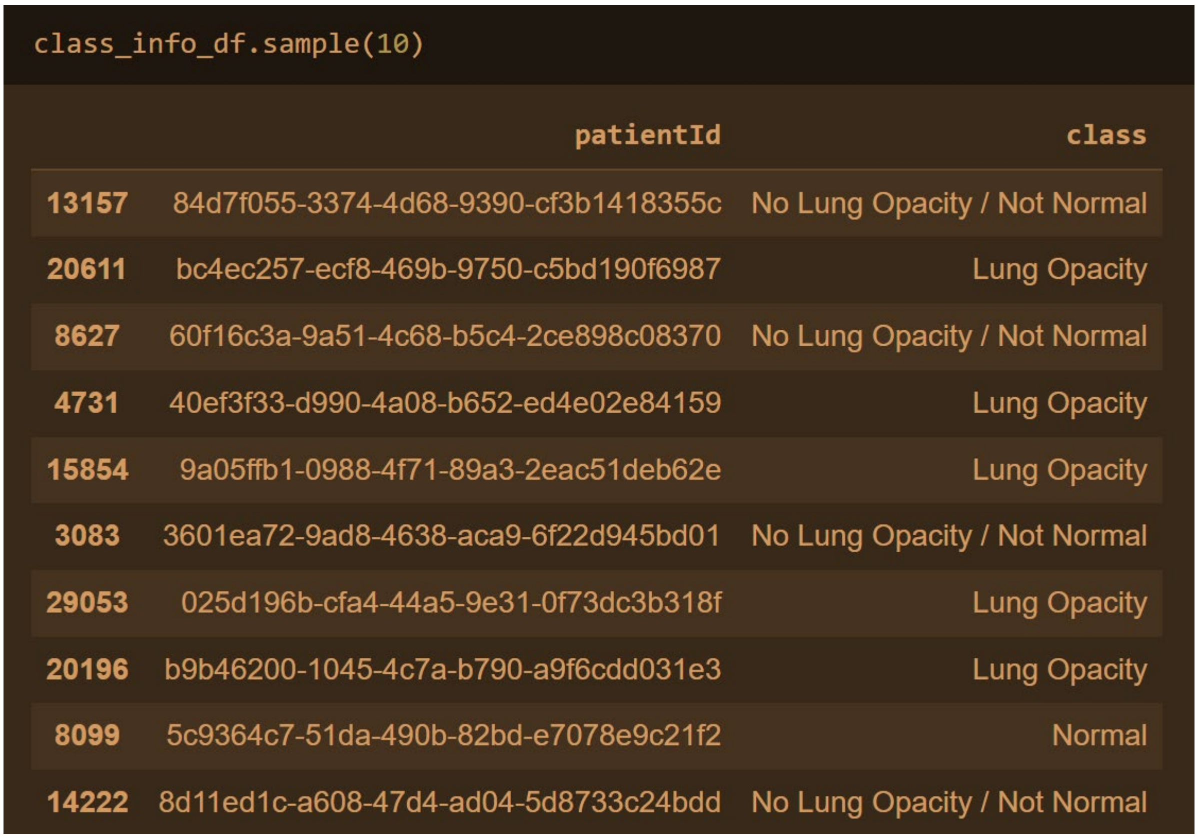
Exploratory Data Analysis (EDA)

After exploring the data, both the tabular and DICOM data, we were able to:

- Discover duplications in the tabular data;
- Explore the DICOM images;
- Extract metadata information from the DICOM data;
- Add features to the tabular data from the meta information in DICOM data;
- Further analyse the distribution of the data with the newly added features from DICOM metadata;

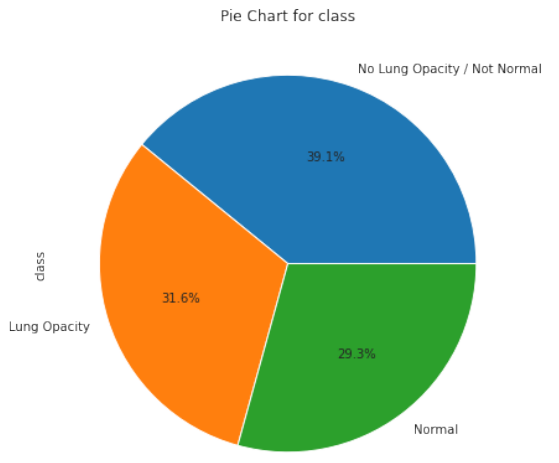
All these findings are useful as a preliminary work for building a model and further improve the performance of various models.

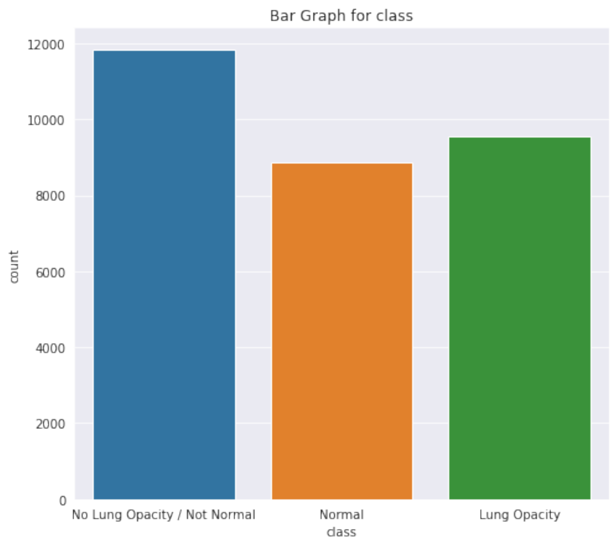
Class distribution from class detailed info:



```
class_info_df.sample(10)
```

	patientId	class
13157	84d7f055-3374-4d68-9390-cf3b1418355c	No Lung Opacity / Not Normal
20611	bc4ec257-ecf8-469b-9750-c5bd190f6987	Lung Opacity
8627	60f16c3a-9a51-4c68-b5c4-2ce898c08370	No Lung Opacity / Not Normal
4731	40ef3f33-d990-4a08-b652-ed4e02e84159	Lung Opacity
15854	9a05ffb1-0988-4f71-89a3-2eac51deb62e	Lung Opacity
3083	3601ea72-9ad8-4638-aca9-6f22d945bd01	No Lung Opacity / Not Normal
29053	025d196b-cfa4-44a5-9e31-0f73dc3b318f	Lung Opacity
20196	b9b46200-1045-4c7a-b790-a9f6cdd031e3	Lung Opacity
8099	5c9364c7-51da-490b-82bd-e7078e9c21f2	Normal
14222	8d11ed1c-a608-47d4-ad04-5d8733c24bdd	No Lung Opacity / Not Normal

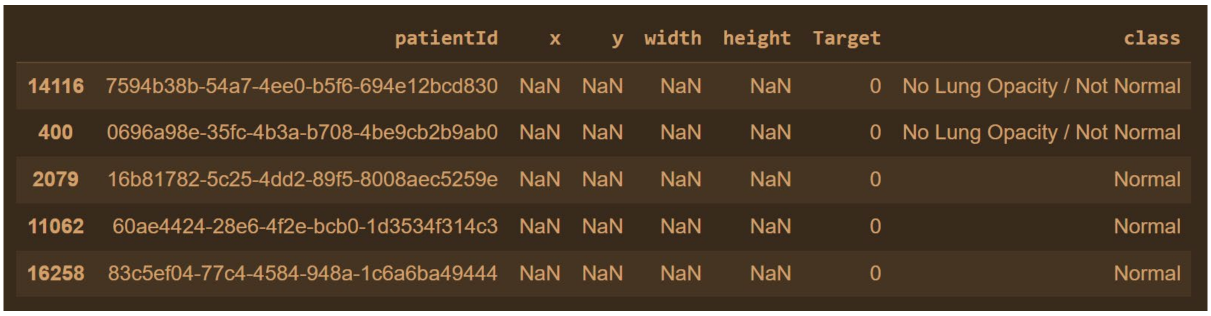




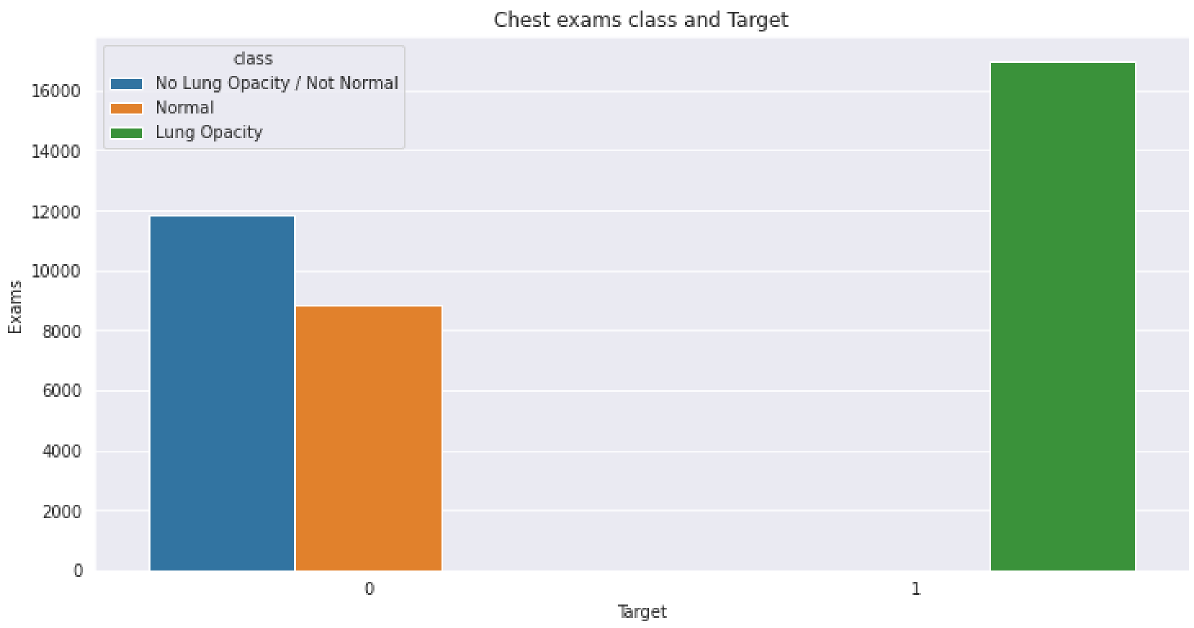
- patientId – Patient ID information; class – Class information of the patient
- **No Lung Opacity / Not Normal** and **Normal** have together the same percent (**69.077%**) as the percent of missing values for target window in class details information.
- In the train set, the percent of data with value for **Target = 1** is therefore **30.92%**.
- **Normal class** - CXR observation is normal, there is no Lung Opacity and abnormalities among 8851 patients
- **Lung Opacity** - CXR shows Lung opacity and this indicates that 16957 Patients have been affected by Pneumonia

- **No Lung Opacity / Not Normal** - CXR observation doesn't show any Lung Opacity but the CXR doesn't look normal in 11821 patients
- There are duplicate entries, but each entry has a corresponding entry in the Labels Data frame.

Merged train and class data info:



	patientId	x	y	width	height	Target	class
14116	7594b38b-54a7-4ee0-b5f6-694e12bcd830	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
400	0696a98e-35fc-4b3a-b708-4be9cb2b9ab0	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal
2079	16b81782-5c25-4dd2-89f5-8008aec5259e	NaN	NaN	NaN	NaN	0	Normal
11062	60ae4424-28e6-4f2e-bcb0-1d3534f314c3	NaN	NaN	NaN	NaN	0	Normal
16258	83c5ef04-77c4-4584-948a-1c6a6ba49444	NaN	NaN	NaN	NaN	0	Normal

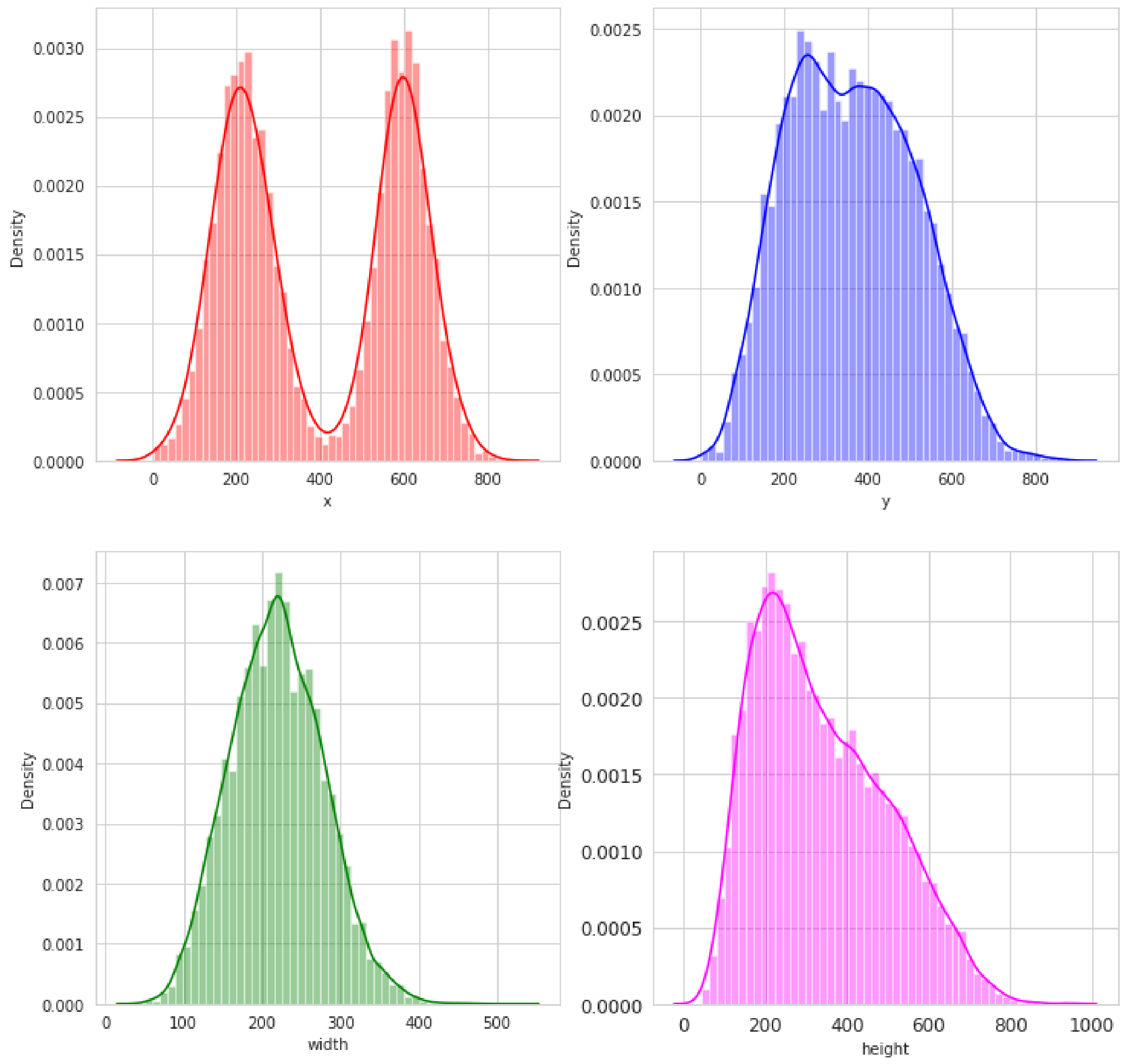


- All chest examinations with Target = 1 (pathology detected) are associated with class: Lung Opacity.
- The chest examinations with Target = 0 (no pathology detected) are either of class: Normal or class: No Lung Opacity/Not Normal.
- Make a note for the classification models: we are using the 3 classes instead of 2.
- x - X coordinate of the infected area
- y - Y coordinate of the infected area
- width - Width of the infected area
- height - Height of the infected area
- Target – Classification
    - label '1': Classification is 'Lung Opacity' and the Opacity area is stored as x, y, width and height column. In case Opacity is found in multiple

places, multiple entries are present in the data with corresponding bounding box.

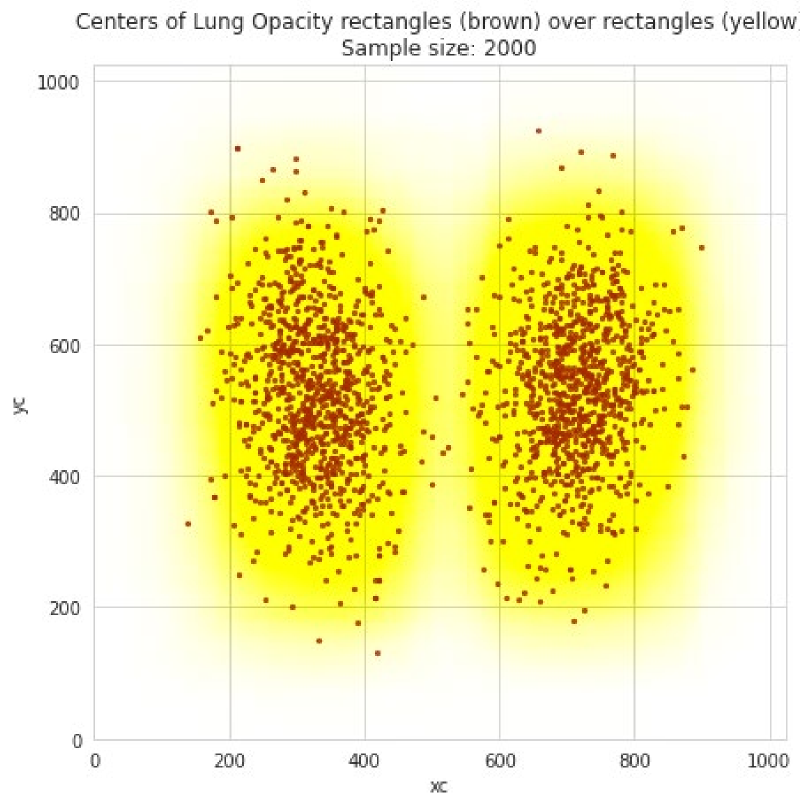
    ◦ label '0’: Classification is either 'Normal' or 'No Lung Opacity / Not Normal'. abnormalities are not stored, so the bounding box is empty.

Detected Lung Opacity window



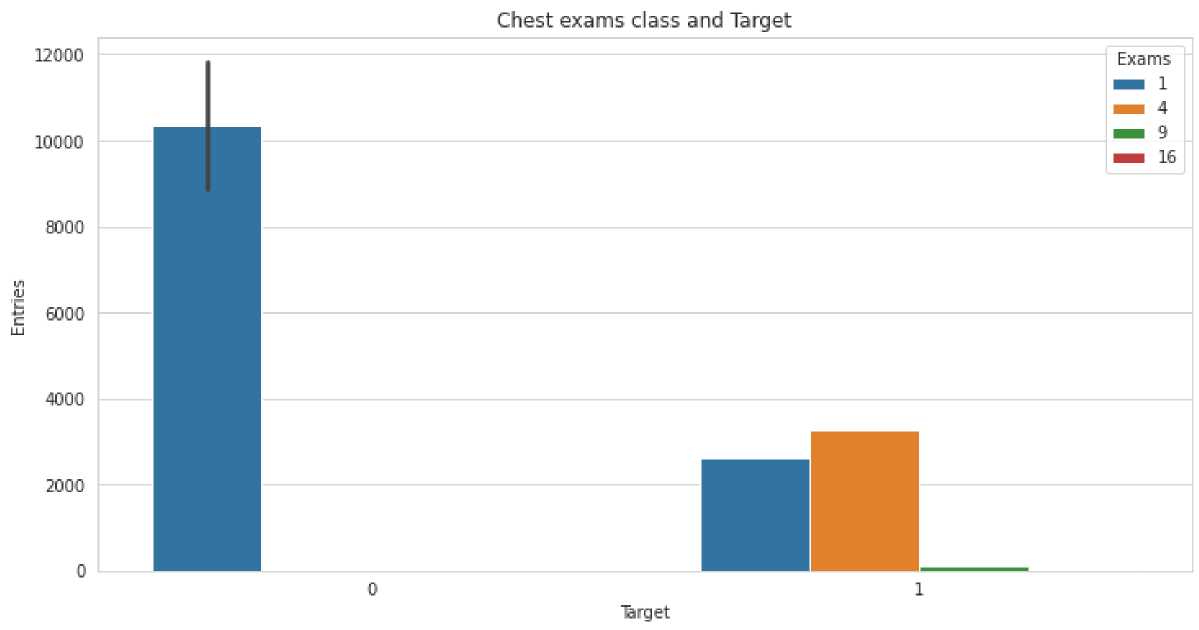
- For the class **Lung Opacity**, corresponding to values of **Target = 1**, we plot the density of **x**, **y**, **width** and **height**.
- We will show a sample of centre points superimposed with the corresponding sample of the rectangles.

Sample of centre points superimposed with the corresponding sample of the rectangles



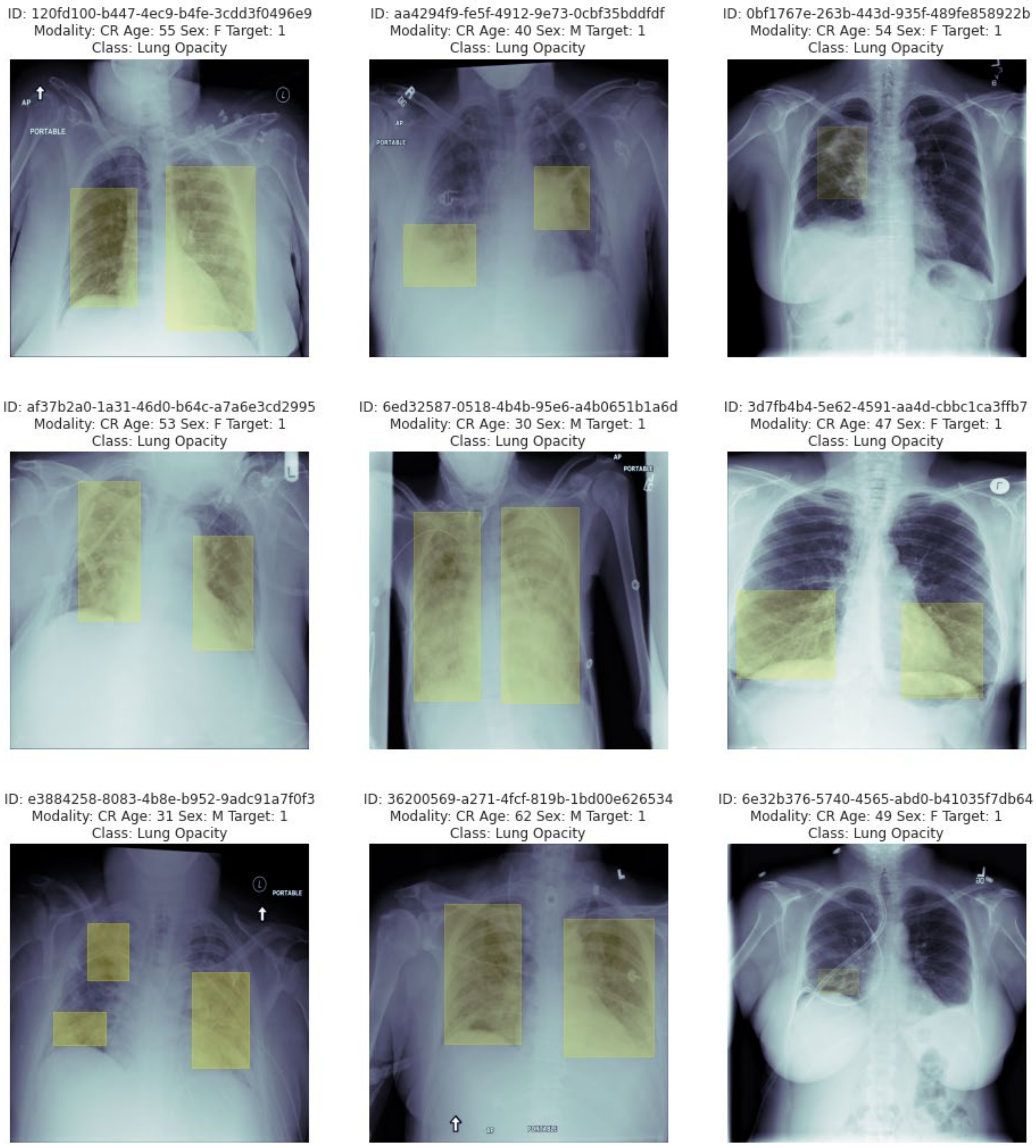
- Lung opacity is more concentrated towards the centre
- More people are having infections in the central lung areas

Understand the target, class and duplicate entries with lung exams



- We have a reduced number of images in the training set (26684), compared with the number of images in the train_df data (30227).
- We can confirm that the number of unique patientId's are equal with the number of DICOM images in the train set.

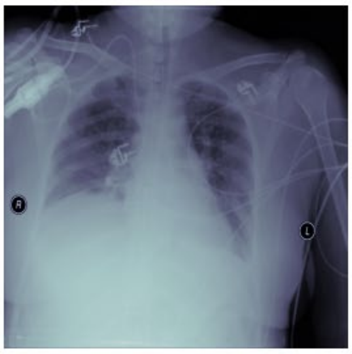
DICOM images with bounding boxes (Target = 1):


ID: 120fd100-b447-4ec9-b4fe-3cdd3f0496e9
Modality: CR Age: 55 Sex: F Target: 1
Class: Lung Opacity
ID: aa4294f9-fe5f-4912-9e73-0cbf35bddfdf
Modality: CR Age: 40 Sex: M Target: 1
Class: Lung Opacity
ID: 0bf1767e-263b-443d-935f-489fe858922b
Modality: CR Age: 54 Sex: F Target: 1
Class: Lung Opacity
ID: af37b2a0-1a31-46d0-b64c-a7a6e3cd2995
Modality: CR Age: 53 Sex: F Target: 1
Class: Lung Opacity
ID: 6ed32587-0518-4b4b-95e6-a4b0651b1a6d
Modality: CR Age: 30 Sex: M Target: 1
Class: Lung Opacity
ID: 3d7fb4b4-5e62-4591-aa4d-cbbc1ca3ffb7
Modality: CR Age: 47 Sex: F Target: 1
Class: Lung Opacity
ID: e3884258-8083-4b8e-b952-9adc91a7f0f3
Modality: CR Age: 31 Sex: M Target: 1
Class: Lung Opacity
ID: 36200569-a271-4fcf-819b-1bd00e626534
Modality: CR Age: 62 Sex: M Target: 1
Class: Lung Opacity
ID: 6e32b376-5740-4565-abd0-b41035f7db64
Modality: CR Age: 49 Sex: F Target: 1
Class: Lung Opacity

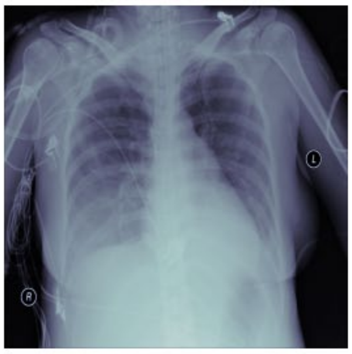
- We can see that bounding boxes are present for only patients with a lung opacity i.e. the patients having pneumonia.
- From DICOM file structure, I could understand that DICOM file can store many information as header information.
- From the Lung Opacity CXR images, it is clear that infection can be present in multiple area in a single CXR and the same is captured. Infected area can intersect on each other.
- No of opacites and bounding box information can be collected from the DICOM meta data.

DICOM images (Target = 0):

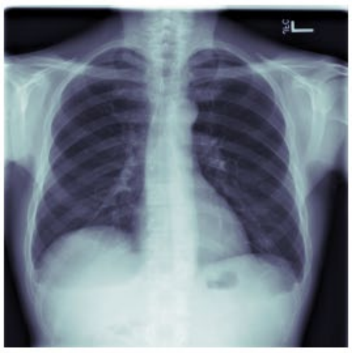
ID: 8041abe6-d563-4d6d-8c5d-2ad2c07aa0f1
Modality: CR Age: 43 Sex: F Target: 0
Class: No Lung Opacity / Not Normal
Window: nan:nan:nan:nan



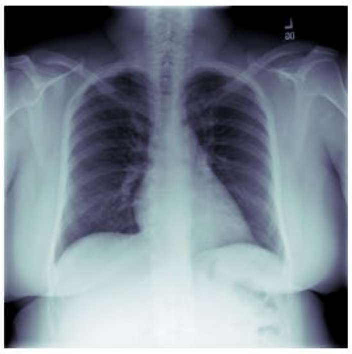
ID: 6ee0b617-c6ea-44d8-a3ef-4789aab257df
Modality: CR Age: 48 Sex: F Target: 0
Class: No Lung Opacity / Not Normal
Window: nan:nan:nan:nan



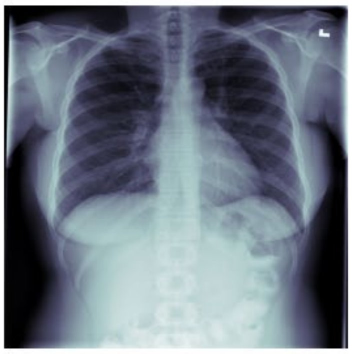
ID: 7d88aae7-e8a9-4f56-985a-72f238754764
Modality: CR Age: 66 Sex: M Target: 0
Class: Normal
Window: nan:nan:nan:nan



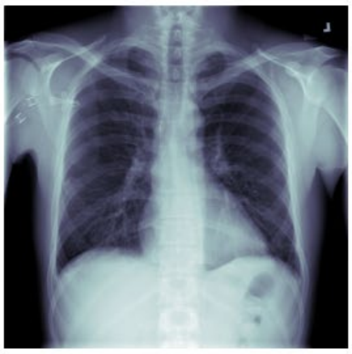
ID: fcabeae4-2631-494c-b7f9-f96d6da5e0ab
Modality: CR Age: 64 Sex: F Target: 0
Class: Normal
Window: nan:nan:nan:nan



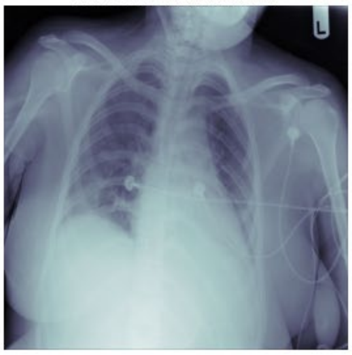
ID: d5915e55-c9c1-42bb-ad84-6ff7202aca34
Modality: CR Age: 38 Sex: F Target: 0
Class: Normal
Window: nan:nan:nan:nan



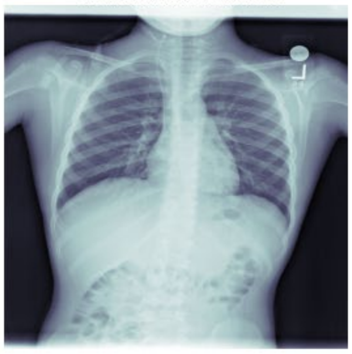
ID: 3a5327d8-8830-4ae2-bd6b-293f5aa42d4b
Modality: CR Age: 36 Sex: M Target: 0
Class: Normal
Window: nan:nan:nan:nan



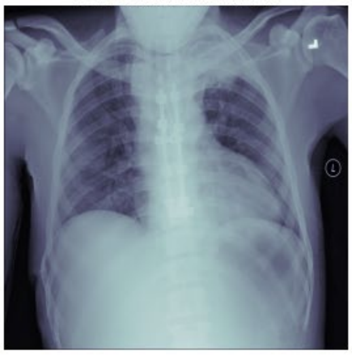
ID: c9081733-79a7-4a7a-8e44-2b5891f9ed51
Modality: CR Age: 25 Sex: F Target: 0
Class: No Lung Opacity / Not Normal
Window: nan:nan:nan:nan



ID: b0b392fd-3cb8-4888-bf0f-3b7ee0a935e2
Modality: CR Age: 10 Sex: M Target: 0
Class: Normal
Window: nan:nan:nan:nan

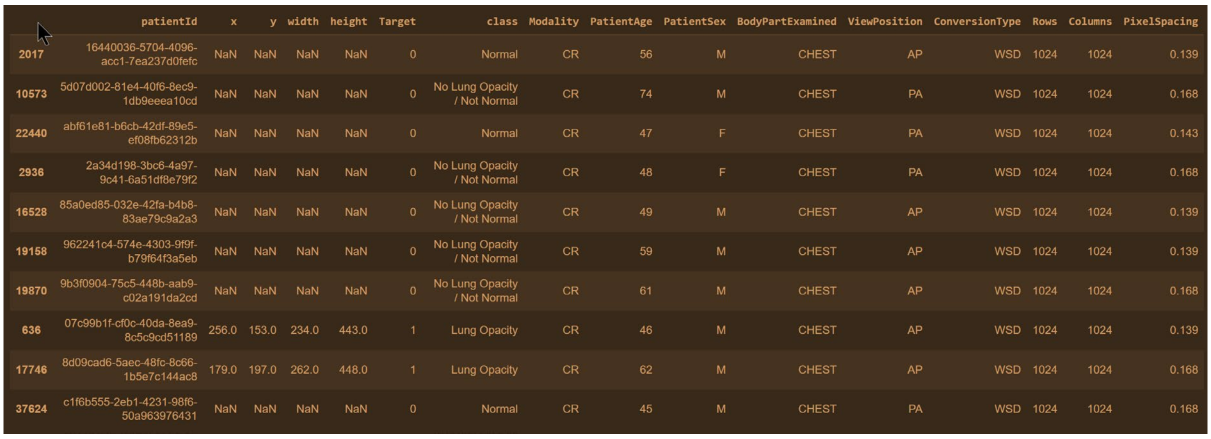


ID: 0d4192cd-b111-45df-86f6-712b04dad4a9
Modality: CR Age: 25 Sex: M Target: 0
Class: No Lung Opacity / Not Normal
Window: nan:nan:nan:nan



- We can see that bounding boxes are not present for patients with normal lungs
- The brightness, contrast and visibility of ribs is more in case of normal lungs.

Train Class dataset with Meta Info:

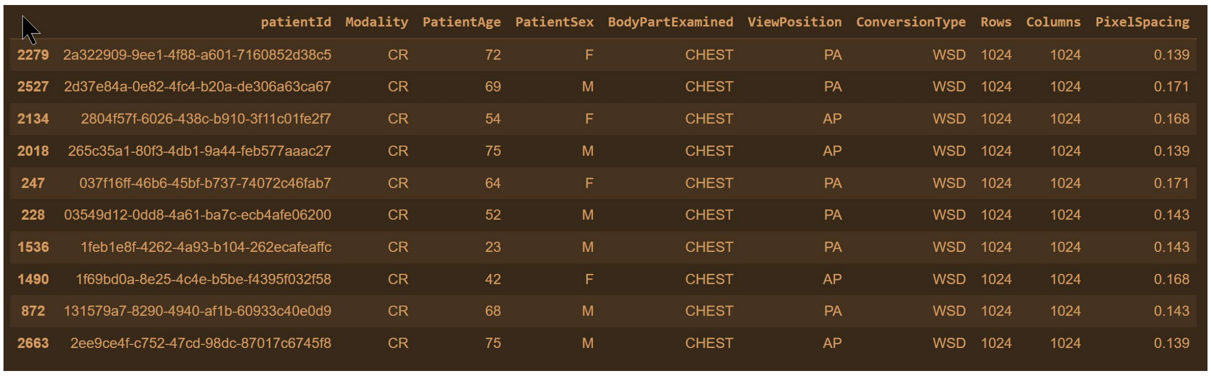


	patientId	x	y	width	height	Target	class	Modality	PatientAge	PatientSex	BodyPartExamined	ViewPosition	ConversionType	Rows	Columns	PixelSpacing
2017	16440036-5704-4096-acc1-7ea237d0fefc	NaN	NaN	NaN	NaN	0	Normal	CR	56	M	CHEST	AP	WSD	1024	1024	0.139
10573	5d07d002-81e4-40f6-8ec9-1db9eeea10cd	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	CR	74	M	CHEST	PA	WSD	1024	1024	0.168
22440	abf61e81-b6cb-42df-89e5-ef08fb62312b	NaN	NaN	NaN	NaN	0	Normal	CR	47	F	CHEST	PA	WSD	1024	1024	0.143
2936	2a34d198-3bc6-4a97-9c41-6a51df8e79f2	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	CR	48	F	CHEST	PA	WSD	1024	1024	0.168
16528	85a0ed85-032e-42fa-b4b8-83ae79c9a2a3	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	CR	49	M	CHEST	AP	WSD	1024	1024	0.139
19158	962241c4-574e-4303-9f9f-b79f64f3a5eb	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	CR	59	M	CHEST	AP	WSD	1024	1024	0.139
19870	9b3f0904-75c5-448b-aab9-c02a191da2cd	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal	CR	61	M	CHEST	AP	WSD	1024	1024	0.168
636	07c99b1f-cf0c-40da-8ea9-8c5c9cd51189	256.0	153.0	234.0	443.0	1	Lung Opacity	CR	46	M	CHEST	AP	WSD	1024	1024	0.139
17746	8d09cad6-5aec-48fc-8c66-1b5e7c144ac8	179.0	197.0	262.0	448.0	1	Lung Opacity	CR	62	M	CHEST	AP	WSD	1024	1024	0.168
37624	c1f6b555-2eb1-4231-98f6-50a963976431	NaN	NaN	NaN	NaN	0	Normal	CR	45	M	CHEST	PA	WSD	1024	1024	0.168

Important and relevant information from the DICOM meta data:

- Study Date & Time - Date and Time of the CXR was captured. Not much useful in our case.
- Modality - DICOM was captured with which equipment CXR, CT scan or MRI etc. Not useful.
- Patient's Name - Patients Name; PatientId is stored as Name. Required of file identification.
- PatientId - Patients ID (Unique)
- PatientSex - Male or Female. no influence on the final classification.
- PatientAge - Age of the Patient. No influence on the final classification.
- BodyPartExamined - Which part of the body is examined. We have got unique value 'CHEST' in this column, so this can be removed.
- Rows & Columns - Image size. Image height and width information is stored. All images are of same size (1024 x 1024). Not much useful.
- PixelSpacing - Actual Image array.
- Filename - DICOM File Name - patient ID is used as File Name, so this column can be dropped.

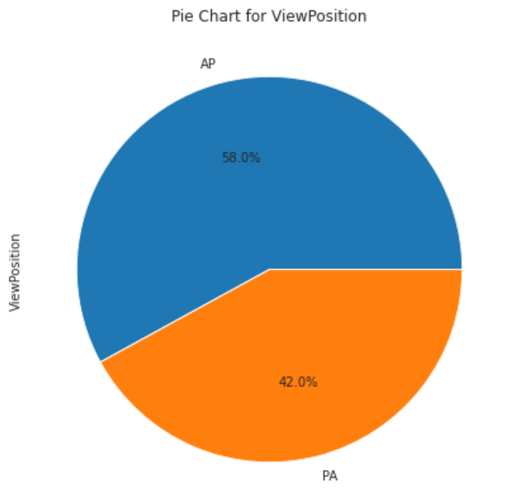
Test Class dataset with Meta info:

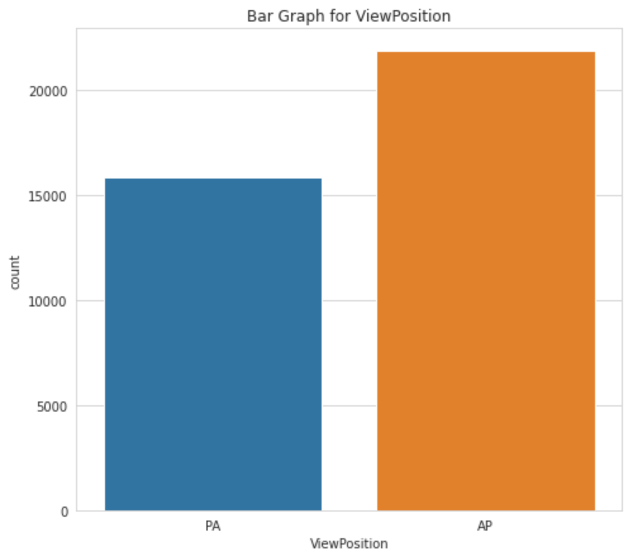


	patientId	Modality	PatientAge	PatientSex	BodyPartExamined	ViewPosition	ConversionType	Rows	Columns	PixelSpacing
2279	2a322909-9ee1-4f88-a601-7160852d38c5	CR	72	F	CHEST	PA	WSD	1024	1024	0.139
2527	2d37e84a-0e82-4fc4-b20a-de306a63ca67	CR	69	M	CHEST	PA	WSD	1024	1024	0.171
2134	2804f57f-6026-438c-b910-3f11c01fe2f7	CR	54	F	CHEST	AP	WSD	1024	1024	0.168
2018	265c35a1-80f3-4db1-9a44-feb577aaac27	CR	75	M	CHEST	AP	WSD	1024	1024	0.139
247	037f16ff-46b6-45bf-b737-74072c46fab7	CR	64	F	CHEST	PA	WSD	1024	1024	0.171
228	03549d12-0dd8-4a61-ba7c-ecb4afe06200	CR	52	M	CHEST	PA	WSD	1024	1024	0.143
1536	1feb1e8f-4262-4a93-b104-262ecafeaffc	CR	23	M	CHEST	PA	WSD	1024	1024	0.143
1490	1f69bd0a-8e25-4c4e-b5be-f4395f032f58	CR	42	F	CHEST	AP	WSD	1024	1024	0.168
872	131579a7-8290-4940-af1b-60933c40e0d9	CR	68	M	CHEST	PA	WSD	1024	1024	0.143
2663	2ee9ce4f-c752-47cd-98dc-87017c6745f8	CR	75	M	CHEST	AP	WSD	1024	1024	0.139

(Description is same as for Train Class dataset with Meta info.)

ViewPosition Attribute

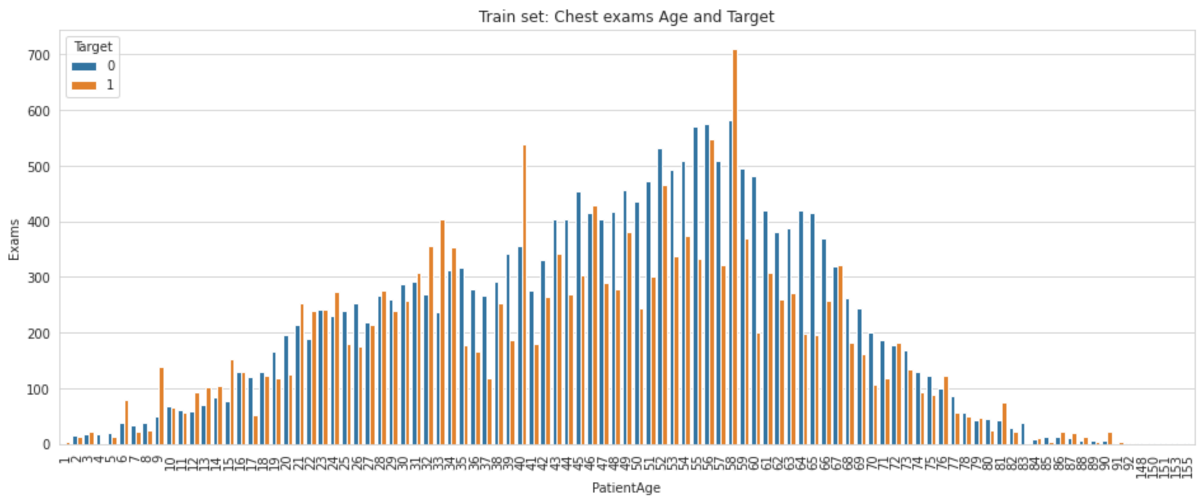


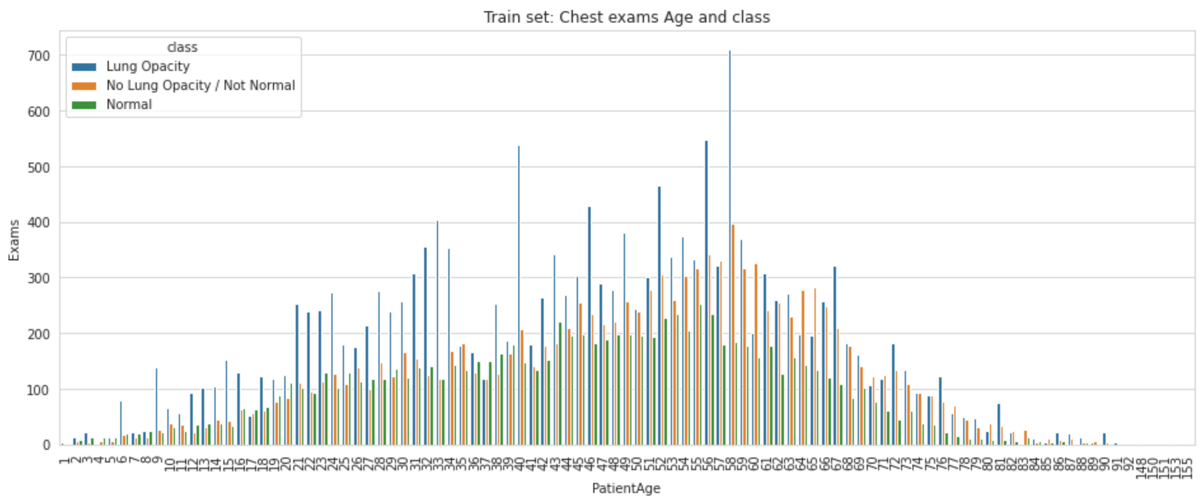


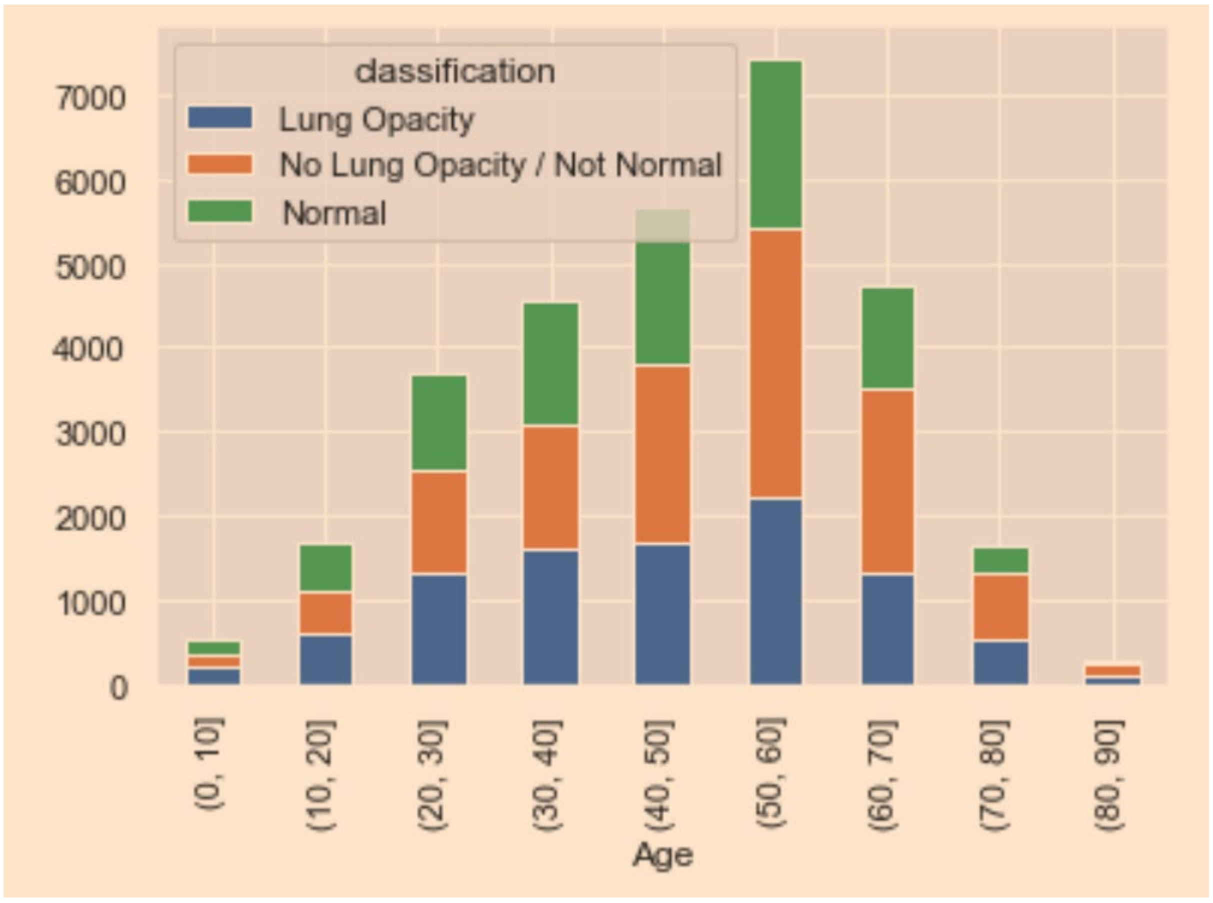
Both **AP** and **PA** body positions are present in the data. The meaning of these view positions are:

- **AP** - Anterior/Posterior: From back to front: When a chest x-ray is taken with the front against the film plate and the x-ray machine in back of the patient it is called an posteroanterior (PA) view.
- **PA** - Posterior/Anterior: From front to back. When a chest x-ray is taken with the back against the film plate and the x-ray machine in front of the patient it is called an anteroposterior (AP) view.

PatientAge Attribute

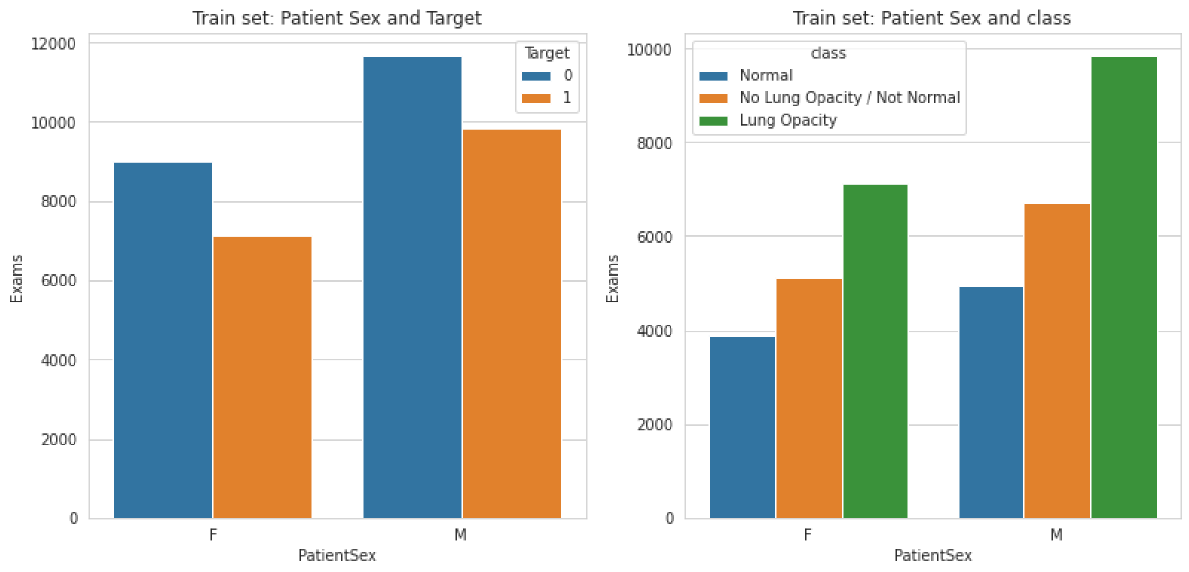






- There is a good distribution of people from all ages in the data. From the plot, it looks is normally distributed with majority of people around ages 40 to 60.
- Average age of people who are normal and who are affected by pneumonia are the same at around 47.
- Distribution of people's age is quite similar for both Normal and Pneumonia affected.
- Average age of people across different classes (Normal, Lung Opacity, No Lung Opacity/Not Normal) are same at around 47.
- Distribution of people's age is quite similar for class types.

PatientGender Attribute



- Gender - Target -> Number of male patients is quite similar in count as the number of female patients for both Normal and Pneumonia affected, with male patients slightly higher.
- Within Male and Female group, the rate of Pneumonia risk is same. So, Sex doesn't have any influence on the classification.

Model Evaluation and Comparison

Dataset Processing

In 2018, the Radiological Society of North America (RSNA) released a dataset on the detection and localization of pneumonia in chest X-rays. The dataset is from the National Institutes of Health (National Computational and Mathematical Methods in Medicine Institutes of Health) public chest X-ray images, with radiologist annotations. The detailed information of the RSNA pneumonia detection dataset can be found on the Kaggle website. The RSNA pneumonia data used in this experiment contains data of 26684 cases, of which only 6012 pneumonia images (22.03%), and the remaining 8851 normal images (31.19%) and 11821 images (44.77%), an image that is abnormal or has no turbidness in the lungs. Since the patient's chest pneumonia may have more than one location, there may be one to four locations.

Due to the characteristics of low brightness and low contrast of the chest X-ray image, to better detect the target area of pneumonia and improve the detection accuracy, the chest X-ray image can be pre-processed. In CXR images, normal lungs will not absorb X-rays, so it will appear black. The location of the pneumonia is a Gray dashed shadow or a cloudy area. To improve the recognition rate, contrast and brightness enhancement operations can be performed on X-ray images.

To improve the effect of model training, it is necessary to enhance the training set using data augmentation methods like Translation, Rotation, Flipped, Zooming, Stretched, Sheared, Elastic deformation etc. Due to time and computing resources limitation, we could not perform data augmentation part here in this experiment.

We have used 2400 images for training, and 600 images for testing our classification models. For this analysis, we resized the original 1024x1024x3 images to 128x128x3. For object detection models, we have used 2000 image for training and 2000 images for testing. For this analysis, we resized the original 1024x1024x3 images to 224x224x3.

Operations of data augmentation and brightness enhancement over the images

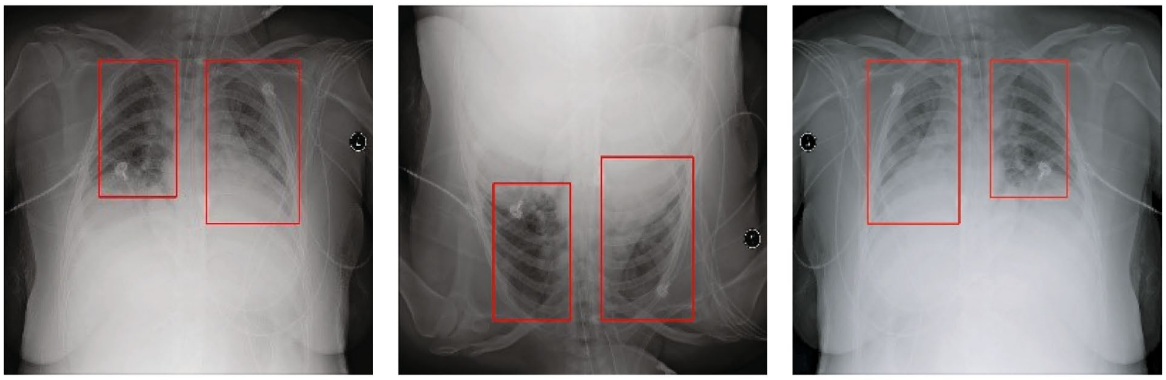


FIGURE 1: Flip processed image.

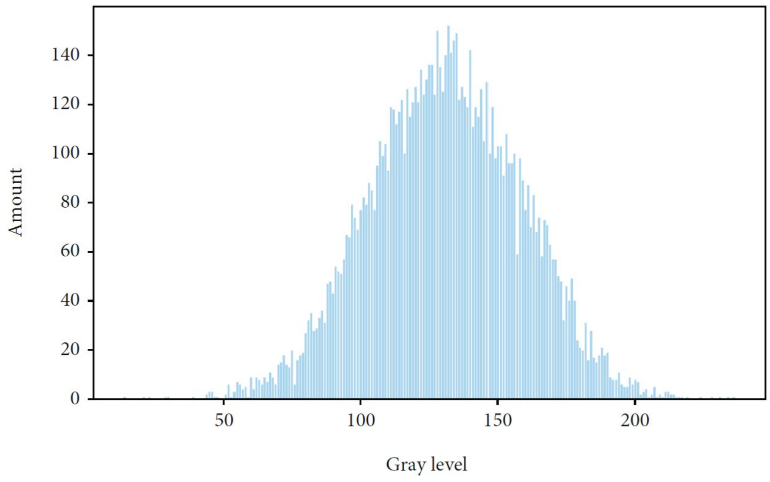


FIGURE 2: Average gray value distribution histogram.

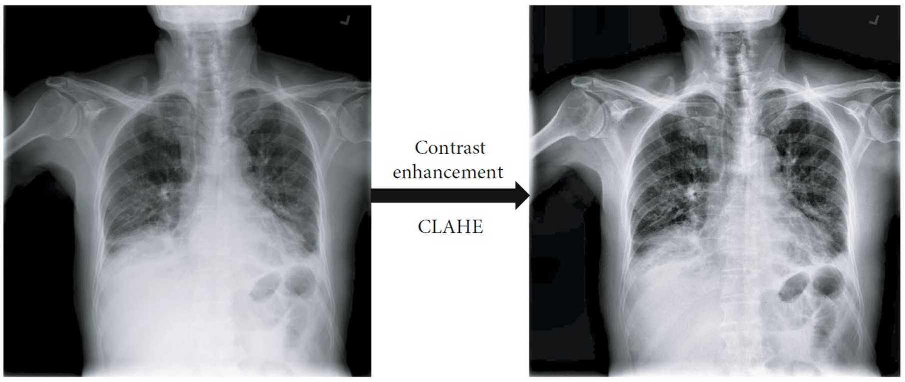


FIGURE 3: Chest X-ray image preprocessing.

Evaluation Measures

Confusion Matrix: Confusion Matrix usually causes a lot of confusion even in those who are using them regularly. Terms used in defining a confusion matrix are TP, TN, FP, and FN.

Use case: Let's take an example of a patient who has gone to a doctor with certain symptoms. Since it's the season of Covid, let's assume that he went with fever, cough, throat ache, and cold. These are symptoms that can occur during any seasonal changes too. Hence, it is tricky for the doctor to do the right diagnosis.

True Positive (TP):

Let's say the patient was actually suffering from Covid and on doing the required assessment, the doctor classified him as a Covid patient. This is called TP or True Positive. This is because the case is positive in real and at the same time the case was classified correctly. Now, the patient can be given appropriate treatment which means, the decision made by the doctor will have a positive effect on the patient and society.

False Positive (FP):

Let's say the patient was not suffering from Covid and he was only showing symptoms of seasonal flu but the doctor diagnosed him with Covid. This is called FP or False Positive. This is because the case was actually negative but was falsely classified as positive. Now, the patient will end up getting admitted to the hospital or home and will be given treatment for Covid. This is an unnecessary inconvenience for him and others as he will get unwanted treatment and quarantine. This is called **Type I Error**.

True Negative (TN):

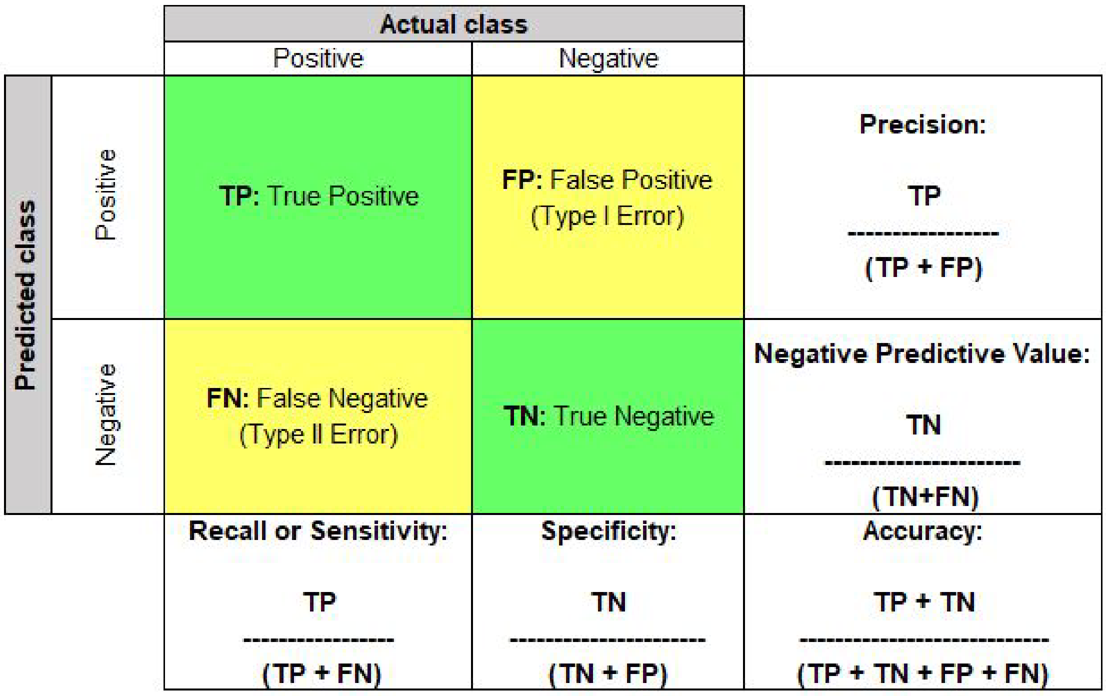
Let's say the patient was not suffering from Covid and the doctor also gave him a clean chit. This is called TN or True Negative. This is because the case was actually negative and was also classified as negative which is the right thing to do. Now the patient will get treatment for his actual illness instead of taking Covid treatment.

False Negative (FN):

Let's say the patient was suffering from Covid and the doctor did not diagnose him with Covid. This is called FN or False Negative as the case was actually positive but was falsely classified as negative. Now the patient will not get the right treatment and also, he will spread the disease to others. This is a highly dangerous situation in this example. This is also called **Type II Error.**

Summary: In this particular example, both FN and FP are dangerous and the classification model which has the lowest FN and FP values needs to be chosen for implementation. But in case there is a tie between few models which score very similar when it comes to FP and FN, in this scenario the model with the least FN needs to be chosen. This is because we simply cannot afford to have FNs! The goal of the hospital would be to not let even one patient go undiagnosed (no FNs) even if some patients get diagnosed wrongly (FPs) and asked to go under quarantine and special care.

Here is how a confusion matrix looks like:



		Actual class		
		Positive	Negative	
Predicted class	Positive	**TP:** True Positive	**FP:** False Positive (Type I Error)	**Precision:** $\frac{TP}{(TP + FP)}$
	Negative	**FN:** False Negative (Type II Error)	**TN:** True Negative	**Negative Predictive Value:** $\frac{TN}{(TN+FN)}$
		Recall or Sensitivity: $\frac{TP}{(TP + FN)}$	**Specificity:** $\frac{TN}{(TN + FP)}$	**Accuracy:** $\frac{TP + TN}{(TP + TN + FP + FN)}$

1. Accuracy:

Accuracy = (TP + TN) / (TP + FP +TN + FN)

This term tells us how many right classifications were made out of all the classifications. In other words, how many TPs and TNs were done out of TP + TN + FP + FNs. It tells the ratio of "True"s to the sum of "True"s and "False"s.

Use case: Out of all the patients who visited the doctor, how many were correctly diagnosed as Covid positive and Covid negative.

2. Precision:

Precision = TP / (TP + FP)

Out of all that were marked as positive, how many are actually truly positive.

Use case: Let's take another example of a classification algorithm that marks emails as spam or not. Here, if emails that are of importance get marked as positive, then useful emails will end up going to the "Spam" folder, which is dangerous. Hence, the classification model which has the least FP value needs to be selected. In other words, a model that has the highest precision needs to be selected among all the models.

3. Recall or Sensitivity:

Recall = TP/ (TN + FN)

Out of all the actual real positive cases, how many were identified as positive.

Use case: Out of all the actual Covid patients who visited the doctor, how many were actually diagnosed as Covid positive. Hence, the classification model which has the least FN value needs to be selected. In other words, a model that has the highest recall value needs to be selected among all the models.

4. Specificity:

Specificity = TN/ (TN + FP)

Out of all the real negative cases, how many were identified as negative.

Use case: Out of all the non-Covid patients who visited the doctor, how many were diagnosed as non-Covid.

5. F1-Score:

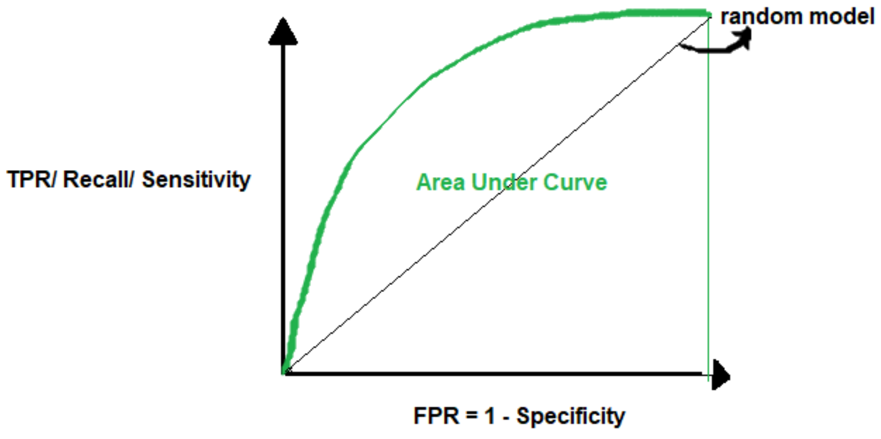
F1 score = 2* (Precision * Recall) / (Precision + Recall)

As we saw above, sometimes we need to give weightage to FP and sometimes to FN. F1 score is a weighted average of Precision and Recall, which means there is equal importance given to FP and FN. This is a very useful metric compared to "Accuracy". The problem with using accuracy is that if we have a highly imbalanced dataset for training (for example, a training dataset with 95% positive class and 5% negative class), the model will end up learning how to predict the positive class properly and will not learn how to identify the negative class. But the model will still have very high accuracy in the test dataset too as it will know how to identify the positives really well.

Use case: Let's take an example where we must give equal importance to both the classes – classify an email as Spam and non-Spam. Let's assume that the model was trained only a highly imbalanced training dataset. Here, Spam is "positive" and non-Spam is "negative" and the training dataset was 90% spam emails and 10% non-spam emails. A model with high accuracy will know to correctly identify all the spam emails but will have trouble identifying non-spam emails. Hence, a lot of important emails will end up going to the spam folder. But if we select a model that has a high F1 score, it would perform better in classifying non-spam from spam.

6. Area Under Curve (AUC) and ROC Curve:

AUC or Area Under Curve is used in conjecture with **ROC Curve** which is **Receiver Operating Characteristics Curve**. AUC is the area under the ROC Curve. So let's first understand the ROC Curve.



A ROC Curve is drawn by plotting TPR or True Positive Rate or Recall or Sensitivity (which we saw above) in the y-axis against FPR or False Positive Rate in the x-axis. FPR = 1- Specificity (which we saw above).

TPR = TP/ (TP + FN)

FPR = 1 – TN/ (TN+FP) = FP/ (TN + FP)

If we use a random model to classify, it has a 50% probability of classifying the positive and negative classes correctly. Here, the AUC = 0.5. A perfect model has a 100% probability of classifying the positive and negative classes correctly. Here, the AUC = 1. So, when we want to select the best model, we want a model that is closest to the perfect model. In other words, a model with AUC close to 1. When we say a model has a high AUC score, it means the model's ability to separate the classes is very high (high separability). This is a very important metric that should be checked while selecting a classification model.

7. Kappa

The **kappa** statistic compares the observed accuracy to an expected accuracy or the accuracy expected from random chance. One of the flaws of pure accuracy is that if a class is imbalanced then making predictions at random could give a high accuracy score. Kappa accounts for this by comparing the model accuracy to the expected accuracy based on the number of instances in each class.

Essentially it tells us how the model is performing compared to a model that classifies observations at random according to the frequency of each class.

$$kappa = (ObservedAccuracy - ExpectedAccuracy)/(1 - ExpectedAccuracy)$$

Kappa returns a value at or below 1, negative values are possible. One drawback of this statistic is that there is no agreed standard to interpret its values. Although, a general interpretation of the metric was given by Landis and Koch in 1977.

K	Interpretation
<0	Poor agreement
0.0-0.20	Slight agreement
0.21-0.40	Fair agreement
0.41-0.60	Moderate agreement
0.61-0.80	Substantial agreement
0.81-1.0	Almost perfect agreement

8. MCC

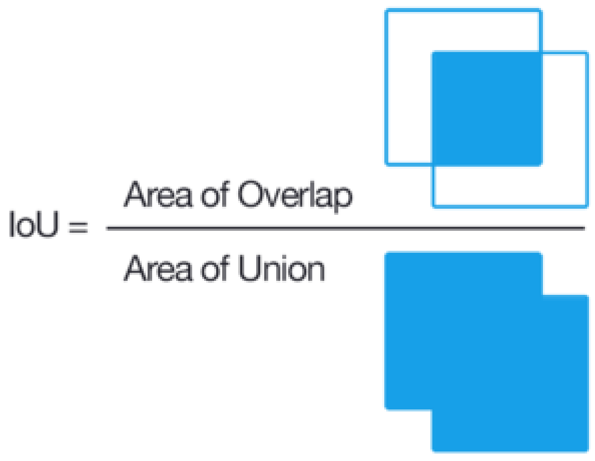
MCC (Matthews Correlation Coefficient) is generally considered one of the best measurements of performance for a classification model. This is largely because, unlike any of the previously mentioned metrics, it takes all possible prediction outcomes into account. If there are imbalances in the classes this will therefore be accounted for.

The MCC is essentially a correlation coefficient between the observed and predicted classifications. As with any correlation coefficient, its value will lie between -1.0 and +1.0. A value of +1 would indicate a perfect model.

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

9. Intersection-Over-Union (IoU, Jaccard Index)

The Intersection-Over-Union (IoU), also known as the Jaccard Index, is one of the most commonly used metrics in semantic segmentation… and for good reason. The IoU is a very straightforward metric that's extremely effective.



IoU calculation visualized. Source: Wikipedia

Before reading the following statement, take a look at the image to the left. Simply put, the IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth, as shown on the image to the left. This metric ranges from 0–1 (0–100%) with 0 signifying no overlap and 1 signifying perfectly overlapping segmentation.

For binary (two classes) or multi-class segmentation, the mean IoU of the image is calculated by taking the IoU of each class and averaging them. (It's implemented slightly differently in code).

10. Dice Coefficient (F1 Score)

Simply put, the **Dice Coefficient is 2 * the Area of Overlap divided by the total number of pixels in both images.** (See explanation of area of union).

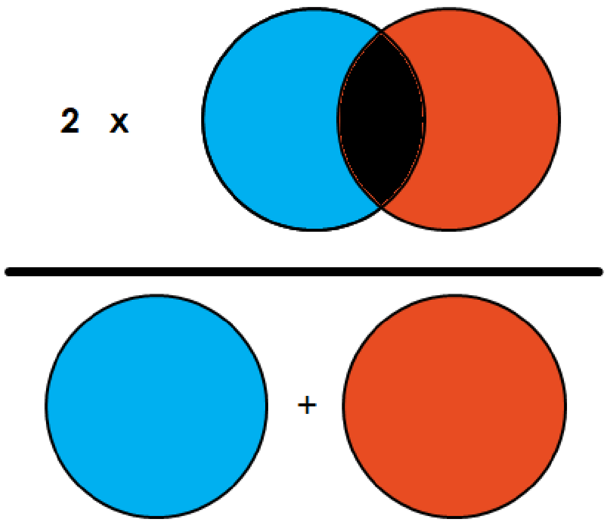


Illustration of Dice Coefficient. 2xOverlap/Total number of pixels

The Dice coefficient is very similar to the IoU. They are positively correlated, meaning if one says model A is better than model B at segmenting an image, then the other will say the same. Like the IoU, they both range from 0 to 1, with 1 signifying the greatest similarity between predicted and truth.

Summary for evaluation measures:

We have seen a various performance or evaluation measures that can be used to choose the best classification model. We also saw that depending upon the problem, we will use the appropriate metric. While accuracy is one of the widely used scores, in the case of imbalanced datasets, we will go for F1-score. Also, we saw the importance of selecting a model using the AUC value.

Assessing the performance of a classifier is generally not straightforward and highly dependent on the use case and available dataset. It is particularly important to understand the risk of being wrong in a particular direction so that you can produce a truly useful model.

Methods Used

As per the Project brief, the objective is to find the Pneumonia in the given CXR. Finding potential Pneumonia is critical in order to prioritize and expedite the clinician's review. As a next step, verifying the reason behind the abnormalities ('No Lung Opacity / Not Normal') is important as lead time should not cause fatalities among these patients. So I would suggest to build a model which produces 3 classification (0 - Normal, 1 - Pneumonia, 2- abnormalities).

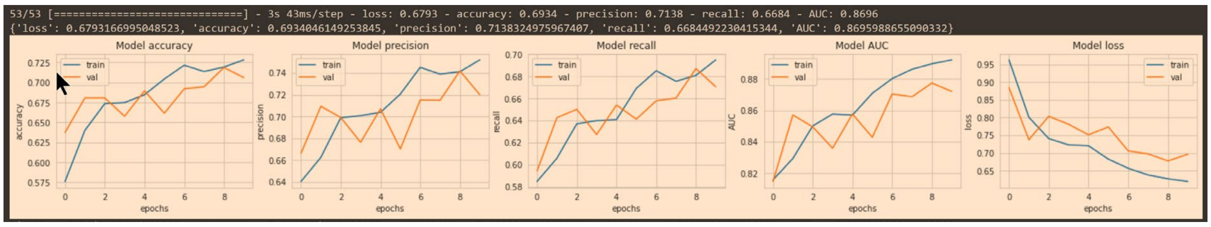
For image classification model training, we started with a simple fully connected neural network, then increased the complexity of CNN models to higher levels. We also experimented with state-of-the-art transfer learning models using ImageNet weights like MobileNetV2, VGG16, ResNet50, and EfficientNetV2B3. We also tried the hyperparameter tuning of a CNN model with smaller number of parameters, and observed improved performance measures with respect to base model and MobileNetV2 model.

For object detection model training, we used the UNet architecture with MobileNet and ResNet50 transfer learning models. We also realized that for semantic segmentation (binary classification) UNet is better and simpler than Faster R-CNN. We found that U-Net is also widely used in medical applications and AI models.

Results

In image classification: With a simple Fully Connected NN, we achieved a decent accuracy: 0.6328, precision: 0.6537, recall: 0.6235, AUC: 0.7454, and F1 Score: 0.6283. Base model training time was nearabout 23s. The highest performance was recorded with EfficientNetV2B3 model: accuracy: 0.6934, precision: 0.6945, recall: 0.7022, AUC: 0.8696, and F1 Score: 0.6961. The model training time was nearabout 1min.

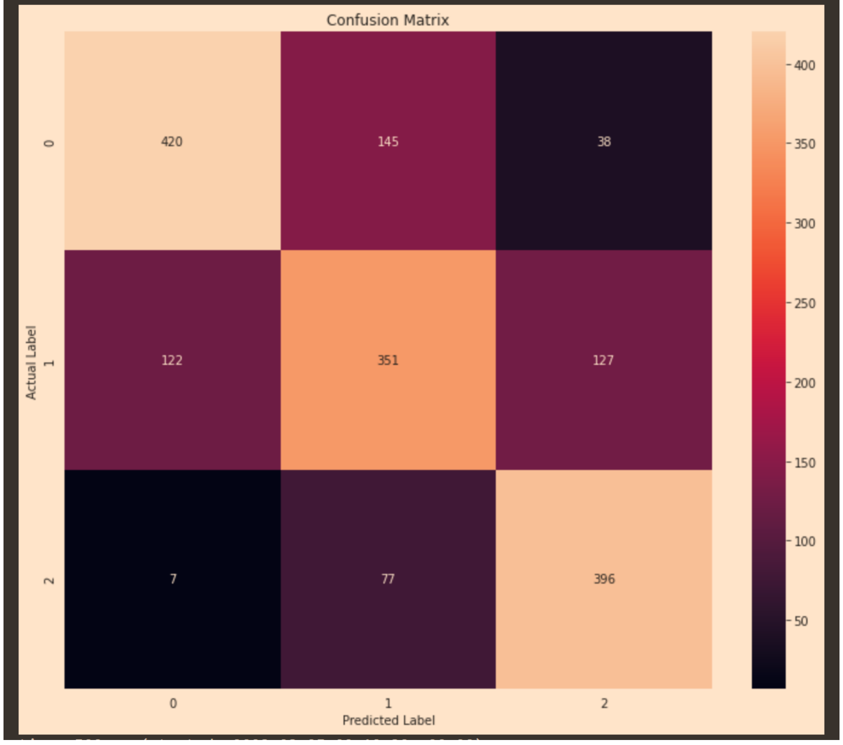
Model-7 (EfficientNetV2B3): Plots for accuracy, precision, recall, AUC, and loss



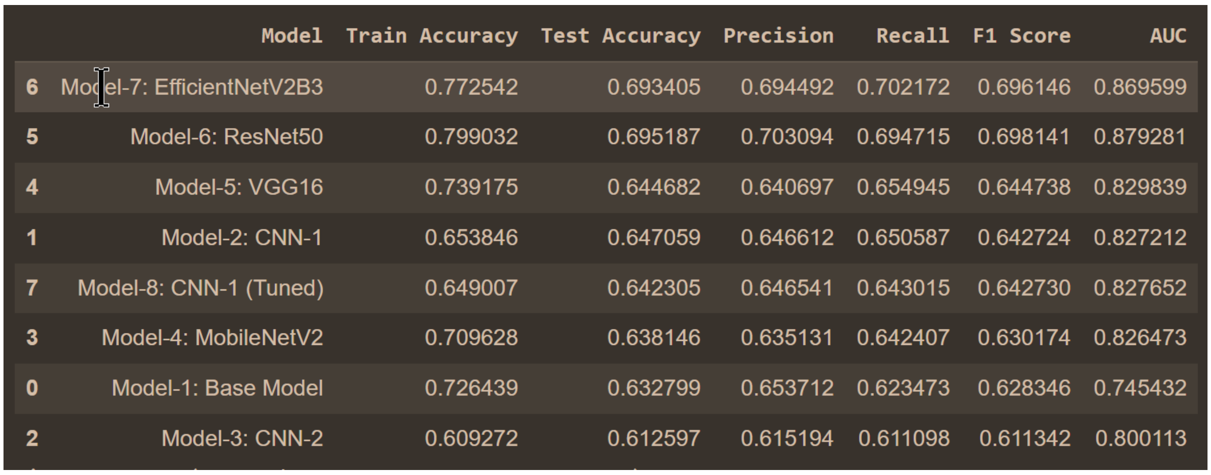
```
53/53 [==============================] - 3s 43ms/step - loss: 0.6793 - accuracy: 0.6934 - precision: 0.7138 - recall: 0.6684 - AUC: 0.8696
{'loss': 0.6793166995048523, 'accuracy': 0.6934046149253845, 'precision': 0.7138324975967407, 'recall': 0.6684492230415344, 'AUC': 0.8695988655090332}
```

Model-7 (EfficientNetV2B3): Classification Report and Confusion Matrix

```
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       603
           1       0.61      0.58      0.60       600
           2       0.71      0.82      0.76       480

    accuracy                           0.69      1683
   macro avg       0.69      0.70      0.70      1683
weighted avg       0.69      0.69      0.69      1683

Confusion Matrix Chart:
```

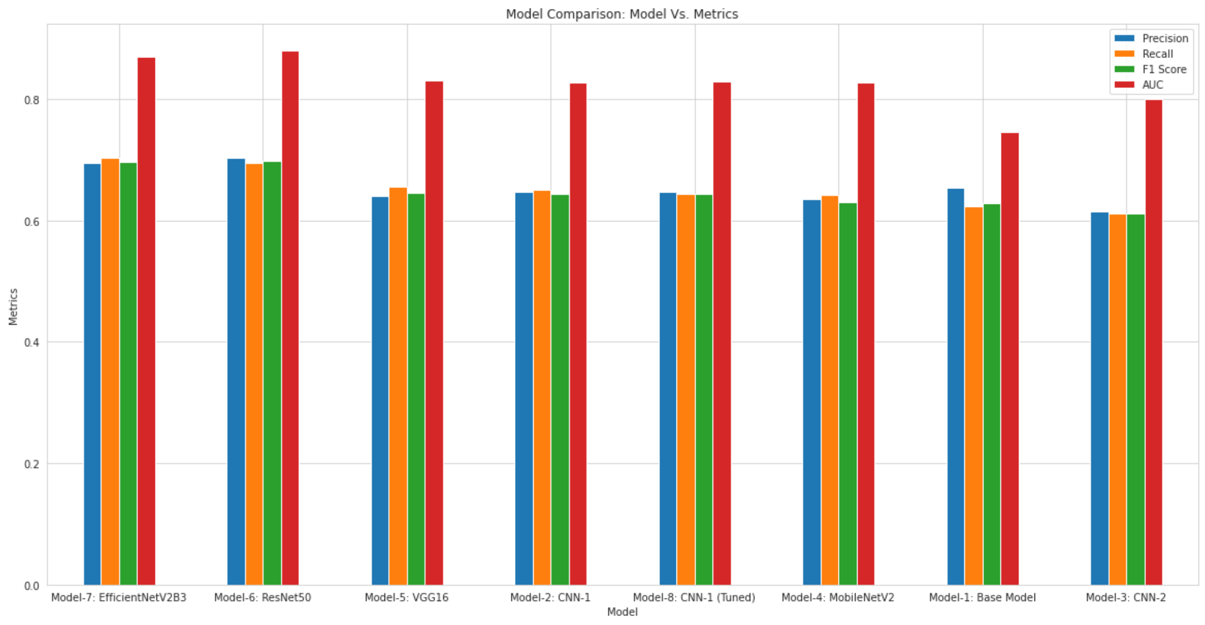


Model Comparison: We compared the performance of various classification models using the Accuracy (Train/Test), Precision, Recall, F1 Score, AUC metrics. (Sorted by F1 score)



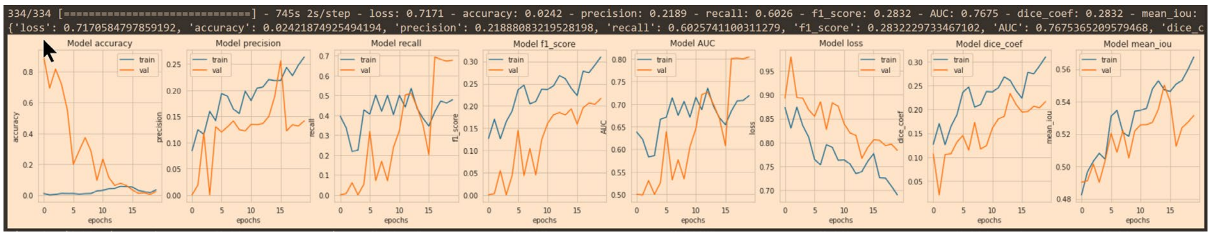
	Model	Train Accuracy	Test Accuracy	Precision	Recall	F1 Score	AUC
6	Model-7: EfficientNetV2B3	0.772542	0.693405	0.694492	0.702172	0.696146	0.869599
5	Model-6: ResNet50	0.799032	0.695187	0.703094	0.694715	0.698141	0.879281
4	Model-5: VGG16	0.739175	0.644682	0.640697	0.654945	0.644738	0.829839
1	Model-2: CNN-1	0.653846	0.647059	0.646612	0.650587	0.642724	0.827212
7	Model-8: CNN-1 (Tuned)	0.649007	0.642305	0.646541	0.643015	0.642730	0.827652
3	Model-4: MobileNetV2	0.709628	0.638146	0.635131	0.642407	0.630174	0.826473
0	Model-1: Base Model	0.726439	0.632799	0.653712	0.623473	0.628346	0.745432
2	Model-3: CNN-2	0.609272	0.612597	0.615194	0.611098	0.611342	0.800113

Model Comparison: Bar graph for Model Vs. Metrics



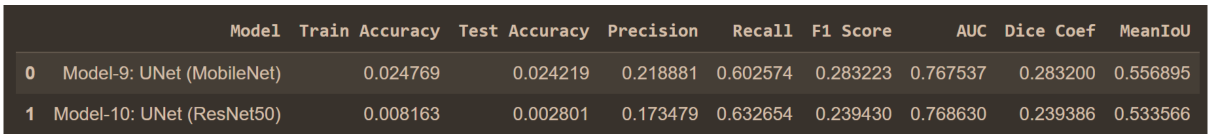
In semantic segmentation: We were able to achieve the highest performance with UNet (MobileNet) model: precision: 0.2188, recall: 0.6026, AUC: 0.7675, F1 Score: 0.2832, Dice Coef: 0.2832, and MeanIoU: 0.5569.

Model-9 (UNet - MobileNet): Plots for accuracy, precision, recall, AUC, F1 Score, loss, Dice Coef, and MeanIoU



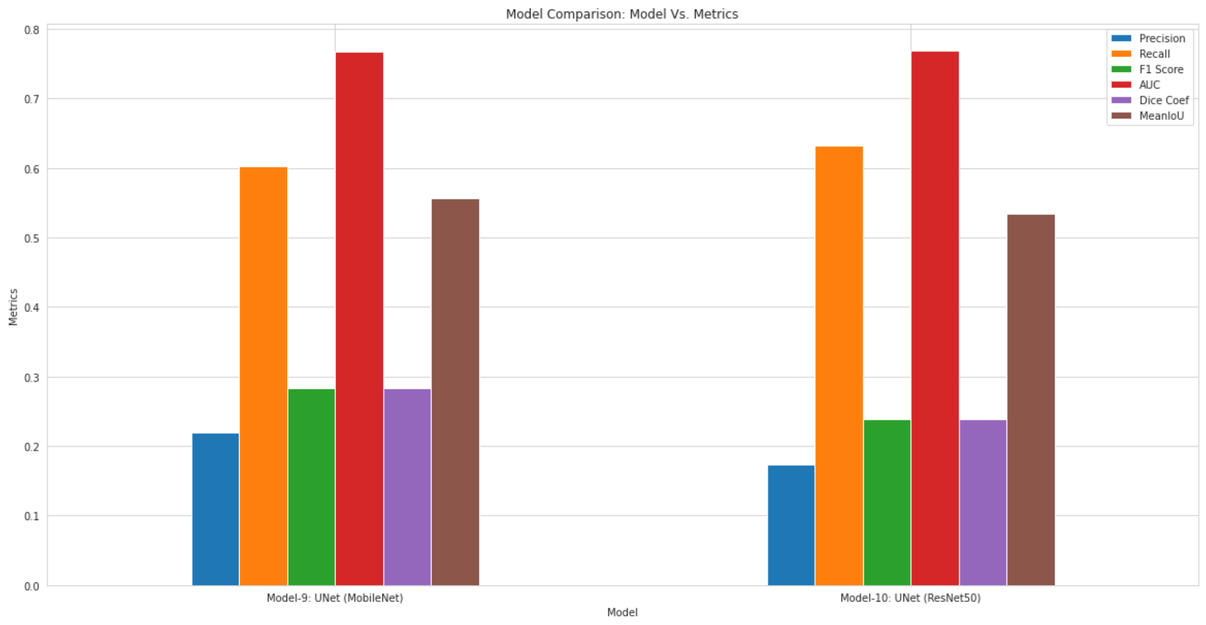
```
334/334 [==============================] - 745s 2s/step - loss: 0.7171 - accuracy: 0.0242 - precision: 0.2189 - recall: 0.6026 - f1_score: 0.2832 - AUC: 0.7675 - dice_coef: 0.2832 - mean_iou:
{'loss': 0.7170584797859192, 'accuracy': 0.02421874925494194, 'precision': 0.21888083219528198, 'recall': 0.6025741100311279, 'f1_score': 0.2832229733467102, 'AUC': 0.7675365209579468, 'dice_c
```

Model Comparison: We compared the performance of object detection models using the Accuracy (Train/Test), Precision, Recall, F1 Score, AUC, Dice Coef, and MeanIoU metrics. (Sorted by Dice Coef and MeanIoU)



	Model	Train Accuracy	Test Accuracy	Precision	Recall	F1 Score	AUC	Dice Coef	MeanIoU
0	Model-9: UNet (MobileNet)	0.024769	0.024219	0.218881	0.602574	0.283223	0.767537	0.283200	0.556895
1	Model-10: UNet (ResNet50)	0.008163	0.002801	0.173479	0.632654	0.239430	0.768630	0.239386	0.533566

Model Comparison: Bar graph for Model Vs. Metrics



Overview of Image Classification Methods

Image classification refers to a process in computer vision that can classify an image according to its visual content and features.

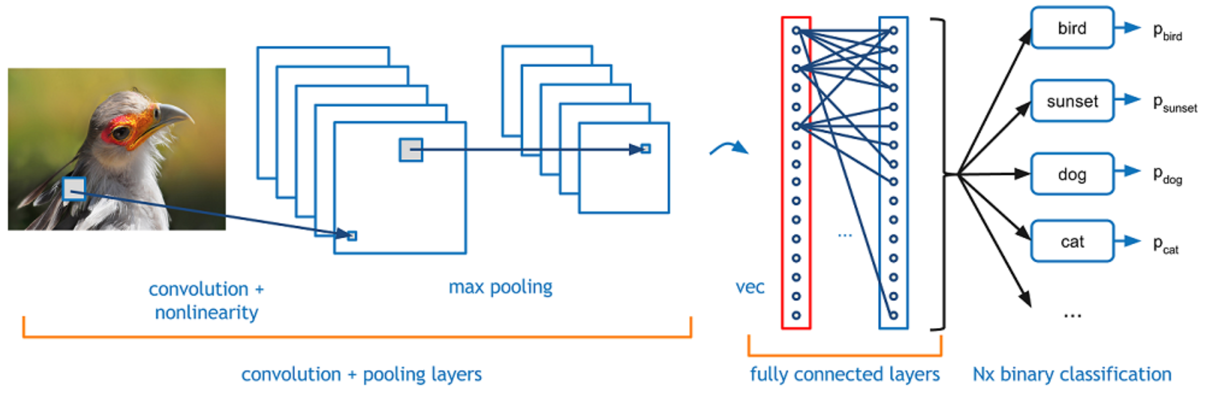
Supervised Classification: Supervised classification uses classification algorithms and regression techniques to develop predictive models. The algorithms include **linear regression, logistic regression, neural networks, decision tree, support vector machine, random forest, naive Bayes, and k-nearest neighbour.**

Unsupervised Classification: The user must have knowledge of the area being classified when the groupings of pixels with common characteristics produced by the computer have to be related to actual features on the ground**. Some of the most common algorithms used in unsupervised learning include cluster analysis, anomaly detection, neural networks, and approaches for learning latent variable models.**

Artificial Neural Network: Inspired by the properties of biological neural networks, Artificial Neural Networks are statistical learning algorithms and are used for a variety of tasks, from relatively simple classification tasks to computer vision and speech recognition. Artificial neural networks are implemented as a system of interconnected processing elements, called nodes, which are functionally analogous to biological neurons. The connections between different nodes have numerical values, called weights, and by altering these values in a systematic way, the network is eventually able to approximate the desired function. The hidden layers can be thought of as individual feature detectors, recognizing more and more complex patterns in the data as it is propagated throughout the network. For example, if the network is given a task to recognize a face, the first hidden layer might act as a line detector, the second hidden takes these lines as input and puts them together to form a nose, the third hidden layer takes the nose and matches it with an eye and so on, until finally the whole face is constructed. This hierarchy enables the network to eventually recognize very complex objects. **The different types of artificial neural network are convolutional neural network, feedforward neural network, probabilistic neural network, time delay neural**

network, deep stacking network, radial basis function network, and recurrent neural network.

Convolutional Neural Network: Convolutional Neural Network (CNN, or ConvNet) are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal pre-processing. It is a special architecture of artificial neural networks. Convolutional neural network uses some of its features of visual cortex and have therefore achieved state of the art results in computer vision tasks. Convolutional neural networks are comprised of two very simple elements, namely convolutional layers and pooling layers. Although simple, there are near-infinite ways to arrange these layers for a given computer vision problem. The elements of a convolutional neural network, such as convolutional and pooling layers, are relatively straightforward to understand. The challenging part of using convolutional neural networks in practice is how to design model architectures that best use these simple elements. The reason why convolutional neural network is hugely popular is because of their architecture, the best thing is there is no need of feature extraction. The system learns to do feature extraction and the core concept is, it uses convolution of image and filters to generate invariant features which are passed on to the next layer. The features in next layer are convoluted with different filters to generate more invariant and abstract features and the process continues till it gets final feature/output which is invariant to occlusions. **The most commonly used architectures of convolutional neural network are LeNet, AlexNet, ZFNet, GoogLeNet, VGGNet, and ResNet.**



Overview of Object Detection Methods

R-CNN

The R-CNN method uses a process called selective search to find out the objects from the image. This algorithm generates a large number of regions and collectively works on them. These collections of regions are checked for having objects if they contain any object. The success of this method depends on the accuracy of the classification of objects.

Fast-RCNN

The Fast-RCNN method uses the structure of R-CNN along with the SPP-net (Spatial Pyramid Pooling) to make the slow R-CNN model faster. The Fast-RCNN uses the SPP-net to calculate the CNN representation for the whole image only once. It then uses this representation to calculate the CNN representation for each patch generated by the selective search approach of R-CNN. The Fast-RCNN makes the process train from end-to-end.

The Fast-RCNN model also includes the bounding box regression along with the training process. This makes both the processes of localization and classification in a single process, making the process faster.

Faster-RCNN

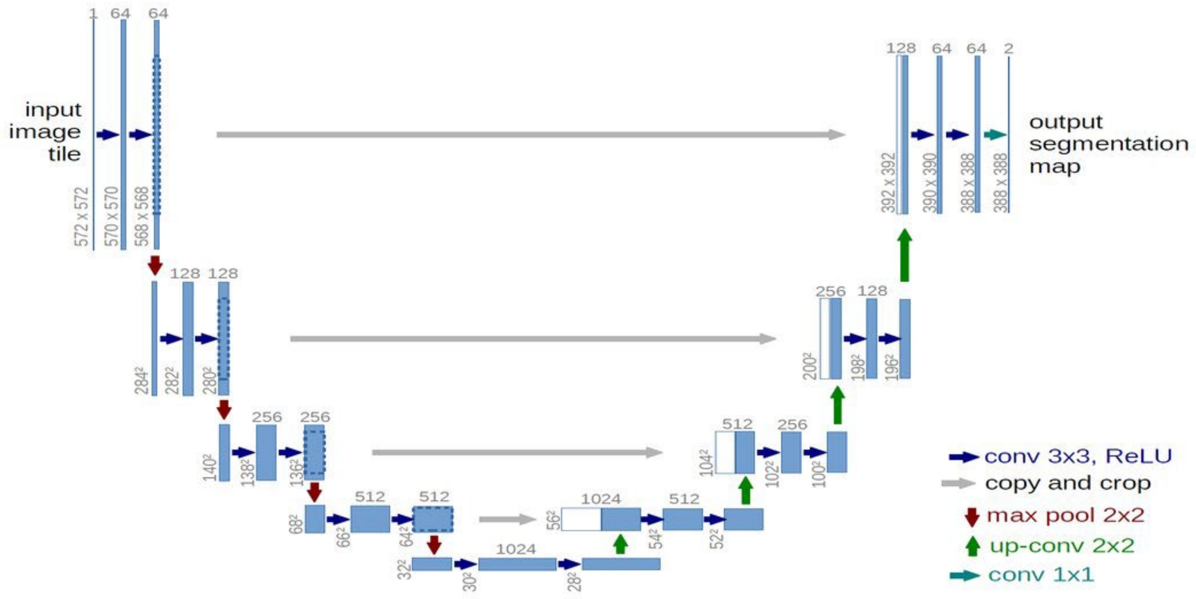
The Faster-RCNN method is even faster than the Fast-RCNN. The Fast-RCNN was fast but the process of selective search and this process is replaced in Faster-RCNN by implementing RPN (Region Proposal Network). The RPN makes the process of selection faster by implementing a small convolutional network, which in turn, generates regions of interest. Along with RPN, this method also uses Anchor Boxes to handle the multiple aspect ratios and scale of objects. Faster-RCNN is one of the most accurate and efficient object detection algorithms.

You Look Only Once (YOLO) Family

The R-CNN approach that we saw above focuses on the division of a visual into parts and focus on the parts that have a higher probability of containing an object, whereas the YOLO framework focuses on the entire image as a whole and predicts the bounding boxes, then calculates its class probabilities to label the boxes. The family of YOLO frameworks is very fast object detectors. Different YOLO versions: YOLOv1, YOLOv2, YOLOv3

Image segmentation with UNet

U-Net gets its name from its architecture. The “U” shaped model comprises convolutional layers and two networks. First is the encoder, which is followed by the decoder. With the U-Net, we can solve the two questions of segmentation: “what” and “where.”



Model Architecture   Source: https://arxiv.org/abs/1505.04597

The encoder network is also called the contracting network. This network learns a feature map of the input image and tries to solve our first question- “what” is in the image? It is similar to any classification task we perform with convolutional neural networks except for the fact that in a U-Net, we do not have any fully connected layers in the end, as the output we require now is not the class label but a mask of the same size as our input image.

This encoder network consists of 4 encoder blocks. Each block contains two convolutional layers with a kernel size of 3*3 and valid padding, followed by a Relu activation function. This is inputted to a max pooling layer with a kernel size of 2*2. With the max pooling layer, we have halved the spatial dimensions learned, thereby reducing the computation cost of training the model.

In between the encoder and decoder network, we have the bottleneck layer. This is the bottommost layer, as we can see in the model above. It consists of 2 convolutional layers followed by Relu. The output of the bottleneck is the final feature map representation.

Now, what makes U-Net so good at image segmentation is skip connections and decoder networks. What we have done till now is similar to any CNN. The skip connections and decoder network separate the U-Net from other CNNs.

The decoder network is also called the expansive network. Our idea is to upsample our feature maps to the size of our input image. This network takes the feature map from the bottleneck layer and generates a segmentation mask with the help of skip connections. The decoder network tries to solve our second question-"where" is the object in the image? It consists of 4 decoder blocks. Each block starts with a transpose convolution (indicated as up-conv in the diagram) with a kernel size of 2*2. This output is concatenated with the corresponding skip layer connection from the encoder block. After which, two convolutional layers with a kernel size of 3*3 are used, followed by a Relu activation function.

Skip connections are indicated with a grey arrow in the model architecture. Skip connections help us use the contextual feature information collected in the encoder blocks to generate our segmentation map. The idea is to use our high-resolution features learned from the encoder blocks (through skip connections) to help us project our feature map (output of the bottleneck layer). This helps us answer "where" is our object in the image?

A 1*1 convolution follows the last decoder block with sigmoid activation which gives the output of a segmentation mask containing pixel-wise classification. This way, it could be said that the contracting path passes across information to the expansive path. And thus, we can capture both the feature information and localization with the help of a U-Net.

In conclusion, we have understood the following about image segmentation and U-Net:

1. Image segmentation can be thought of as a combination of classification and localization tasks.
2. We wish to answer 2 questions in image segmentation – "what" and "where"?
3. The encoder path of the U-Net answers "what" is in the image and acts similarly to any CNN.
4. The decoder path of the U-Net answers "where" is the object in the image and produces a mask of the size of the original image.
5. Skip connections enable us to use the features learned in the encoder network to help generate our output mask.

We have implemented a U-Net for biomedical (Pneumonia detection) segmentation tasks through the above application. At the same time, we have an idea of image segmentation and how a U-Net approaches any segmentation task. We have gone in-depth on the model architecture and the function of each layer.

This model can further be improved by having pre-trained weights for our encoder network. Several variations of the U-Net have also come about, but the basic intuition and working remain the same.

Conclusion and Next Steps

Limitations

While we develop the **Convolutional Neural Networks** (CNN) to classify the images, it is often observed the model starts overfitting when we try to improve the accuracy. Very frustrating, hence we list down the following techniques which would improve the model performance without overfitting the model on the training data:

1. **Data normalization**
   We may normalize the image tensors by subtracting the mean and dividing by the standard deviation of pixels across each channel. Normalizing the data prevents the pixel values from any one channel from disproportionately affecting the losses and gradients.

2. **Data augmentation**
   Of course, it would be great if we could get hold of additional high-quality training data to expand our dataset, but that is not always feasible. We can try to increase our data by data augmentation. Rotation, flipping, random cropping, introducing colour variations, etc. are some methods that might help.

   Also, keep in mind that an artificially inflated dataset is never as good as one of the same size that contains all authentic, real images. A mirrored image contains much of the same information as its original, we merely hope it will delay the network from overfitting and hope that it will learn the important patterns instead.

3. **Batch normalization**
   After each convolutional layer, we can add a batch normalization layer, which normalizes the outputs of the previous layer. This is somewhat similar to data normalization, except it’s applied to the outputs of a layer, and the mean and standard deviation are learned parameters.

4. **Learning rate scheduling**
   Instead of using a fixed learning rate, we can use a learning rate scheduler, which will change the learning rate after every batch of training. There are many strategies for varying the learning rate during training.

5. **Weight Decay:** We may add weight decay to the optimizer, yet another regularization technique which prevents the weights from becoming too large by adding an additional term to the loss function.

6. **Gradient clipping**: We can also add gradient clip pint, which helps to limit the values of gradients to a small range to prevent undesirable changes in model parameters due to large gradient values during training.
7. **Adam optimizer**: Instead of Adam, we can use SGD and other optimizers which uses techniques like momentum and adaptive learning rates for faster training. There are many other optimizers to choose from and experiment with.
8. **Lower the learning rate:** This is a bit of side note, but try lowering the learning rate. Your network seems to overfit in only a few epochs which is very fast. Obviously, lowering the learning rate will not combat overfitting but it will happen more slowly. This means that we can hopefully find an epoch with better overall performance before overfitting takes place. Note that a lower learning rate will never magically make a bad-performing network good. It's just one way to locate a set of parameters that performs a tad bit better.
9. **Randomize the training data order:** During training, the training data is presented in batches to the network. This often happens in a fixed order over all iterations. This may lead to certain biases in the network. First of all, make sure that the training data is shuffled at least once. We do not want to present the classes one by one, for example first all plane images, then all chairs, etc... This could lead to the network unlearning much of the first class by the end of each epoch. Also, reshuffle the training data between epochs. This will again avoid potential minor biases because of training data order.
10. **Improve the network design:** You've designed a convolutional neural network with only two convolution layers and two fully connected layers. Maybe this model is too shallow to learn to differentiate between the different classes. Know that the convolution layers tend to first pick up small visual features and then tend to combine these in higher level patterns. So maybe adding a third convolution layer may help the network identify more meaningful patterns. Obviously, network design is something you'll have to experiment with and making networks overly deep or complex is also a pitfall to watch out for!
11. **Transfer Learning**: We can make use of pre-trained models and fine tune it (complete model or later layers only) for our application.
12. **Ensembling:** We can use multiple models and combine their results to get better results.
13. Increase the number of epochs: more training, better performance.
14. Update the model with optimum settings: e.g. change the optimizer, learning rate, loss function etc.
15. Also try ML approaches with DL; Sometimes ML outperforms the DL.

Implications

Traditional deep networks reduce partial information when extracting features, and this affects their capacities for image detection. The residual neural network structure of the low complexity dilated bottleneck avoids the computational delay associated with network depth and the problems involved in the large numbers of parameters associated with network width. We can extract deep feature semantic information from an image, thereby avoiding loss of information, the Soft-NMS operation is performed on props generated by the RPN; useful information is retained while filtering the extra box, and finally the model is able to detect pneumonia to a high level of precision.

The AI models that we have built can affect the bigger healthcare and biomedical industry in many ways like:

- Assist research activities and may reduce physician stress
- Streamlines tasks and provides the real time data
- Increased efficiency of the diagnostic process
- Reduced overall costs of running the business
- Saves time and resources
- Enhanced patient care
- Easy information sharing
- Better prevention care
- Assist research and may reduce physician stress

Closing Reflections

1. We understood the value of CRISP-DM project management framework in the realm of data science domain.
2. We have completed the EDA, and analysed the important and relevant attributes of our dataset in greater details.
3. We have done the pre-processing of the DICOM images, and collected additional information from the images. In the process, we learned to access, analyse and process the images for data modelling.
4. We have built the various CNN models for classification with and without transfer learning. In the process, we learned to use various state-of-the art pre trained models with ImageNet weights.
5. We have built UNET models for semantic segmentation of images.
6. We learned to differentiate and make use of various evaluation metrics to understand the quality of our models.
7. Given the time and computing resources, we can also experiment with Mask RCNN, Yolo, CheXNet and other more advanced models.
8. We can improve the Dice Coef. and MeanIoU by image augmentation, increasing the sample size, image size and hyperparameter tuning.
9. We would have done some up sampling of the 'Lung Opacity' class using augmentations. This would have helped us improving Recall value for 'Lung Opacity' class.

10. Classification error to understand the level of confidence of our various metrics is also a great factor to consider while quoting the performance metrics of various models.
11. Understanding Kappa, MCC and mAP are also great factors to consider for model evaluation measures.
12. We can also implement the Bayesian optimization techniques for hyperparameter tuning.
13. Overall, it was a great experience working on this project and we learned a lot in the journey.

References:

1. [Towards Data Science](https://towardsdatascience.com)
2. [Kaggle. Kaggle Code](https://www.kaggle.com/code)
3. [KdNuggets](https://www.kdnuggets.com/)
4. [AnalyticsVidhya](https://www.analyticsvidhya.com/blog/)
5. [Wikipedia](https://en.wikipedia.org/)
6. [Numpy](https://numpy.org/)
7. [Pandas](https://pandas.pydata.org/)
8. [SciPy](https://scipy.org/)
9. [MatplotLib](https://matplotlib.org/)
10. [Seaborn](https://seaborn.pydata.org/)
11. [Python](https://www.python.org/)
12. [Plotly](https://plotly.com/)
13. [Bokeh](https://docs.bokeh.org/en/latest/)
14. [RStudio](https://www.rstudio.com/)
15. [MiniTab](https://www.minitab.com/en-us/)
16. [Anaconda](https://www.anaconda.com/)
17. [PapersWithCode](https://paperswithcode.com/)