Name: Anik Manik

Email address: iamanik4@gmail.com

Contact number: 9477672426

Anydesk address: 400 728 410

Years of Work Experience: 2.6 years

Date: 24th Jan 2021

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from PIL import Image
import cv2
import glob
import glob2
%matplotlib inline
from tqdm import tqdm
import shutil
```

In [ ]:
```
!pip install pydicom
import pydicom
```

```
Collecting pydicom
  Downloading https://files.pythonhosted.org/packages/f4/15/df16546bc59bfca390cf072d473fb2c8acd423163
6f64356593a63137e55/pydicom-2.1.2-py3-none-any.whl (1.9MB)
     |████████████████████████████████| 1.9MB 15.4MB/s
Installing collected packages: pydicom
Successfully installed pydicom-2.1.2
```

https://www.kaggle.com/seesee/siim-train-test (https://www.kaggle.com/seesee/siim-train-test)

In [ ]:
```
# download the dataset
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36" --header="Accept: t
ext/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: http
s://www.kaggle.com/" --header="Cookie: ext_name=ojplmecpdpgccookcobabopnaifgidhf" --header="Connectio
n: keep-alive" "https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archive.zip?X-Goo
g-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%
2F20210208%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210208T141344Z&X-Goog-Expires=259199&X-Goog-
SignedHeaders=host&X-Goog-Signature=23ee9f5a6583d7a24693e852051d1caed22b1b383d36cc2cccce0c5fdd1278f6a7
82480e292f768b8712f7266e70b4c9d73081c111af3a1615e76399fdca4d805157d2c9ab14d52b5f8d994670b3a291a8f24956
1f41758d5c1fdd42daac1388cac1bbda4d42bf30878dfbb01421c7714a1101f630c3881bee0db37906ac71bf2dc7b97b3365e0
1a14cf450c59e96eee1908d632930dda5447da2daf78e48d851ccaf371e13428c536a6e5864b7740b2da1b602fedbc6708e9e3
abce94c6d2695dabfba0dca03338209ca77c5078d40a94514547175ada4d8c1d91219cffceafcdc8d7c1373137196056609 4d9
33238bbdd297bc7af5977a8c402626f0aab3a8" -c -O 'archive.zip'
```

```
--2021-02-08 14:14:03--  https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archiv
e.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gservice
account.com%2F20210208%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210208T141344Z&X-Goog-Expires=2
59199&X-Goog-SignedHeaders=host&X-Goog-Signature=23ee9f5a6583d7a24693e852051d1caed22b1b383d36cc2cccce
0c5fdd1278f6a782480e292f768b8712f7266e70b4c9d73081c111af3a1615e76399fdca4d805157d2c9ab14d52b5f8d99467
0b3a291a8f249561f41758d5c1fdd42daac1388cac1bbda4d42bf30878dfbb01421c7714a1101f630c3881bee0db37906ac71
bf2dc7b97b3365e01a14cf450c59e96eee1908d632930dda5447da2daf78e48d851ccaf371e13428c536a6e5864b7740b2da1
b602fedbc6708e9e3abce94c6d2695dabfba0dca03338209ca77c5078d40a94514547175ada4d8c1d91219cffceafcdc8d7c1
3731371960566094d933238bbdd297bc7af5977a8c402626f0aab3a8
Resolving storage.googleapis.com (storage.googleapis.com)... 172.253.62.128, 172.217.9.208, 172.253.1
22.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.253.62.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2059765561 (1.9G) [application/zip]
Saving to: ‘archive.zip’

archive.zip         100%[===================>]   1.92G  33.4MB/s    in 19s

2021-02-08 14:14:22 (106 MB/s) - ‘archive.zip’ saved [2059765561/2059765561]
```

In [ ]:
```
# unzip the data
!unzip -qq 'archive.zip'
```

In [ ]:
```
# read the given train csv file
train_df = pd.read_csv('siim/train-rle.csv')
train_df.head()
```

Out[ ]:

	ImageId	EncodedPixels
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...

This csv file contains the ImageID and RLE encoded masks.

In [ ]:
```
# check the properties of the train dataframe
train_df.describe()
```

Out[ ]:

	ImageId	EncodedPixels
count	12954	12954
unique	12047	3577
top	1.2.276.0.7230010.3.1.4.8323329.1851.151787516...	-1
freq	10	9378

Out of 12954 imageID, 12047 are unique. It means there are duplicates.

In [ ]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12954 entries, 0 to 12953
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ImageId         12954 non-null  object
 1    EncodedPixels  12954 non-null  object
dtypes: object(2)
memory usage: 202.5+ KB
```

There are no null values in this dataset.

Drop the duplicate ImageIDs

In [ ]:
```
# drop the duplicate ImageIDs
train_df.drop_duplicates(subset ="ImageId", keep = 'first', inplace = True)
```

In [ ]:
```
train_df.describe()
```

Out[ ]:

	ImageId	EncodedPixels
count	12047	12047
unique	12047	2670
top	1.2.276.0.7230010.3.1.4.8323329.13474.15178752...	-1
freq	1	9378

Now there are no duplicates left in the dataset, all 12047 ImageIDs are unique.

In [ ]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12047 entries, 0 to 12953
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ImageId         12047 non-null  object
 1    EncodedPixels  12047 non-null  object
dtypes: object(2)
memory usage: 282.4+ KB
```

Create 2 separate directory to store Train and Test images

In [ ]:
```
from tqdm import tqdm
import shutil

# create 2 separate directories for train and test dicom images
train_images_dicom = 'siim/train_images_dicom/'
test_images_dicom = 'siim/test_images_dicom/'

if not os.path.isdir(train_images_dicom):
    os.makedirs(train_images_dicom)
if not os.path.isdir(test_images_dicom):
    os.makedirs(test_images_dicom)
```

Move the images to respective directory

In [ ]:
```
# move all train dicom images from 'dicom-images-train' to 'train_images_dicom' in a single directory
train_path = 'siim/dicom-images-train/'
train_dcm_list = glob2.glob(os.path.join(train_path, '**/*.dcm'))
for filename in tqdm(train_dcm_list):
  shutil.move(str(filename), train_images_dicom)

# move all test dicom images from 'dicom-images-test' to 'test_images_dicom' in a single directory
test_path = 'siim/dicom-images-test/'
test_dcm_list = glob2.glob(os.path.join(test_path, '**/*.dcm'))
for filename in tqdm(test_dcm_list):
  shutil.move(str(filename), test_images_dicom)
```

```
100%|██████████| 12089/12089 [00:00<00:00, 25946.12it/s]
100%|██████████| 3205/3205 [00:00<00:00, 29636.99it/s]
```

Create a dataframe containing ImageId, EncodedPixels and ImagePath

In [ ]:
```
# create a dataframe containing ImageId, EncodedPixels and ImagePath
# dicom images has '.dcm' extension, concating '.dcm' with ImageId I get filename
# train_df['DICOM_Path'] = os.path.join(train_images_dicom, (train_df['ImageId']+'.dcm'))
train_df['Image_Path'] = train_images_dicom + train_df['ImageId']+'.dcm'
train_df.head()
```

Out[ ]:

	ImageId	EncodedPixels	Image_Path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/train_images_dicom/1.2.276.0.7230010.3.1....

In [ ]:
```
# save the .csv file for further use
train_df.to_csv('train_dicom_imageid_path.csv', index=False)
```

In [ ]:
```
# create test dataframe from the test ImageIDs given
test_df = pd.DataFrame(columns = ['ImageId', 'Image_Path'])
test_data = os.listdir(test_images_dicom)
for file in test_data:
  test_df = test_df.append({'ImageId':file[:-4], 'Image_Path':test_images_dicom+file}, ignore_index =
True)
test_df.head()
```

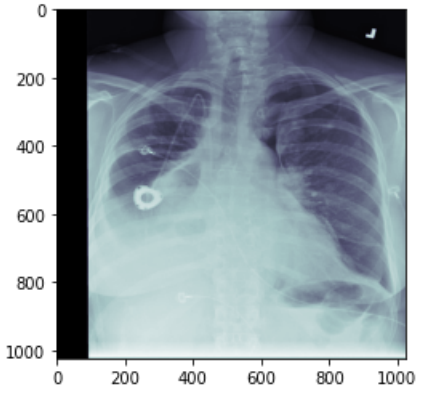
Out[ ]:

	ImageId	Image_Path
0	ID_6a5c8492a	siim/test_images_dicom/ID_6a5c8492a.dcm
1	ID_ba5a9ad4f	siim/test_images_dicom/ID_ba5a9ad4f.dcm
2	ID_2c87f564e	siim/test_images_dicom/ID_2c87f564e.dcm
3	ID_ac43e5bb6	siim/test_images_dicom/ID_ac43e5bb6.dcm
4	ID_eaad30cb8	siim/test_images_dicom/ID_eaad30cb8.dcm

Display a sample image

In [ ]:
```
# displaying a sample image
# https://stackoverflow.com/questions/48185544/read-and-open-dicom-images-using-python
image_row = 4
sample_img = pydicom.read_file(train_df['Image_Path'][image_row]).pixel_array
print('ImageId = '+str(train_df['ImageId'][image_row]))
plt.imshow(sample_img, cmap='bone')
plt.show()
```

```
ImageId = 1.2.276.0.7230010.3.1.4.8323329.10016.1517875220.992175
```



Read metadata from dicom image and print

In [ ]:
```
# read metadata from dicom image and print
# https://medium.com/@ashkanpakzad/reading-editing-dicom-metadata-w-python-8204223a59f6
sample_metadata = pydicom.dcmread(train_df['Image_Path'][image_row])
print(sample_metadata)
```

```
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: 1.2.276.0.7230010.3.1.4.8323329.10016.151787522
0.992175
(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: 1.2.276.0.7230010.3.1.4.8323329.10016.151787522
0.992175
(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: 'c844072c-3ba6-48d2-b009-4eb329fecd26'
(0010, 0020) Patient ID                          LO: 'c844072c-3ba6-48d2-b009-4eb329fecd26'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'F'
(0010, 1010) Patient's Age                       AS: '57'
(0018, 0015) Body Part Examined                  CS: 'CHEST'
(0018, 5101) View Position                       CS: 'PA'
(0020, 000d) Study Instance UID                  UI: 1.2.276.0.7230010.3.1.2.8323329.10016.151787522
0.992174
(0020, 000e) Series Instance UID                 UI: 1.2.276.0.7230010.3.1.3.8323329.10016.151787522
0.992173
(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: [0.139, 0.139]
(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 128324 elements
```

Extract sample informations from metadata

In [ ]:
```
# extract few informations from metadata
print("Patient Age = " +str(sample_metadata.PatientAge))
print("Patient Sex = " +str(sample_metadata.PatientSex))
print("Patient Modality = " +str(sample_metadata.Modality))
print("Patient BodyPart = " +str(sample_metadata.BodyPartExamined))
print("Patient ViewPosition = " +str(sample_metadata.ViewPosition))
```

```
Patient Age = 57
Patient Sex = F
Patient Modality = CR
Patient BodyPart = CHEST
Patient ViewPosition = PA
```

Extract informations from DICOM image

In [ ]:
```
age = []
sex = []
modality = []
bodyPart = []
viewPosition = []

for file in tqdm(train_df['Image_Path']):
  metadata = pydicom.dcmread(file)
  age.append(metadata.PatientAge)
  sex.append(metadata.PatientSex)
  modality.append(metadata.Modality)
  bodyPart.append(metadata.BodyPartExamined)
  viewPosition.append(metadata.ViewPosition)
```

```
100%|██████████| 12047/12047 [00:10<00:00, 1194.85it/s]
```

Add these informations in dataframe column

In [ ]:
```
train_df['Age'] = age
train_df['Sex'] = sex
train_df['Modality'] = modality
train_df['BodyPart'] = bodyPart
train_df['ViewPosition'] = viewPosition
train_df.head()
```

Out[ ]:

	ImageId	EncodedPixels	Image_Path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/train_images_dicom/1.2.276.0.7230010.3.1....

In [ ]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12047 entries, 0 to 12953
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ImageId         12047 non-null  object
 1    EncodedPixels  12047 non-null  object
 2   Image_Path      12047 non-null  object
 3   Age             12047 non-null  object
 4   Sex             12047 non-null  object
 5   Modality        12047 non-null  object
 6   BodyPart        12047 non-null  object
 7   ViewPosition    12047 non-null  object
dtypes: object(8)
memory usage: 1.1+ MB
```

In [ ]:
```
train_df.describe()
```

Out[ ]:

	ImageId	EncodedPixels	Image_
count	12047	12047	12047
unique	12047	2670	12047
top	1.2.276.0.7230010.3.1.4.8323329.13474.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3
freq	1	9378	1

In [ ]:
```
# remove extra space in EncodedPixels column
train_df.rename(columns = {' EncodedPixels':'EncodedPixels'}, inplace = True)
```

Add a column whether the image is with pneumothorax or without pneumothorax

In [ ]:
```
# add a column whether the image is with pneumothorax or without pneumothorax
train_df['Whether_Pneumothorax'] = np.where(train_df['EncodedPixels']=='-1', 'No', 'Yes')
train_df.head(10)
```

Out[ ]:

	ImageId	EncodedPixels	Image_Path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/train_images_dicom/1.2.276.0.7230010.3.1....
5	1.2.276.0.7230010.3.1.4.8323329.11444.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
6	1.2.276.0.7230010.3.1.4.8323329.32219.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
7	1.2.276.0.7230010.3.1.4.8323329.32395.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
8	1.2.276.0.7230010.3.1.4.8323329.10623.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....
9	1.2.276.0.7230010.3.1.4.8323329.12095.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1....

In [ ]:
```
# save the .csv file for further use
train_df.to_csv('train_data_info.csv', index=False)
```

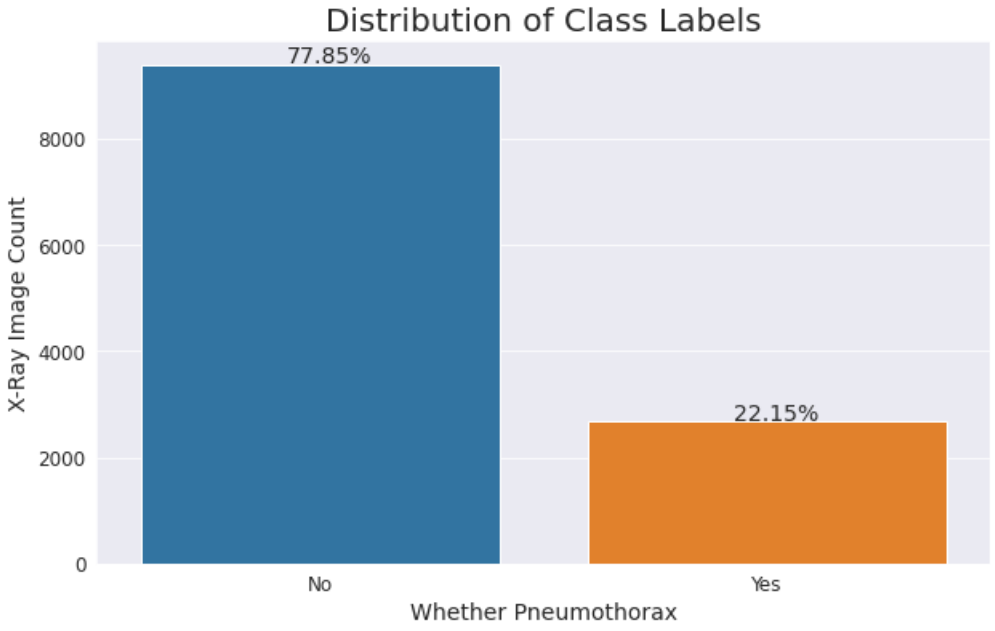
Define a function to display percentage in graph

In [ ]:
```
# https://stackoverflow.com/questions/31749448/how-to-add-percentages-on-top-of-bars-in-seaborn
def percentage_with_hue(plot, feature, Number_of_categories, hue_categories, x_pos=0, y_pos=0, fontsiz
e = 12):
    '''
    This function prints percentage of a barplot when plotted with hue.
    '''
    # get the height of each bar
    a = [p.get_height() for p in plot.patches]
    patch = [p for p in plot.patches]
    for i in range(Number_of_categories):
        total = feature.value_counts().values[i]
        for j in range(hue_categories):
            # calculate percentage of the category
            percentage = '{:.2f}%'.format(100 * a[(j*Number_of_categories + i)]/total)
            # set the x and y position of the percentage value to look good
            x = patch[(j*Number_of_categories + i)].get_x() + patch[(j*Number_of_categories + i)].get_
width() / 2 + x_pos
            y = patch[(j*Number_of_categories + i)].get_y() + patch[(j*Number_of_categories + i)].get_
height() + y_pos
            ax.annotate(percentage, (x, y), fontsize = fontsize)
    plt.show()

def percentage_without_hue(ax, feature, x_pos=0, y_pos=0, fontsize = 12 ):
    '''
    This function prints percentage of a barplot when plotted without hue.
    '''
    total = len(feature)
    for p in ax.patches:
        # calculate percentage of the category
        percentage = '{:.2f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width()/2 + x_pos # To adjust the position of the percentage value
        y = p.get_y() + p.get_height() + y_pos # To adjust the position of the percentage value
        ax.annotate(percentage, (x, y),ha='center', size = fontsize)
    plt.show()
```

Check distribution of class labels i.e. Pneumothorax or not

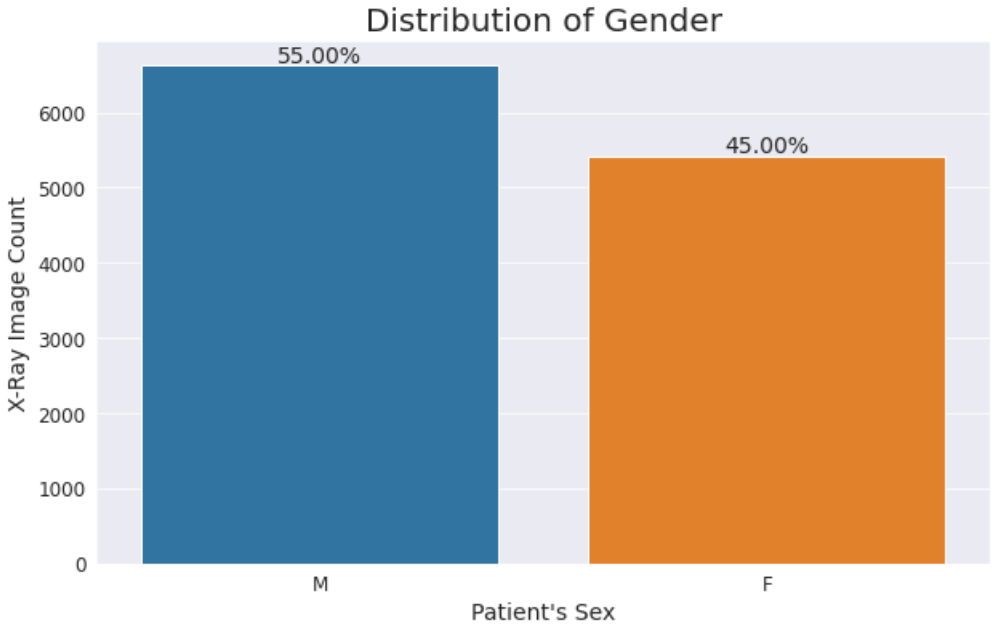
In [ ]:
```
# Check distribution of class labels i.e. Pneumothorax or not
plt.figure(figsize=(10,6))
sns.set_style('darkgrid')
ax = sns.countplot(x='Whether_Pneumothorax',data=train_df)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.title('Distribution of Class Labels', fontsize=20)
plt.xlabel('Whether Pneumothorax', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_without_hue(ax, train_df['Whether_Pneumothorax'], 0.02,40,14)
```



Among all the x-ray images 77.85% are without Pneumothorax and 22.15% are with pneumothorax

Check distribution of Gender

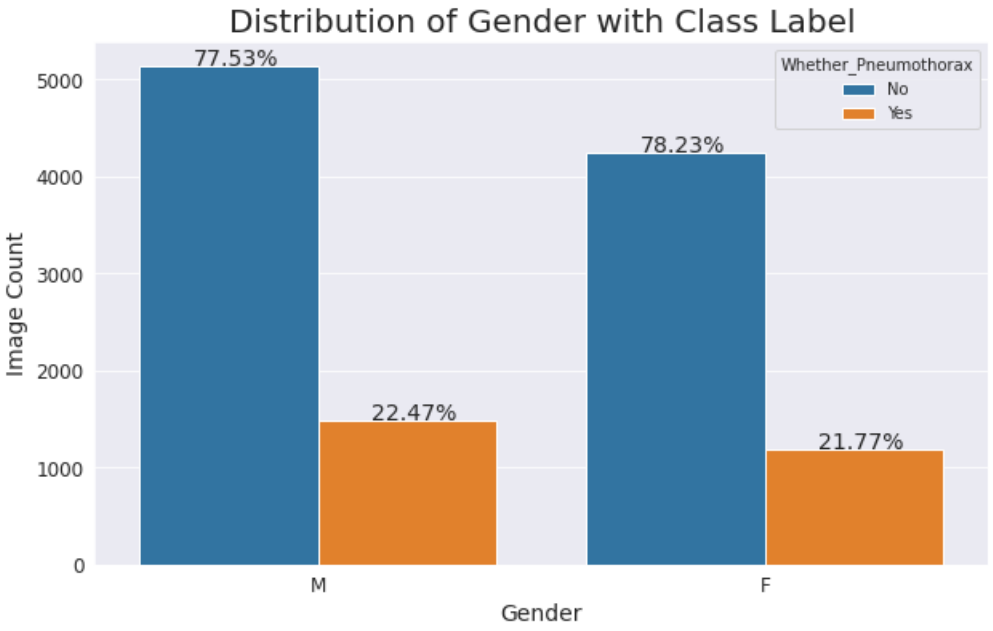
In [ ]:
```
# Check distribution of Gender
plt.figure(figsize=(10,6))
sns.set_style('darkgrid')
ax = sns.countplot(x='Sex',data=train_df)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.title('Distribution of Gender', fontsize=20)
plt.xlabel('Patient\'s Sex', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_without_hue(ax, train_df['Sex'], 0,40,14)
```



There are 55% male patients and 45% female patients in the given dataset.

Distribution of Gender along with class label

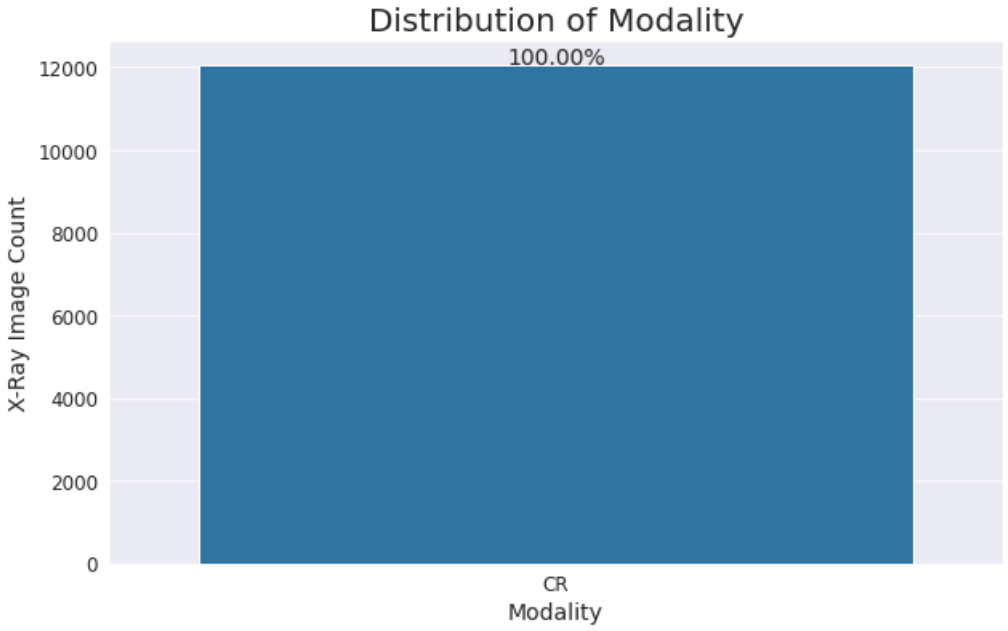
In [ ]:
```
# Distribution of Gender along with class label
plt.figure(figsize=(10, 6))
ax = sns.countplot(x='Sex',data=train_df, hue = 'Whether_Pneumothorax')
plt.title('Distribution of Gender with Class Label', size = 20)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.xlabel('Gender', size = 14)
plt.ylabel('Image Count', size = 14)
percentage_with_hue(ax, train_df['Sex'], 2,2,-0.08,10, 14)
```



Among all the male patients 77.53% are without pneumothorax and 22.47% are with pneumothorax and among all the female patients 78.23% are without pneumothorax and 21.77% are with pneumothorax. Pneumothorax distribution is almost similar for both male and female patients.

Check distribution of Modality

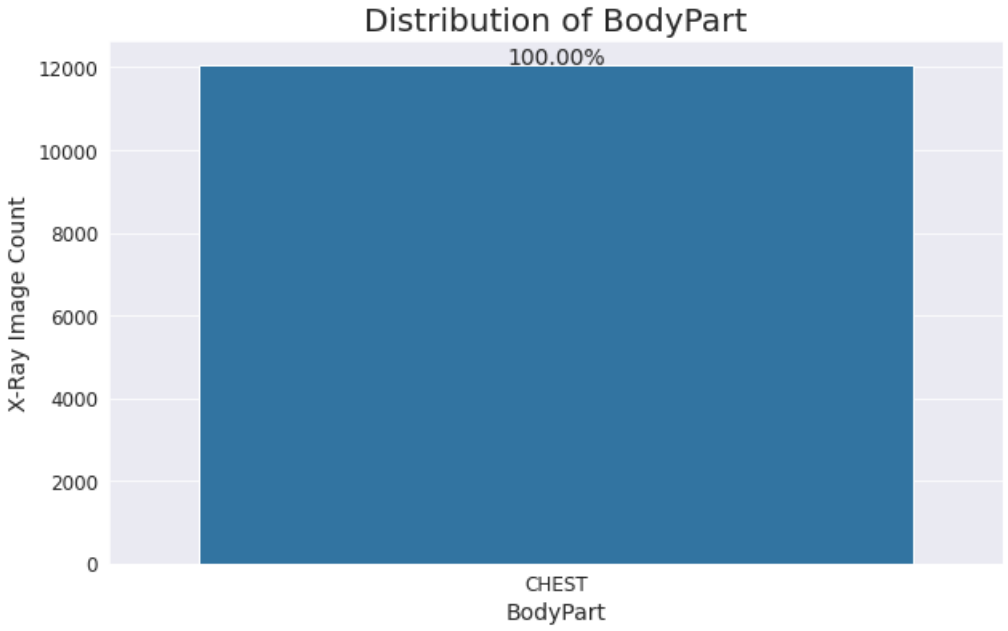
In [ ]:
```
# Check distribution of Modality
plt.figure(figsize=(10,6))
sns.set_style('darkgrid')
ax = sns.countplot(x='Modality',data=train_df)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.title('Distribution of Modality', fontsize=20)
plt.xlabel('Modality', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_without_hue(ax, train_df['Modality'], 0,40,14)
```



For all the patients modality is CR

Check distribution of BodyPart the image contains

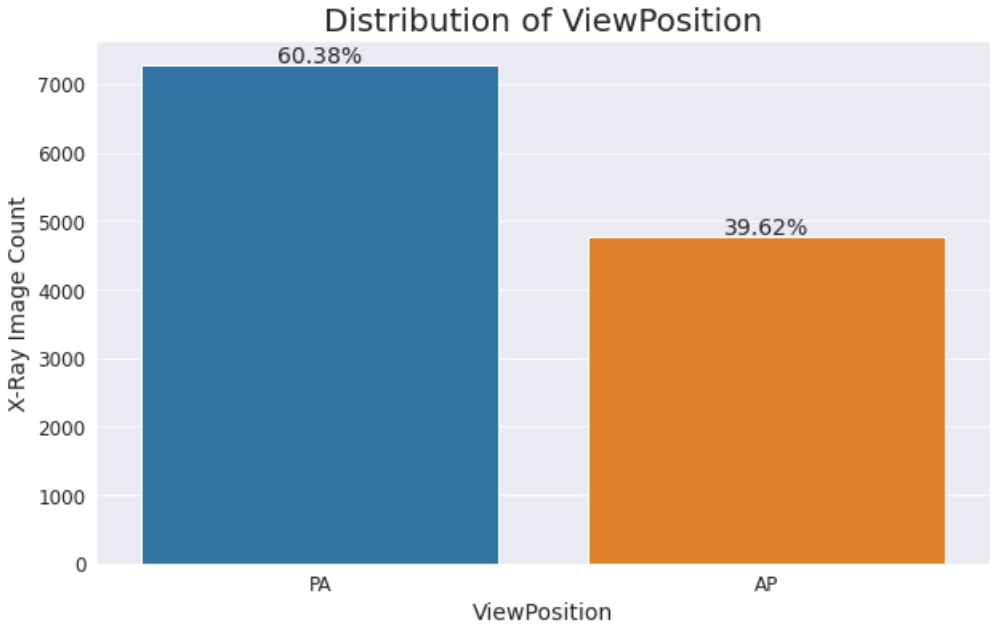
In [ ]:
```
# Check distribution of BodyPart the image contains
plt.figure(figsize=(10,6))
sns.set_style('darkgrid')
ax = sns.countplot(x='BodyPart',data=train_df)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.title('Distribution of BodyPart', fontsize=20)
plt.xlabel('BodyPart', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_without_hue(ax, train_df['BodyPart'], 0,40,14)
```



As pneumothorax is a lung disease, all the images contain chest x-rays only.

Check distribution of ViewPosition

In [ ]:
```
# Check distribution of ViewPosition
plt.figure(figsize=(10,6))
sns.set_style('darkgrid')
ax = sns.countplot(x='ViewPosition',data=train_df)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.title('Distribution of ViewPosition', fontsize=20)
plt.xlabel('ViewPosition', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_without_hue(ax, train_df['ViewPosition'], 0,40,14)
```



Source: https://en.wikipedia.org/wiki/Chest_radiograph (https://en.wikipedia.org/wiki/Chest_radiograph)

Posteroanterior view (PA):

The x-ray source is positioned so that the x-ray beam enters through the posterior (back) aspect of the chest and exits out of the anterior (front) aspect, where the beam is detected.

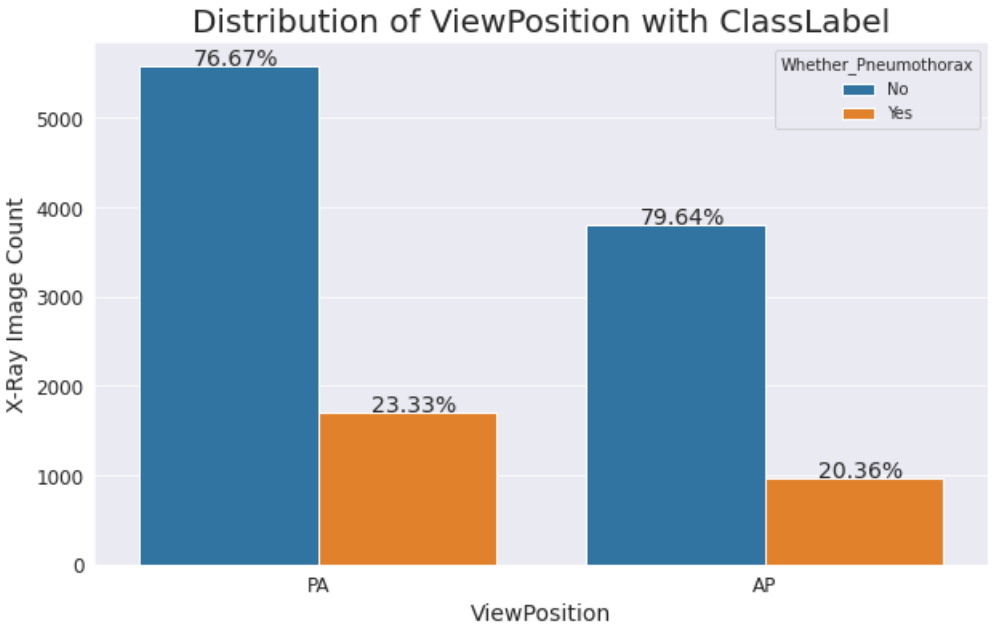
Anteroposterior view (AP):

The x-ray source and detector are reversed: the x-ray beam enters through the anterior aspect and exits through the posterior aspect of the chest. AP chest x-rays are harder to read than PA x-rays and are therefore generally reserved for situations where it is difficult for the patient to get an ordinary chest x-ray, such as when the patient is bedridden.

In the dataset for 60.38% images view position is PA and 39.62% images view position is AP

Distribution of ViewPosition with ClassLabel

In [ ]:
```
# Distribution of ViewPosition with ClassLabel
plt.figure(figsize=(10, 6))
ax = sns.countplot(x='ViewPosition',data=train_df, hue = 'Whether_Pneumothorax')
plt.title('Distribution of ViewPosition with ClassLabel', size = 20)
plt.xticks(size = 12)
plt.yticks(size = 12)
plt.xlabel('ViewPosition', size = 14)
plt.ylabel('X-Ray Image Count', size = 14)
percentage_with_hue(ax, train_df['ViewPosition'], 2,2,-0.08,10, 14)
```



For viewposition PA 76.67% are without pneumothorax and 23.33% are without pneumothorax and for viewposition AP 79.64% are without pneumothorax and 20.36% are without pneumothorax.

In [ ]:
```
#convert the Age column to int
train_df["Age"] = pd.to_numeric(train_df["Age"])

# sort the age of the patients in ascending order
sorted_ages = np.sort(train_df["Age"].values)
print(sorted_ages)
```

```
[  1   2   2 ...  94 148 413]
```

In [ ]:
```
# print last 10 ages to check if there is any abnormal behaviour
sorted_ages[-10:]
```

Out[ ]:
```
array([ 87,  88,  88,  89,  89,  89,  90,  94, 148, 413])
```

Patient's age 413 and 148 seems abnomal. So, I will discard these.

Print different percentile values of patient's age.

In [ ]:
```
for p in range(25,101,25):
    print(str(p)+"th = " + str(np.percentile(train_df["Age"], p)), end=" || ")
```

```
25th = 35.0 || 50th = 49.0 || 75th = 59.0 || 100th = 413.0 ||
```

In [ ]:
```
for p in range(90,101,1):
    print(str(p)+"th = " + str(np.percentile(train_df["Age"], p)), end=" || ")
```

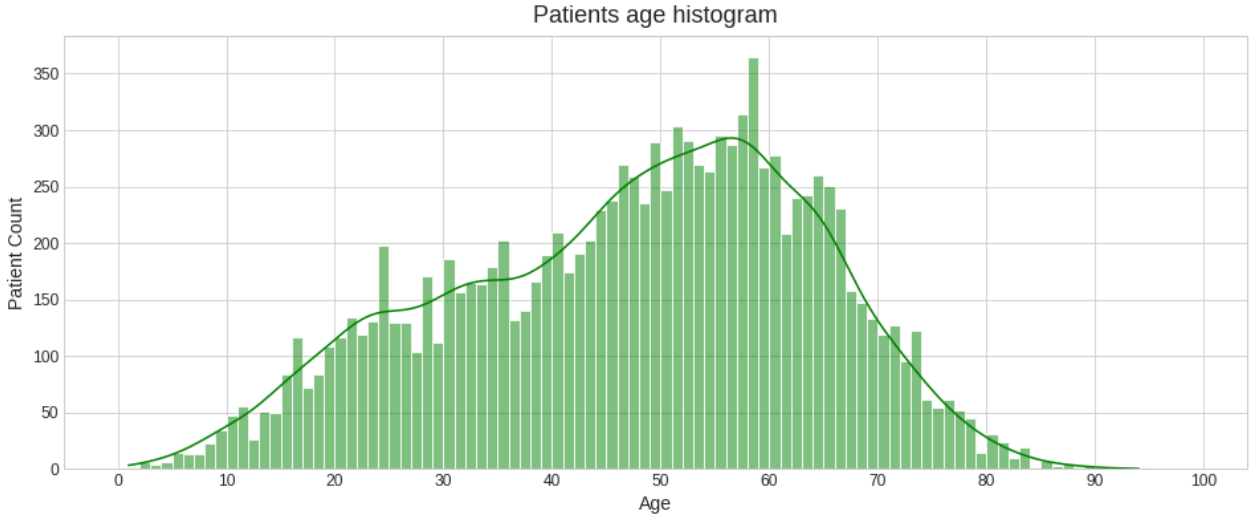
```
90th = 67.0 || 91th = 68.0 || 92th = 69.0 || 93th = 70.0 || 94th = 71.0 || 95th = 72.0 || 96th = 73.0
|| 97th = 74.0 || 98th = 76.0 || 99th = 79.0 || 100th = 413.0 ||
```

In [ ]:
```
for p in [99.1,99.2,99.3,99.4,99.5,99.6,99.7,99.8,99.9,100]:
  print(str(p)+"th = " + str(np.percentile(train_df["Age"], p)), end=" || ")
```

```
99.1th = 80.0 || 99.2th = 80.0 || 99.3th = 80.0 || 99.4th = 81.0 || 99.5th = 81.0 || 99.6th = 82.0 ||
99.7th = 83.0 || 99.8th = 84.0 || 99.9th = 87.0 || 100th = 413.0 ||
```

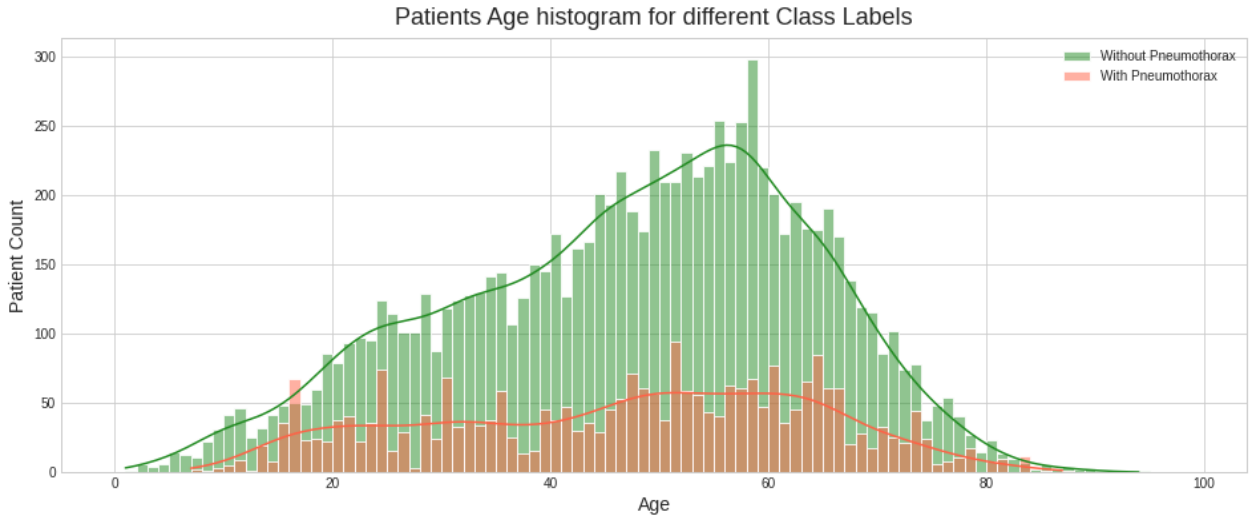
Patients age histogram

In [ ]:
```
import seaborn as sns
plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(16, 6))
# discard last 2 values as these are outliers
sns.histplot(sorted_ages[:-2], bins=[i for i in range(100)], kde=True, color ='green')
plt.title("Patients age histogram", fontsize=18, pad=10)
plt.xticks([i for i in range(0,101,10)], size = 12)
plt.yticks(size = 12)
plt.xlabel("Age", fontsize=14)
plt.ylabel("Patient Count", fontsize=14)
plt.show()
```



Plot the histogram of age for patients with and without pneumothorax separately

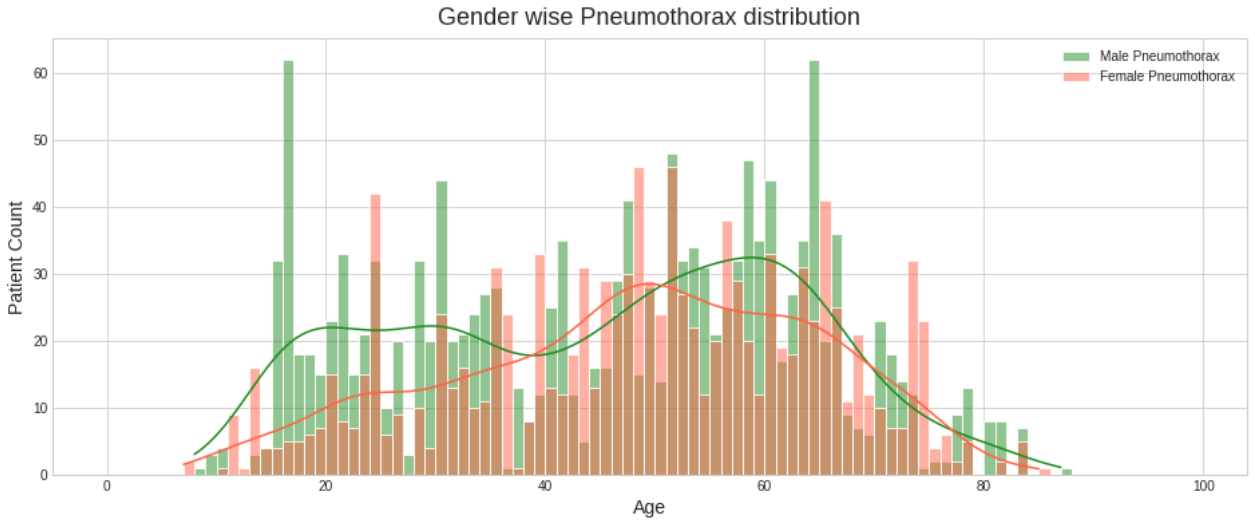
In [ ]:
```
plt.figure(figsize=(16, 6))
# consider the patient's age less than 120
sns.histplot( train_df[(train_df.Whether_Pneumothorax == 'No') & (train_df.Age <= 120)]['Age'], bins=[
i for i in range(100)], color="forestgreen", kde=True, label="Without Pneumothorax")
sns.histplot( train_df[(train_df.Whether_Pneumothorax == 'Yes') & (train_df.Age <= 120)]['Age'], bins=
[i for i in range(100)], kde=True, color="tomato", label="With Pneumothorax")
plt.title("Patients Age histogram for different Class Labels", fontsize=18, pad=10)
plt.xlabel("Age", fontsize=14)
plt.ylabel("Patient Count", fontsize=14)
plt.legend()
plt.show()
```



1. Patients within 0-6 years and 90-100 years are not suffering in Pneumothorax.
2. For patients with 16 years of age pneumothorax count is more than without pneumothorax.

Gender wise Pneumothorax distribution

In [ ]:
```
# Gender wise Pneumothorax distribution
plt.figure(figsize=(16, 6))
sns.histplot( train_df[(train_df.Whether_Pneumothorax == 'Yes') & (train_df['Sex'] == 'M') & (train_df
.Age <= 120)]['Age'], color="forestgreen", bins=[i for i in range(100)], kde=True, label="Male Pneumot
horax")
sns.histplot( train_df[(train_df.Whether_Pneumothorax == 'Yes') & (train_df['Sex'] == 'F') & (train_df
.Age <= 120)]['Age'], color="tomato", bins=[i for i in range(100)], kde=True, label="Female Pneumothor
ax")
plt.title("Gender wise Pneumothorax distribution", fontsize=18, pad=10)
plt.xlabel("Age", fontsize=14)
plt.ylabel("Patient Count", fontsize=14)
plt.legend()
plt.show()
```



create 2 separate directories for train and test png images

In [ ]:
```
# create 2 separate directories for train and test png images
train_images_png = 'siim/train_images_png/'
test_images_png = 'siim/test_images_png/'

if not os.path.isdir(train_images_png):
    os.makedirs(train_images_png)
if not os.path.isdir(test_images_png):
    os.makedirs(test_images_png)
```

Define function to convert dicom images to png

In [ ]:
```
# Define function to convert dicom images to png
def convert_images_to_png(filename, outdir):
  ds = pydicom.read_file(str(filename))
  img = ds.pixel_array
  img = cv2.resize(img, (256, 256))
  cv2.imwrite(outdir + filename.split('/')[-1][:-4] + '.png', img)
```

In [ ]:
```
for file in tqdm(train_df['Image_Path']):
  convert_images_to_png(file, train_images_png)
```

```
100%|██████████| 12047/12047 [01:38<00:00, 122.83it/s]
```

In [ ]:
```
for file in tqdm(test_df['Image_Path']):
  convert_images_to_png(file, test_images_png)
```

```
100%|██████████| 3205/3205 [00:25<00:00, 124.65it/s]
```

In [ ]:
```
# Define function to convert RLE to mask, provided by organizers
def rle2mask(rle, width, height):
    mask= np.zeros(width* height)
    array = np.asarray([int(x) for x in rle.split()])
    starts = array[0::2]
    lengths = array[1::2]

    current_position = 0
    for index, start in enumerate(starts):
        current_position += start
        mask[current_position:current_position+lengths[index]] = 1
        current_position += lengths[index]

    return mask.reshape(width, height)
```

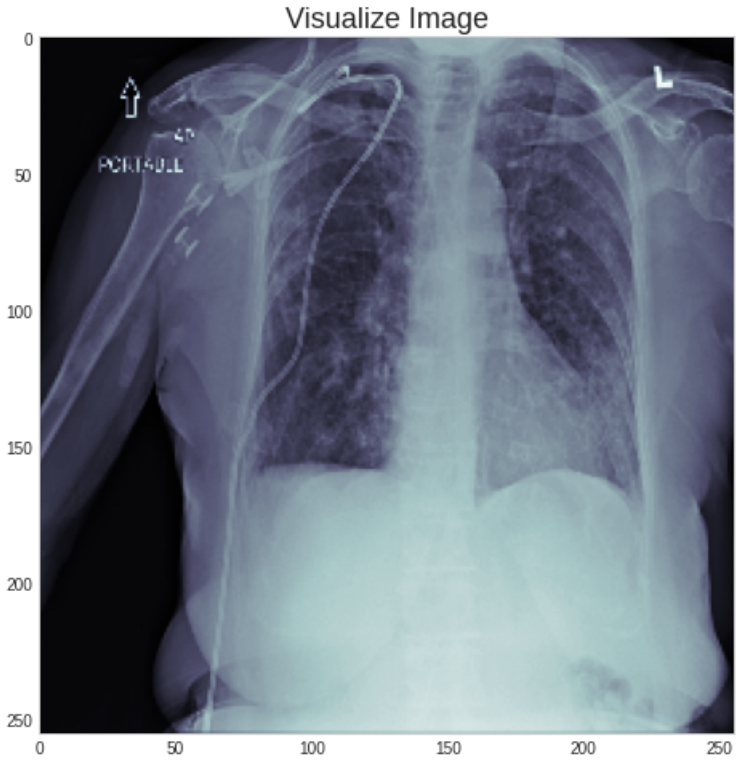
In [ ]:
```
train_df.iloc[20:30]
```

Out[ ]:

	ImageId	EncodedPixels	Image_Path
20	1.2.276.0.7230010.3.1.4.8323329.32060.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
21	1.2.276.0.7230010.3.1.4.8323329.11391.15178752...	618697 45 974 53 969 55 966 59 963 61 962 62 9...	siim/train_images_dicom/1.2.276.0.7230010.3.1...
22	1.2.276.0.7230010.3.1.4.8323329.14520.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
23	1.2.276.0.7230010.3.1.4.8323329.11541.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
24	1.2.276.0.7230010.3.1.4.8323329.32561.15178751...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
25	1.2.276.0.7230010.3.1.4.8323329.12598.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
26	1.2.276.0.7230010.3.1.4.8323329.12084.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
27	1.2.276.0.7230010.3.1.4.8323329.10251.15178752...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
28	1.2.276.0.7230010.3.1.4.8323329.6873.151787520...	-1	siim/train_images_dicom/1.2.276.0.7230010.3.1...
29	1.2.276.0.7230010.3.1.4.8323329.1857.151787516...	378999 2 1018 8 1013 13 1009 15 1007 17 1006 1...	siim/train_images_dicom/1.2.276.0.7230010.3.1...

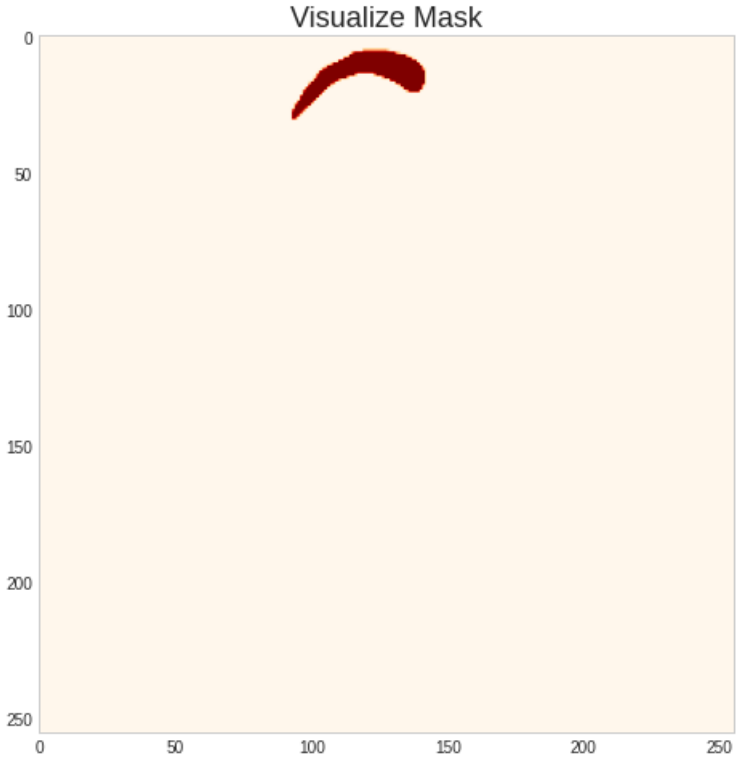
Visualize Image

In [ ]:
```
# plot original image
plt.figure(figsize=(8,8))
image_row = 29
# read the original image
ds = pydicom.read_file(train_df['Image_Path'][image_row])
img = cv2.resize(ds.pixel_array, (256, 256))
plt.imshow(img, cmap=plt.cm.bone)
plt.title("Visualize Image", fontsize=18)
plt.grid(False)
plt.show()
```



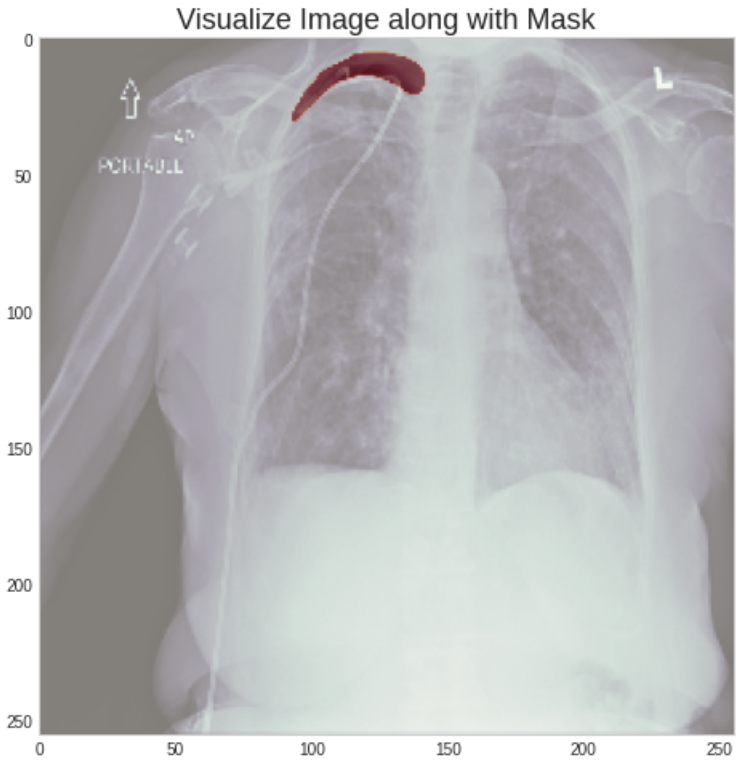
Visualize Masks

In [ ]:
```
# plot mask
plt.figure(figsize=(8,8))
image_bytes = rle2mask(train_df['EncodedPixels'][image_row], 1024, 1024).T
image_bytes = cv2.resize(image_bytes, (256, 256))
plt.imshow(image_bytes, cmap='OrRd', alpha=1)
plt.title("Visualize Mask", fontsize=18)
plt.grid(False)
plt.show()
```



Visualize Image along with Mask

In [ ]:
```
# Visualize Image along with it's Mask
plt.figure(figsize=(8,8))
plt.imshow(image_bytes, cmap='OrRd')
plt.imshow(img, cmap=plt.cm.bone, alpha=0.5)
plt.title("Visualize Image along with Mask", fontsize=18)
plt.grid(False)
plt.show()
```



Create Directories for mask png files

In [ ]:
```
# Create Directories for mask png files
train_mask_png = 'siim/train_mask_png/'
# test_mask_png = 'siim/test_mask_png'

if not os.path.isdir(train_mask_png):
    os.makedirs(train_mask_png)
# if not os.path.isdir(test_mask_png):
#     os.makedirs(test_mask_png)
```

In [ ]:
```
def convert_images_to_png(filename, outdir):
  ds = pydicom.read_file(str(filename))
  img = ds.pixel_array
  img = cv2.resize(img, (256, 256))
  cv2.imwrite(outdir + filename.split('/')[-1][:-4] + '.png', img)
```

In [ ]:
```
# define function to convert mask to png image
def convert_masks_to_png(data, outdir):
  for img_id, enc_pix in tqdm(data.values):
    mask_path = outdir + str(img_id) + '_mask.png'
    # print(mask_path)
    if enc_pix != "-1":
      image_bytes = rle2mask(enc_pix, 1024, 1024).T
      mask = cv2.resize(image_bytes, (256, 256))
      cv2.imwrite(mask_path, mask)
    else:
      mask = np.zeros((256, 256), dtype=np.uint8)
      cv2.imwrite(mask_path, mask)
convert_masks_to_png(train_df[['ImageId', 'EncodedPixels']], train_mask_png)
```

```
100%|██████████| 12047/12047 [00:17<00:00, 704.76it/s]
```

create a dataframe containing full image path and mask path

In [ ]:
```
# create a dataframe containing full image path and mask path
train_image_mask_df = pd.DataFrame(columns = ['ImageId', 'Image_Path', 'Mask_Path'])
for img_id in tqdm(train_df['ImageId']):
  # print(img_id)
  image_path = train_images_png + str(img_id) + '.png'
  mask_path = train_mask_png + str(img_id) + '_mask.png'
  # print(image_path)
  # print(mask_path)
  train_image_mask_df = train_image_mask_df.append({'ImageId':img_id, 'Image_Path':image_path, 'Mask_P
ath':mask_path}, ignore_index = True)

train_image_mask_df.head()
```

```
100%|██████████| 12047/12047 [00:28<00:00, 421.45it/s]
```

Out[ ]:

	ImageId	Image_Path	
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	siim/train_images_png/1.2.276.0.7230010.3.1.4....	siim/train_mask_

In [ ]:
```
test_image_df = pd.DataFrame(columns = ['ImageId', 'Image_Path'])
for img_id in tqdm(test_df['ImageId']):
  image_path = test_images_png + str(img_id) + '.png'
  test_image_df = test_image_df.append({'ImageId':img_id, 'Image_Path':image_path}, ignore_index = Tru
e)

test_image_df.head()
```

```
100%|██████████| 3205/3205 [00:06<00:00, 482.30it/s]
```

Out[ ]:

	ImageId	Image_Path
0	ID_6a5c8492a	siim/test_images_png/ID_6a5c8492a.png
1	ID_ba5a9ad4f	siim/test_images_png/ID_ba5a9ad4f.png
2	ID_2c87f564e	siim/test_images_png/ID_2c87f564e.png
3	ID_ac43e5bb6	siim/test_images_png/ID_ac43e5bb6.png
4	ID_eaad30cb8	siim/test_images_png/ID_eaad30cb8.png

Now I can use the image and mask to train Unet model.