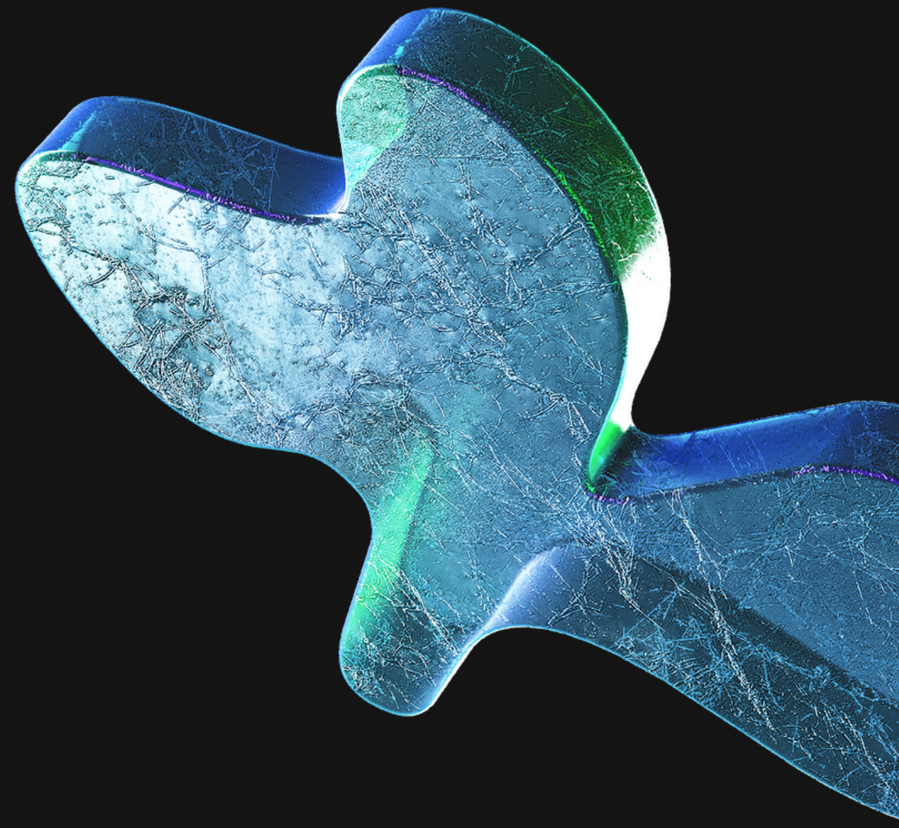
SC1015 Mini Project

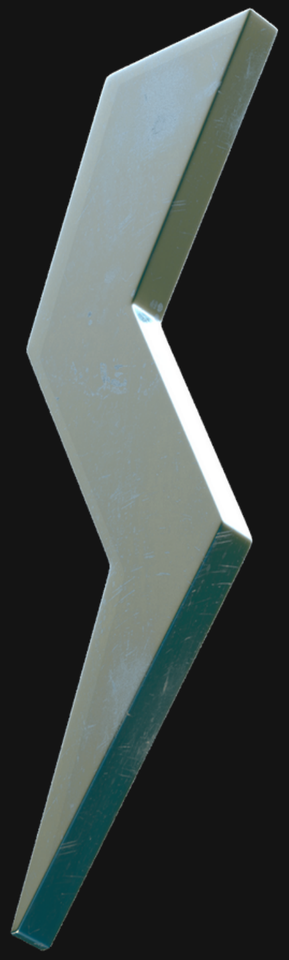
Classifying Lung X-Ray



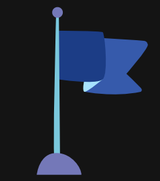
Anson, Tushar and Viral

Agenda

What you need to know

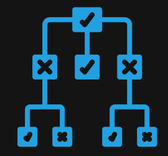


- Motivation

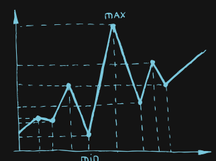


- Data Collection, Preparation, Exploration

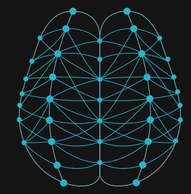
- Classifying using Decision Trees



- Classifying using Logistic Regression



- Classifying using CNN

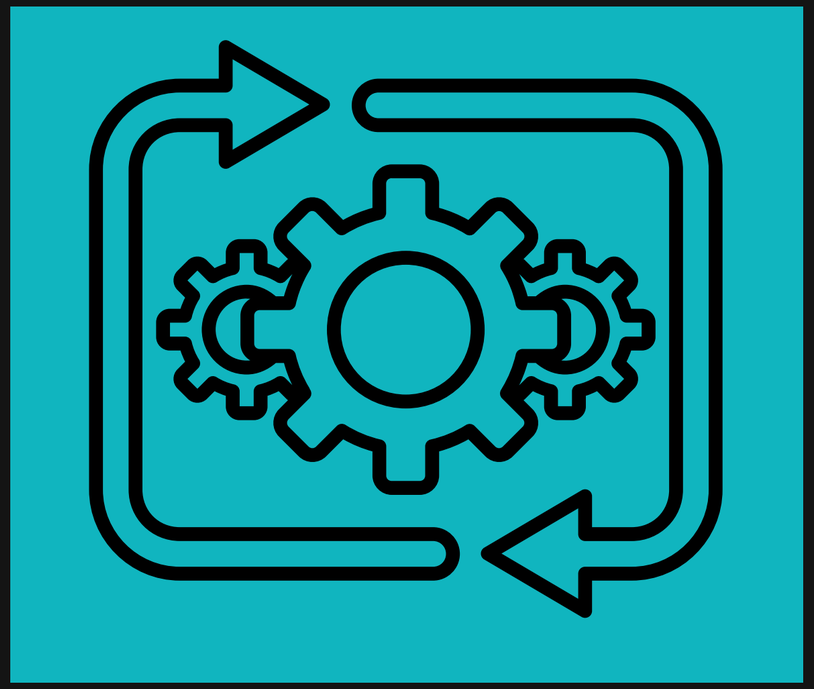


- Comparing and Evaluating Models

- Improvements to our Methods

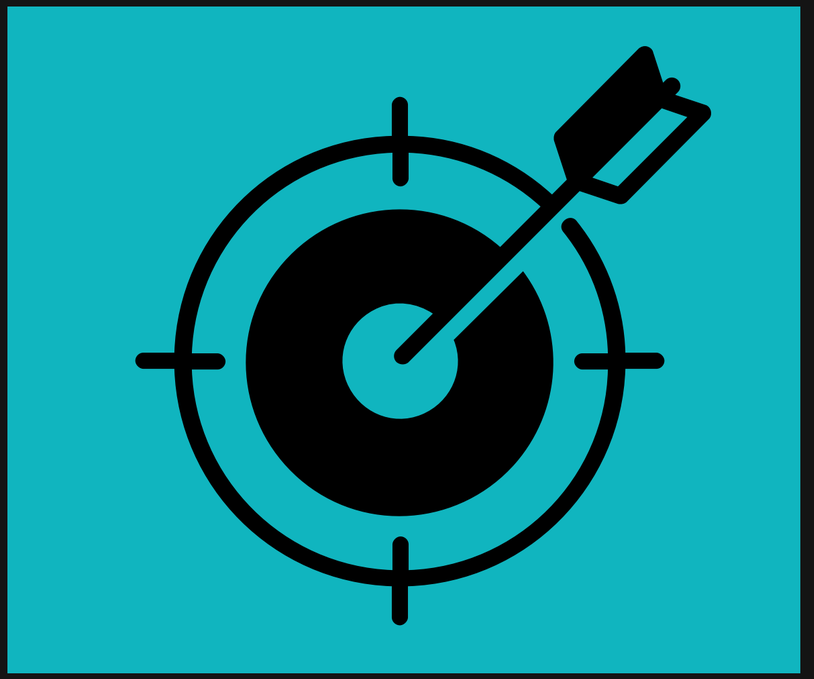
Motivation

AUTOMATED DIAGNOSTICS



PROVIDE DEEPER INSIGHTS INTO LUNG DISEASES

BETTER ACCURACY



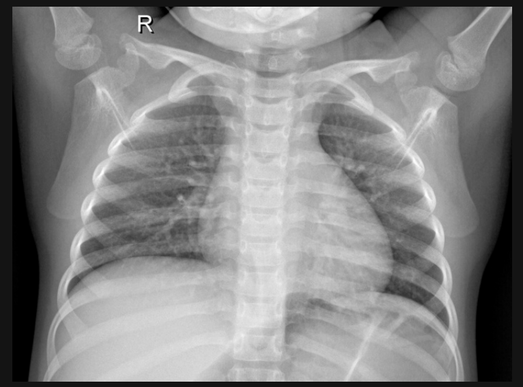
CXR had a pooled sensitivity of 0.77 and a specificity of 0.91.

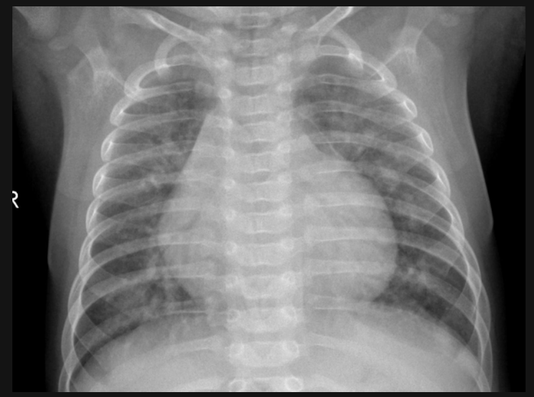
Data Collection

Mendeley Data

Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images

By: Daniel Kermany, Kang Zhang, Michael Goldbaum



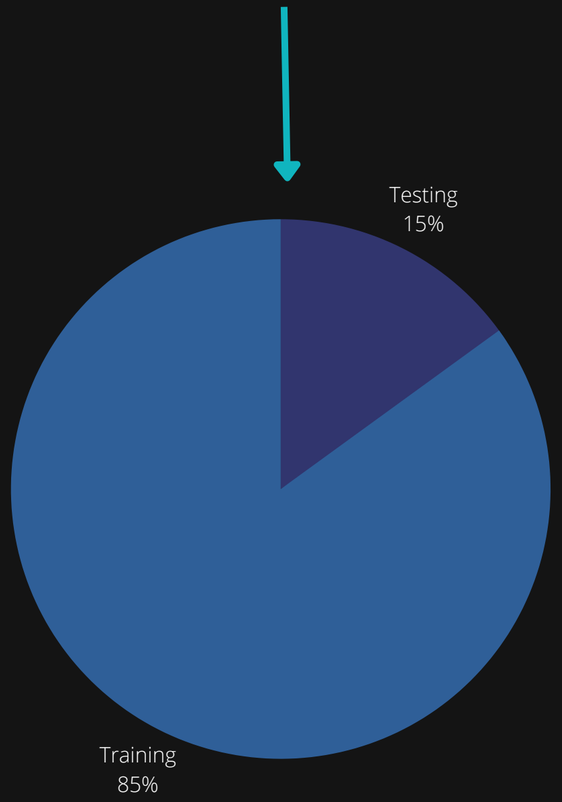


Data Preparation

5856 images

4273 Pneumonia

1583 Normal



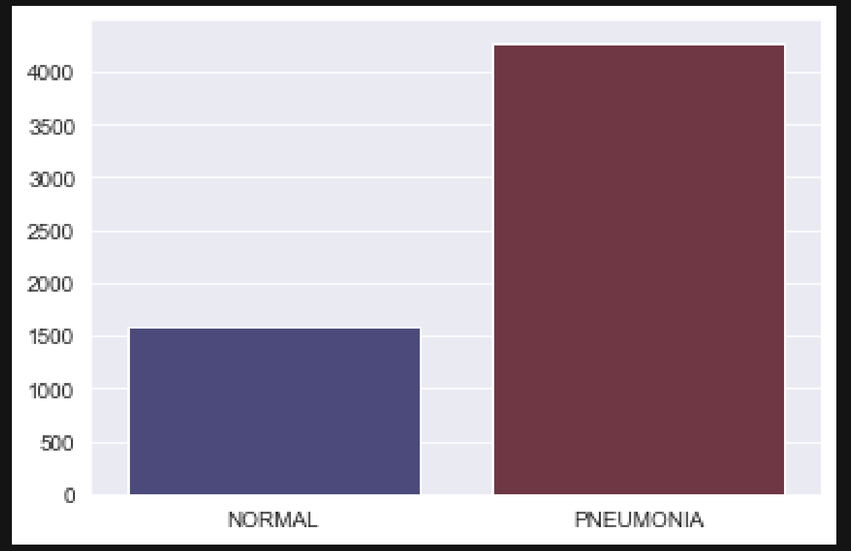


IMAGE DATA IN JPEG

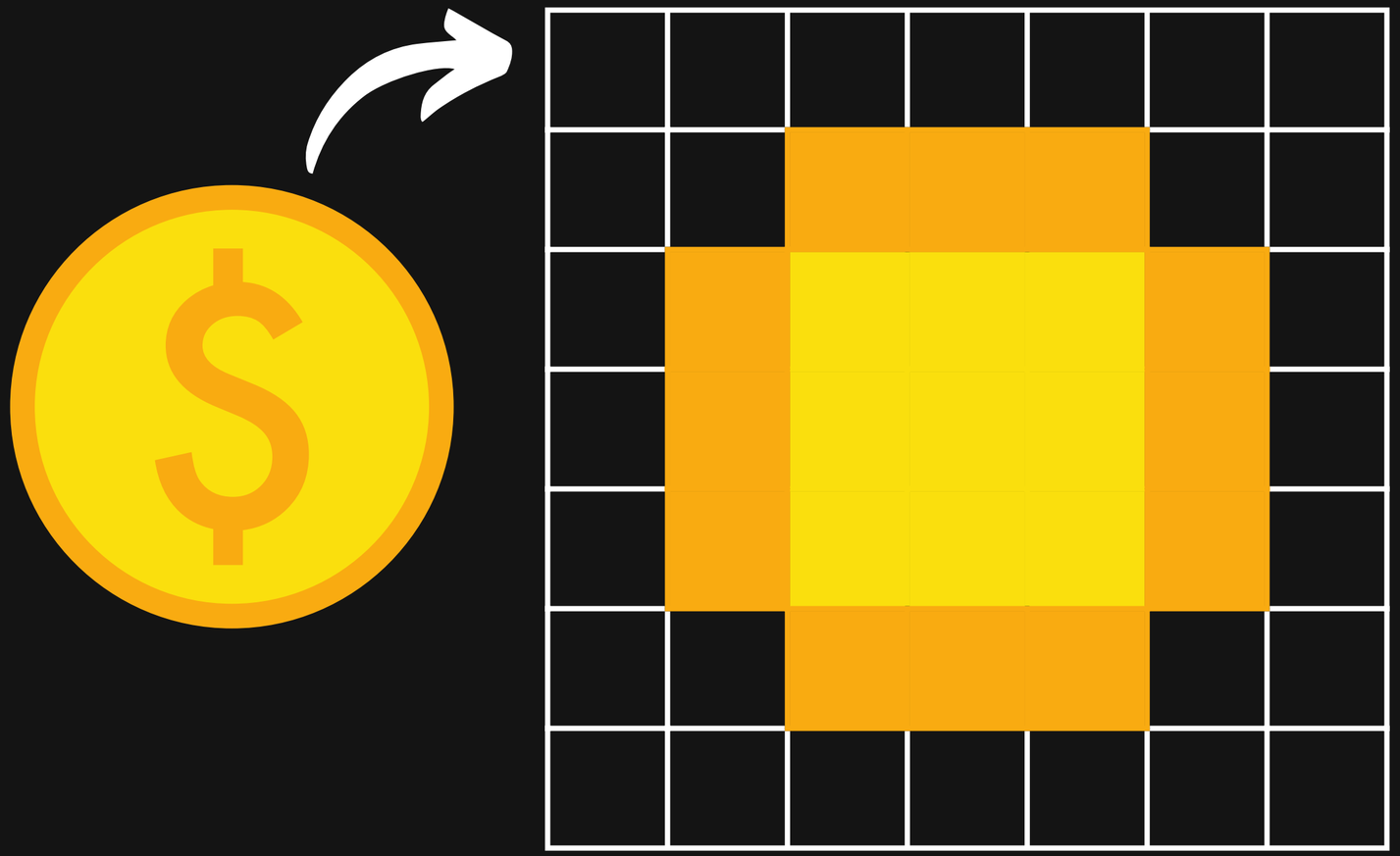
Each image is a Chest X-ray.
Each image is labeled as Normal or Pneumonia.
Different Dimensions for each one.

Data Preparation

TRANSLATING IMAGES TO NUMBERS

Standardize the size scale for each image to 100x100.
Converting the Image to Grayscale. So each Pixel has a value from 0 to 255 of Grayscale. 0 is white and 255 is Pitch Black.
Create a 2D Array.

scale to 6x6

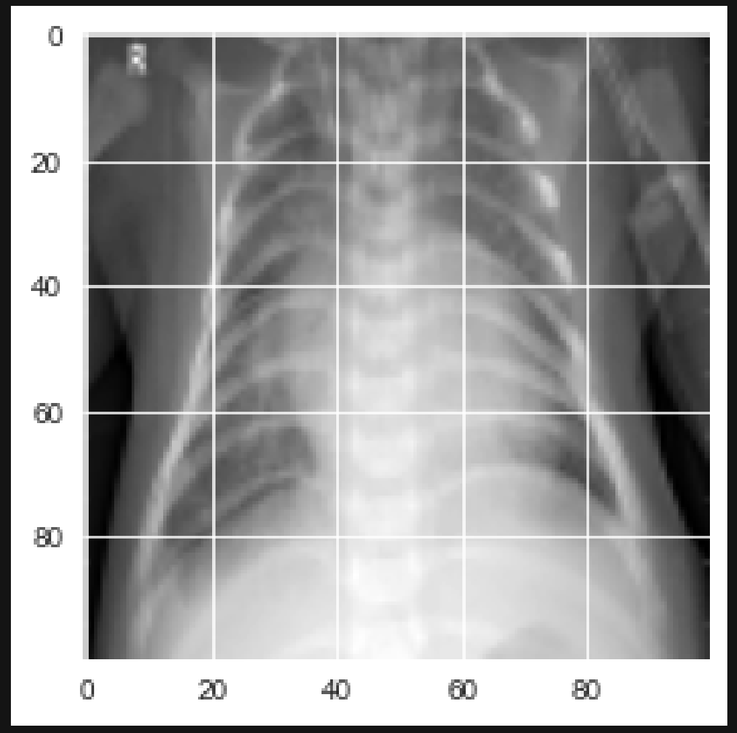


255	255	255	255	255	255	255
255	255	204	204	204	255	255
255	204	102	102	102	204	255
255	204	102	102	102	204	255
255	204	102	102	102	204	255
255	255	204	204	204	255	255
255	255	255	255	255	255	255

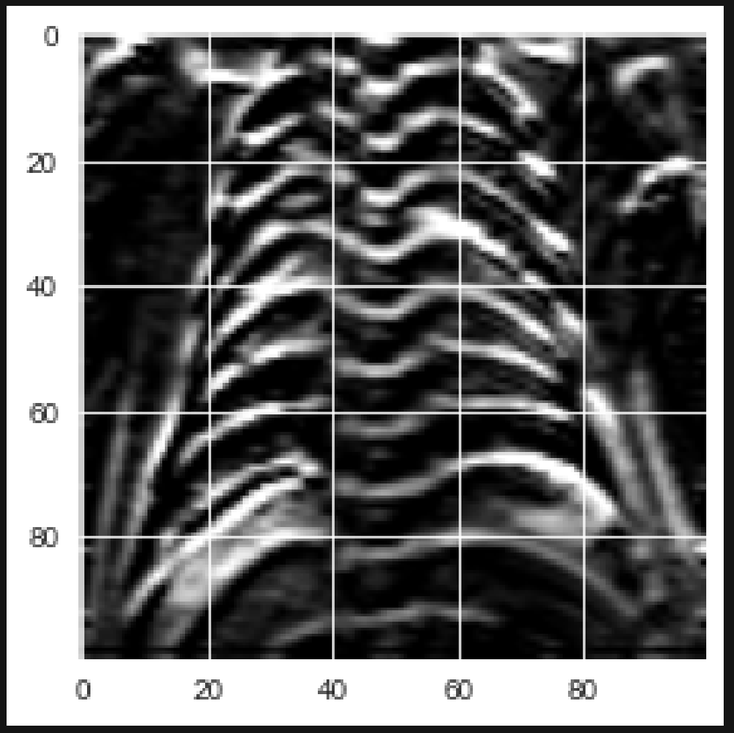
Data Preparation

FILTERING THE IMAGE THROUGH CONVOLUTION MATRICES

Applying these filter by multiplying the matrices using OpenCV functions. Highlights horizontal and vertical lines.




```
mat_y = np.array([[ -1, -2, -1],
                  [ 0, 0, 0],
                  [ 1, 2, 1]])
mat_x = np.array([[ -1, 0, 1],
                  [ 0, 0, 0],
                  [ 1, 2, 1]])

filtered_image = cv2.filter2D(gray, -1, mat_y)
filtered_image = cv2.filter2D(filtered_image, -1, mat_x)
```



Data Preparation

CONVERTING 2D DATA TO 1D.

This allows us to see an image as a list of pixels. Also helps in predictions, as each pixel would be a feature.

Filtered 2D Image Data 6x6

255	255	255	255	255	255	255
255	255	51	51	51	255	255
255	51	204	204	204	51	255
255	51	204	204	204	51	255
255	51	204	204	204	51	255
255	255	51	51	51	255	255
255	255	255	255	255	255	255

Flatten

1D array of 36 elements

255	255	255	255	255	255	255	255	255	51	51	51	255	255	255	51	204	204	204	51	255

Data Preparation

INTO A DATAFRAME

Each image is a subject or data, where the pixels are the features and the label is the Y-values.

	0	1	2	3	4	5	6	7	8	9	...	9991	9992	9993	9994	9995	9996	9997	9998	9999	label
0	0	0	0	0	0	6	18	30	18	0	...	0	0	0	0	0	2	6	8	8	NORMAL
1	0	0	0	0	0	94	202	140	32	0	...	2	0	0	0	0	0	0	0	0	NORMAL
2	208	255	255	255	255	238	0	0	0	0	...	36	60	52	16	0	0	0	0	0	NORMAL
3	0	0	4	12	8	0	0	6	32	52	...	255	166	0	0	0	0	0	0	0	NORMAL
4	0	0	0	0	0	6	6	0	0	0	...	0	0	0	0	0	0	0	0	0	NORMAL
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4268	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	PNEUMONIA
4269	0	2	22	36	28	12	0	0	0	0	...	168	158	120	48	8	0	0	0	0	PNEUMONIA
4270	255	255	255	150	244	255	255	255	138	0	...	2	0	2	6	6	2	0	0	0	PNEUMONIA
4271	0	0	0	0	0	166	255	255	255	130	...	18	12	4	0	0	4	4	0	0	PNEUMONIA
4272	200	204	154	66	22	192	255	255	255	255	...	0	0	0	0	0	0	0	0	0	PNEUMONIA

5856 rows × 10001 columns

Data Exploration

EXPLORING THE LABELS

Instead of exploring each image, we are going to simplfy and derive a general representation of each label by finding the average mean.

	0	1	2	3	4	5	6	7	8	9	...	9991	9992	9993	9994	9995	9996	9997	9998	9999	label
0	0	0	0	0	0	6	18	30	18	0	...	0	0	0	0	0	2	6	8	8	NORMAL
1	0	0	0	0	0	94	202	140	32	0	...	2	0	0	0	0	0	0	0	0	NORMAL
2	208	255	255	255	255	238	0	0	0	0	...	36	60	52	16	0	0	0	0	0	NORMAL
3	0	0	4	12	8	0	0	6	32	52	...	255	166	0	0	0	0	0	0	0	NORMAL
4	0	0	0	0	0	6	6	0	0	0	...	0	0	0	0	0	0	0	0	0	NORMAL
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4268	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	PNEUMONIA
4269	0	2	22	36	28	12	0	0	0	0	...	168	158	120	48	8	0	0	0	0	PNEUMONIA
4270	255	255	255	150	244	255	255	255	138	0	...	2	0	2	6	6	2	0	0	0	PNEUMONIA
4271	0	0	0	0	0	166	255	255	255	130	...	18	12	4	0	0	4	4	0	0	PNEUMONIA
4272	200	204	154	66	22	192	255	255	255	255	...	0	0	0	0	0	0	0	0	0	PNEUMONIA

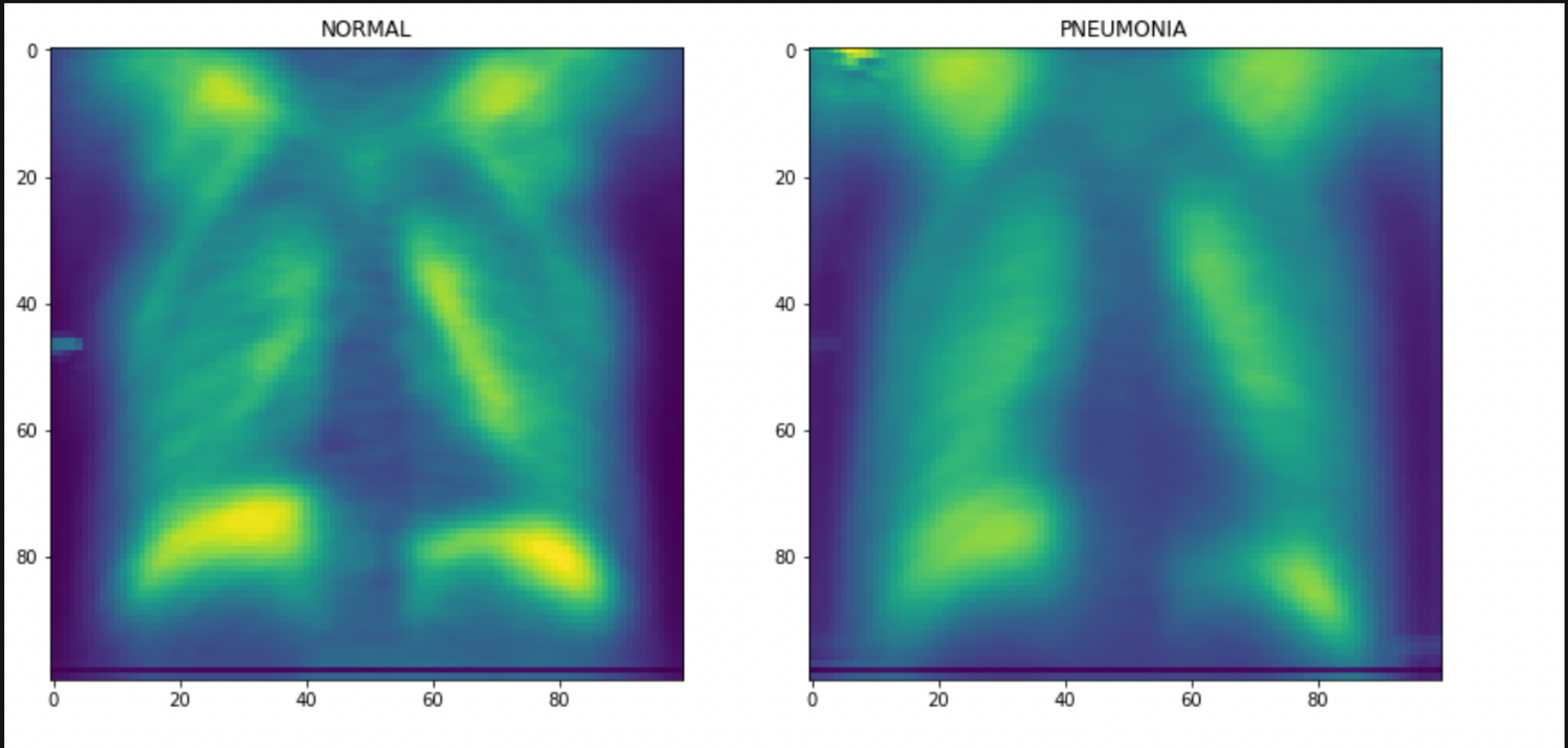
5856 rows × 10001 columns

Avg. Normal

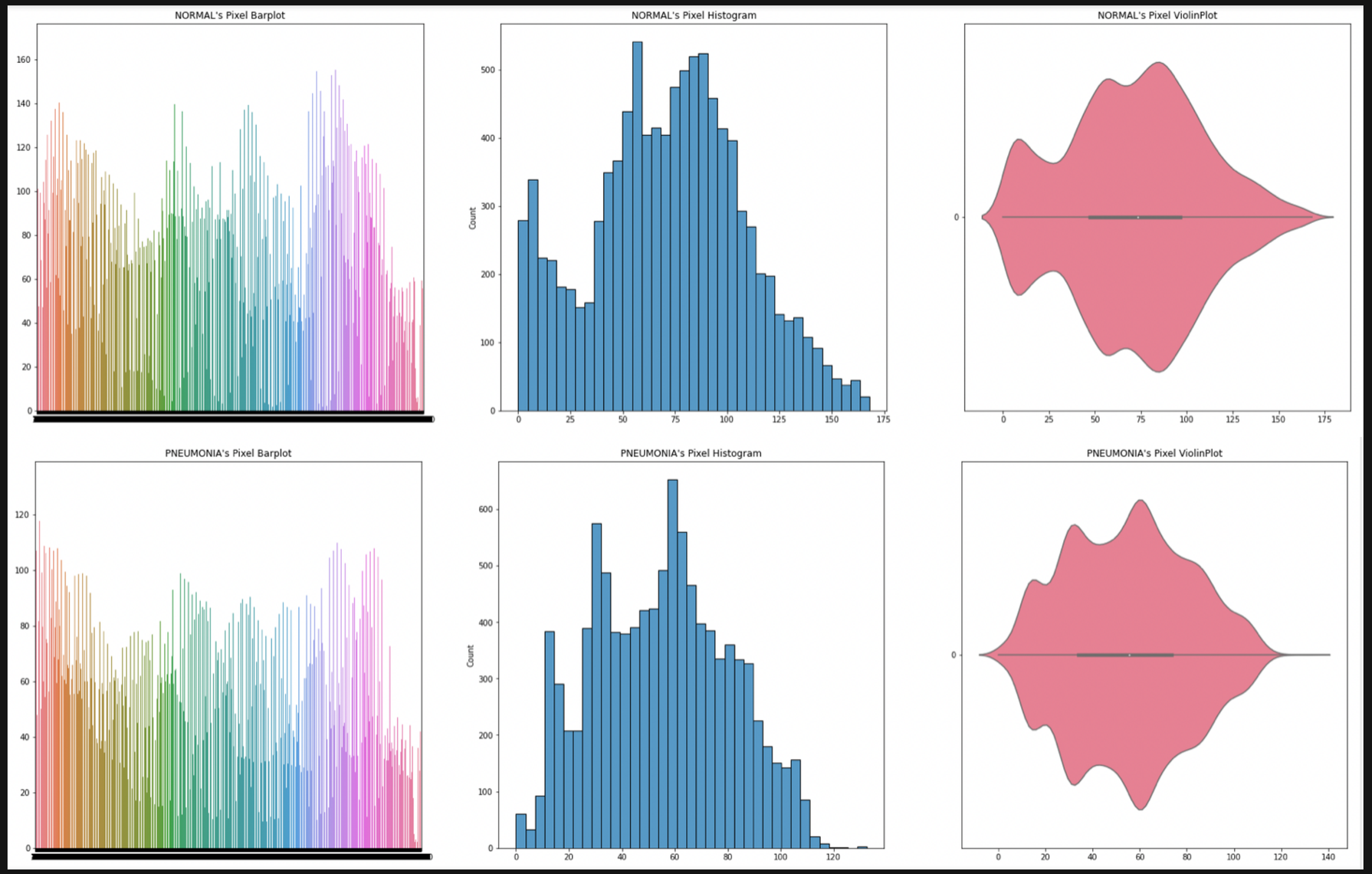
Avg. Pneumonia

Data Exploration

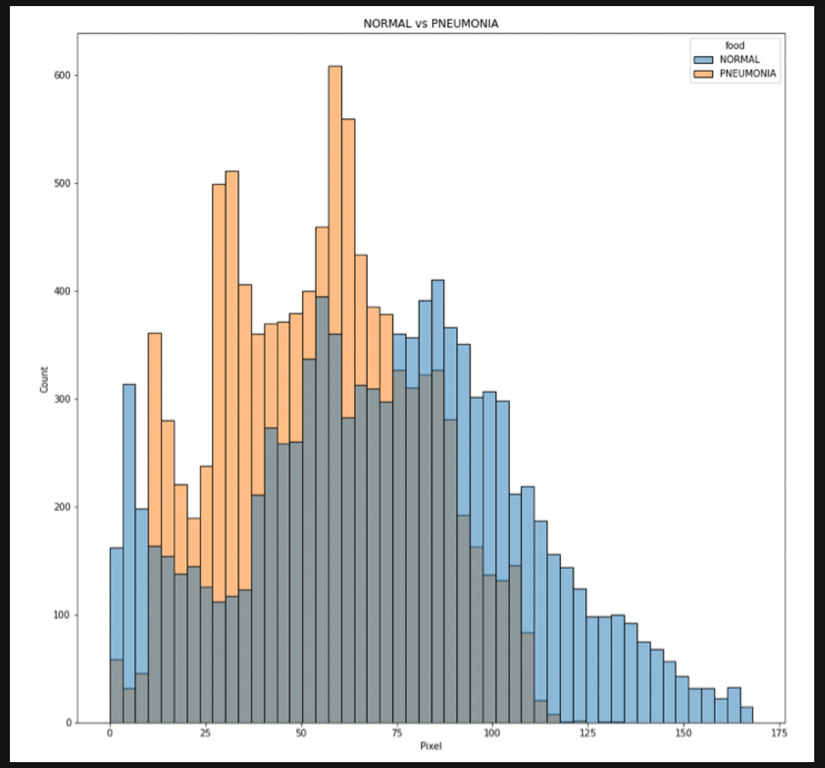
AVERAGE LABELS IN IMAGES

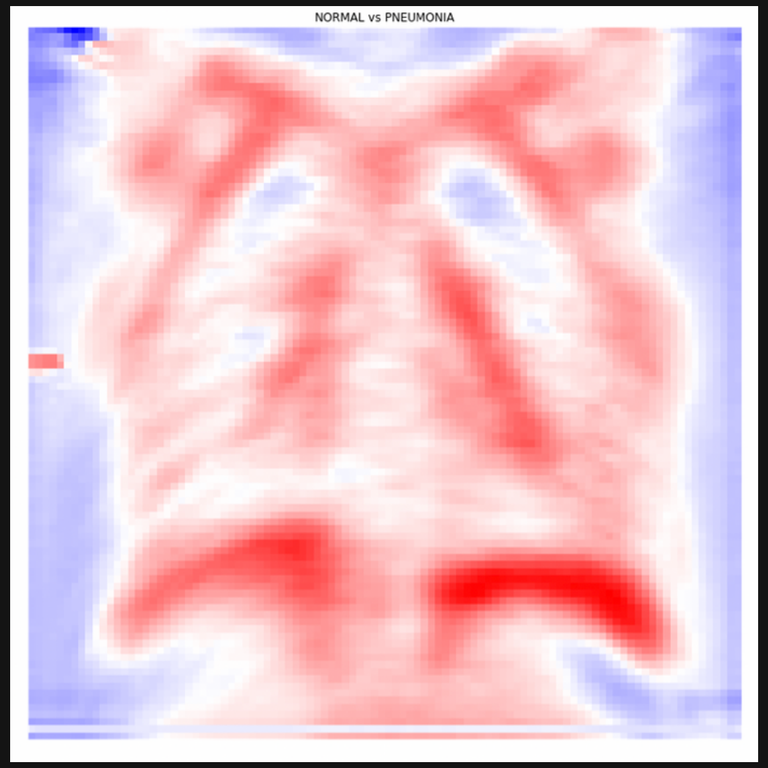


Data Exploration



Data Exploration





Kolmogorov-Smirnov tests

```
import scipy
scipy.stats.ks_2samp(foodavg['NORMAL'], foodavg['PNEUMONIA'])
```

```
KstestResult(statistic=0.2504, pvalue=1.2368469271974868e-275)
```

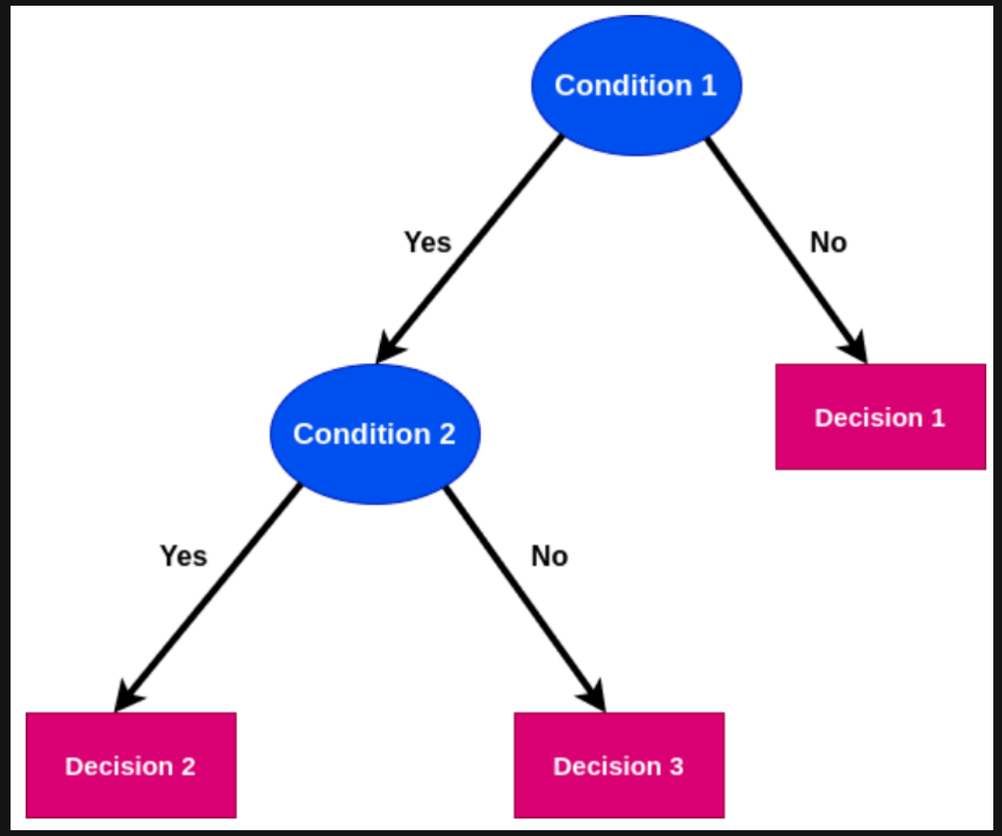
Rejects Null Hypothesis, since p-value <0.05

Classification with Decision Tree

Graphical representation of all possible solutions to a decision based on certain conditions

Import Decision Tree Classifier and Confusion Matrix

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
```



Classification with Decision Tree

Training decision tree to use values in X_train to determine category in y_train

```
X_train = traindf.drop(['type'],axis = 1)
```

```
y_train = pd.DataFrame(traindf['type'].astype('category'))
```

Train DF:

	0	1	2	3	4	5	6	7	8	9	...	9991	9992	9993	9994	9995	9996	9997	9998	9999	type
0	8	8	0	0	6	6	0	72	208	208	...	0	0	0	0	0	0	0	0	0	normal
1	255	158	0	0	214	255	255	255	255	134	...	0	0	0	0	0	0	0	0	0	normal
2	0	0	0	0	44	88	90	108	150	184	...	84	112	108	90	92	82	38	6	6	normal
3	0	0	0	0	180	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	normal
4	0	0	0	0	0	0	0	0	18	66	...	0	0	0	0	0	0	0	0	0	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5227	152	152	152	156	164	164	156	160	184	216	...	0	0	0	28	52	40	28	20	20	pneumonia
5228	255	255	255	0	28	66	88	106	108	104	...	4	0	0	0	0	0	0	0	0	pneumonia
5229	0	0	0	0	0	0	0	0	4	12	...	0	46	84	38	0	12	58	106	106	pneumonia
5230	8	0	0	0	0	0	14	30	52	80	...	54	42	18	4	2	4	2	0	0	pneumonia
5231	4	0	0	0	0	0	0	0	0	0	...	78	72	24	0	4	28	50	50	50	pneumonia

Classification with Decision Tree

Using values in X_test in decision tree to determine category y_test

```
X_test = testdf.drop(['type'],axis = 1)
```

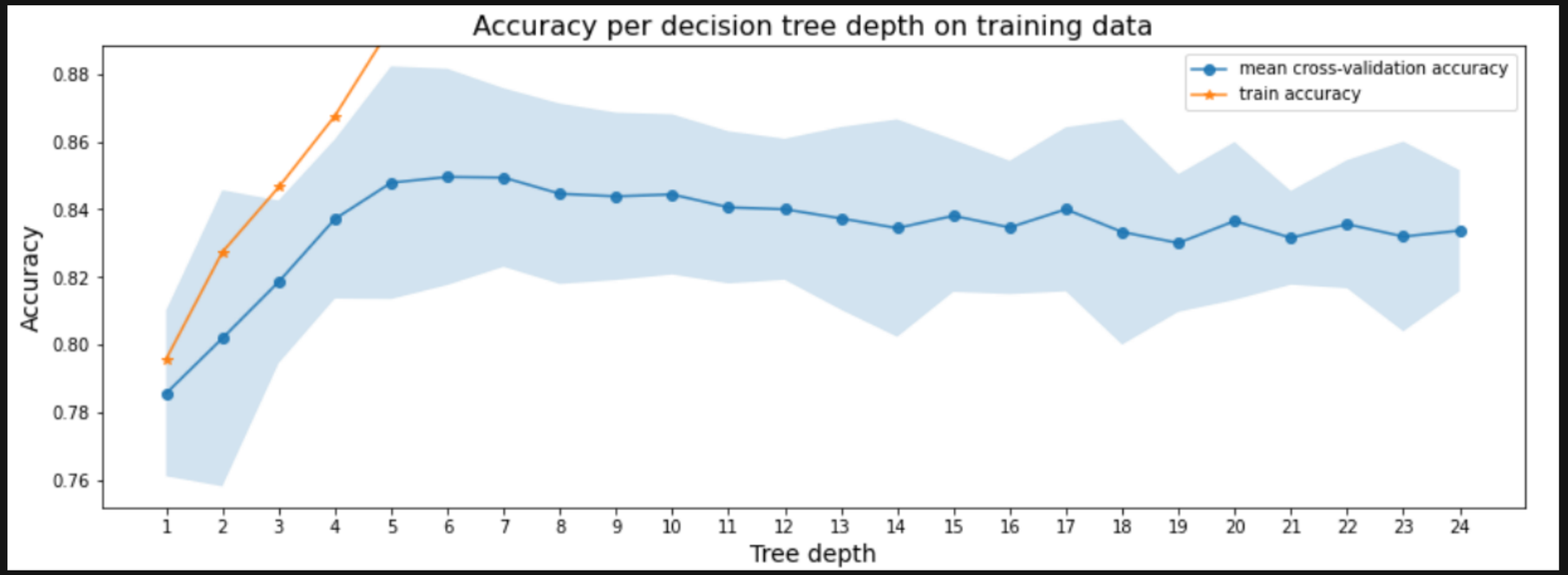
```
y_test = pd.DataFrame(testdf['type'].astype('category'))
```

Test DF:

	0	1	2	3	4	5	6	7	8	9	...	9991	9992	9993	9994	9995	9996	9997	9998	9999	type
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	2	6	6	2	2	normal
1	0	0	0	0	0	0	0	0	0	0	...	0	8	10	2	0	0	0	0	0	normal
2	0	0	0	12	32	36	22	6	0	0	...	58	14	0	0	0	0	0	0	0	normal
3	0	0	124	255	255	255	255	255	255	255	...	0	0	0	0	0	0	0	0	0	normal
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	10	18	12	8	8	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
619	46	222	255	154	0	0	8	22	14	0	...	194	112	128	104	56	20	2	0	0	pneumonia
620	42	14	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	2	6	6	pneumonia
621	118	178	226	216	184	190	224	232	220	226	...	98	78	70	60	24	0	0	0	0	pneumonia
622	14	255	255	255	70	0	0	0	0	0	...	0	0	0	0	0	0	24	72	72	pneumonia
623	0	154	255	255	255	0	0	0	0	0	...	12	2	0	0	0	0	0	0	0	pneumonia

Classification with Decision Tree

Selecting depth of Classification Tree by running cross validation on the trees from depth of range 1 to 24



We selected depth 7 through this method
- avoiding overfitting
- better chance to reproduce the accuracy and generalize the model

```
from sklearn.model_selection import cross_val_score
```

Classification with Decision Tree

```
dectree = DecisionTreeClassifier(max_depth = 7)
dectree.fit(X_train, y_train)
y_train_pred = dectree.predict(X_train)
y_test_pred = dectree.predict(X_test)
```

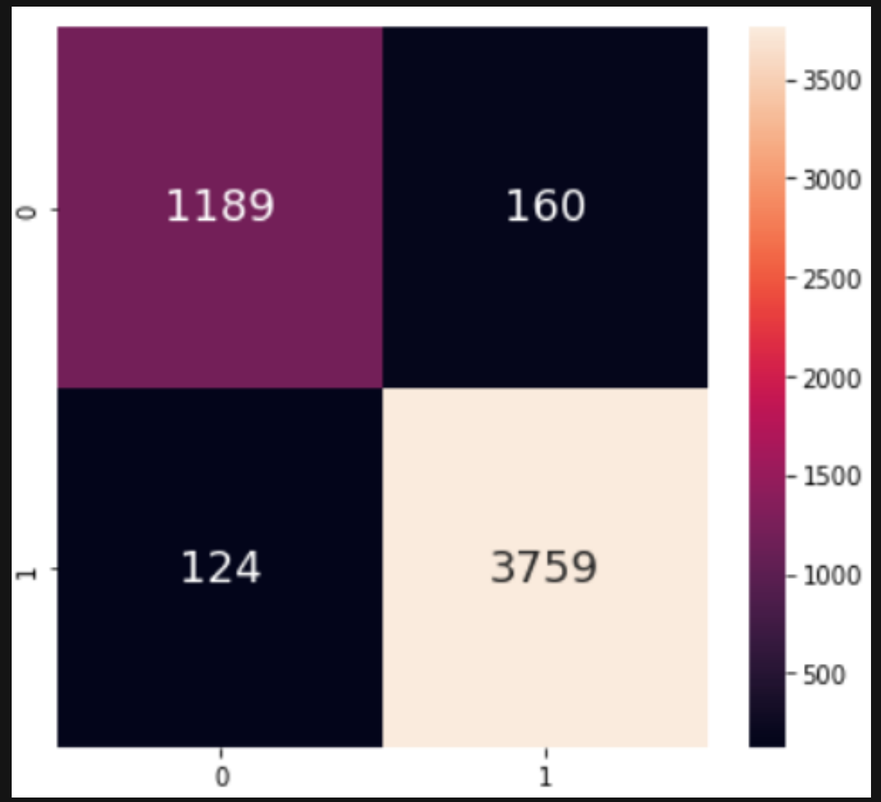
```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 0.9243119266055045

Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.7339743589743589
```

Classification with Decision Tree

Train Set

False Positive Rate:
160/1349 ≈ 11.9%

False Negative Rate:
124/3883 ≈ 3.2%



True Positive Rate:
3759/3883 ≈ 96.8%

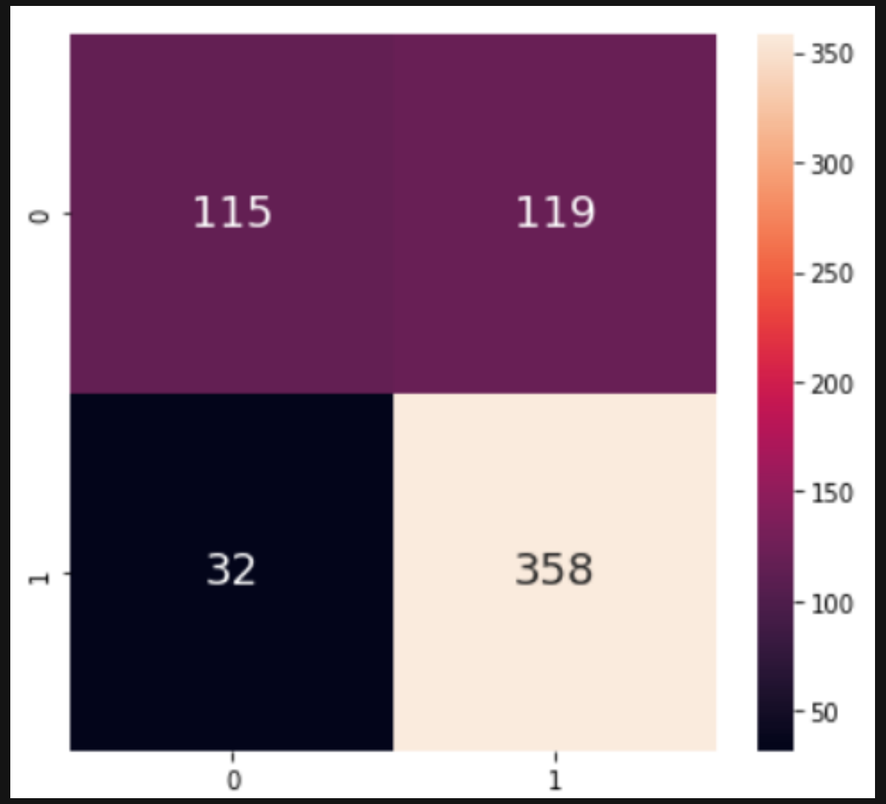
True Negative Rate:
1189/1349 ≈ 88.1%

Classification with Decision Tree

Test Set

False Positive Rate:
119/234 ≈ 50.9%

False Negative Rate:
32/390 ≈ 8.2%



True Positive Rate:
358/390 ≈ 91.8%

True Negative Rate:
115/234 ≈ 49.1%

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

sklearn is one of the most used libraries for machine learning

It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis.

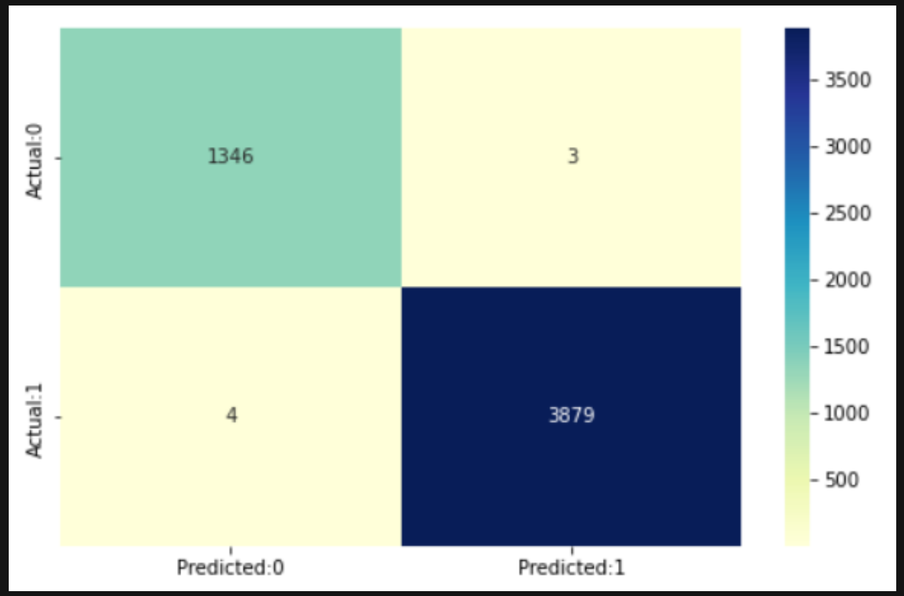
Precision and Accuracy Scores of Train and Test Data

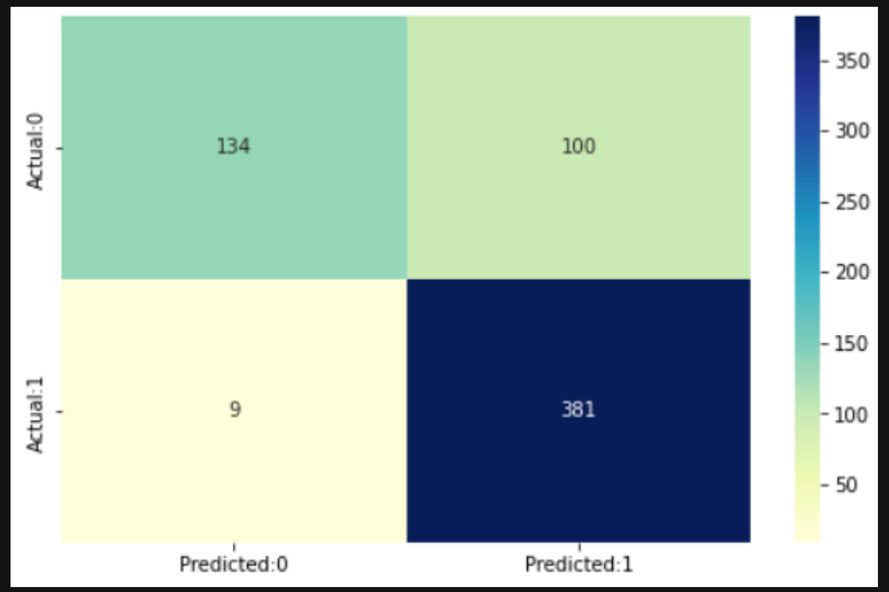
```
logmodel = LogisticRegression(solver='lbfgs', max_iter=26)
logmodel.fit(X_train, np.ravel(y_train))


predictions1 = logmodel.predict(X_train)
print("\t\tTrain Data")
print("Precision Score \t: ",precision_score(y_train, predictions1,
                                                 average='weighted'))
print("Accuracy\t\t: ",accuracy_score(y_train, predictions1))

print("\t\tTest Data")
predictions = logmodel.predict(X_test)
print("Precision Score \t: ",precision_score(y_test, predictions,
                                                 average='weighted'))
print("Accuracy\t\t: ",accuracy_score(y_test, predictions))
```

```
                Train Data
Precision Score         :  0.9986624981203052
Accuracy                :  0.9986620795107034
                Test Data
Precision Score         :  0.8464609714609715
Accuracy                :  0.8253205128205128
```

Using Confusion Matrix for comparison

```
                          TRAIN DATA
The acuuracy of the model          = TP+TN/(TP+TN+FP+FN)          =  0.9986620795107034
 The Missclassification            = 1-Accuracy                   =  0.0013379204892965957
 Sensitivity or True Positive Rate = TP/(TP+FN)                   =  0.9989698686582539
 Specificity or True Negative Rate = TN/(TN+FP)                   =  0.9977761304670126
 Positive Predictive value         = TP/(TP+FP)                   =  0.999227202472952
 Negative predictive Value         = TN/(TN+FN)                   =  0.997037037037037
```






```
                          TEST DATA
The acuuracy of the model          = TP+TN/(TP+TN+FP+FN)          =  0.8253205128205128
 The Missclassification            = 1-Accuracy                   =  0.17467948717948723
 Sensitivity or True Positive Rate = TP/(TP+FN)                   =  0.9769230769230769
 Specificity or True Negative Rate = TN/(TN+FP)                   =  0.5726495726495726
 Positive Predictive value         = TP/(TP+FP)                   =  0.7920997920997921
 Negative predictive Value         = TN/(TN+FN)                   =  0.9370629370629371
```

Predicting Training and Testing Accuracy

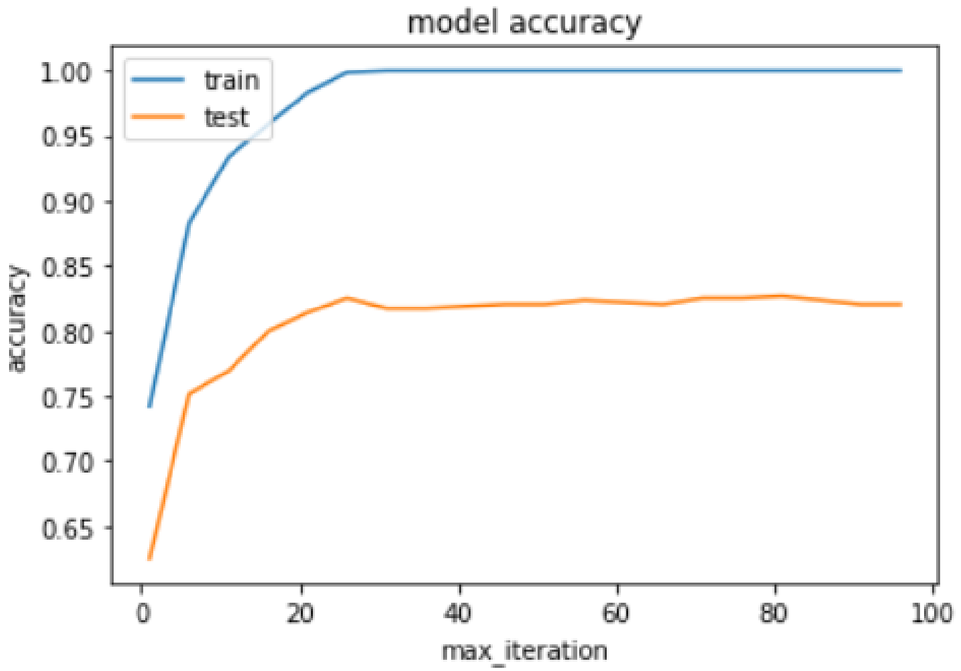
Python predict() function enables us to predict the labels of the data values on the basis of the trained model.

```
# predict - Predict class labels for samples in X
log_reg.predict(X_train)
y_pred = log_reg.predict(X_train)

# predict_proba - Probability estimates
pred_proba = log_reg.predict_proba(X_train)

# coef_ - Coefficient of the features in the decision function
log_reg.coef_
```

```
# Accuracy on Train
print("The Training Accuracy is: ", log_reg.score(X_train, y_train))

# Accuracy on Test
print("The Testing Accuracy is: ", log_reg.score(X_test, y_test))
```

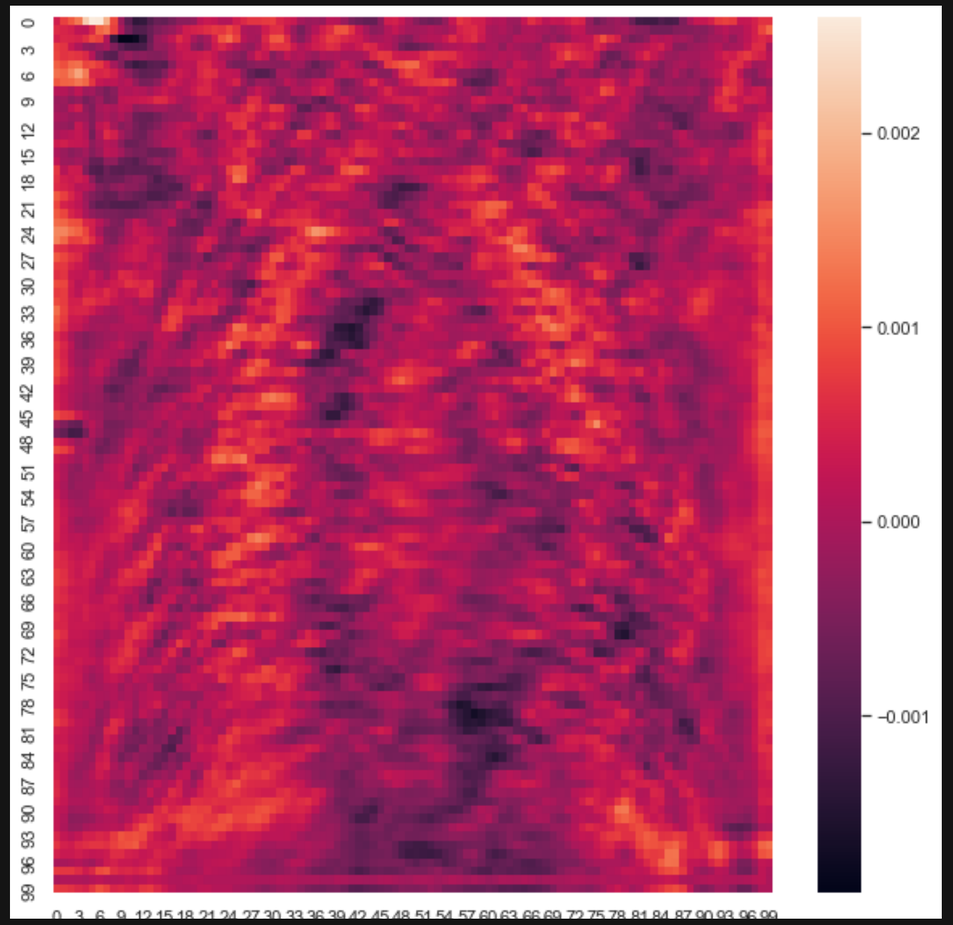
```
The Training Accuracy is:  0.9986620795107034
The Testing Accuracy is:  0.8253205128205128
```

```
In [35]: log_reg.classes_

Out[35]: array(['normal', 'pneumonia'], dtype=object)

In [36]: cm.sum(axis=1)
         cm_norm

Out[36]: array([[0.99777613, 0.00222387],
                [0.00103013, 0.99896987]])

In [37]: cm

Out[37]: array([[1346,    3],
                [   4, 3879]], dtype=int64)

In [38]: cm.sum(axis=0)

Out[38]: array([1350, 3882], dtype=int64)

In [39]: np.diag(cm)

Out[39]: array([1346, 3879], dtype=int64)
```



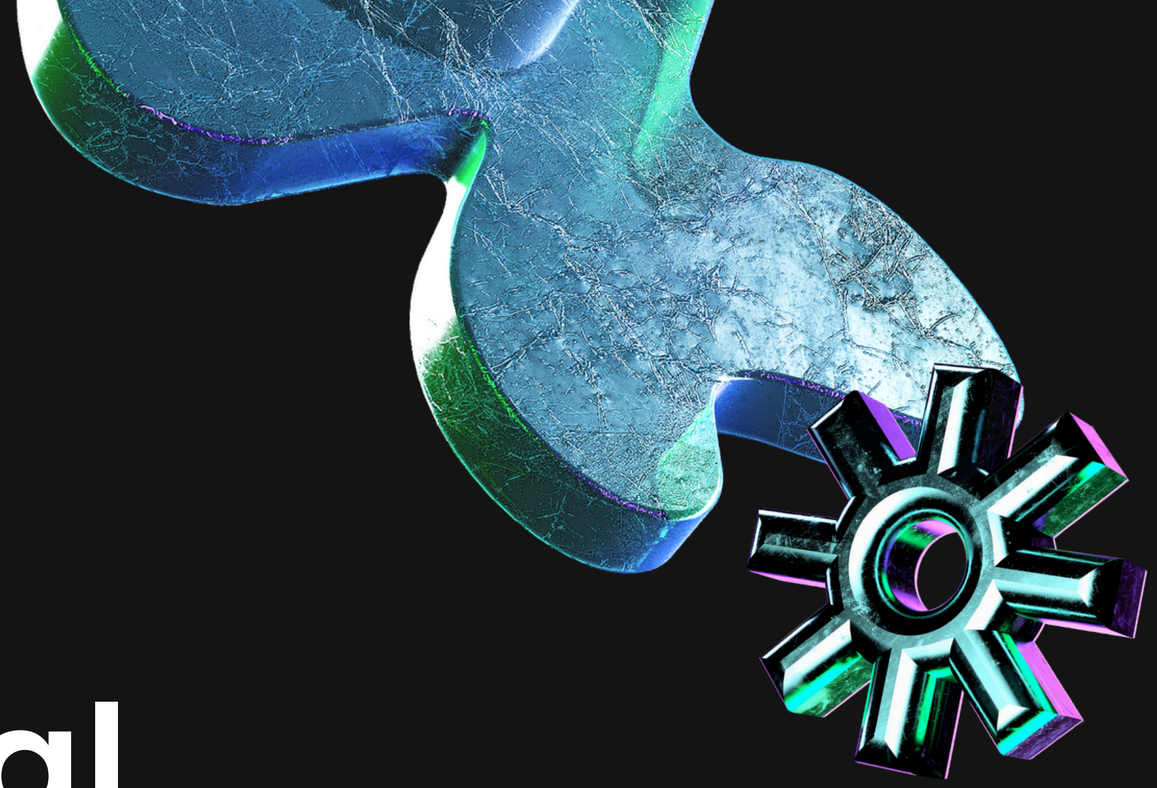

```
#logmodel with 26 iteration
logmodel = LogisticRegression(solver='lbfgs', max_iter=26)
logmodel.fit(X_train, np.ravel(y_train))

predictions = logmodel.predict(X_test)

print("Accuracy: ",accuracy_score(y_test, predictions))
```

```
Accuracy:  0.8253205128205128
```

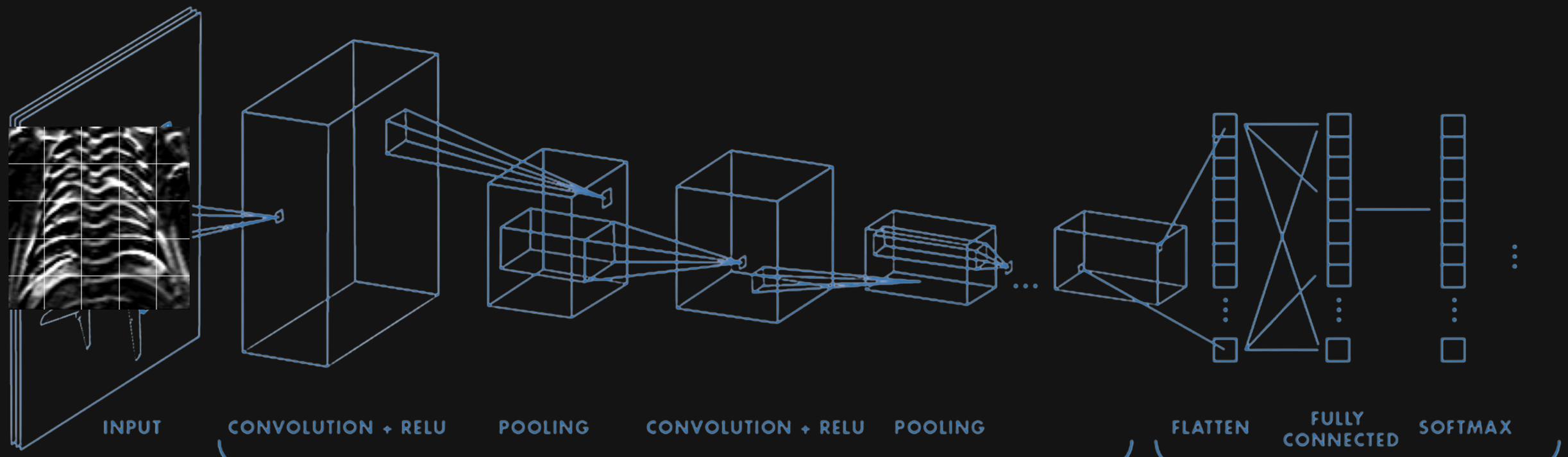


Seeing what our Model sees.

The heatplot of the coefficients of the model.
Clearly it looks a little like the lungs images.
Darker spots are (negative values)->Normal
Lighter Spots (orange to white) are (positive values)-> Pneumonia



Using Convolutional Neural Network


INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
Feature Learning
Structural Condition Classification

Transfer Learning using Resnet

- Resnet50 is a powerful trained CNN model.
- So we used the same convolution layer as Resnet, but change the output layers to predict Normal or Pneumonia and input layers for the 224x224 values.

```
base_model = tensorflow.keras.applications.ResNet152V2(
    weights='imagenet',
    input_shape=(224, 224, 1),
    include_top=False)

base_model.trainable = False

def get_pretrained():

    #Input shape = [width, height, color channels]
    inputs = layers.Input(shape=(224, 224, 1))

    x = base_model(inputs)

    # Head
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(128, activation='relu')(x)
    x = layers.Dropout(0.1)(x)

    #Final Layer (Output)
    output = layers.Dense(1, activation='sigmoid')(x)

    model = keras.Model(inputs=[inputs], outputs=output)

    return model
```

Pre-Image Preparation

ZOOMING, SHIFTING AND ETC... FOR MORE ROBUST MODEL

```
train_datagen = ImageDataGenerator(rescale=1./255, zoom_range = 0.1,
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1)
test_datagen= ImageDataGenerator(rescale=1./255)
```

VALIDATION DATASET

TO check for overfit during training.

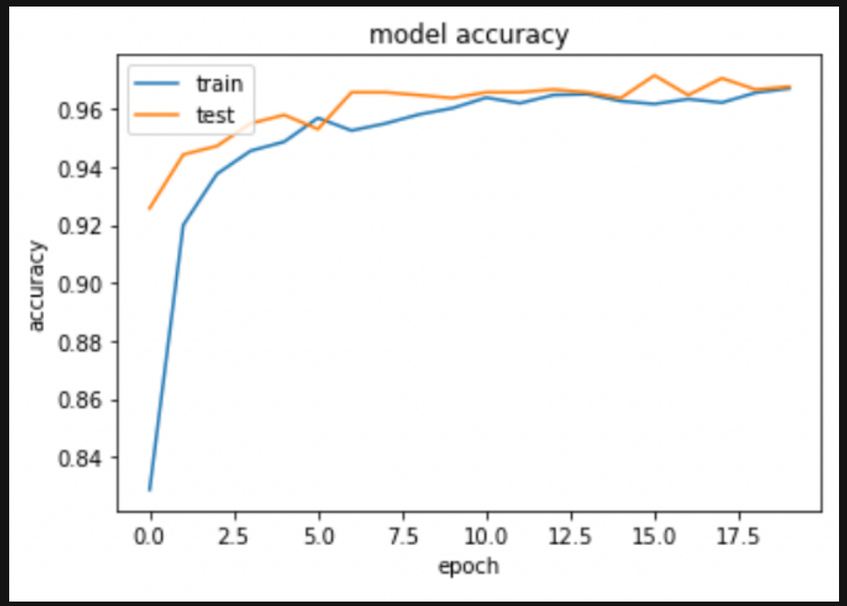
75% of total data for training
10% for Validation
15% for Testing

Training the Model

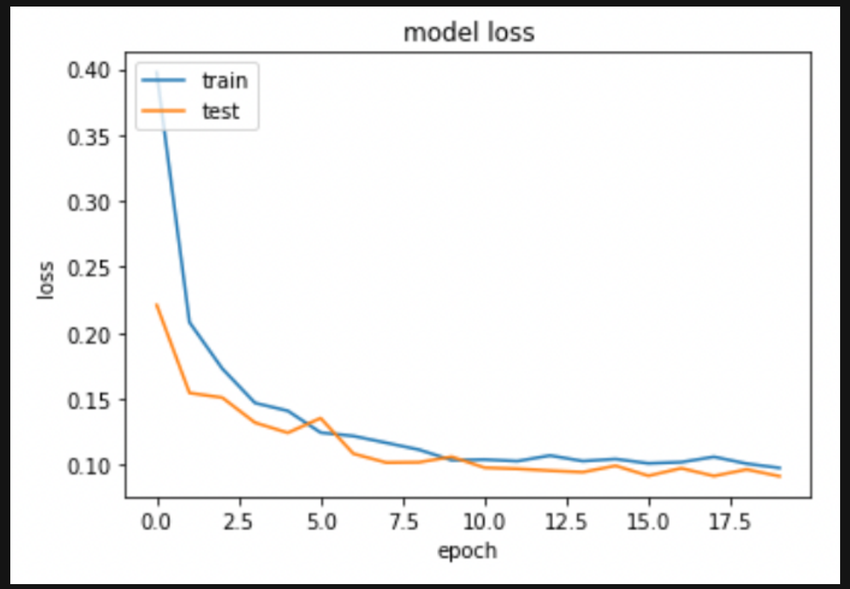
Validation: Yes
Loss Model: Binary Cross-entropy
Metrics to Obsere: Binary Accuracy
Optimizer: Adam
Epochs: 20 (trial and error)

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 resnet152v2 (Functional)    (None, 7, 7, 2048)        58331648

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 58,594,049
Trainable params: 262,401
Non-trainable params: 58,331,648
_________________________________________________________________
```

Accuracy of Training and Val.



Loss of Validation and Val.

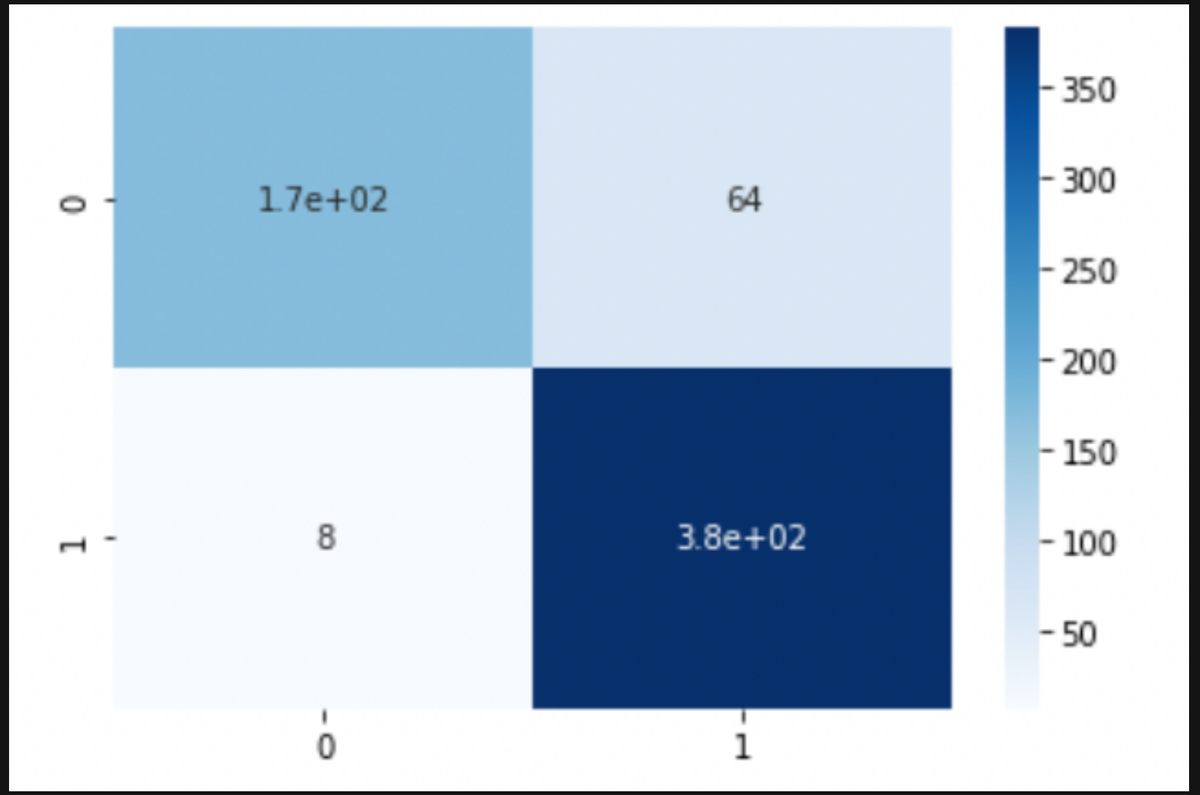


Evaluating the Model

Using Testing Data

TOTAL ACCURACY: 88.46%
LOSS: 0.2846

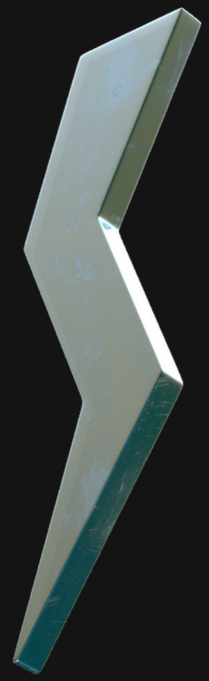
```
True Positive Rate: 98.0 %.
True Negative Rate: 73.0 %.
False Positive Rate: 2.0 %.
False Negative Rate: 27.0 %.
```



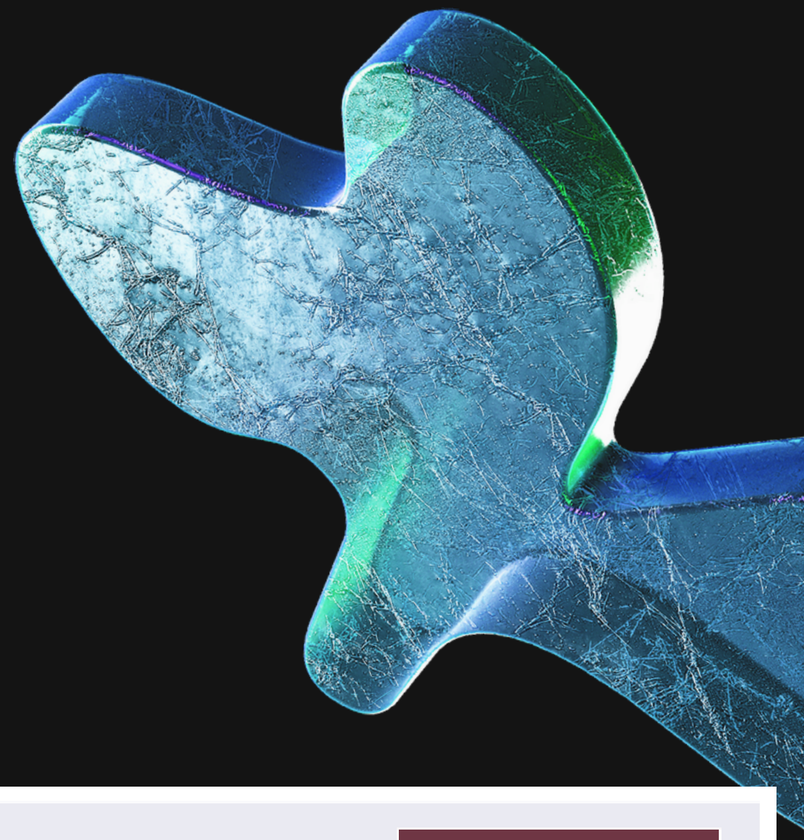
Best Model to use

Clearly Convolution Neural Network

However, models like Logistic Regression could be used for less complex training, as it took more than 20 minutes to train for the CNN, while less than a minute for the both models.



Limitation



The True Negative Rate or the Accuracy is low throughout the models.

WHY?

Imbalance Class data.

Pneumonia has 2x more data than normal.
Solution:

1. Reduce the Pneumonia to be same size of Normal, but we lose a lot of data.
2. Modify the Loss function during training such that the penalty for misclassifying "normal" is more.