FINAL PROJECT

Abnormal Tuberculosis Detection using Deep Learning

Darnell Kikoo - Binus University

TABLE OF CONTENTS

BUSINESS BACKGROUND

EDA AND MODEL SELECTION

MODELLING AND EVALUATION

CONCLUSION AND RECCOMENDATION

Business Background

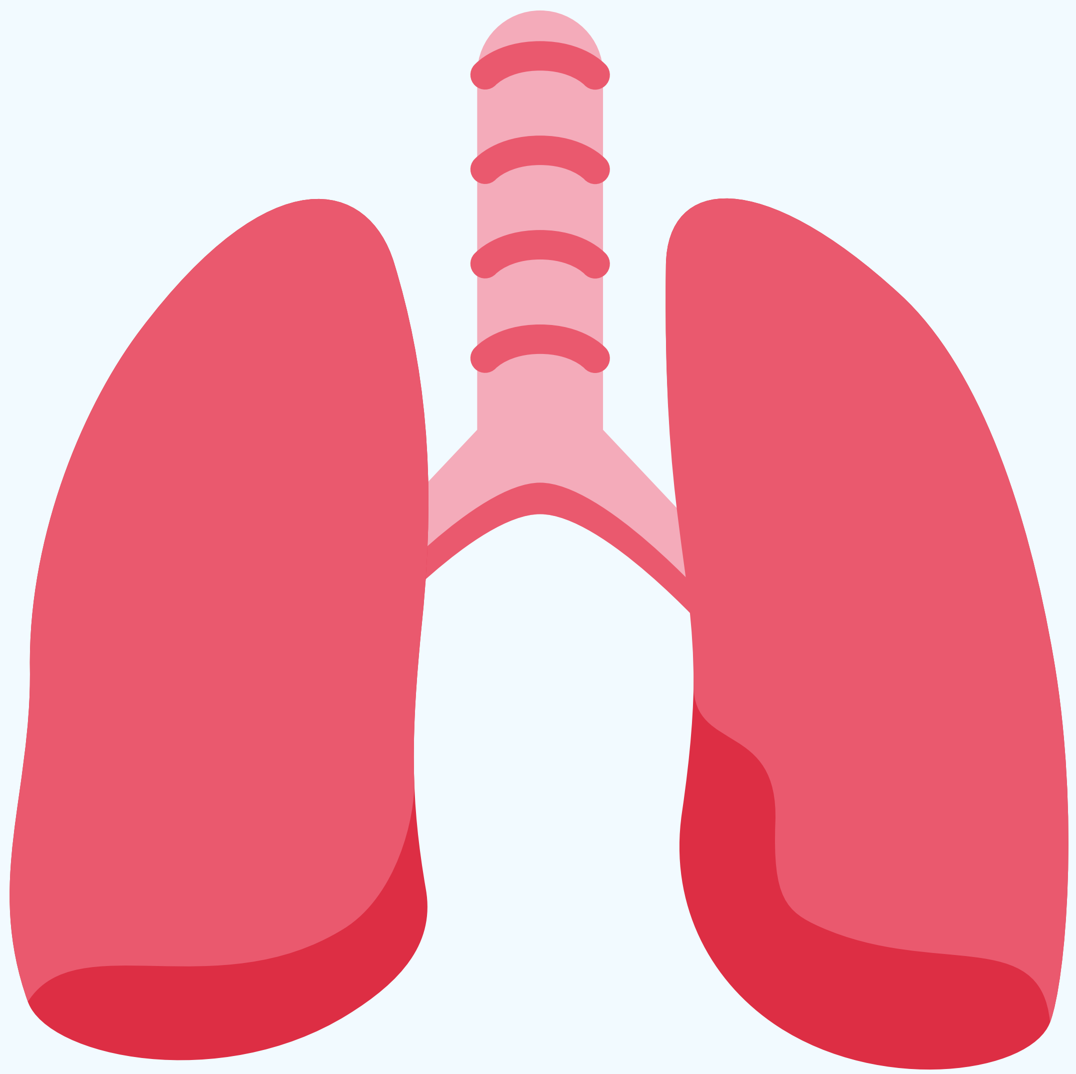
BACKGROUND

I am a Data Scientist working at one of the Health Organization in China

Doctors here still manually assess the patients' lung condition manually. Hence, I was asked to create a model which could reduce the time wasted on X-Ray Image Scanning

BUSINESS OBJECTIVE

The **Business Objective** of this project is to create a model which can classify the condition of patients' lungs based on their X-Ray scan, in order to reduce the time wasted on manual assessment from doctors



OUTPUT

The output of this project is a **model** that can accurately predict a patient's lung condition based on their X-Ray Image

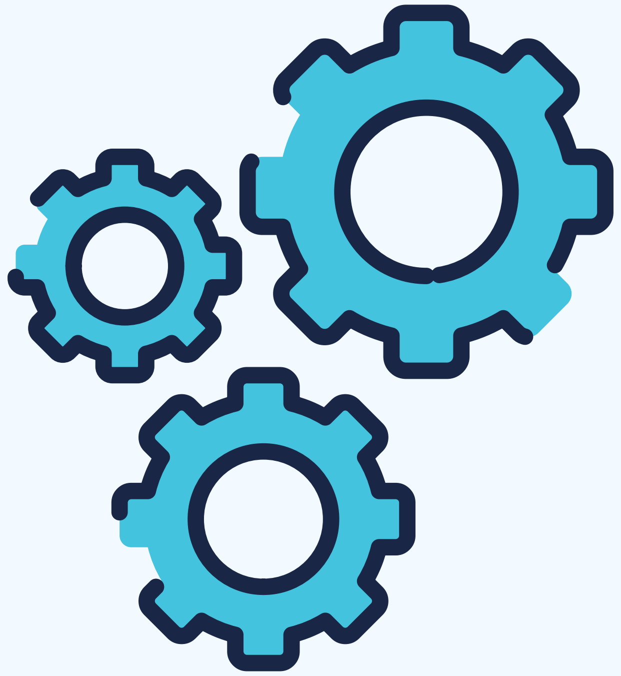
The data consists of 800 Images of patients' X-Ray Image Scan

PROJECT LIMITATION

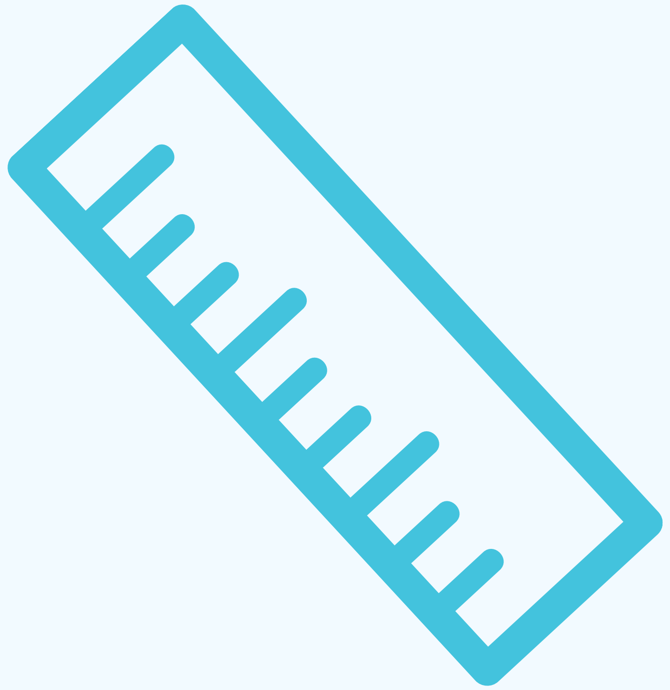
Limited Time and GPU Supply

Limited number of resources
(Only 800 images are available)

ANALYTIC APPROACH



Using Deep Learning
(Convolutional Neural Network)
for **Supervised Learning**



Metrics : **Recall**,
Supported by Accuracy and
AUC for measurement support

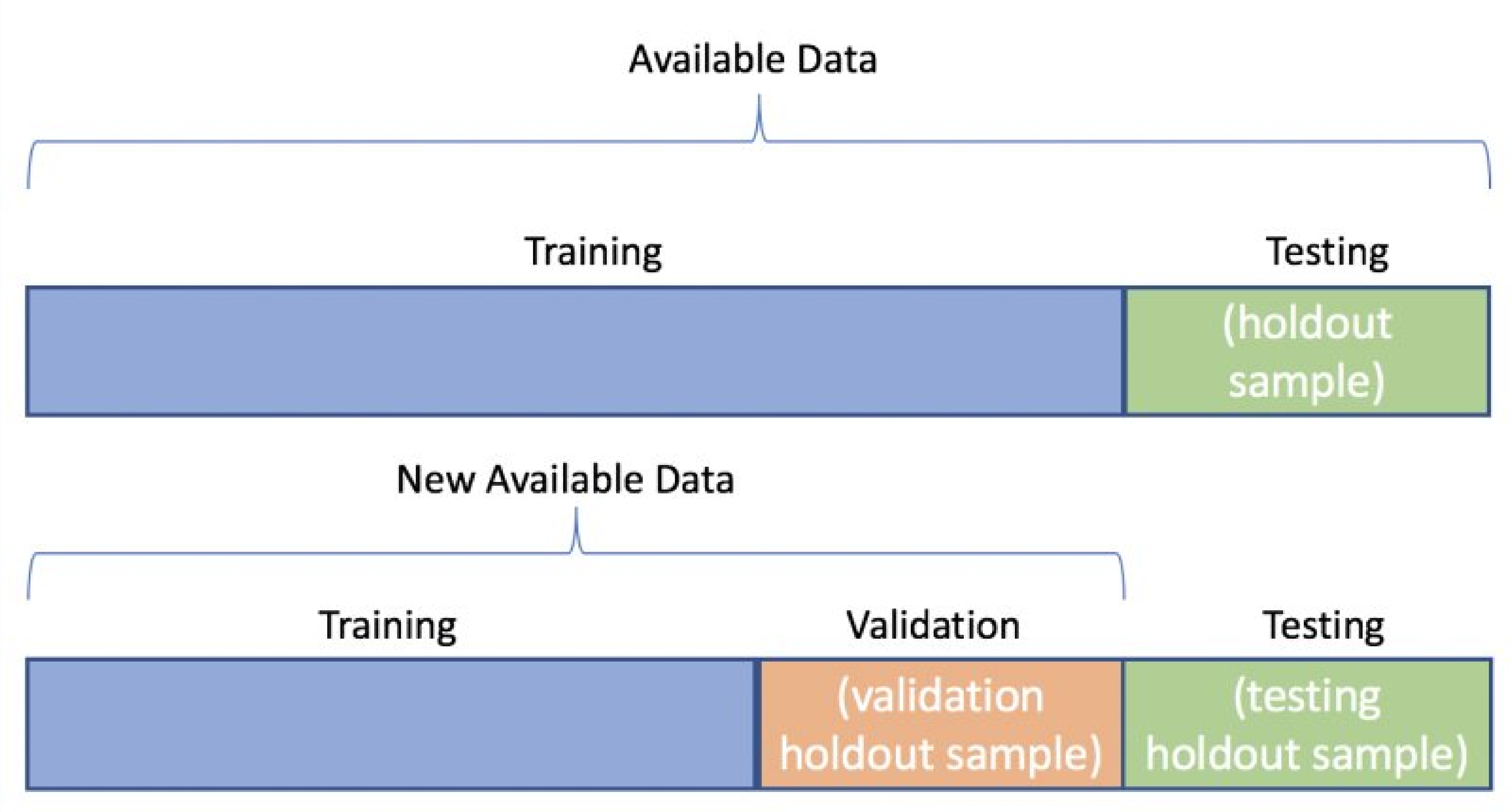
Exploratory Data Analysis and Model Selection

DATA COLLECTION

kaggle™

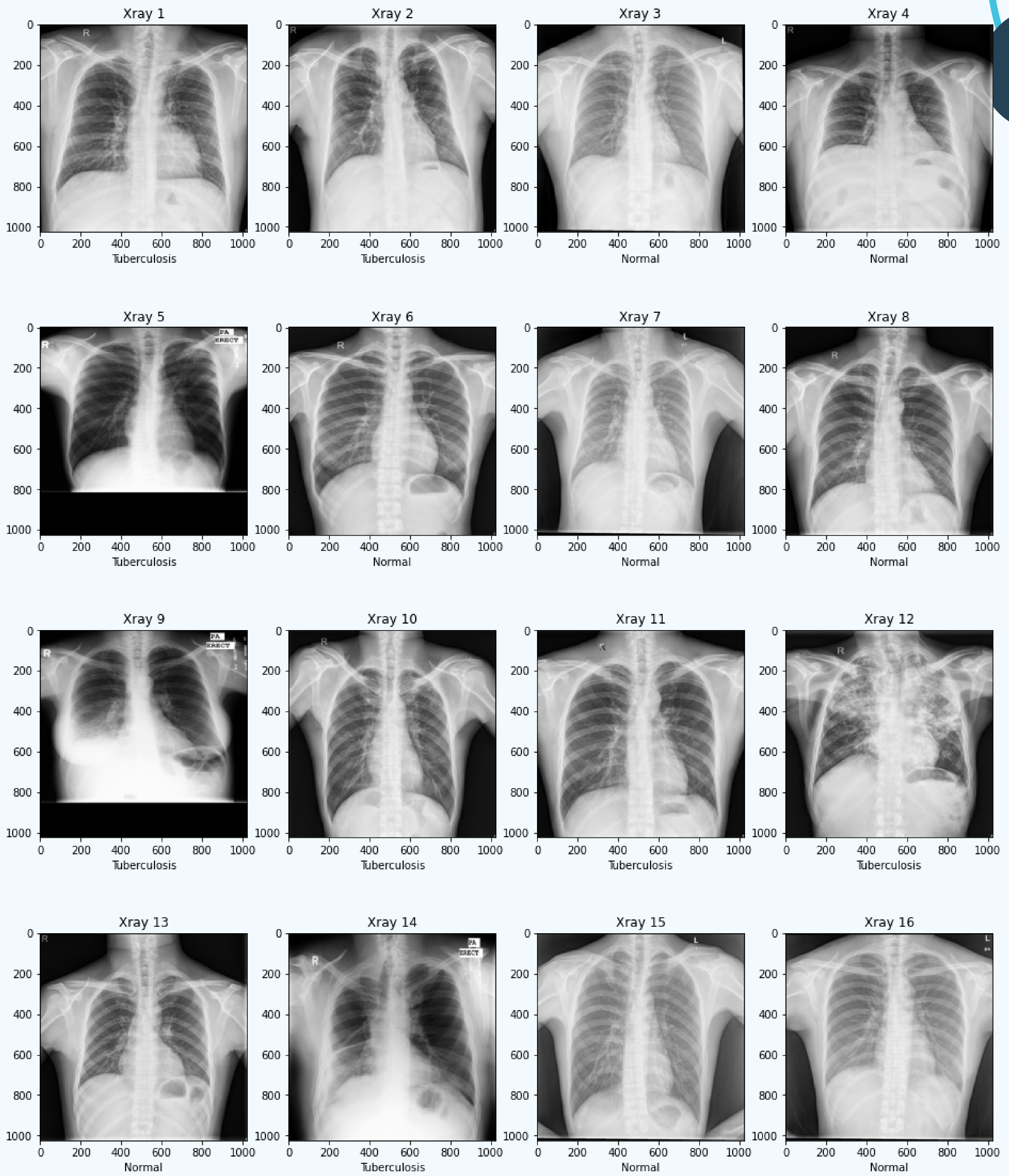
Dataset is taken from **Kaggle,** with the title **"Pulmonary Chest X-Ray Abnormalities",** with 800 images

SPLITTING DATASET



The dataset is split into **80:20** for training and testing, then the Training Data is split again into **80:20** for training and validation

SAMPLING IMAGES

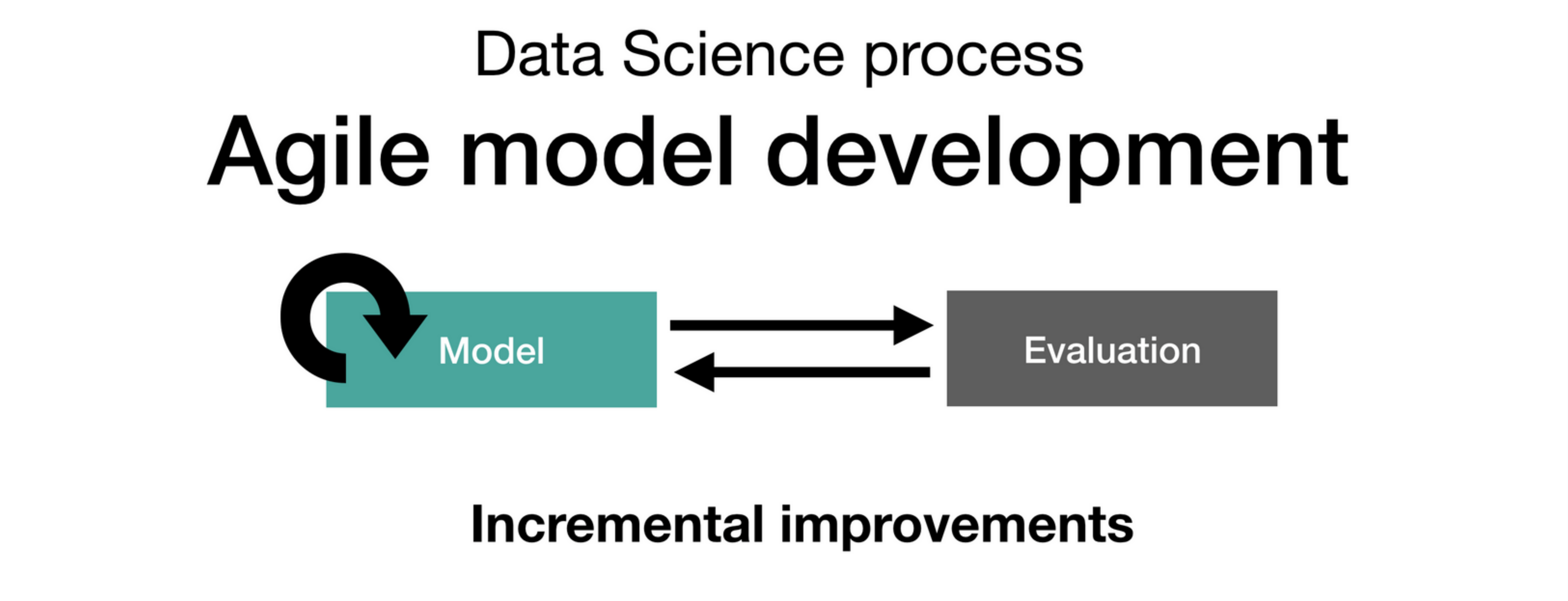


DATA DISTRIBUTION

```
Train Value Counts
0    260
1    252
Name: Label, dtype: int64
Validation DF Value Counts
0    65
1    63
Name: Label, dtype: int64
Test DF Value Counts
0    81
1    79
Name: Label, dtype: int64
```

The dataset seems to be **well-distributed** between normal and abnormal lungs

BASELINE MODEL




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 10)      280
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 10)      910
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 218, 218, 10)      910
_________________________________________________________________
flatten (Flatten)            (None, 475240)            0
_________________________________________________________________
dense (Dense)                (None, 1)                 475241
=================================================================
Total params: 477,341
Trainable params: 477,341
Non-trainable params: 0
_________________________________________________________________
```

Making baseline value is important, so that we can use it as an measurement to improve and get our best model. In this case, we created a baseline model consisting of **3 Convolutional Layers**

AUGMENTING DATA

```
Epoch 14/20
16/16 [==============================] - 67s 4s/step - loss: 0.0388 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - auc: 1.0000
- val_loss: 0.4088 - val_accuracy: 0.8594 - val_precision: 0.8689 - val_recall: 0.8413 - val_auc: 0.9330
val / train : 10.63
```

From the result, the baseline model got overfit towards the training data. Hence, we decided to augment the training data to prevent overfitting
Some other reasons may include:
- The size of each images are different, the look of the images may look different if it's reshaped
- Augmenting with Zooming or shifting will help in looking detailed futures.

AUGMENTING DATA

We decided to augment the data by:

zoom_range = 0.05 (Zooming the image by 5%)
width_shift_range = 0.05 (Shifting the image's width by 5%)
height_shift_range = 0.05, (Shifting the image's height by 5%)

AUGMENTING DATA

Result

```
Epoch 20/20
16/16 [==============================] - 71s 5s/step - loss: 0.5037 - accuracy: 0.7403 - precision: 0.7248 - recall: 0.8048 - auc: 0.8383
- val_loss: 0.4786 - val_accuracy: 0.7656 - val_precision: 0.7619 - val_recall: 0.7619 - val_auc: 0.8556
val / train : 0.88
```

After augmenting the training data, the model was able to reduce overfitting by 20%++

Choosing Pretrained Model

A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. You either use the pretrained model as is or use transfer learning to customize this model to a given task.

Using a pretrained model has been proven to reduce the time needed to construct a new architecture, and achieving promising results

PRETRAINED MODEL OPTIONS

RESNET V2

50 and 151 Version

R

EFFICIENTNET

B0 and B1 Version

e

DENSENET

DenseNet121 Version

D

MODEL SELECTION

Comparison Result

Model	Loss	Acc	Precision	Recall	AUC
Baseline	0.422	0.820	0.770	0.904	0.915
ResNet50 V2	0.381	0.859	0.8571	0.8571	0.916
ResNet 151 V2	0.399	0.804	0.806	0.793	0.901
EfficientNetB0	0.428	0.843	0.890	0.777	0.902
EfficientNetB1	0.411	0.820	0.822	0.809	0.912
DenseNet121	0.522	0.757	0.7105	0.8571	0.842

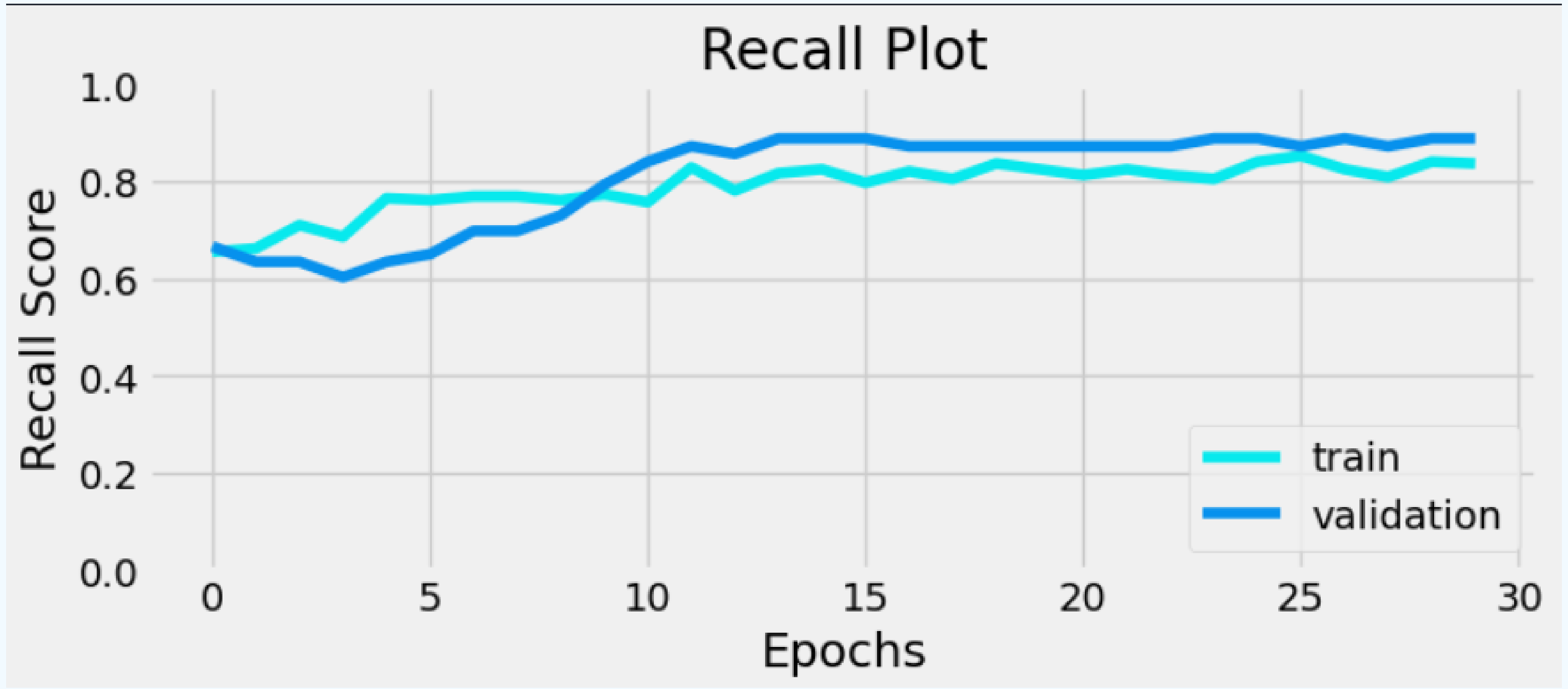
After doing evaluation for the Validation Data, the ResNet50 V2 Model outperformed all the metrics. Hence, This pretrained model will be used for further tuning and evaluation

Modelling and Evaluation

EXPERIMENT 1

ADDED BATCH NORMALIZATION AND DROPOUT LAYER

```
Epoch 30/30
16/16 [==============================] - 72s 5s/step - loss: 0.3245 - accuracy: 0.8601 - auc: 0.9384 - recall: 0.8740 - val_loss: 0.3322 - val_accuracy: 0.8594 - val_auc:
0.9317 - val_recall: 0.8889
val / train : 0.94
```

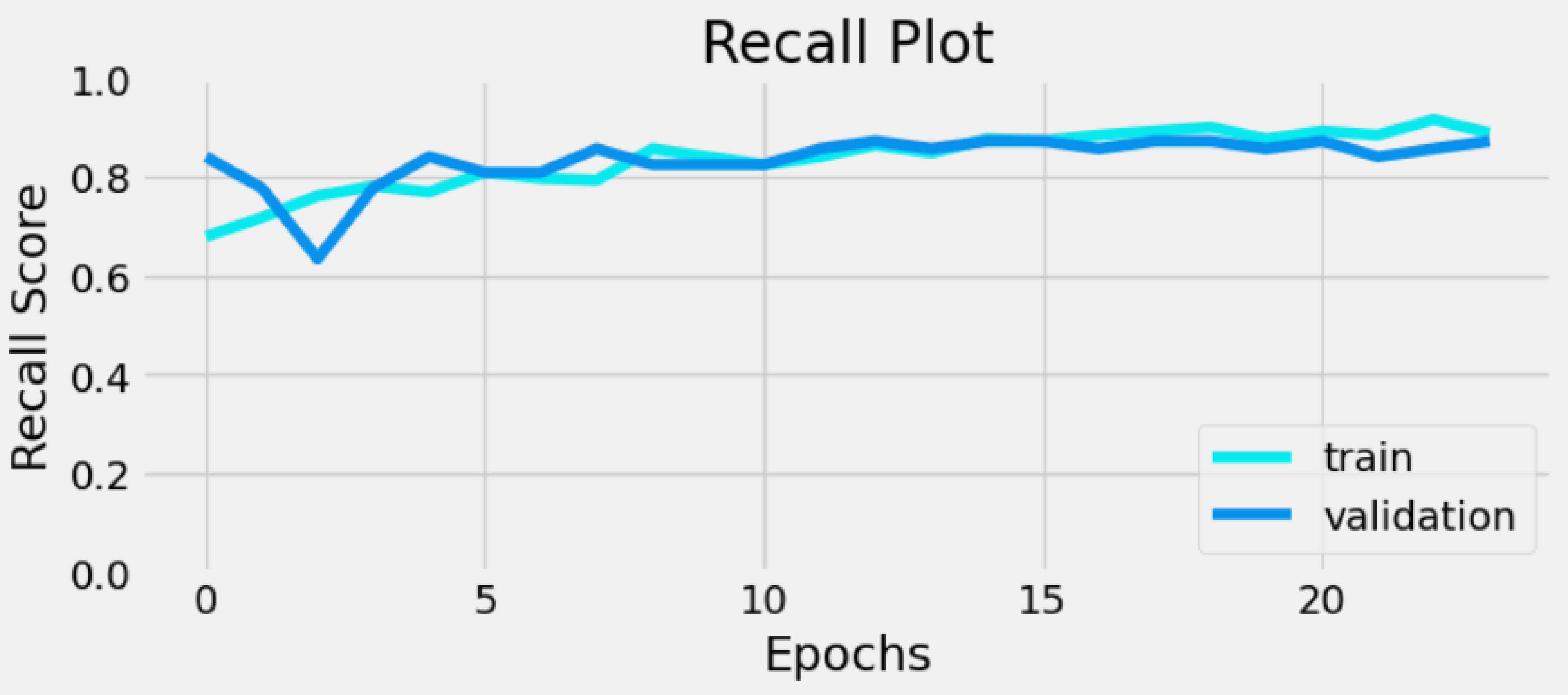


The model seems to have improved compared to the original architecture, and we will move on to the next experiment

EXPERIMENT 2

ADDED 3 DENSE LAYERS WITH X NEURONS

```
Epoch 24/30
16/16 [==============================] - 72s 5s/step - loss: 0.2799 - accuracy: 0.8803 - auc: 0.9561 - recall: 0.8777 - val_loss: 0.3802 - val_accuracy: 0.8203 - val_auc: 0.9228 - val_recall: 0.8730
val / train : 1.56
```



The model became more stable, and will be used for hyperparameter tuning.

HYPERPARAMETER TUNING

HyperBand Algorithm

Hyperparameter tuning is done to get a better performance from the model by looking for the best hyperparameters through some searching from the tuning algorithms. In this case, I used HyperBand Algorithm for the tuning.

The hyperparameters tuned are :

Number of neurons for each dense layer (units)

Dropout Rate

Activation Function (ReLU or eLU)

Learning Rate

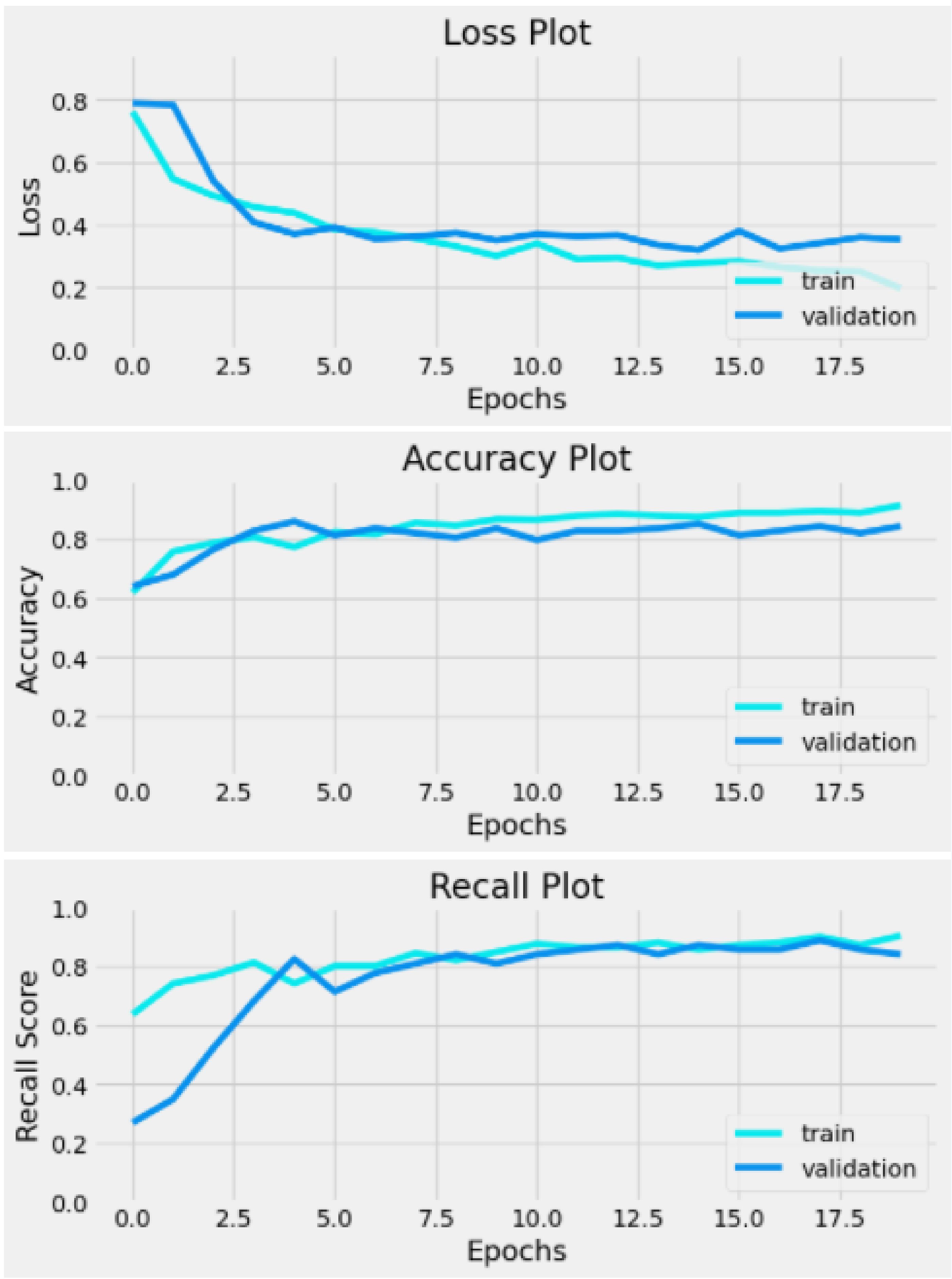
'UNITS': 160,
'DROPOUTRATE': 0.3,
'ACTIVATION': 'ELU',
'LEARNING_RATE': 0.0005,

RESULT

EVALUATION

After fitting with the tuned model, the result:

-The model is quite stable and does not overfit
-Has a lower Loss
- Better Recall



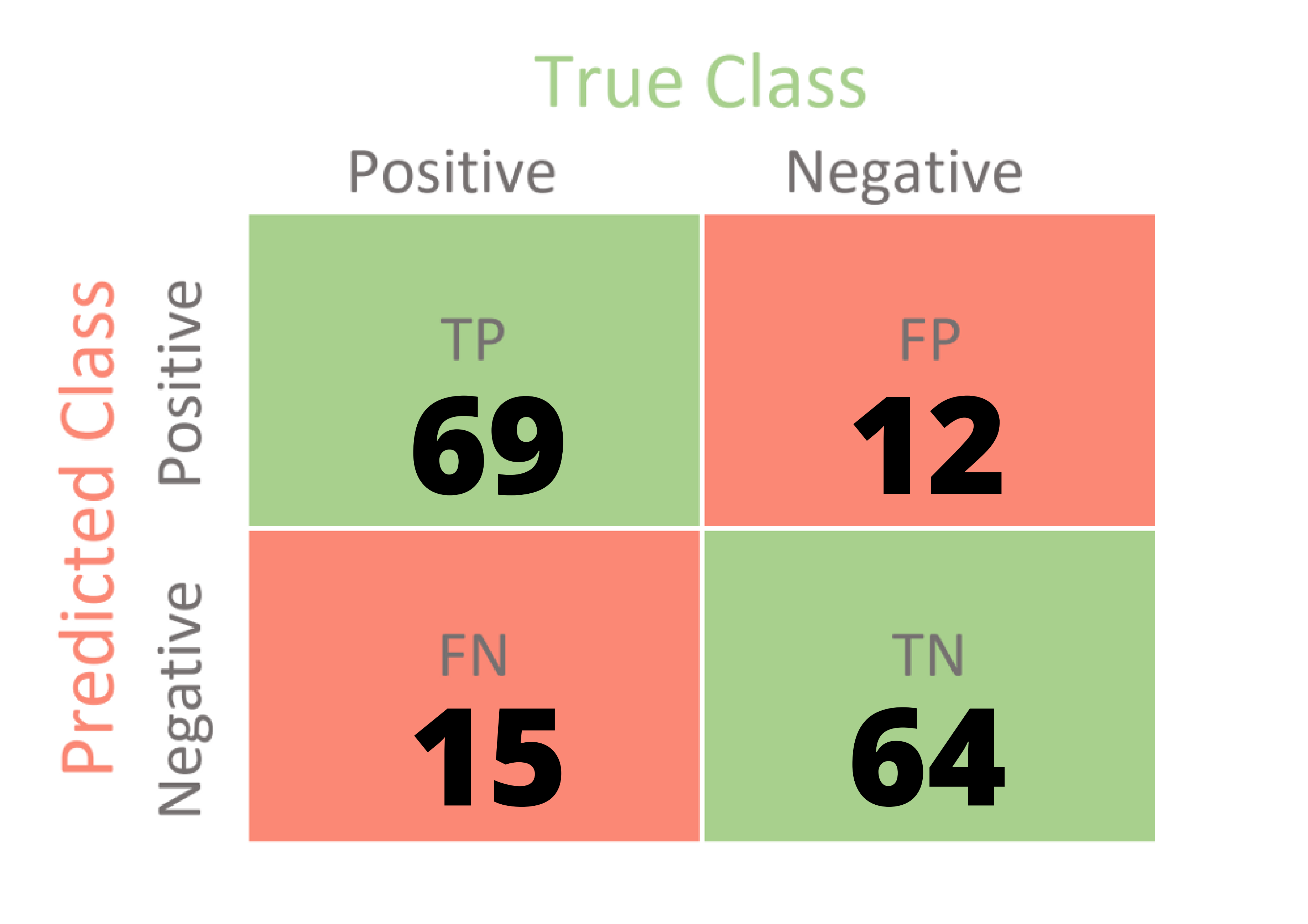
EVALUATION

Prediction on Unseen Dataset

Accuracy score : 0.83125

Recall score : 0.83125

	precision	recall	f1-score	support
0	0.82	0.85	0.84	81
1	0.84	0.81	0.83	79
accuracy			0.83	160
macro avg	0.83	0.83	0.83	160
weighted avg	0.83	0.83	0.83	160



Conclusion and Reccomendation

CONCLUSION

Predicting 128 images only took less than 10 seconds. We can speed up the Patient's Lung early diagnosis using Deep Learning model with an average recall of 83%

RECCOMENDATION

Some reccomendations for our model to improve are:

- Add more X-Ray Images
- More GPU Supply
- More time

Take into note that:

Our model have a simply 83% on Recall, so not all X-Ray Scans will be truely predicted.

CONTACT ME:

DARNELL KIKOO
darnellkikoo@gmail.com

TELEPON
+6281218222211

COMPUTER SCIENCE
Binus University, B2024

THANK YOU