

**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 [1]: import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import datetime as dt
from datetime import datetime
from tqdm.notebook import tqdm
from glob import glob
import pandas as pd
import shutil
import glob2
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
import datetime
from sklearn.model_selection import train_test_split
from keras.losses import binary_crossentropy
import keras.backend as K
from keras.models import load_model
```

```
In [2]: # install libraries to read dicom images
!pip install -q tensorflow-io
!pip install pydicom

|████████████████████████████████████████| 25.3MB 128kB/s
Collecting pydicom
  Downloading https://files.pythonhosted.org/packages/f4/15/df16546bc59bfca390cf072d473fb2c8acd423163
6f64356593a63137e55/pydicom-2.1.2-py3-none-any.whl (1.9MB)
  |████████████████████████████████████████| 1.9MB 18.8MB/s
Installing collected packages: pydicom
Successfully installed pydicom-2.1.2
```

```
In [3]: import pydicom as dicom
import tensorflow as tf
import tensorflow_io as tfio
```

```
In [4]: # mount google drive
from google.colab import drive
drive.mount('gdrive',force_remount=True)

Mounted at gdrive
```

## Download the dataset from kaggle

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

```
In [5]: # download the dataset from kaggle
# https://www.kaggle.com/seesee/siim-train-test
!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/89.0.4389.90 Safari/537.36" --header="Accept: te
xt/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
tion/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=G00G4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%
2F20210324%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210324T022759Z&X-Goog-Expires=259199&X-Goog-
SignedHeaders=host&X-Goog-Signature=6537e07b49380396cf2a8773c646d3e4847a77f3f9e6d24612c369ee3962e3aaab
5e69f6e9ea89f09026dea49c0ea2818d9a29f5e713e0b25cba7445cbfe806668b81034ec3b93f88942ec5770e0e69c7c2387a4
fcc6ea770aa548f4e84d1e7f7d789e8581e5a78883165555fc729dbfeeca80c797157680c411dd8e045b95a5eb7b304d91f89
f4e56a9bc25d46f84a416d540b4aef097d7ac0512bcc6ca52495e135a86065aaec9e9fe7f0188a29d89f1c11775b84f8d64d8b
cb3a8641feb1f2e7473c02a91402da8df9784bd889855e0c274a65098a5abcccb5cc0f926f02ed52330b438bc2a538c77d0fb9
492927c1ec7296b0f9828950b2ffe6f6a12e76" -c -O 'archive.zip'

--2021-03-24 12:48:55-- https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archiv
e.zip?X-Goog-Algorithm=G00G4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gservice
account.com%2F20210324%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210324T022759Z&X-Goog-Expires=2
59199&X-Goog-SignedHeaders=host&X-Goog-Signature=6537e07b49380396cf2a8773c646d3e4847a77f3f9e6d24612c3
69ee3962e3aaab5e69f6e9ea89f09026dea49c0ea2818d9a29f5e713e0b25cba7445cbfe806668b81034ec3b93f88942ec577
0e0e69c7c2387a4fcc6ea770aa548f4e84d1e7f7d789e8581e5a78883165555fc729dbfeeca80c797157680c411dd8e045b9
5a5eb7b304d91f89f4e56a9bc25d46f84a416d540b4aef097d7ac0512bcc6ca52495e135a86065aaec9e9fe7f0188a29d89f1
c11775b84f8d64d8bcb3a8641feb1f2e7473c02a91402da8df9784bd889855e0c274a65098a5abcccb5cc0f926f02ed52330b
438bc2a538c77d0fb9492927c1ec7296b0f9828950b2ffe6f6a12e76
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.7.176, 172.217.7.208, 172.217.1
5.112, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.7.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2059765561 (1.9G) [application/zip]
Saving to: 'archive.zip'

archive.zip          100%[=====>] 1.92G  139MB/s   in 19s

2021-03-24 12:49:14 (101 MB/s) - 'archive.zip' saved [2059765561/2059765561]
```

```
In [6]: # unzip the dataset
!unzip -qq 'archive.zip'
```

```
In [7]: # read the given train csv file
image_df = pd.read_csv('siim/train-rle.csv')
image_df.head()
```

Out[7]:

	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...

```
In [8]: # drop the duplicate ImageIDs
image_df.drop_duplicates(subset = "ImageId", keep = 'first', inplace = True)
```

```
In [9]: # create a directory for dicom images
images_dicom = 'siim/images_dicom/'
if not os.path.isdir(images_dicom):
    os.makedirs(images_dicom)

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

```
In [10]: # remove extra space in EncodedPixels column
image_df.rename(columns = {' EncodedPixels':'EncodedPixels'}, inplace = True)

# add a column whether the image is with pneumothorax or without pneumothorax
image_df['is_pneumothorax'] = np.where(image_df['EncodedPixels']=='-1', 0, 1)

image_df.head()
```

Out[10]:

	ImageId	EncodedPixels	is_pneumothorax
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	0
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	1
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	0
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	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...	1

```
In [11]: # split the dataset and use val_df for final prediction
from sklearn.model_selection import train_test_split
train_df, val_df = train_test_split(image_df, test_size=0.2, random_state=42, stratify=image_df['is_pneumothorax'], shuffle=True)
```

```
In [12]: # add full dicom path to image_df
val_df['dicom_path'] = images_dicom + val_df['ImageId']+'.dcm'
val_df.head()
```

Out[12]:

	ImageId	EncodedPixels	is_pneumothorax	
10812	1.2.276.0.7230010.3.1.4.8323329.11636.15178752...	-1	0	siim/images_dicom/1.2.27
7110	1.2.276.0.7230010.3.1.4.8323329.4471.151787518...	278724 1 1020 6 1016 9 1014 11 1011 13 1010 13...	1	siim/images_dicom/1.2.27
5130	1.2.276.0.7230010.3.1.4.8323329.5233.151787518...	-1	0	siim/images_dicom/1.2.27
5131	1.2.276.0.7230010.3.1.4.8323329.11260.15178752...	611609 30 992 33 989 36 987 40 982 44 978 49 9...	1	siim/images_dicom/1.2.27
5297	1.2.276.0.7230010.3.1.4.8323329.14511.15178752...	-1	0	siim/images_dicom/1.2.27

**Load Classification model from google drive**

```
In [13]: from keras.models import load_model
model_2 = load_model("gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/weights-07-0.640
0.hdf5")
```

## Calculate the predicted probabilities from the classification model

```
In [14]: val_image_path = val_df['dicom_path'].values
val_image_prob = []
for file in tqdm(val_image_path):
    size = 256
    image = tf.io.read_file(file)
    image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.squeeze(image,[0])
    image=tf.tile(image, tf.constant([1,1,3], tf.int32))
    image=tf.image.resize(image,size=[size,size])
    image = tf.expand_dims(image,axis=0)

    pred = model_2.predict(image)
    val_image_prob.append(pred[0])
```

## Define function to plot confusion matrix

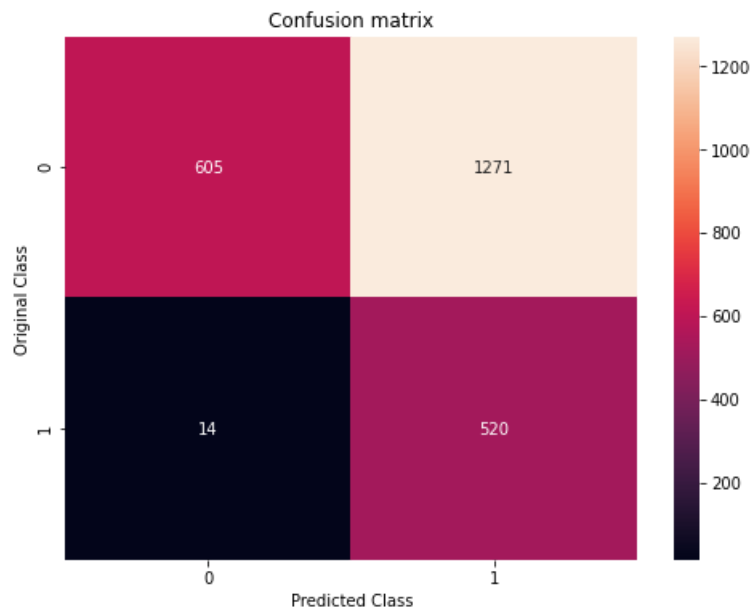
```
In [15]: from sklearn.metrics import confusion_matrix
# define function to plot confusion matrix
def conf_matrix(test_y,predict_y):
    labels = [0,1]
    plt.figure(figsize=(8,6))
    C = confusion_matrix(test_y, predict_y)
    Precision = C[1][1]/(C[0][1] + C[1][1])
    Recall = C[1][1]/(C[1][0] + C[1][1])
    sns.heatmap(C, annot=True, fmt='d')
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title('Confusion matrix')
    plt.show()
    return Precision, Recall
```

## Plot Confusion Matrix for different thresholds

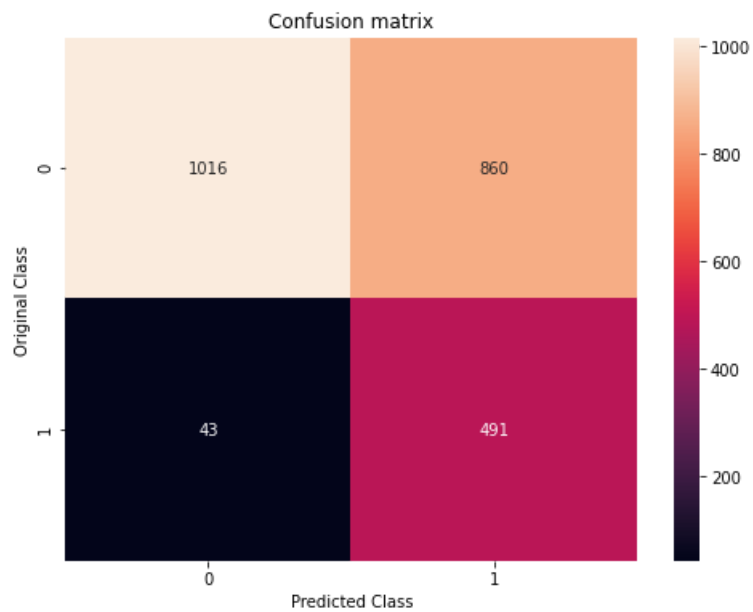
```
In [16]: val_image_label = val_df['is_pneumothorax'].values
threshold_list = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
# threshold_list = [0.5]
for threshold in threshold_list:
    val_pred_label = []
    for ele in val_image_prob:
        if ele > threshold:
            val_pred_label.append(1)
        else:
            val_pred_label.append(0)

    print("\n\n" + "="*20 + " Threshold Value = " + str(threshold) + " " + "="*20)
    Precision, Recall = conf_matrix(val_image_label, val_pred_label)
    print("Precision = {:.4f}, Recall = {:.4f} for Threshold = {}".format(Precision, Recall, threshold))
```

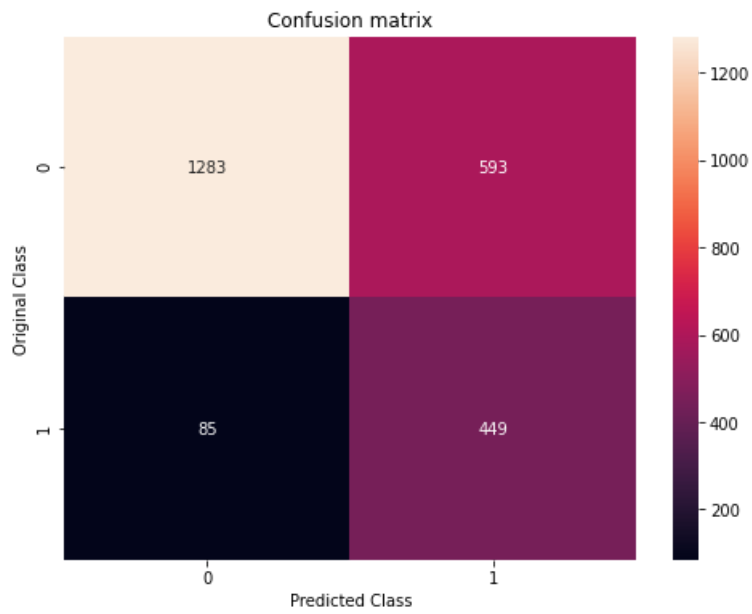
===== Threshold Value = 0.1 =====



===== Threshold Value = 0.2 =====

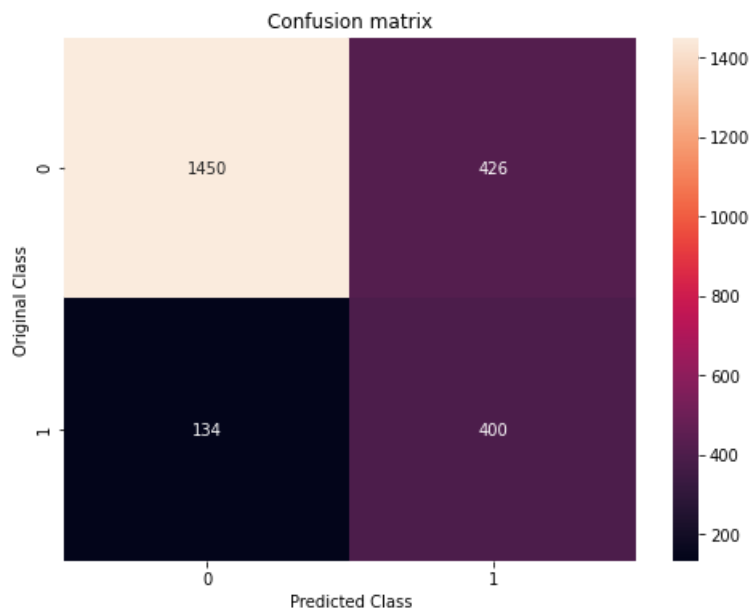


===== Threshold Value = 0.3 =====



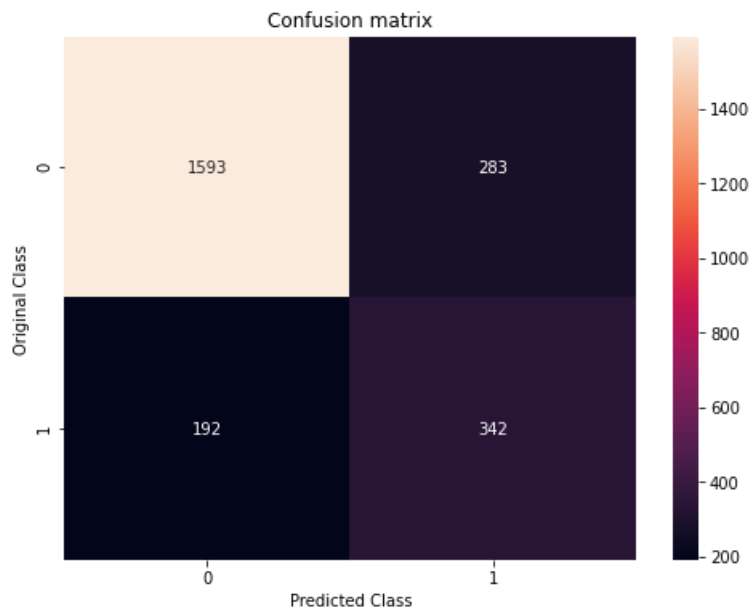
Precision = 0.4309, Recall = 0.8408 for Threshold = 0.3

===== Threshold Value = 0.4 =====



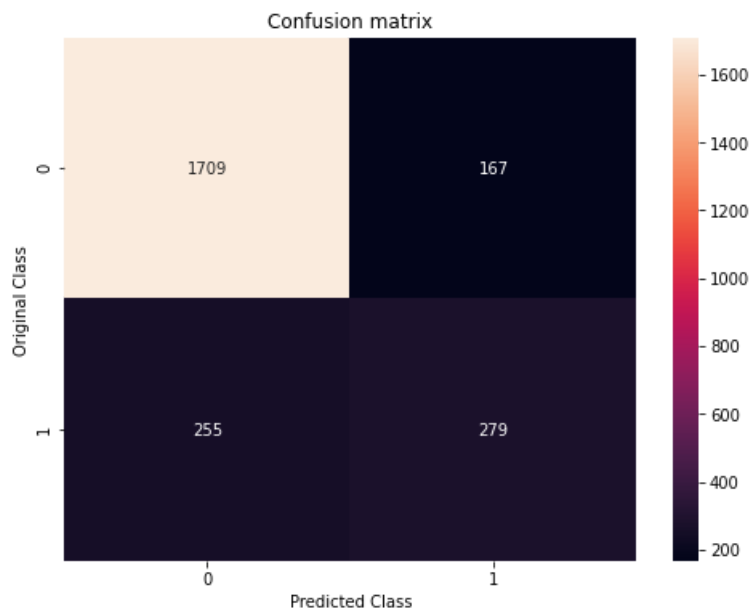
Precision = 0.4843, Recall = 0.7491 for Threshold = 0.4

===== Threshold Value = 0.5 =====



Precision = 0.5472, Recall = 0.6404 for Threshold = 0.5

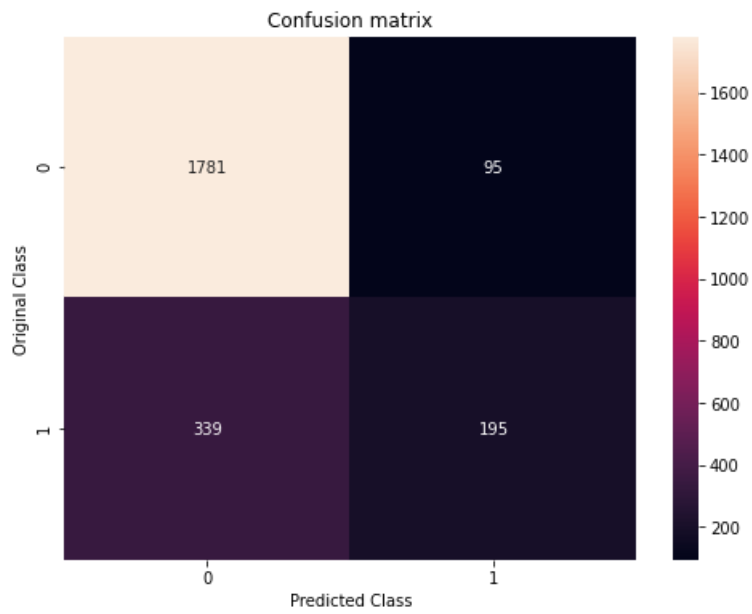
===== Threshold Value = 0.6 =====



Precision = 0.6256, Recall = 0.5225 for Threshold = 0.6

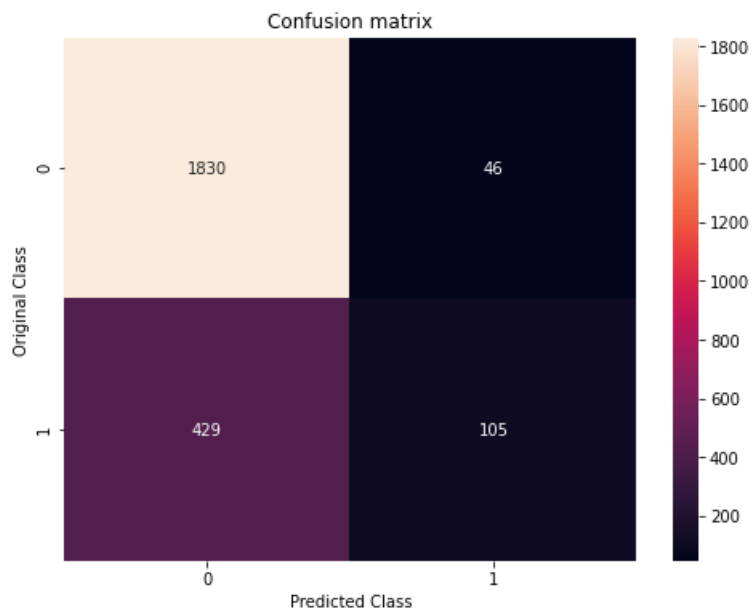
===== Threshold Value = 0.7 =====





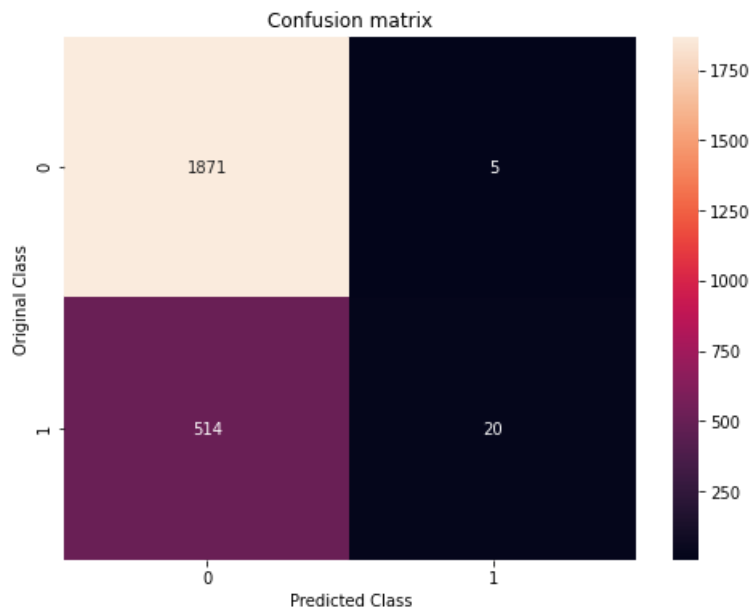
Precision = 0.6724, Recall = 0.3652 for Threshold = 0.7

===== Threshold Value = 0.8 =====



Precision = 0.6954, Recall = 0.1966 for Threshold = 0.8

===== Threshold Value = 0.9 =====



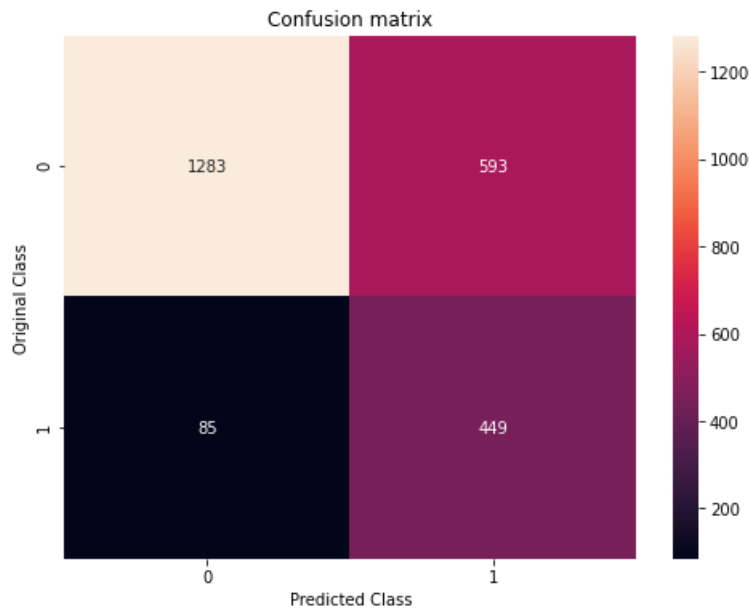
Precision = 0.8000, Recall = 0.0375 for Threshold = 0.9

Looking at all the confusion matrix, we can see that for threshold = 0.1 recall is very high but precision is very low. Considering both precision and recall, I am taking 0.3 as final threshold.

## Predict labels based on the threshold value

```
In [17]: threshold = 0.3
val_pred_label = []
for ele in val_image_prob:
    if ele > threshold:
        val_pred_label.append(1)
    else:
        val_pred_label.append(0)
# print(confusion_matrix(val_image_label, val_pred_label))
print("\n\n" + "="*20 + " Threshold Value = " + str(threshold) + " " + "="*20)
Precision, Recall = conf_matrix(val_image_label, val_pred_label)
print("Precision = {:.4f}, Recall = {:.4f} for Threshold = {}".format(Precision, Recall, threshold))
```

===== Threshold Value = 0.3 =====



Precision = 0.4309, Recall = 0.8408 for Threshold = 0.3

```
In [18]: # add a new column in val_df dataframe with the predicted probabilities and predicted classes
val_image_prob = [ele[0] for ele in val_image_prob]
val_df['pred_prob'] = val_image_prob
val_df['pred_label'] = val_pred_label
val_df.head()
```

Out[18]:

	ImageId	EncodedPixels	is_pneumothorax	
10812	1.2.276.0.7230010.3.1.4.8323329.11636.15178752...	-1	0	siim/images_dicom/1.2.27
7110	1.2.276.0.7230010.3.1.4.8323329.4471.151787518...	278724 1 1020 6 1016 9 1014 11 1011 13 1010 13...	1	siim/images_dicom/1.2.27
5130	1.2.276.0.7230010.3.1.4.8323329.5233.151787518...	-1	0	siim/images_dicom/1.2.27
5131	1.2.276.0.7230010.3.1.4.8323329.11260.15178752...	611609 30 992 33 989 36 987 40 982 44 978 49 9...	1	siim/images_dicom/1.2.27
5297	1.2.276.0.7230010.3.1.4.8323329.14511.15178752...	-1	0	siim/images_dicom/1.2.27

```
In [19]: # save val_df for future use
val_df.to_csv('gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/val_df_with_prob.csv',
index=False)
```

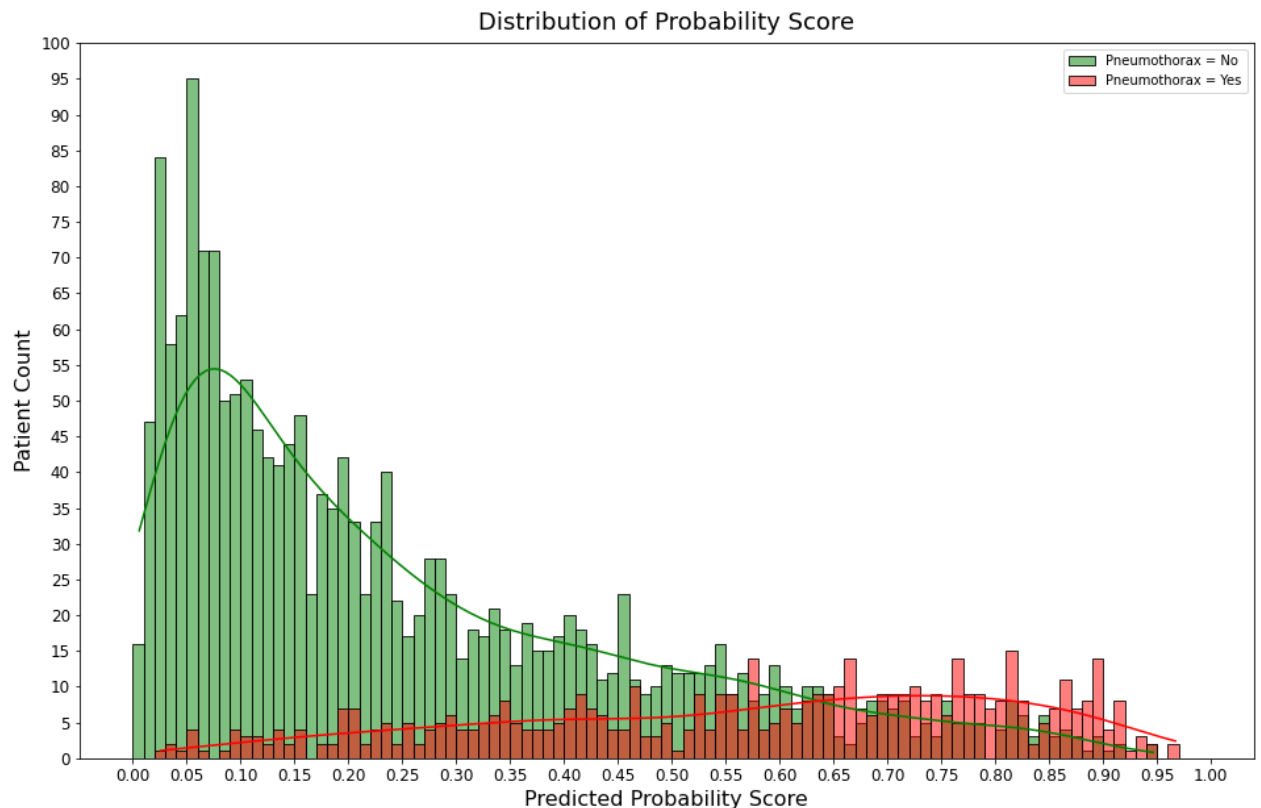
```
In [20]: # Load val_df from drive
val_df = pd.read_csv('gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/val_df_with_prob.csv')
val_df.head()
```

Out[20]:

	ImageId	EncodedPixels	is_pneumothorax	
0	1.2.276.0.7230010.3.1.4.8323329.11636.15178752...	-1	0	siim/images_dicom/1.2.276.0.7
1	1.2.276.0.7230010.3.1.4.8323329.4471.151787518...	278724 1 1020 6 1016 9 1014 11 1011 13 1010 13...	1	siim/images_dicom/1.2.276.0.7
2	1.2.276.0.7230010.3.1.4.8323329.5233.151787518...	-1	0	siim/images_dicom/1.2.276.0.7
3	1.2.276.0.7230010.3.1.4.8323329.11260.15178752...	611609 30 992 33 989 36 987 40 982 44 978 49 9...	1	siim/images_dicom/1.2.276.0.7
4	1.2.276.0.7230010.3.1.4.8323329.14511.15178752...	-1	0	siim/images_dicom/1.2.276.0.7

Distribution of predicted probability score

```
In [65]: # plot the distribution of the predicted probability score
plt.figure(figsize=(16, 10))
probability = [i/100 for i in range(100)]
sns.histplot(val_df[(val_df.is_pneumothorax == 0)]['pred_prob'], bins=[i for i in probability], color="green", kde=True, label="Pneumothorax = No")
sns.histplot(val_df[(val_df.is_pneumothorax == 1)]['pred_prob'], bins=[i for i in probability], color="red", kde=True, label="Pneumothorax = Yes")
plt.xticks([i/20 for i in range(21)], size = 12)
plt.yticks([i*5 for i in range(21)], size = 12)
plt.title("Distribution of Probability Score", fontsize=18, pad=10)
plt.xlabel("Predicted Probability Score", fontsize=16)
plt.ylabel("Patient Count", fontsize=16)
plt.legend()
plt.show()
```



## Conclusion:

1. There is a overlap between the probability scores of positive and negative points.
2. As per the distribution 0.65 is the best threshold but we have taken 0.3 as threshold because patients which contains pneumothorax is our main concern. It is okay to predict a patient without pneumothorax as pneumothorax but the opposite can be dangerous.

## False Negative Analysis

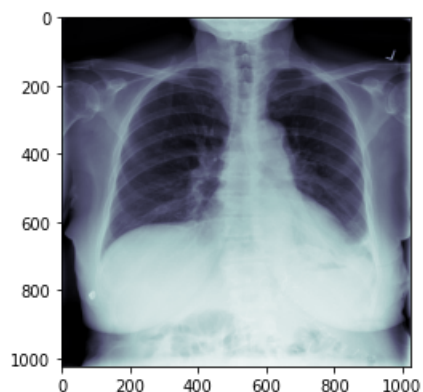
```
In [33]: # put false negative points in false_negative_val_df
false_negative_val_df = val_df[(val_df["is_pneumothorax"]==1) & (val_df["pred_label"]==0)]
false_negative_val_df = false_negative_val_df.sort_values(by=['pred_prob'], ascending=True)
false_negative_val_df.head()
```

Out[33]:

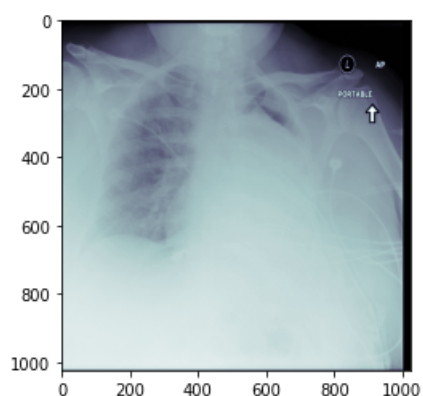
	ImageId	EncodedPixels	is_pneumothorax	
<b>2194</b>	1.2.276.0.7230010.3.1.4.8323329.10336.15178752...	642202 9 1014 11 1013 12 1011 13 1011 13 1010 ...	1	siim/images_dicom/1.2.276
<b>453</b>	1.2.276.0.7230010.3.1.4.8323329.4735.151787518...	551095 2 1019 9 1013 13 1008 19 1003 22 1000 2...	1	siim/images_dicom/1.2.276
<b>1452</b>	1.2.276.0.7230010.3.1.4.8323329.1339.151787516...	594107 3 1015 13 1010 15 1008 17 1006 19 1004 ...	1	siim/images_dicom/1.2.276
<b>267</b>	1.2.276.0.7230010.3.1.4.8323329.4469.151787518...	634025 12 1010 18 1004 22 1001 24 998 26 997 2...	1	siim/images_dicom/1.2.276
<b>382</b>	1.2.276.0.7230010.3.1.4.8323329.31865.15178751...	378947 1 1022 1 1023 1 1022 2 1022 1 1023 1 10...	1	siim/images_dicom/1.2.276

```
In [66]: # displaying first 10 images with Least probability score from false_negative_val_df
for image_row in range(0,10):
    sample_img = dicom.read_file(false_negative_val_df['dicom_path'].iloc[image_row]).pixel_array
    print('ImageId = '+str(false_negative_val_df['dicom_path'].iloc[image_row]))
    print("Original Class = 1 and Predicted Class = 0")
    print('Predicted Probability = ' + str(false_negative_val_df['pred_prob'].iloc[image_row]))
    plt.imshow(sample_img, cmap='bone')
    plt.show()
    print('\n\n')
```

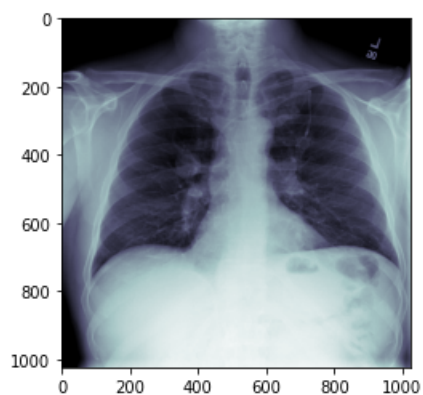
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10336.1517875223.220220.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.02580205164849758



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4735.1517875184.410483.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.03687261417508125



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1339.1517875167.304186.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.03993135690689087

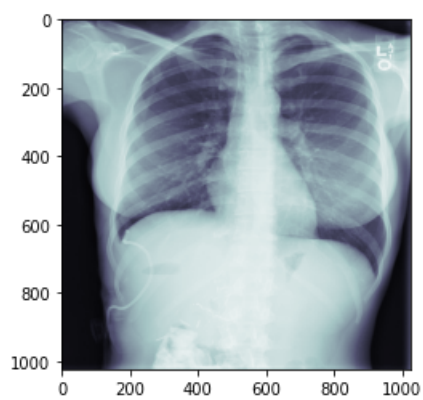


ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4469.1517875183.18796.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.04197678342461586

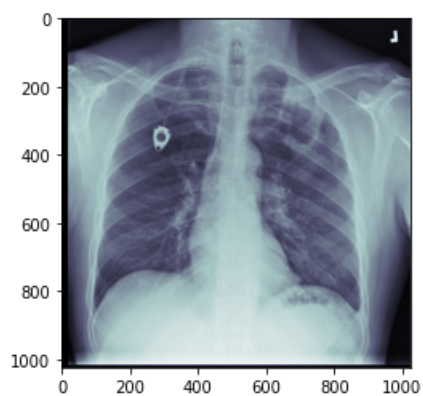




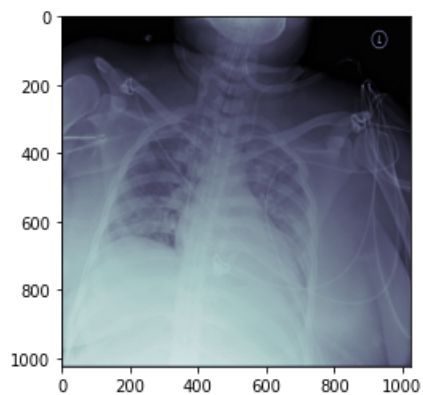
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.31865.1517875157.209292.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.050168614834547036



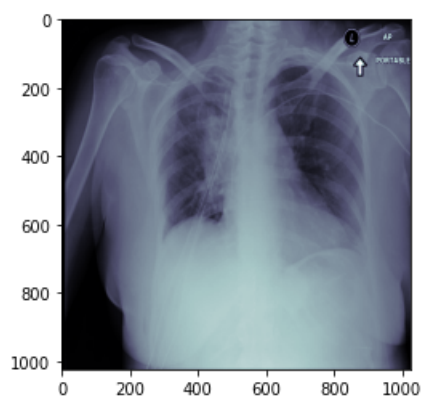
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11990.1517875236.793020.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.05123323202133179



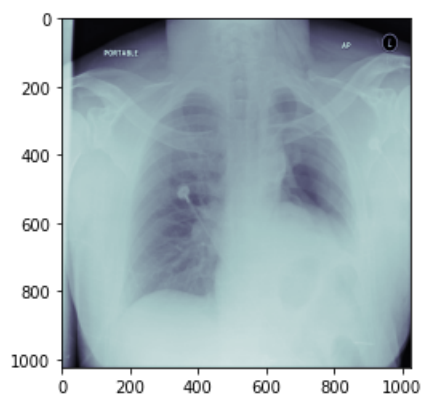
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3209.1517875176.611814.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.057887747883796685



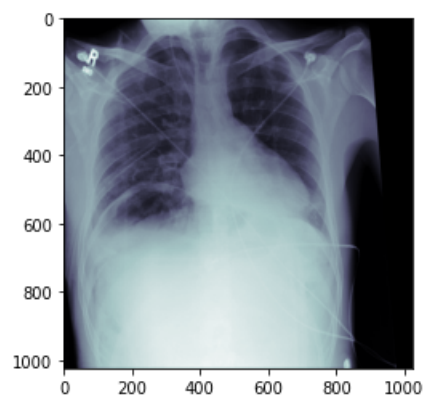
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5009.1517875185.946301.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.05822708085179329



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.2320.1517875172.159451.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.06254734098911285

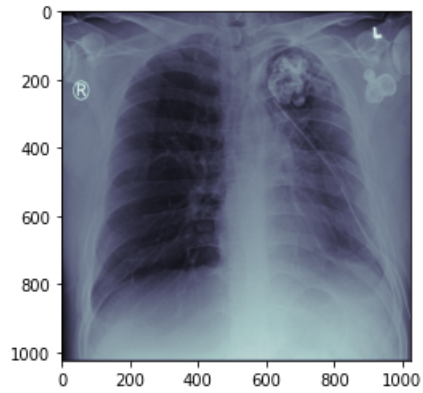


ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12323.1517875238.576790.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.08710384368896484

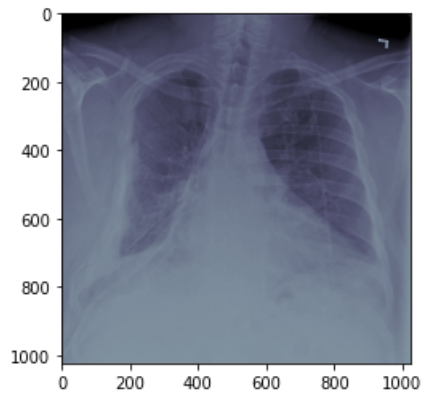


```
In [67]: # displaying last 10 images with max probability score from false_negative_val_df
for image_row in range(len(false_negative_val_df)-10, len(false_negative_val_df)):
    sample_img = dicom.read_file(false_negative_val_df['dicom_path'].iloc[image_row]).pixel_array
    print('ImageId = '+str(false_negative_val_df['dicom_path'].iloc[image_row]))
    print("Original Class = 1 and Predicted Class = 0")
    print('Predicted Probability = ' + str(false_negative_val_df['pred_prob'].iloc[image_row]))
    plt.imshow(sample_img, cmap='bone')
    plt.show()
    print('\n\n')
```

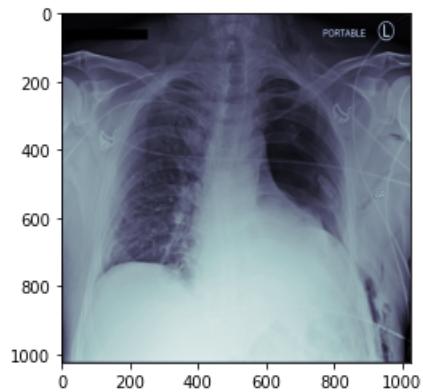
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10374.1517875223.421330.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.2867679297924042



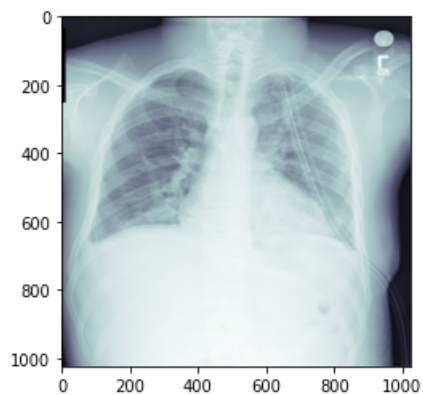
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3125.1517875176.223801.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.28812387585639954



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3123.1517875176.214943.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.288391649723053



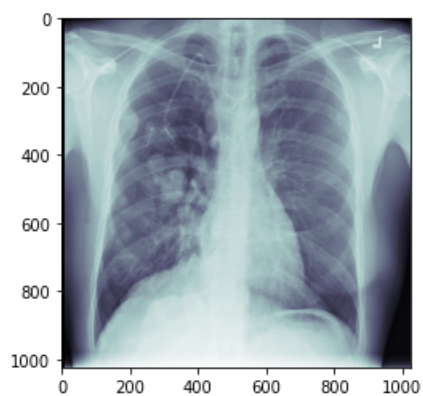
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12097.1517875237.372856.dcm  
Original Class = 1 and Predicted Class = 0  
Predicted Probability = 0.2897959053516388



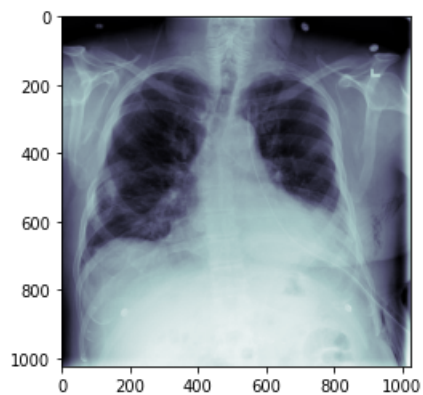
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1677.1517875168.880742.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.2902456521987915



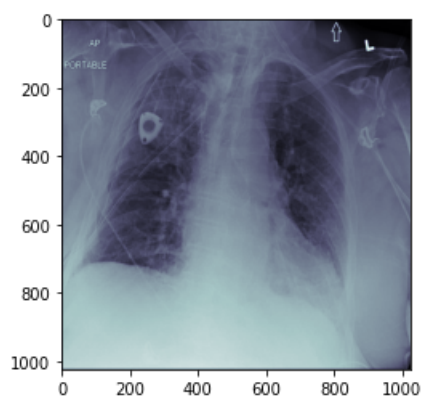
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4429.1517875182.835395.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.29037055373191833



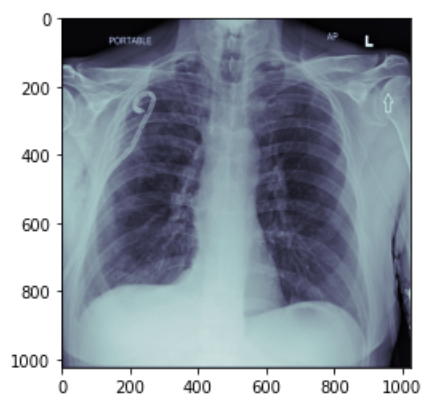
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4958.1517875185.686212.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.29176196455955505



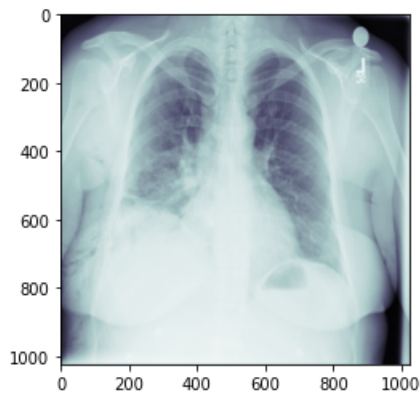
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12422.1517875239.46167.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.2923955023288727



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5202.1517875186.940296.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.295212596654892



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10094.1517875221.969850.dcm  
 Original Class = 1 and Predicted Class = 0  
 Predicted Probability = 0.2954666018486023



## Conclusion:

1. False negative points whose probability score are very low(near zero), are classified completely wrongly. To fix this issue we need to oversample these data so that the model can learn from this similar images.
2. False negative points whose probability score is less than threshold but has little higher value(near about threshold value), even though these are classified wrongly this can be fixed by training our model furthermore.

## False Positive Analysis

```
In [34]: # put false positive points in false_positive_val_df
false_positive_val_df = val_df[(val_df["is_pneumothorax"]==0) & (val_df["pred_label"]==1)]
false_positive_val_df = false_positive_val_df.sort_values(by=['pred_prob'], ascending=True)
false_positive_val_df.head()
```

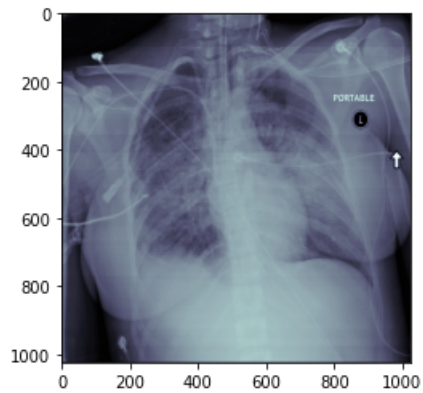
Out[34]:

	Imageld	EncodedPixels	is_pneumothorax	
<b>1750</b>	1.2.276.0.7230010.3.1.4.8323329.3439.151787517...	-1	0	siim/images_dicom/1.2.276
<b>482</b>	1.2.276.0.7230010.3.1.4.8323329.5928.151787519...	-1	0	siim/images_dicom/1.2.276
<b>134</b>	1.2.276.0.7230010.3.1.4.8323329.14318.15178752...	-1	0	siim/images_dicom/1.2.276
<b>1748</b>	1.2.276.0.7230010.3.1.4.8323329.31775.15178751...	-1	0	siim/images_dicom/1.2.276
<b>2067</b>	1.2.276.0.7230010.3.1.4.8323329.1037.151787516...	-1	0	siim/images_dicom/1.2.276

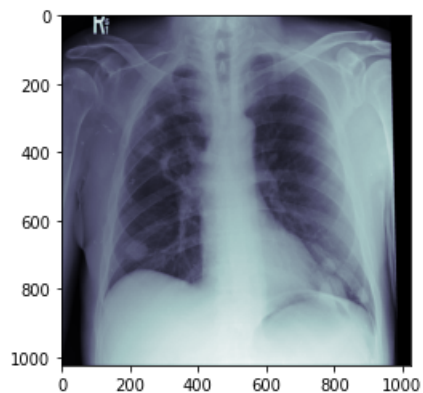


```
In [68]: # displaying first 10 images with least probability score from false_positive_val_df
for image_row in range(0,10):
    sample_img = dicom.read_file(false_positive_val_df['dicom_path'].iloc[image_row]).pixel_array
    print('ImageId = '+str(false_positive_val_df['dicom_path'].iloc[image_row]))
    print("Original Class = 0 and Predicted Class = 1")
    print('Predicted Probability = ' + str(false_positive_val_df['pred_prob'].iloc[image_row]))
    plt.imshow(sample_img, cmap='bone')
    plt.show()
    print('\n\n')
```

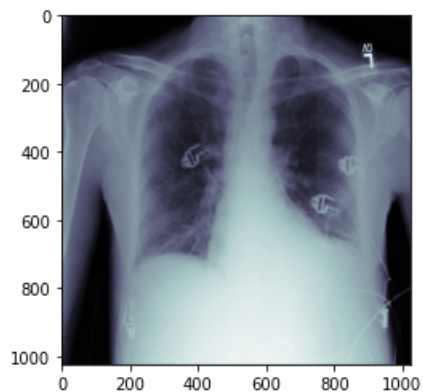
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3439.1517875177.852416.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.3001985251903534



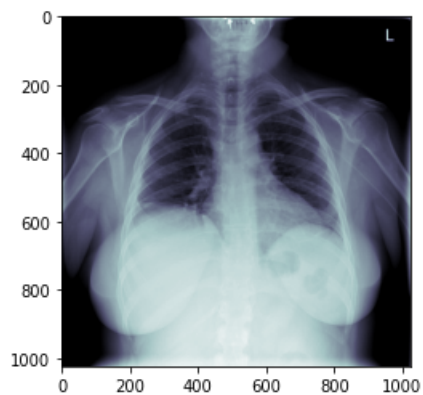
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5928.1517875192.727858.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.3003208935260773



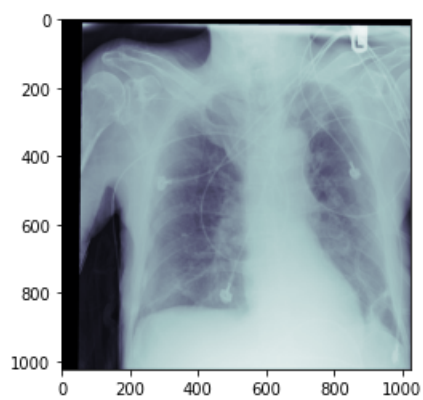
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.14318.1517875251.231260.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.3013767004013061



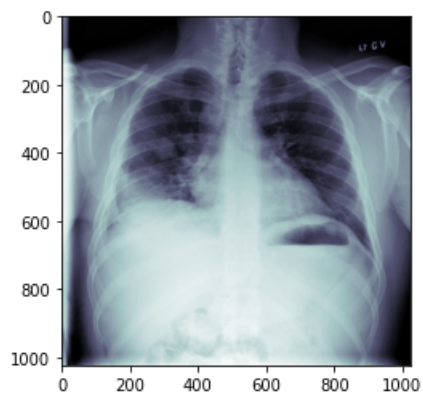
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.31775.1517875156.777338.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.3015292286872864



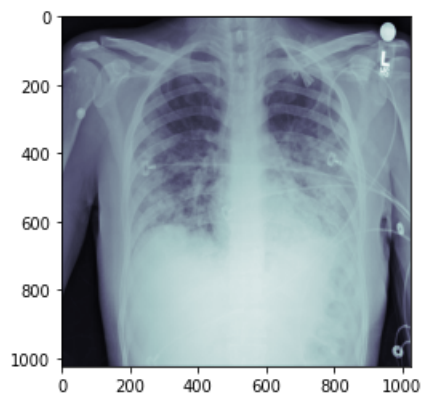
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1037.1517875166.14849.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.3024154007434845



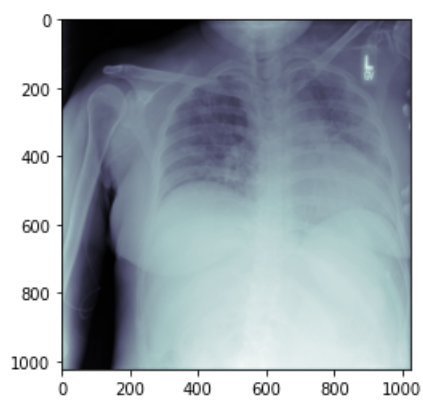
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10189.1517875222.538484.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.30243226885795593



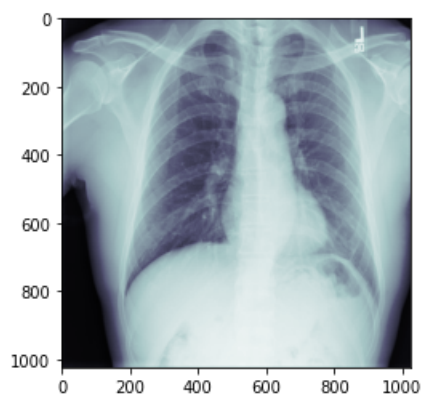
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1576.1517875168.475041.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.30267927050590515



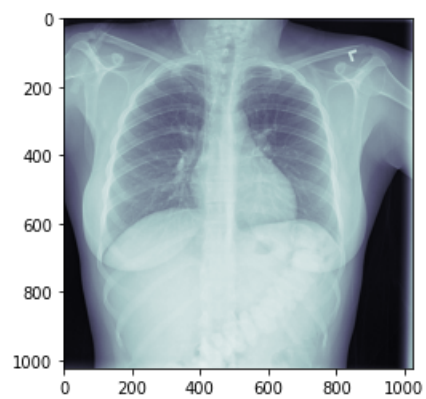
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.2278.1517875171.915442.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.3030185103416443



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.1257.1517875166.994874.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.3031831979751587

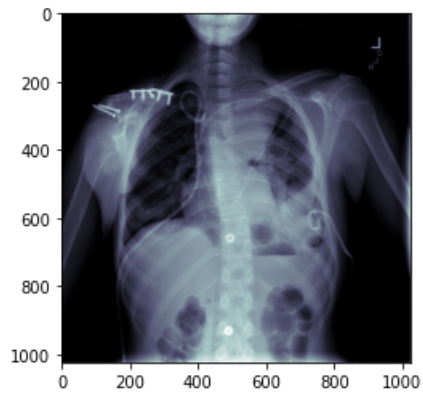


ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.32266.1517875159.299328.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.30321142077445984

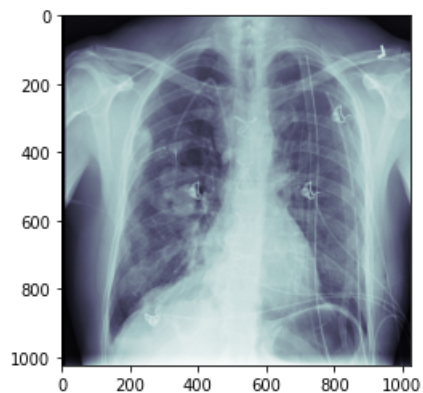


```
In [69]: # displaying last 10 images with max probability score from false_positive_val_df
for image_row in range(len(false_positive_val_df)-10, len(false_positive_val_df)):
    sample_img = dicom.read_file(false_positive_val_df['dicom_path'].iloc[image_row]).pixel_array
    print('ImageId = '+str(false_positive_val_df['dicom_path'].iloc[image_row]))
    print("Original Class = 0 and Predicted Class = 1")
    print('Predicted Probability = ' + str(false_positive_val_df['pred_prob'].iloc[image_row]))
    plt.imshow(sample_img, cmap='bone')
    plt.show()
    print('\n\n')
```

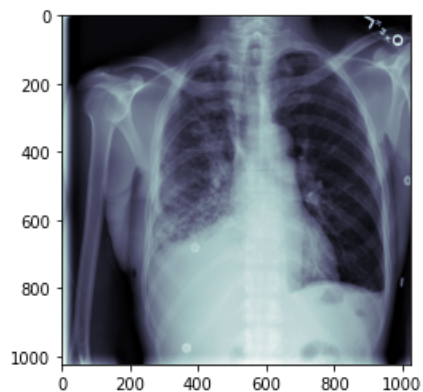
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.3344.1517875177.387601.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.8747736811637878



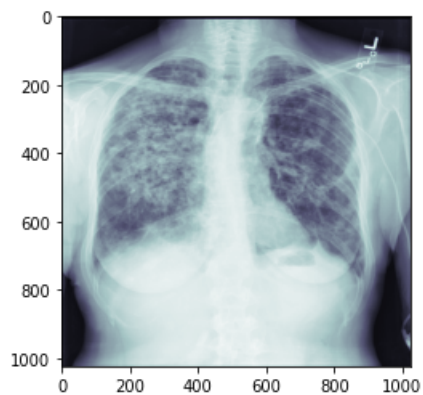
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.13401.1517875245.151502.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.885296106338501



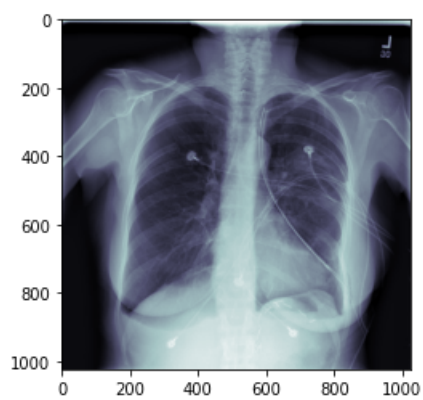
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.11578.1517875233.705534.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.8950964212417603



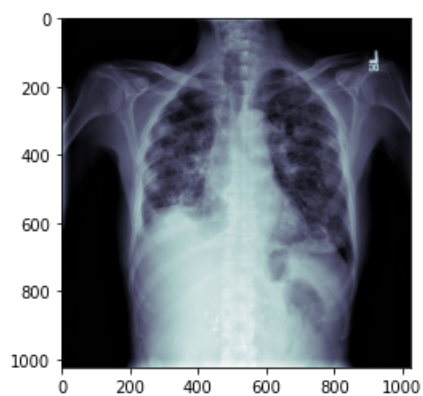
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.12625.1517875240.194497.dcm  
Original Class = 0 and Predicted Class = 1  
Predicted Probability = 0.8977672457695007



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.10521.1517875224.91521.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.8996217846870422

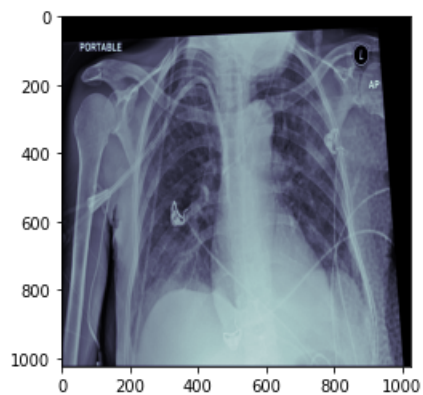


ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5609.1517875189.27613.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.9007509350776672

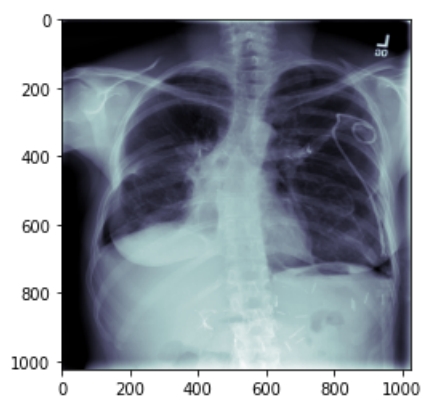


ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.6450.1517875198.300471.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.910106360912323

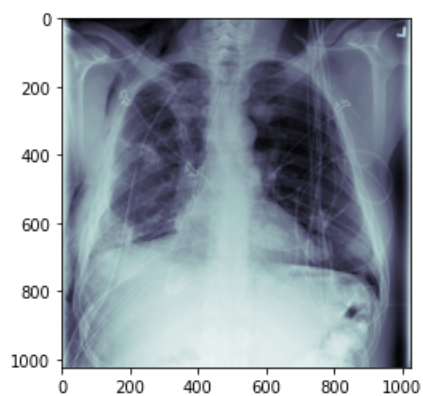




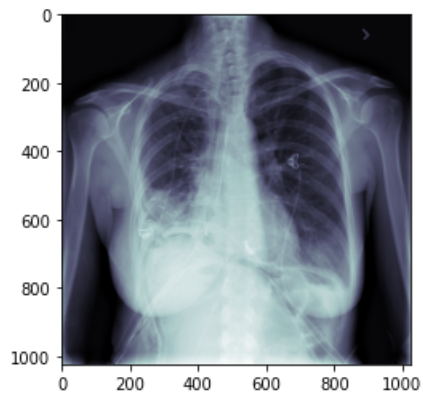
ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.5589.1517875188.912177.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.9112218618392944



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.31670.1517875156.225110.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.940726399421692



ImageId = siim/images\_dicom/1.2.276.0.7230010.3.1.4.8323329.4863.1517875185.163456.dcm  
 Original Class = 0 and Predicted Class = 1  
 Predicted Probability = 0.9462146162986756



## Conclusion:

1. False positive points whose probability score is higher(near one), are classified complete wrongly. We need to oversample these data and train our model to get better result.
2. False positive points whose probability score is lower(near threshold value) but higher than the threshold, even though these points are classified wrongly this can be fixed by training our model further.