Configuration Manual

MSc Research Project
Data Analytics

Tushar Shailesh Dalvi
Student ID: x18134301

School of Computing
National College of Ireland

Supervisor: Mr. Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Tushar Shailesh Dalvi
Student ID:	x18134301
Programme:	MSc. Data Analytics
Year:	2019-20
Module:	Configuration Manual
Supervisor:	Mr. Noel Cosgrave
Submission Due Date:	12/12/2019
Project Title:	Classification of Pneumonia from chest X-rays using Transfer Learning Analysis
Word Count:	807
Page Count:	13

Signature:	
Date:	11th December 2019

Configuration Manual

Tushar Shailesh Dalvi
x18134301

11th December 2019

1 Introduction

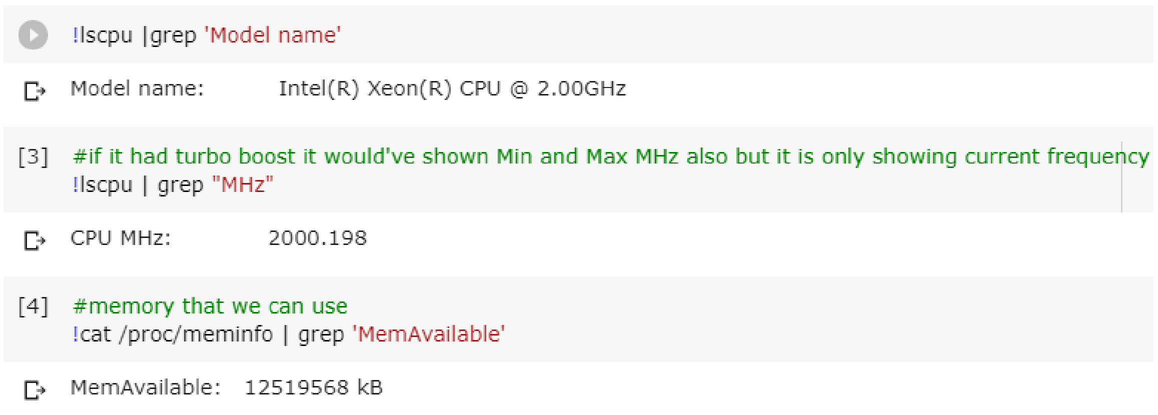
This document is a guide on how the project is implemented. Code which is used for classification of pneumonia from a chest x-ray is mentioned in this document. With the help of various libraries like numpy, keras, matplotlib completed this research, at which step which library is used and how that library used to get the expected output is mentioned in this document.

2 System Specification

This research is carried out on the Python-based environment which is provided by google known as Google Collaboratory. Google Collaboratory is cloud base Python jupyter environment which provides ready to use Jupyter Notebook. System specification for google colab is shown below:.

Google Colab Specification:

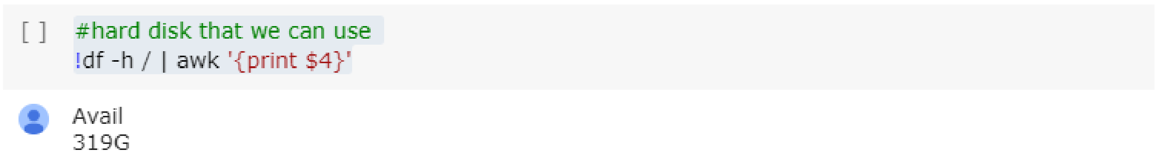
- **CPU Name, CPU Frequency, and Memory:** Intel Xenon(R) based CPU with 2.0 GHz clock speed, with 13 GB Memory



```
!lscpu |grep 'Model name'

Model name:          Intel(R) Xeon(R) CPU @ 2.00GHz

[3] #if it had turbo boost it would've shown Min and Max MHz also but it is only showing current frequency
    !lscpu | grep "MHz"

CPU MHz:             2000.198

[4] #memory that we can use
    !cat /proc/meminfo | grep 'MemAvailable'

MemAvailable:    12519568 kB
```

Figure 1: Google Colab specification

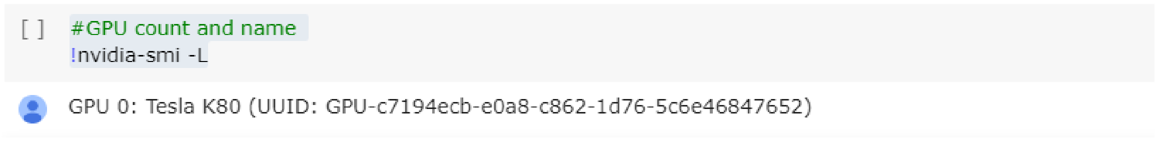
- **Storage Space:** Google Colab providing 319 GB online storage data



```
[ ]  #hard disk that we can use
     !df -h / | awk '{print $4}'

     Avail
     319G
```

Figure 2: Storage Space

- **GPU Specification:** Tesla K80 GPU with 12 GB GDDR5 Memory is provided for More Computational power



```
[ ]  #GPU count and name
     !nvidia-smi -L

     GPU 0: Tesla K80 (UUID: GPU-c7194ecb-e0a8-c862-1d76-5c6e46847652)
```

Figure 3: GPU Specification

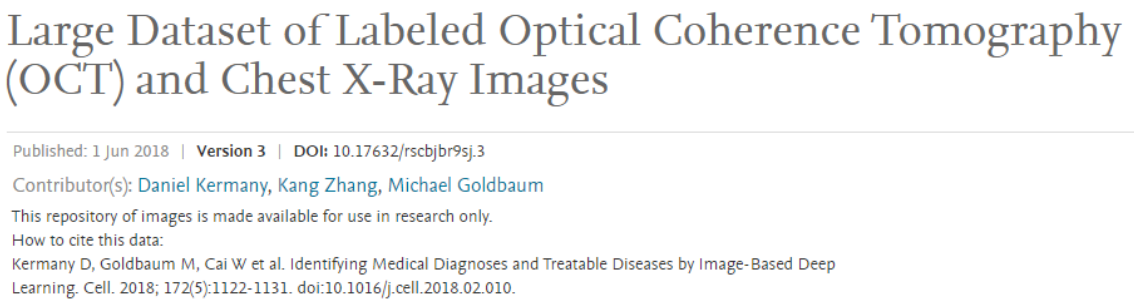
3 Software

To carry out this research used a Python-based environment, which is provided by Google colab

- **Python Version:** Python 3

4 Data Gathering

The data is collected to carry out pneumonia classification is from the Mendeley website [1] , which contains chest X-ray images with both normal and Pneumonia infected class, and this data is freely available for research.



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

Published: 1 Jun 2018 | **Version 3** | **DOI:** 10.17632/rscbjbr9sj.3

Contributor(s): Daniel Kermany, Kang Zhang, Michael Goldbaum

This repository of images is made available for use in research only.
How to cite this data:
Kermany D, Goldbaum M, Cai W et al. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. Cell. 2018; 172(5):1122-1131. doi:10.1016/j.cell.2018.02.010.

Figure 4: Data Collection

[1] https://data.mendeley.com/datasets/rscbjbr9sj/3

5 Data Storage

The data which are downloaded from Mendeley is in image format. it is downloaded to the local system and then uploaded to Google Drive. Below are the steps followed to upload the data in Google Drive and how to start Colab Notebook.

- Login to google Dive using google Email Id.
- Click on new, then select Folder upload.
- Then select the folder from the local drive and click on upload.
- To start colab go to New → File → More → Google Colab.

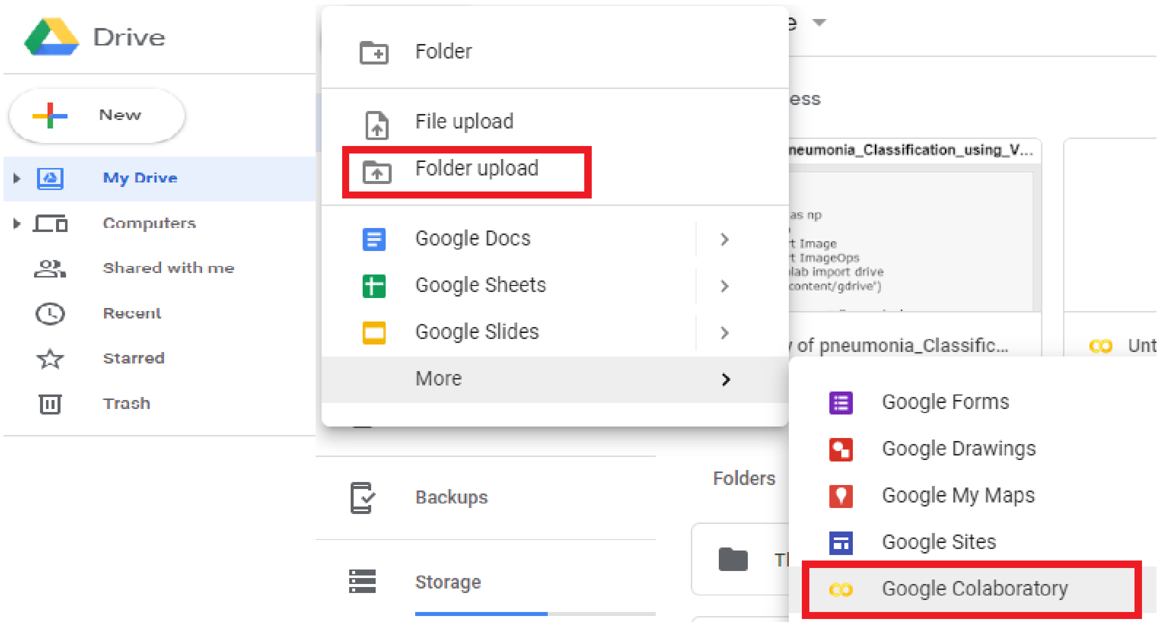


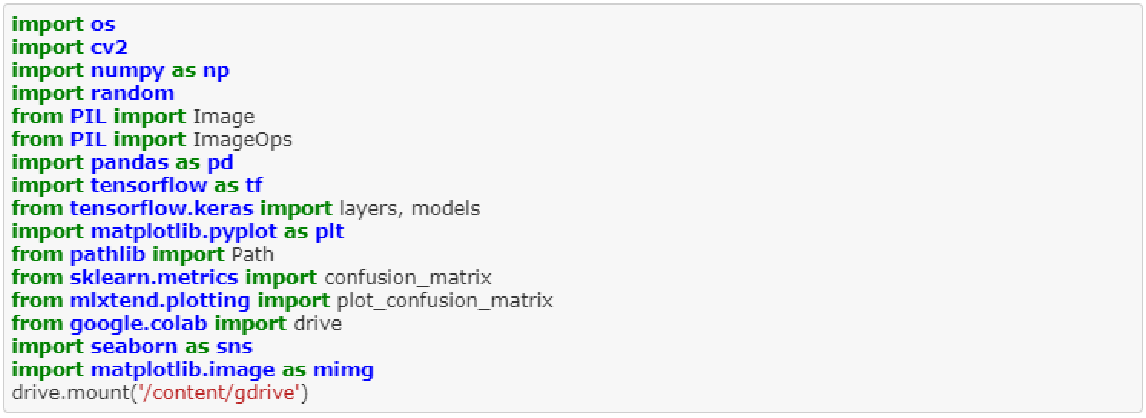
Figure 5: Uploading file to drive and opening colab

6 Implementation

The entire project is uploaded on the GitHub repository [2]. To start Research, need to load useful libraries, throughout Research below Libraries are used.
Required Libraries with Version

- Numpy: 1.17.4
- tensorflow: 1.15.0
- matplotlib: 3.1.2
- mlxtend: 0.14.0
- seaborn: 0.9.0

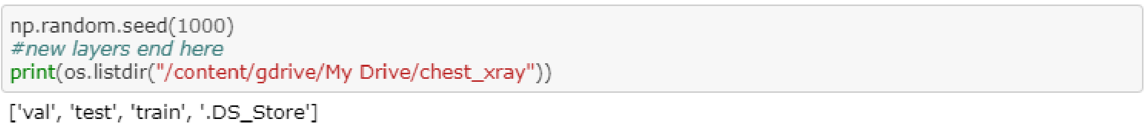
[2]https://github.com/x18134301/x18134301_TusharDalvi



```
import os
import cv2
import numpy as np
import random
from PIL import Image
from PIL import ImageOps
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from google.colab import drive
import seaborn as sns
import matplotlib.image as mimg
drive.mount('/content/gdrive')
```

Figure 6: List of Libraries

- **Setting Data Path and setting Seed:** setting seed to 1000 to produce the same output every-time. Function os.listdir is used to confirm that we are using the same path where data is located.
  Library Used: os

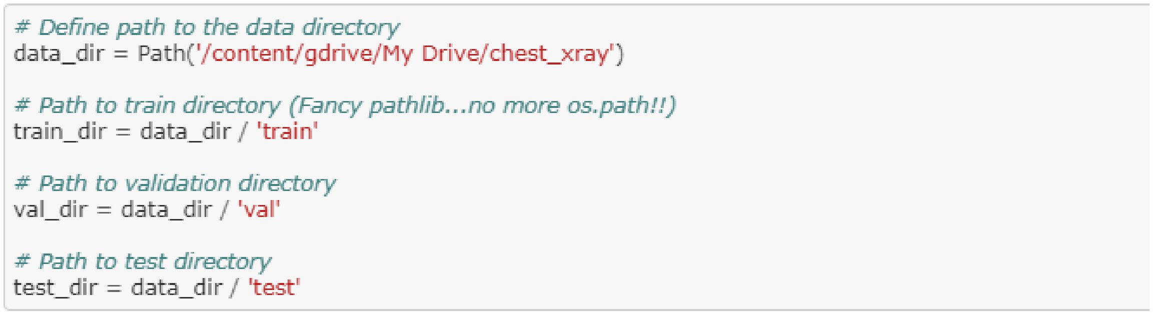


```
np.random.seed(1000)
#new layers end here
print(os.listdir("/content/gdrive/My Drive/chest_xray"))
```

```
['val', 'test', 'train', '.DS_Store']
```

Figure 7: Setting Random Seed and Path

- Setting data directory Created data directory for training, testing and validation datasets
  Library used: Pathlib as Path

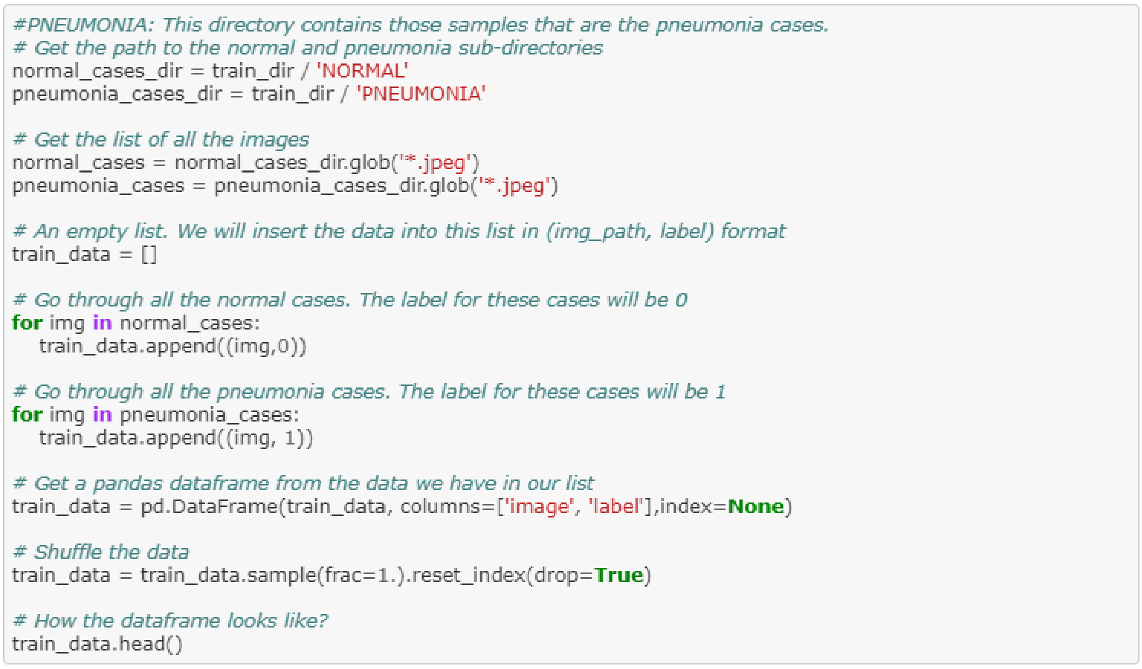


```
# Define path to the data directory
data_dir = Path('/content/gdrive/My Drive/chest_xray')

# Path to train directory (Fancy pathlib...no more os.path!!)
train_dir = data_dir / 'train'

# Path to validation directory
val_dir = data_dir / 'val'

# Path to test directory
test_dir = data_dir / 'test'
```

Figure 8: Training Testing and Validation Data Path

7 Exploring Data

- Initially every image from the training directory inserted into the list with the respected class to identify the class imbalance.

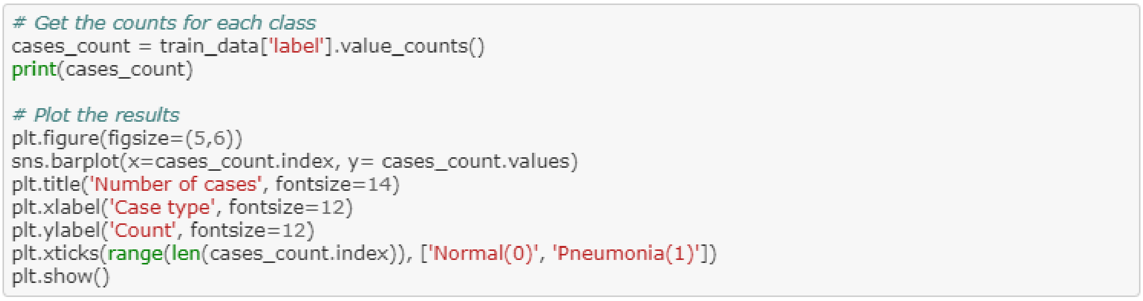


```
#PNEUMONIA: This directory contains those samples that are the pneumonia cases.
# Get the path to the normal and pneumonia sub-directories
normal_cases_dir = train_dir / 'NORMAL'
pneumonia_cases_dir = train_dir / 'PNEUMONIA'

# Get the list of all the images
normal_cases = normal_cases_dir.glob('*.jpeg')
pneumonia_cases = pneumonia_cases_dir.glob('*.jpeg')

# An empty list. We will insert the data into this list in (img_path, label) format
train_data = []

# Go through all the normal cases. The label for these cases will be 0
for img in normal_cases:
    train_data.append((img,0))

# Go through all the pneumonia cases. The label for these cases will be 1
for img in pneumonia_cases:
    train_data.append((img, 1))

# Get a pandas dataframe from the data we have in our list
train_data = pd.DataFrame(train_data, columns=['image', 'label'],index=None)

# Shuffle the data
train_data = train_data.sample(frac=1.).reset_index(drop=True)

# How the dataframe looks like?
train_data.head()
```

Figure 9: combining training images with class

- Plotting Training Dataset Graph

  Library Used: matplotlib.pyplot as plt, seaborn as sns



```
# Get the counts for each class
cases_count = train_data['label'].value_counts()
print(cases_count)

# Plot the results
plt.figure(figsize=(5,6))
sns.barplot(x=cases_count.index, y= cases_count.values)
plt.title('Number of cases', fontsize=14)
plt.xlabel('Case type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(range(len(cases_count.index)), ['Normal(0)', 'Pneumonia(1)'])
plt.show()
```

Figure 10: Plotting Trainning Graph

- Plotting random 3 images from each class from Training Data.
  Library Used: matplotlib.pyplot as plt, skimage.io, skimage.transform

```python
from skimage.io import imread
from skimage.transform import resize


# Get few samples for both the classes
pneumonia_samples = (train_data[train_data['label']==1]['image'].iloc[:5]).tolist()
normal_samples = (train_data[train_data['label']==0]['image'].iloc[:5]).tolist()

# Concat the data in a single list and del the above two list
samples = pneumonia_samples + normal_samples
del pneumonia_samples, normal_samples

# Plot the data
f, ax = plt.subplots(2,3, figsize=(20,8))
for i in range(6):
    img = imread(samples[i])
    ax[i//3, i%3].imshow(img, cmap='gray')
    if i<3:
        ax[i//3, i%3].set_title("Pneumonia")
    else:
        ax[i//3, i%3].set_title("Normal")
    ax[i//3, i%3].axis('off')
    ax[i//3, i%3].set_aspect('auto')
plt.show()
```

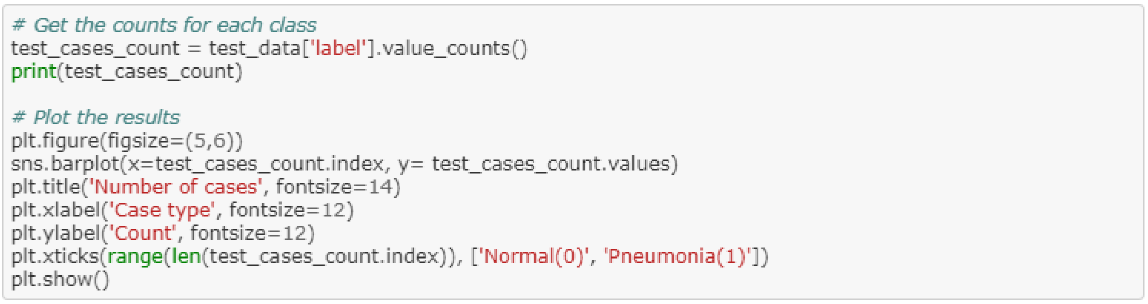
Figure 11: Plotting images from each class

- In this section, testing images from each class are Checked, and combining images from each class from the testing directory is done.

```python
#PNEUMONIA: This directory contains those samples that are the pneumonia cases.
# Get the path to the normal and pneumonia sub-directories
normal_test_cases_dir = test_dir / 'NORMAL'
pneumonia_test_cases_dir = test_dir / 'PNEUMONIA'

# Get the list of all the images
test_normal_cases = normal_test_cases_dir.glob('*.jpeg')
test_pneumonia_cases = pneumonia_test_cases_dir.glob('*.jpeg')

# An empty list. We will insert the data into this list in (img_path, label) format
test_data = []

# Go through all the normal cases. The label for these cases will be 0
for img in test_normal_cases:
    test_data.append((img,0))

# Go through all the pneumonia cases. The label for these cases will be 1
for img in test_pneumonia_cases:
    test_data.append((img, 1))

# Get a pandas dataframe from the data we have in our list
test_data = pd.DataFrame(test_data, columns=['image', 'label'],index=None)

# Shuffle the data
test_data = test_data.sample(frac=1.).reset_index(drop=True)

# How the dataframe looks like?
test_data.head()
```

Figure 12: combining testing images with class

- Plotting Testing Dataset Graph
  Library Used: matplotlib.pyplot as plt, seaborn as sns

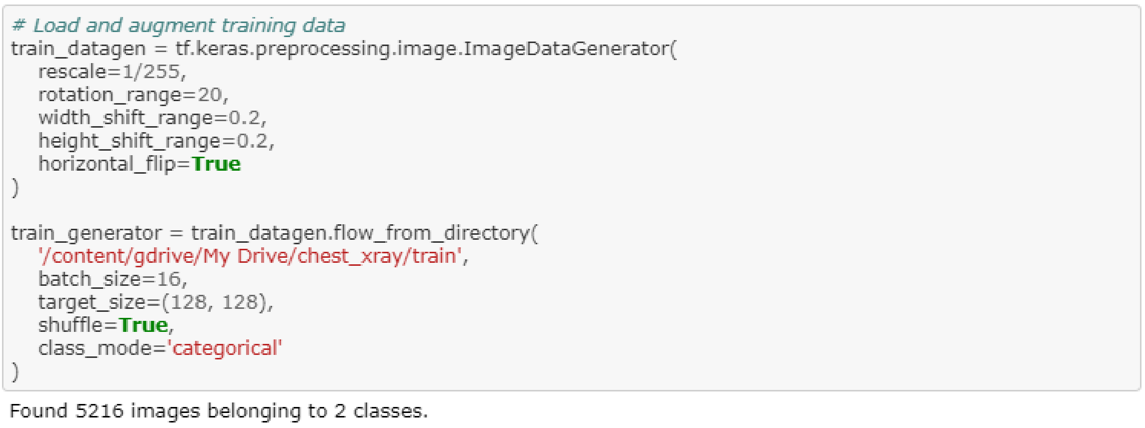


```
# Get the counts for each class
test_cases_count = test_data['label'].value_counts()
print(test_cases_count)

# Plot the results
plt.figure(figsize=(5,6))
sns.barplot(x=test_cases_count.index, y= test_cases_count.values)
plt.title('Number of cases', fontsize=14)
plt.xlabel('Case type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(range(len(test_cases_count.index)), ['Normal(0)', 'Pneumonia(1)'])
plt.show()
```

Figure 13: Plotting testing image count graph

8 Data Preparation

As we saw from previous graphs, training data have a class imbalance. Thus, the Data Augmentation technique is used to removed class imbalance. In which the ImageDataGenerator function is used [3].



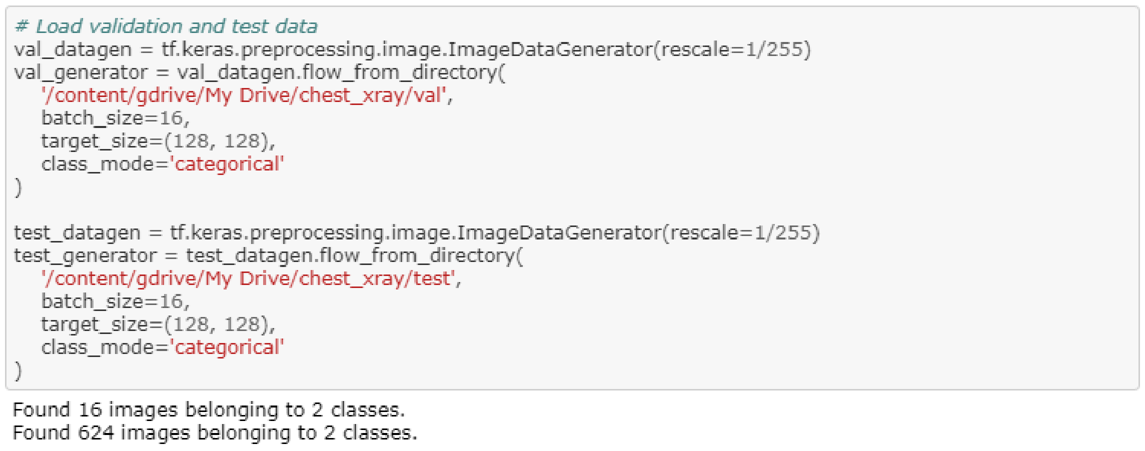
```
# Load and augment training data
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1/255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    '/content/gdrive/My Drive/chest_xray/train',
    batch_size=16,
    target_size=(128, 128),
    shuffle=True,
    class_mode='categorical'
)
```

```
Found 5216 images belonging to 2 classes.
```

Figure 14: DAta Augmentation

- Preparation of testing data and validation data

[3]https://keras.io/preprocessing/image/



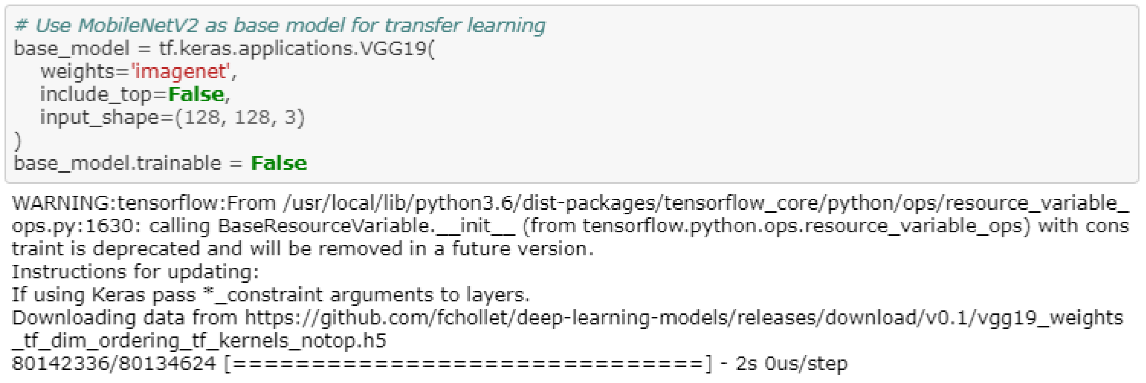
```
# Load validation and test data
val_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
val_generator = val_datagen.flow_from_directory(
    '/content/gdrive/My Drive/chest_xray/val',
    batch_size=16,
    target_size=(128, 128),
    class_mode='categorical'
)

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
test_generator = test_datagen.flow_from_directory(
    '/content/gdrive/My Drive/chest_xray/test',
    batch_size=16,
    target_size=(128, 128),
    class_mode='categorical'
)
```

```
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

Figure 15: Testing and Validation data Preparation

9 Model Building

- In this section, the model building code is shown in which initial layers are removed. **Library used:** tensorflow as tf

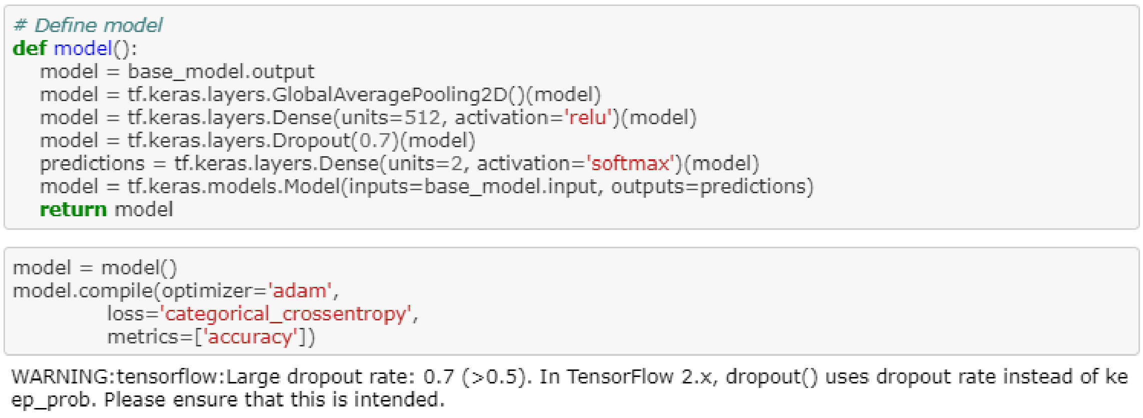


```
# Use MobileNetV2 as base model for transfer learning
base_model = tf.keras.applications.VGG19(
    weights='imagenet',
    include_top=False,
    input_shape=(128, 128, 3)
)
base_model.trainable = False
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/resource_variable_
ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with cons
traint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights
_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 2s 0us/step
```

Figure 16: Using base Model as VGG19

- In this subsection adding extra dense layers and compiling model are shown[4]
  Used library: tensorflow as tf



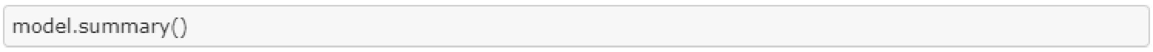
```
# Define model
def model():
    model = base_model.output
    model = tf.keras.layers.GlobalAveragePooling2D()(model)
    model = tf.keras.layers.Dense(units=512, activation='relu')(model)
    model = tf.keras.layers.Dropout(0.7)(model)
    predictions = tf.keras.layers.Dense(units=2, activation='softmax')(model)
    model = tf.keras.models.Model(inputs=base_model.input, outputs=predictions)
    return model
```

```
model = model()
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead of ke
ep_prob. Please ensure that this is intended.
```

Figure 17: Adding Extra Dense Layers

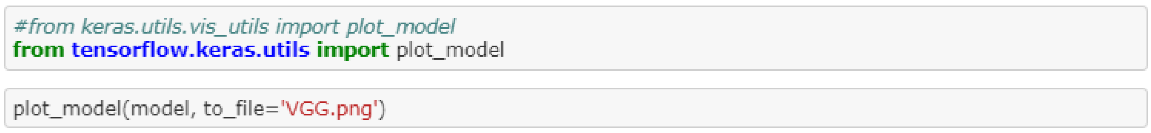
- Model Summary showed using the function.
  Function Name: model.summary()



```
model.summary()
```

Figure 18: checking Model Summary

- Using Keras model layers are plotted for review
  Library Used: from tensorflow.keras.utils import plot_model



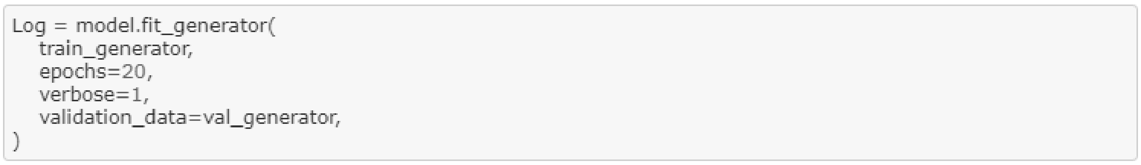
```
#from keras.utils.vis_utils import plot_model
from tensorflow.keras.utils import plot_model
```

```
plot_model(model, to_file='VGG.png')
```

Figure 19: Plotting Model

- In this subsection using training data training started. The total number of epochs set to 20 and validating model with validation data.
  Function used: model.fit_generator()

[4]https://keras.io/getting-started/sequential-model-guide/

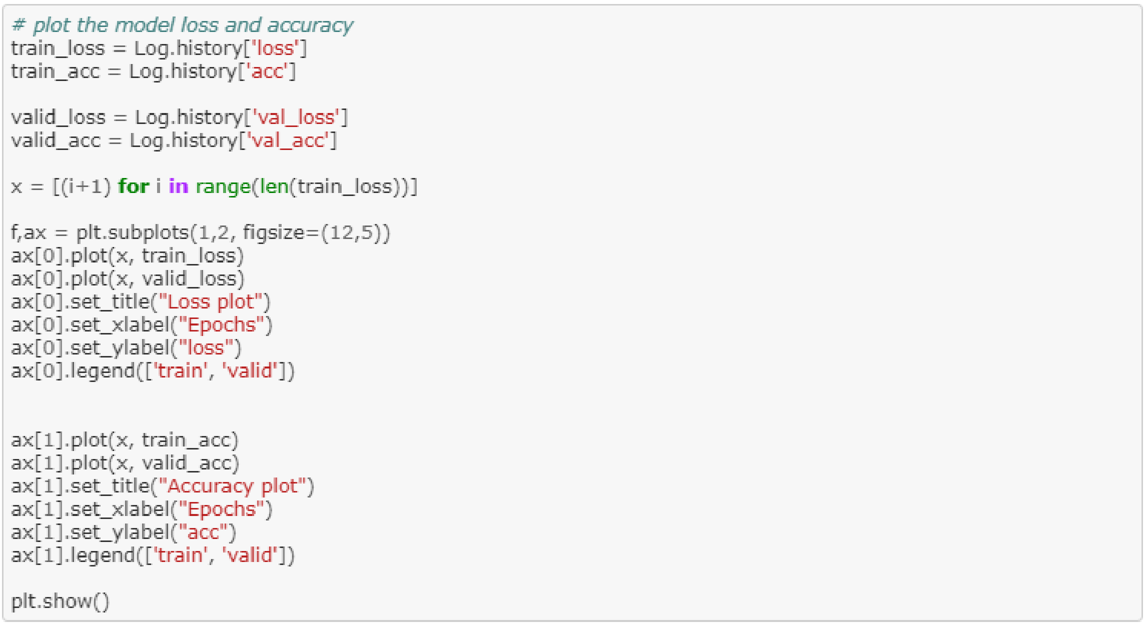


```
Log = model.fit_generator(
    train_generator,
    epochs=20,
    verbose=1,
    validation_data=val_generator,
)
```

Figure 20: Model Training

10 model Evaluation

- To check how the model performed while training Accuracy and Loss plot are plotted[5] .
  Library Used: matplotlib.pyplot as plt

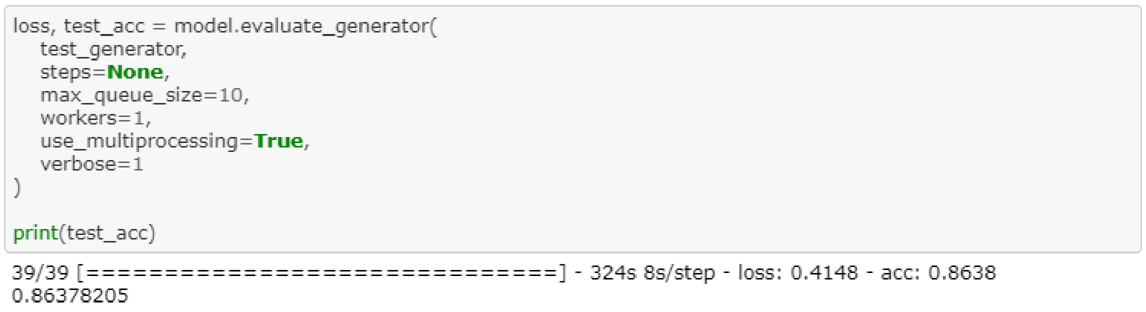


```
# plot the model loss and accuracy
train_loss = Log.history['loss']
train_acc = Log.history['acc']

valid_loss = Log.history['val_loss']
valid_acc = Log.history['val_acc']

x = [(i+1) for i in range(len(train_loss))]

f,ax = plt.subplots(1,2, figsize=(12,5))
ax[0].plot(x, train_loss)
ax[0].plot(x, valid_loss)
ax[0].set_title("Loss plot")
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("loss")
ax[0].legend(['train', 'valid'])


ax[1].plot(x, train_acc)
ax[1].plot(x, valid_acc)
ax[1].set_title("Accuracy plot")
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("acc")
ax[1].legend(['train', 'valid'])

plt.show()
```

Figure 21: Loss and Accuracy plot

- Evaluating model on the testing dataset to check test accuracy.
  Function used: model.evaluate_generator()

[5]https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html

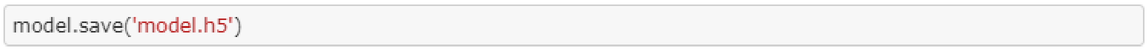


```
loss, test_acc = model.evaluate_generator(
    test_generator,
    steps=None,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=True,
    verbose=1
)

print(test_acc)
```

```
39/39 [==============================] - 324s 8s/step - loss: 0.4148 - acc: 0.8638
0.86378205
```

Figure 22: model evaluation

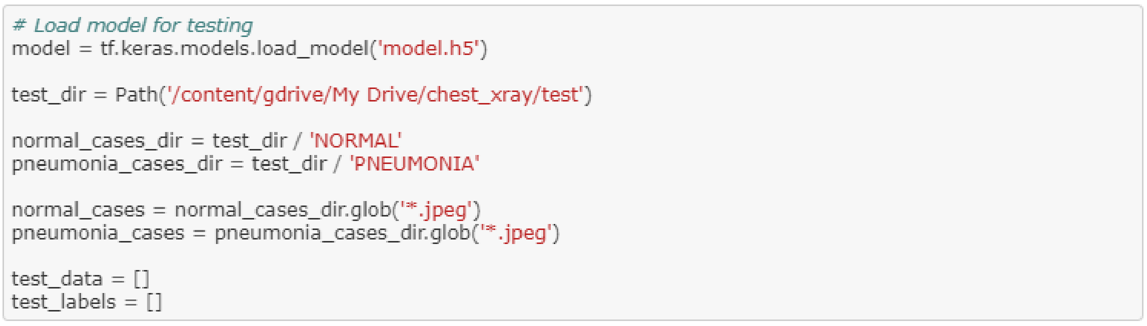
- Saving the model for future use.



```
model.save('model.h5')
```

Figure 23: Saving Model

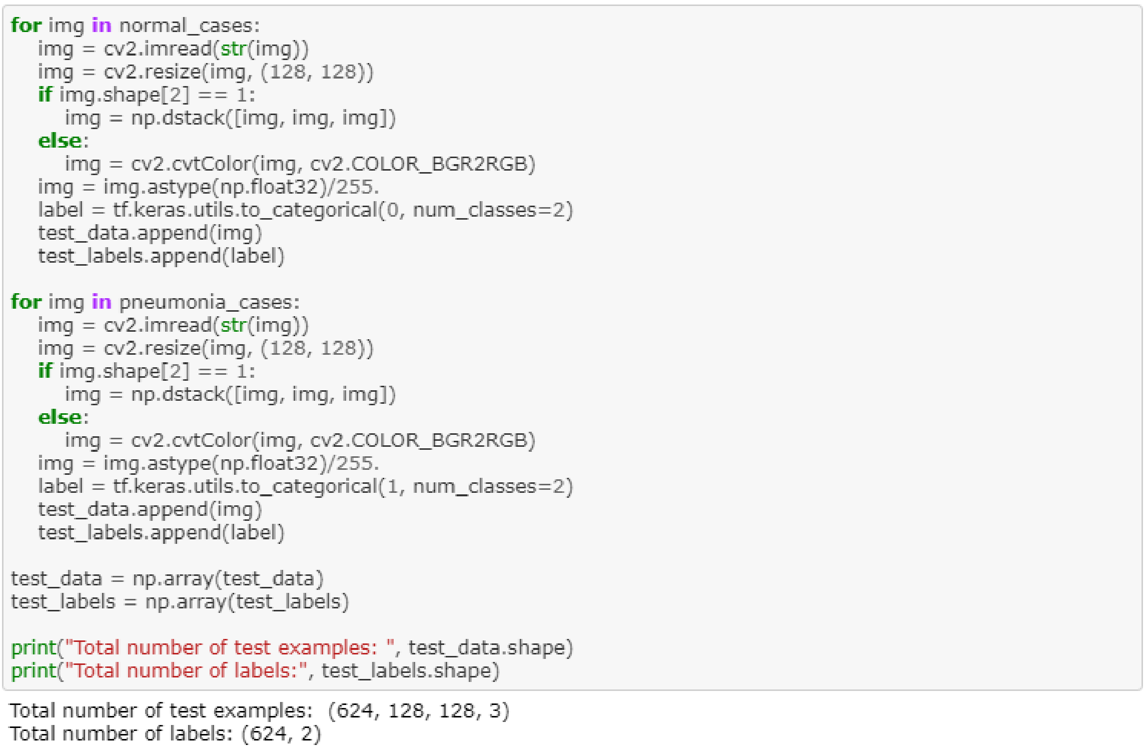
- Loading model that is saved earlier and loading normal images and pneumonia images for the testing model.



```
# Load model for testing
model = tf.keras.models.load_model('model.h5')

test_dir = Path('/content/gdrive/My Drive/chest_xray/test')

normal_cases_dir = test_dir / 'NORMAL'
pneumonia_cases_dir = test_dir / 'PNEUMONIA'

normal_cases = normal_cases_dir.glob('*.jpeg')
pneumonia_cases = pneumonia_cases_dir.glob('*.jpeg')

test_data = []
test_labels = []
```

Figure 24: Loading Model collecting testing data

- Loading normal cases and pneumonia cases in the test_data list and adding their labels in test_labels.
  Library Used: tensorflow as tf, numpy as np, cv2



```
for img in normal_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (128, 128))
    if img.shape[2] == 1:
        img = np.dstack([img, img, img])
    else:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = tf.keras.utils.to_categorical(0, num_classes=2)
    test_data.append(img)
    test_labels.append(label)

for img in pneumonia_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (128, 128))
    if img.shape[2] == 1:
        img = np.dstack([img, img, img])
    else:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = tf.keras.utils.to_categorical(1, num_classes=2)
    test_data.append(img)
    test_labels.append(label)

test_data = np.array(test_data)
test_labels = np.array(test_labels)

print("Total number of test examples: ", test_data.shape)
print("Total number of labels:", test_labels.shape)
```

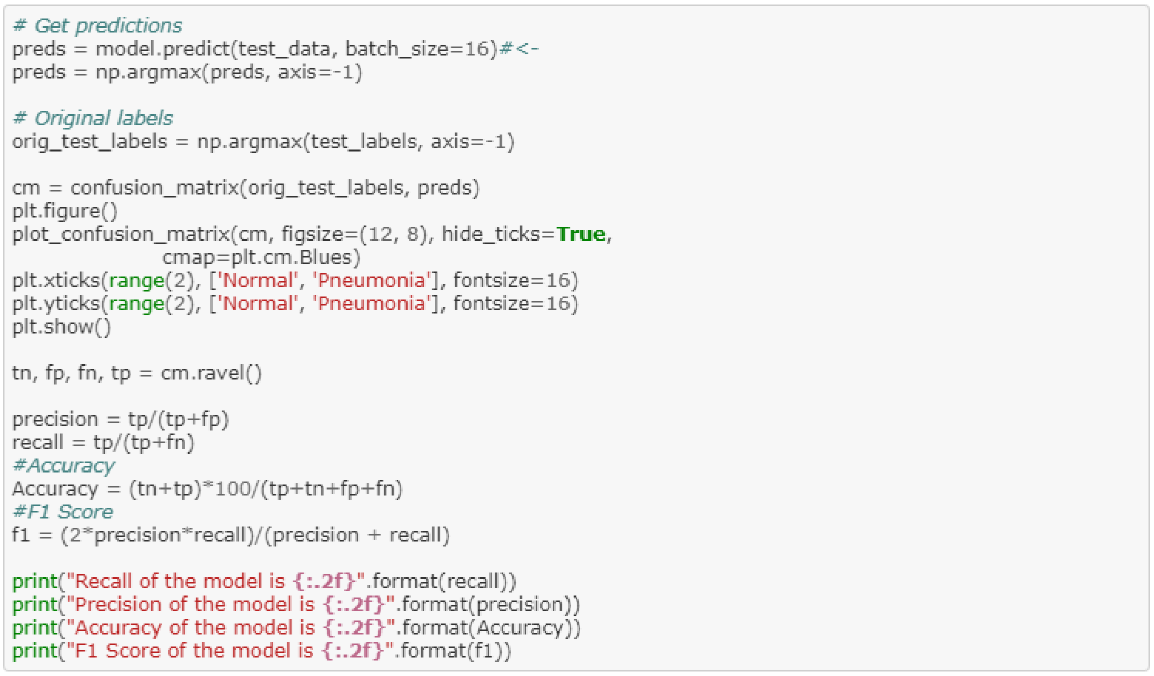
```
Total number of test examples:  (624, 128, 128, 3)
Total number of labels: (624, 2)
```

Figure 25: labeling images from testing dataset

11 Testing Model on Testing Dataset

- Using predict function testing data provided to model to test how much accurately model classifies images between pneumonia and Normal.
- Performing a comparison between predicted data and original data.
- Using confusion matrix measuring recall and precision, Accuracy and F1 Score[6]. **Library Used:** numpy as np, from mlxtend.plotting import plot_confusion_matrix, from sklearn.metrics import confusion matrix

[6]https://medium.com/hugoferreiras-blog/confusionmatrixandothermetricsinmachinelearning-894688cb1c0a



```
# Get predictions
preds = model.predict(test_data, batch_size=16)#<-
preds = np.argmax(preds, axis=-1)

# Original labels
orig_test_labels = np.argmax(test_labels, axis=-1)

cm = confusion_matrix(orig_test_labels, preds)
plt.figure()
plot_confusion_matrix(cm, figsize=(12, 8), hide_ticks=True,
                      cmap=plt.cm.Blues)
plt.xticks(range(2), ['Normal', 'Pneumonia'], fontsize=16)
plt.yticks(range(2), ['Normal', 'Pneumonia'], fontsize=16)
plt.show()

tn, fp, fn, tp = cm.ravel()

precision = tp/(tp+fp)
recall = tp/(tp+fn)
#Accuracy
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
#F1 Score
f1 = (2*precision*recall)/(precision + recall)

print("Recall of the model is {:.2f}".format(recall))
print("Precision of the model is {:.2f}".format(precision))
print("Accuracy of the model is {:.2f}".format(Accuracy))
print("F1 Score of the model is {:.2f}".format(f1))
```

Figure 26: Plotting Confusion matrix

12 conclusion

- Throughout this document step by step explained how coding part is implemented, also total code is shared on GitHub Repository and the GitHub link is shared in this document.

References