FDA Submission

Your Name: Sushmita Maltare

Name of your Device: Pneumonia detector from Chest X-Rays

Algorithm Description

1. General Information

Intended Use Statement: This algorithm is intended for use on men and women from the ages of 1-95 years whose chest and rib-cage X-Rays have been acquired in PA or AP positions.

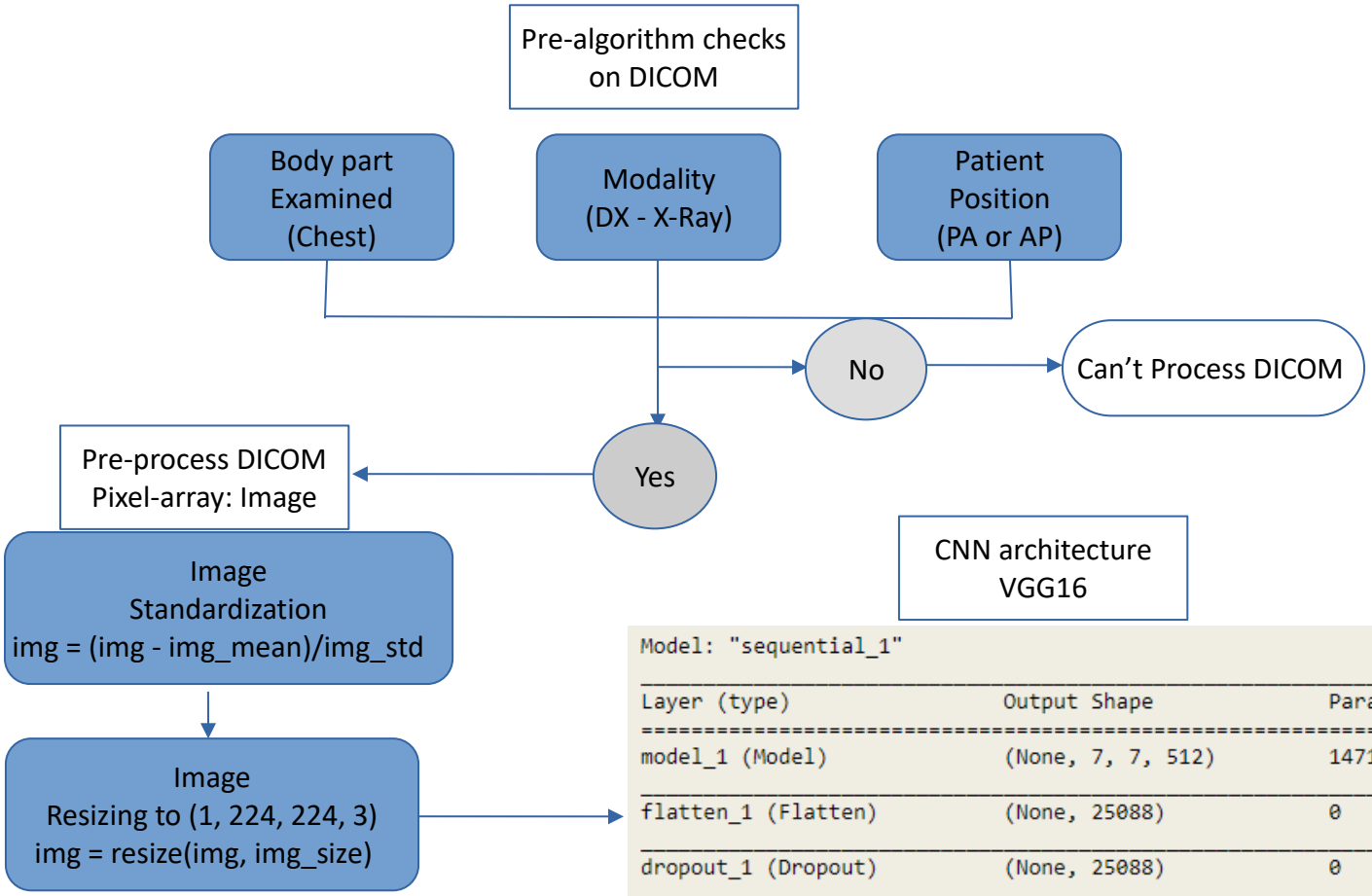
Indications for Use: To be used by radiologists as a tool to identify pneumonia in chest X-rays acquired in PA or AP positions.

Device Limitations: The results above indicate that the presence of other diseases in a chest x-ray is a limitation of this algorithm, and that the algorithm performs very poorly on the accurate detection of pneumonia in the presence of other conditions such as, pneumothorax, and effusion. The presence of infiltration and edema have a slight impact on the algorithm's sensitivity and may reduce the ability to detect pneumonia, while the presence of effusion, mass, nodule has a slight impact on specificity and may increase the number of false positive pneumonia classifications.

Clinical Impact of Performance: A false positive for pneumonia may lead to misdiagnosis as presence of other diseases can be falsely labelled as pneumonia. This will lead to inappropriate treatment, and likely worsening of patient condition. A false negative for pneumonia will lead to delay in diagnosis and treatment of pneumonia patients.

2. Algorithm Design and Function

<< Insert Algorithm Flowchart >>




```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_1 (Model)              (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              25691136
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 41,062,209
Trainable params: 26,347,521
Non-trainable params: 14,714,688
```

DICOM Checking Steps: The algorithm has certain conditions for images which need to be met before being fed to model.predict() function. Since my algorithm is only applicable for X-Ray images of Chest which are acquired in PA or AP positions, I first check the DICOM headers for these three conditions. If they all are met by the DICOM file, then the pixel-array from DICOM is sent to be per-processed.

Preprocessing Steps: Pixel-array is standardized and resized to (1, 224, 224, 3) size which is the size that VGG16 accepts.

CNN Architecture: I used VGG16, which is a pre-trained network downloaded from Keras. Weights from 'imagenet' were used and all but the last convolutional layer were frozen. Last layer was fine-tuned as can be seen in above flowchart.

3. Algorithm Training

Parameters:

* Types of augmentation used during training: I used ImageDataGenerator package from keras and used following arguments for training data:

```
preprocessing_function = preprocess_input,
horizontal_flip = True,
vertical_flip = False,
height_shift_range= 0.1,
width_shift_range=0.1,
rotation_range=20,
shear_range = 0.1,
zoom_range=0.1
```

For validation data, I did not use augmentation but pre-processed it using only preprocess_function argument inside ImageDataGenerator.

* Batch size: 64 for training data and 100 for validation data
* Optimizer learning rate: 0.0001
* Layers of pre-existing architecture that were frozen:

VGG16 is the pre-existing architecture used in this project. All but the last maxpool2D layer were frozen, below are the layers that were frozen (labelled False):

input_1 False
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False

block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 False
block4_pool False
block5_conv1 False
block5_conv2 False
block5_conv3 False
block5_pool True

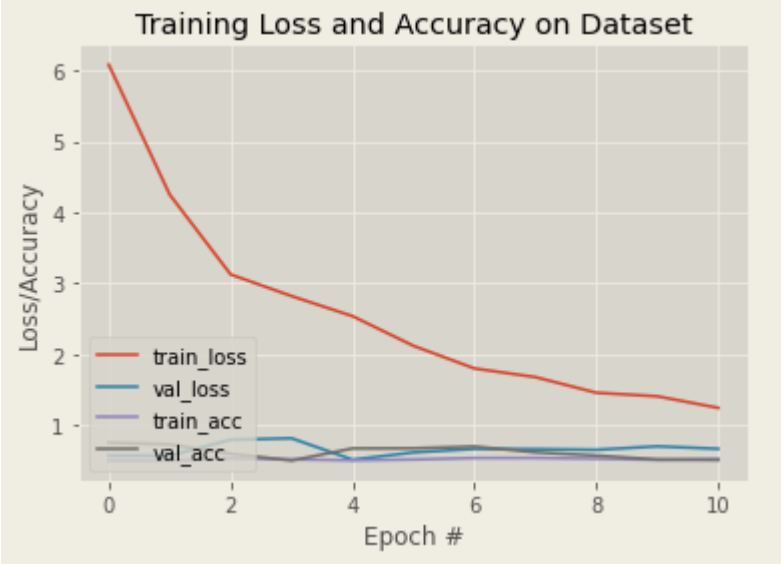
* Layers of pre-existing architecture that were fine-tuned
The last layer of the VGG16 (block5_pool) was fine-tuned for the purpose of this project.

* Layers added to pre-existing architecture: Below are the layers added to pre-existing architecture(see the architecture diagram in flowchart on page 2):

1. Flatten the output of the VGG16 model because it is from a convolutional layer.
    my_model.add(Flatten())
2. Added a dropout-layer which may prevent overfitting and improve generalization ability to unseen data e.g. the test-set.
    my_model.add(Dropout(0.5))
3. Added a dense (aka. fully-connected) layer. This is for combining features that the VGG16 model has recognized in the image.
    my_model.add(Dense(1024,activation='relu'))
4. Another dropout layer
    my_model.add(Dropout(0.5))
5. Another dense layer.
    my_model.add(Dense(512,activation='relu'))
6. Another dropout layer
    my_model.add(Dropout(0.5))
7. Another dense layer
    my_model.add(Dense(256,activation='relu'))
8. Another dropout layer
    my_model.add(Dropout(0.5))

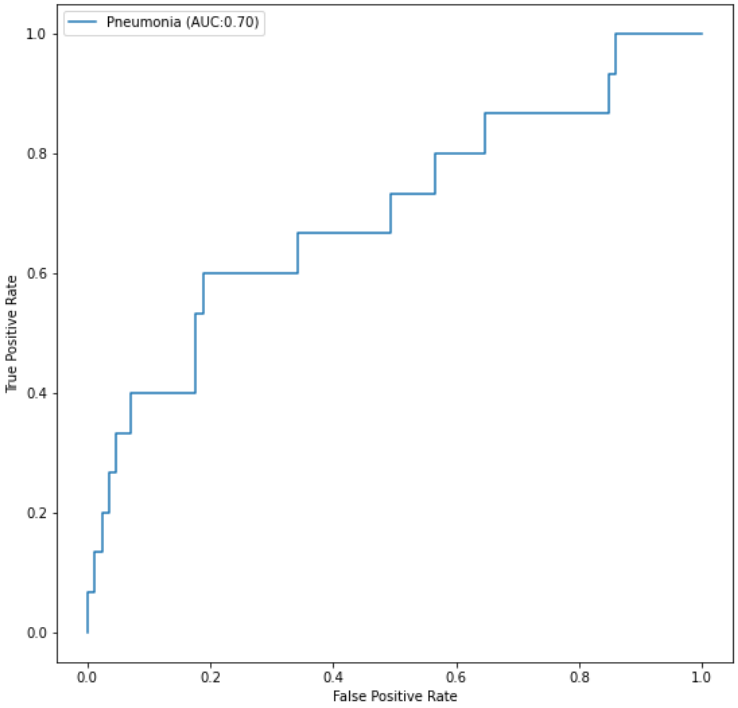
9. Added a dense (aka. fully-connected) layer. Changed the activation function to sigmoid, so output of the last layer is in the range of [0,1]
    my_model.add(Dense(1,activation='sigmoid'))

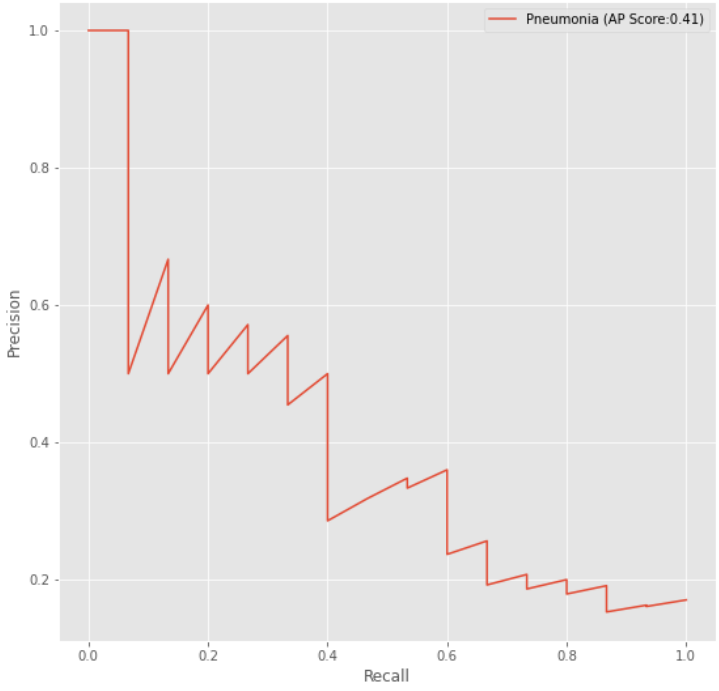
<< Insert algorithm training performance visualization >>



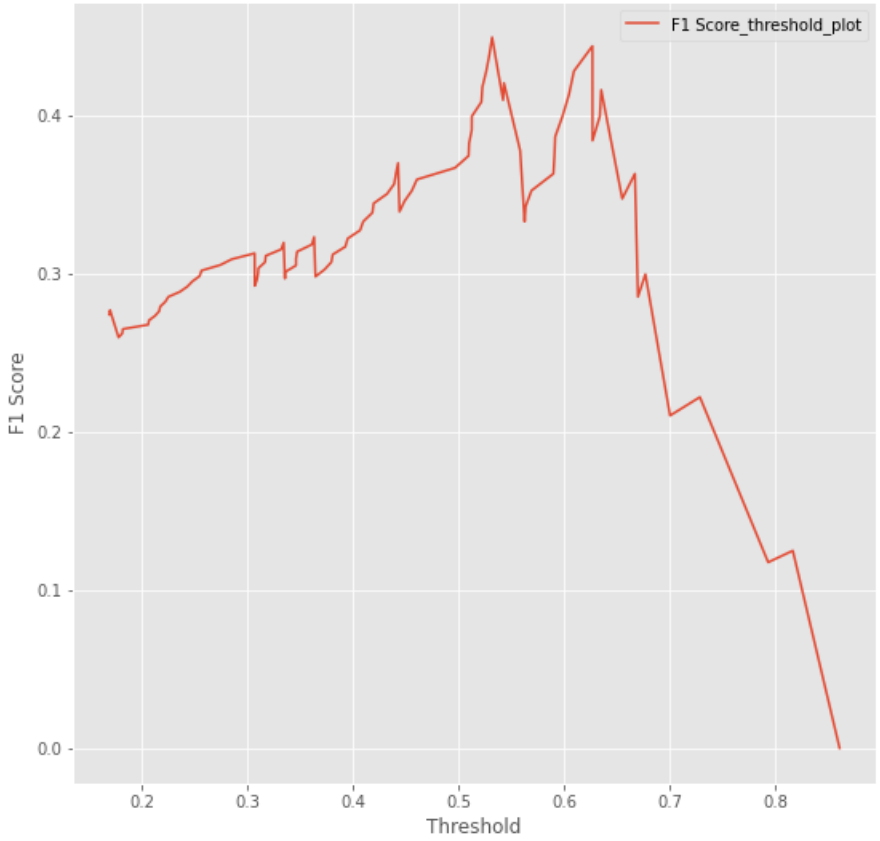
Training performance: validation loss decreased up to 5th epoch, while training loss kept going down. So, the algorithm was overfitting and would require more work for the algorithm to prove useful.

<< Insert P-R curve >>





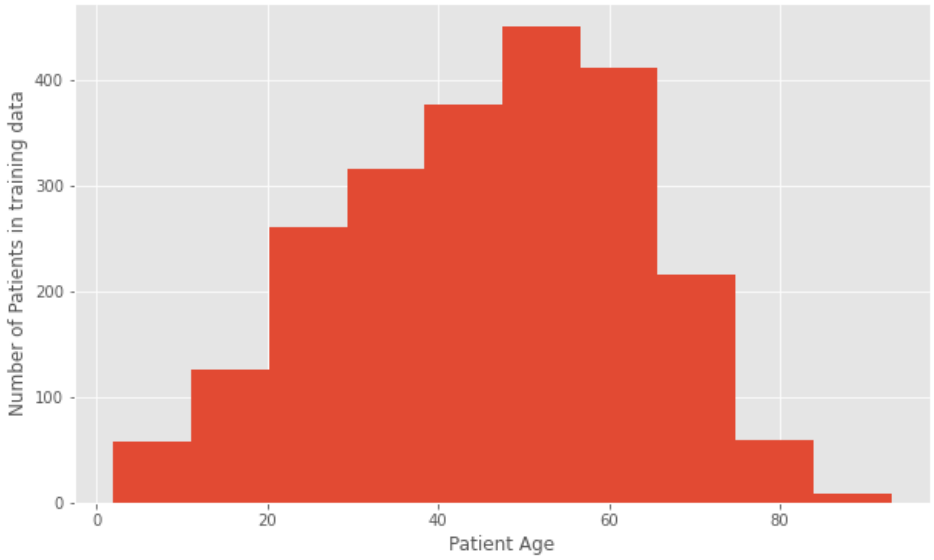
Final Threshold and Explanation:

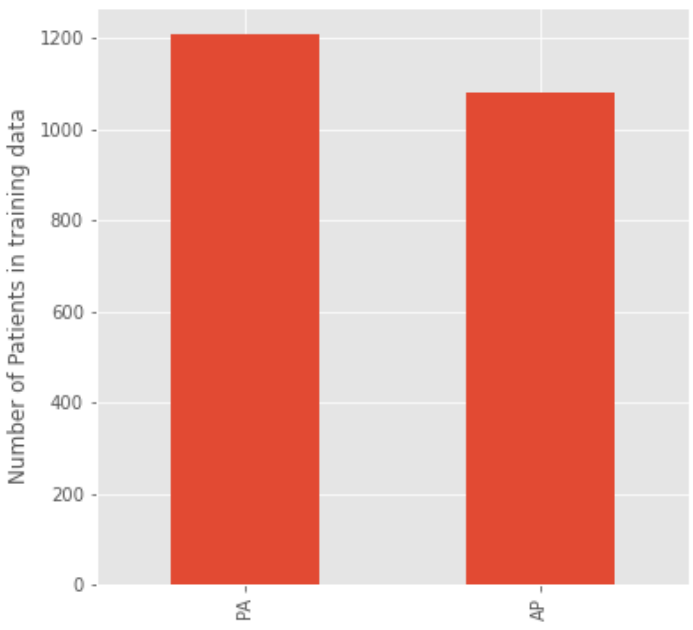


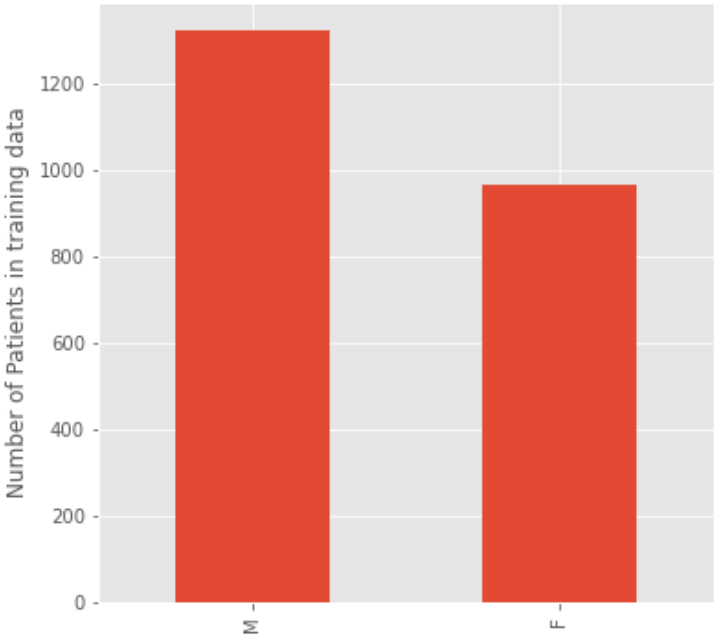
Final threshold: I selected a threshold corresponding to the best F1-score.
Best F1-Score: 0.45
Best threshold: 0.53166294

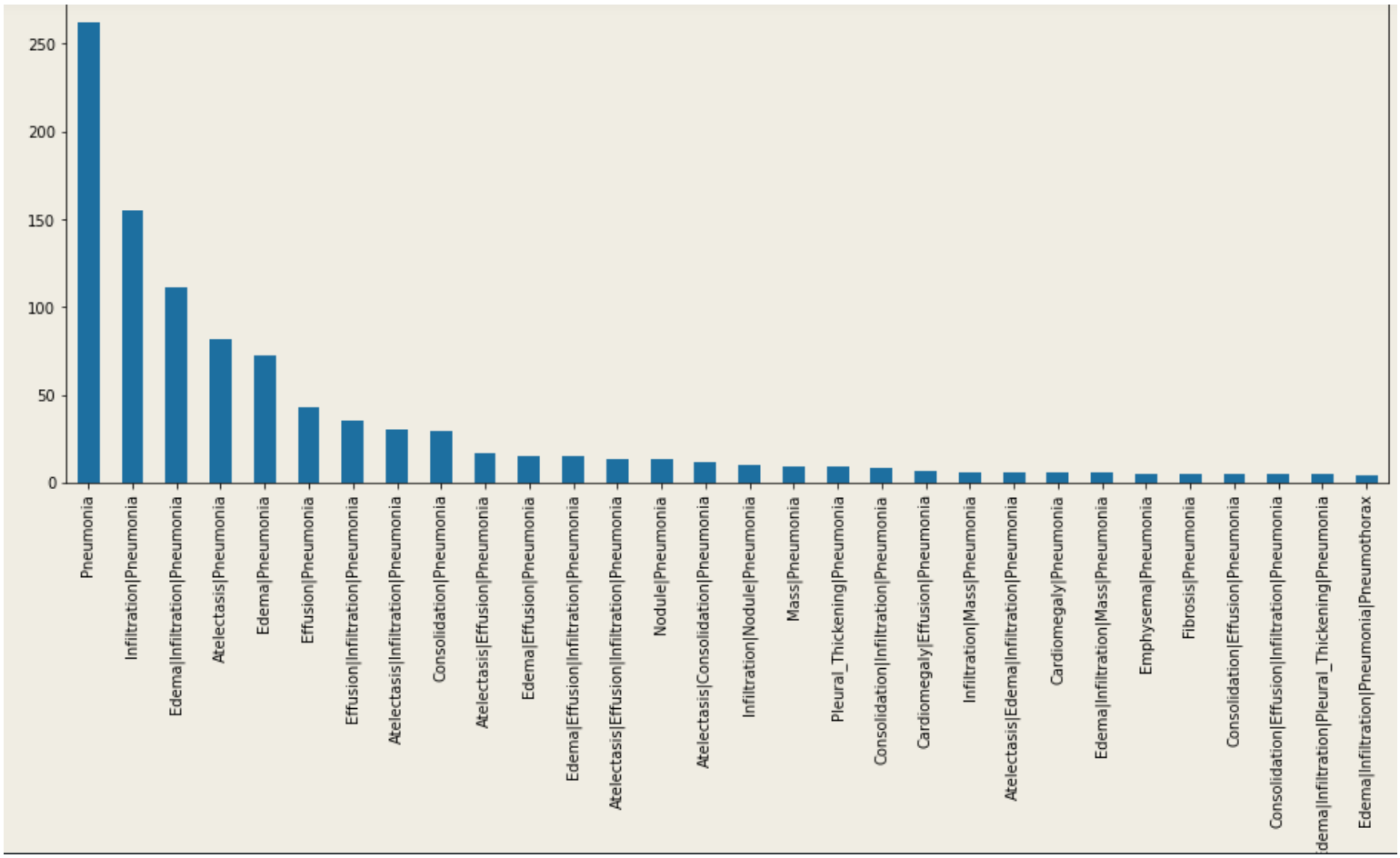
4. Databases
 (For the below, include visualizations as they are useful and relevant)
I removed age outliers (max age incorrectly shows 414), this resulted in a dataset of length 112104.
- Training and validation datasets were first split 80:20 ratio with same proportion of pneumonia in both
-Then, to get equal number of positive and negative pneumonia cases in training dataset, some negative data was removed. Also, validation was processed to have 20% positive cases

Description of Training Dataset:
Size of training dataset: 2288
Ratio of positive to negative cases: 1:1
The patient demographic data (as it is available)
The radiologic techniques used and views taken: X-Ray, two views were present in data, PA & AP
The co-occurrence frequencies of pneumonia with other diseases

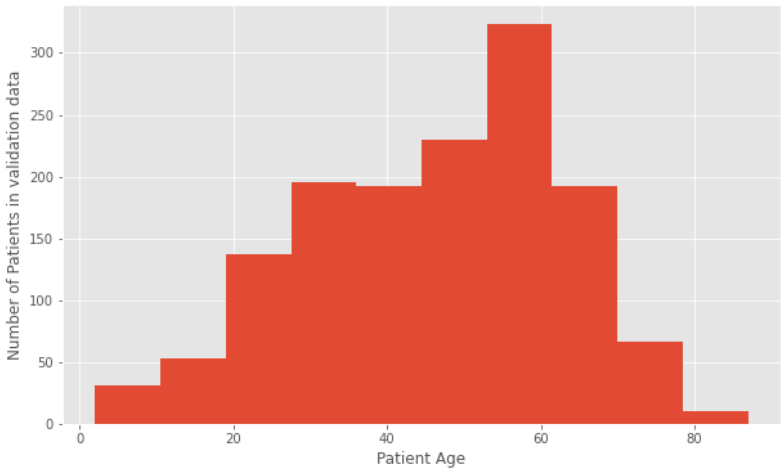


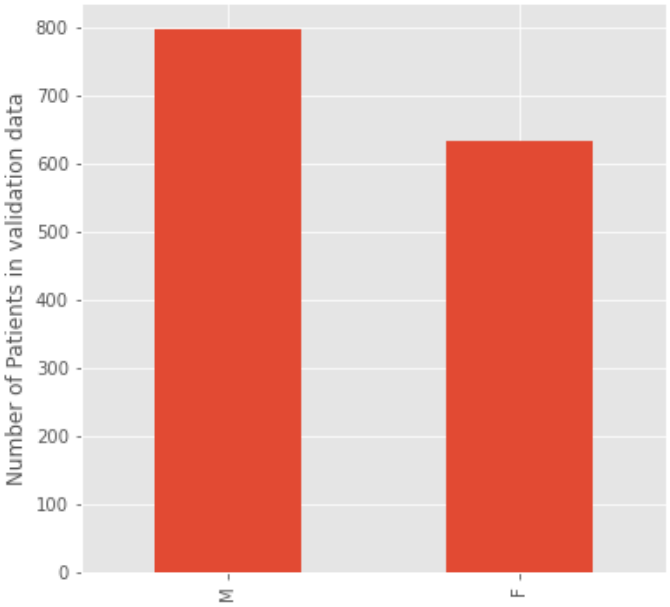


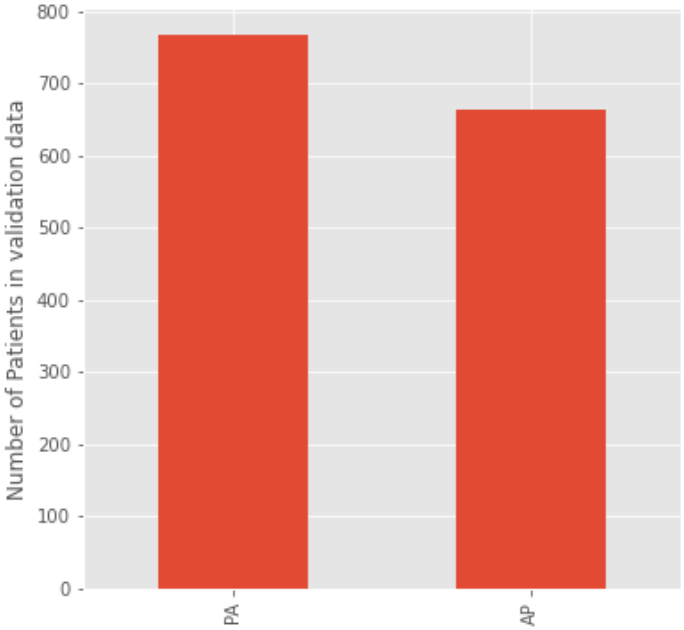


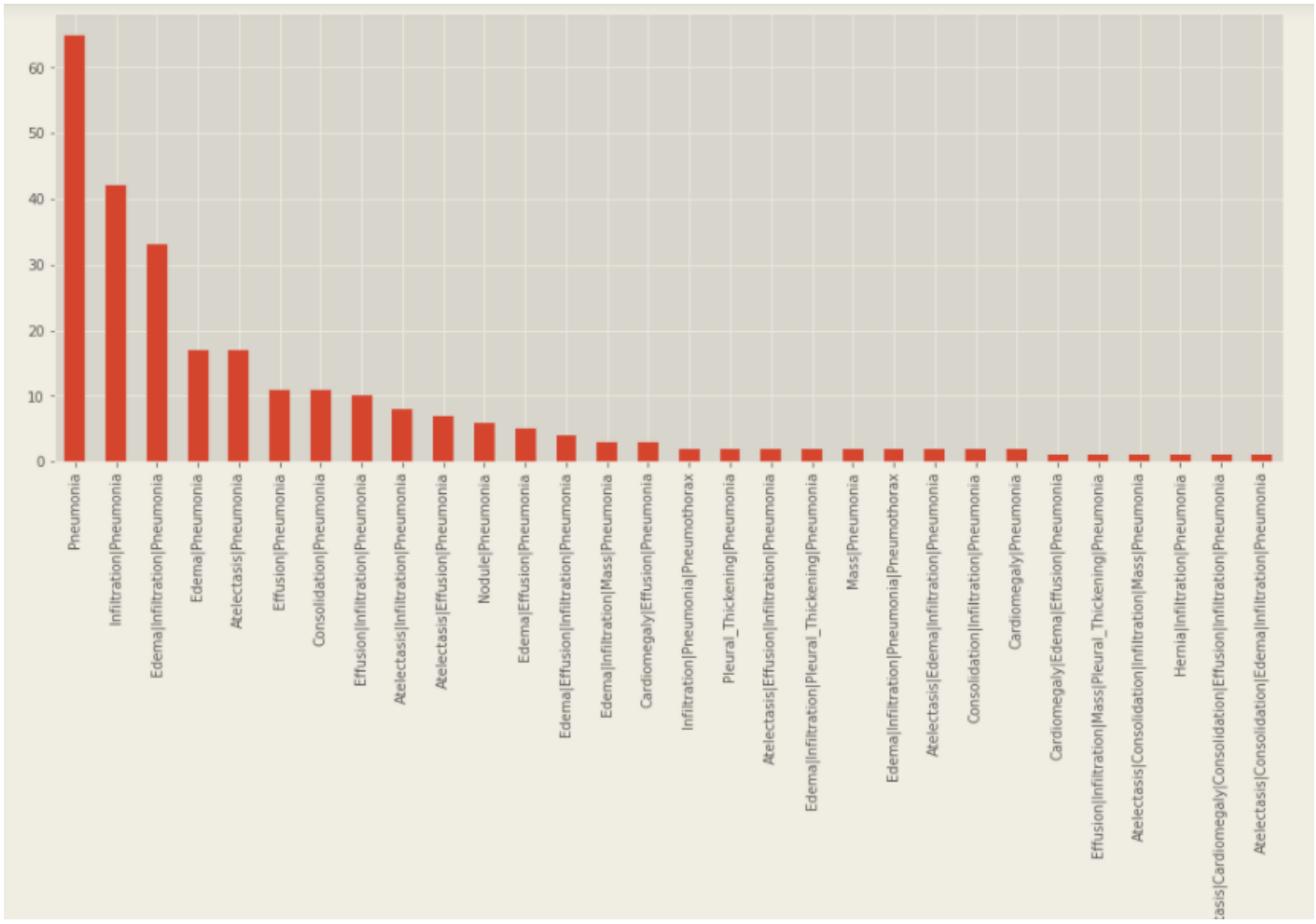


Description of Validation Dataset:
Size of validation dataset: 1430
Ratio of positive to negative cases: 1:4
The patient demographic data (as it is available): graph below
The radiologic techniques used and views taken: X-Ray, two views were present in data, PA & AP
The co-occurrence frequencies of pneumonia with other diseases




Number of Patients in validation data
800
700
600
500
400
300
200
100
0
M
F


Number of Patients in validation data
800
700
600
500
400
300
200
100
0
PA
AP


60
50
40
30
20
10
0
Pneumonia
Infiltration|Pneumonia
Edema|Infiltration|Pneumonia
Edema|Pneumonia
Atelectasis|Pneumonia
Effusion|Pneumonia
Consolidation|Pneumonia
Effusion|Infiltration|Pneumonia
Atelectasis|Infiltration|Pneumonia
Atelectasis|Effusion|Pneumonia
Nodule|Pneumonia
Edema|Effusion|Pneumonia
Edema|Effusion|Infiltration|Pneumonia
Edema|Infiltration|Mass|Pneumonia
Cardiomegaly|Effusion|Pneumonia
Infiltration|Pneumonia|Pneumothorax
Pleural_Thickening|Pneumonia
Atelectasis|Effusion|Infiltration|Pneumonia
Edema|Infiltration|Pleural_Thickening|Pneumonia
Mass|Pneumonia
Edema|Infiltration|Pneumonia|Pneumothorax
Atelectasis|Edema|Infiltration|Pneumonia
Consolidation|Infiltration|Pneumonia
Cardiomegaly|Pneumonia
Cardiomegaly|Edema|Effusion|Pneumonia
Effusion|Infiltration|Mass|Pleural_Thickening|Pneumonia
Atelectasis|Consolidation|Infiltration|Mass|Pneumonia
Hernia|Infiltration|Pneumonia
Atelectasis|Consolidation|Effusion|Infiltration|Pneumonia
Atelectasis|Consolidation|Edema|Infiltration|Pneumonia

5. Ground Truth

Looking at the original paper, these labels were extracted from radiology reports using natural language processing algorithm, they have reported >90% accuracy. I looked at the discussion on original kaggle dataset and some radiologists have also mentioned that a lot of the labels that they looked at are incorrect, just by looking at the X-Ray. Also, final diagnostic is made by looking at various other reports such as blood work, BAL samples, microbiological assays, patient condition, medical history etc.

6. FDA Validation Plan

Patient Population Description for FDA Validation Dataset:
Below are based on EDA on the dataset:

Age ranges : 1-95 years
Sex: Male and Female
Type of imaging modality: X-Ray
Body part imaged: Chest
Prevalence of disease of interest: 20%

Ground Truth Acquisition Methodology:
Weighted score of a group of 3-4 radiologists' labels, as this is the industry standard.

Algorithm Performance Standard:
Based on the research paper provided, I chose F1-score. In the CheXNet study paper, average radiologist F1 score (0.387) seemed to be acceptable, so my algorithm should be able to have an F1-score above that.