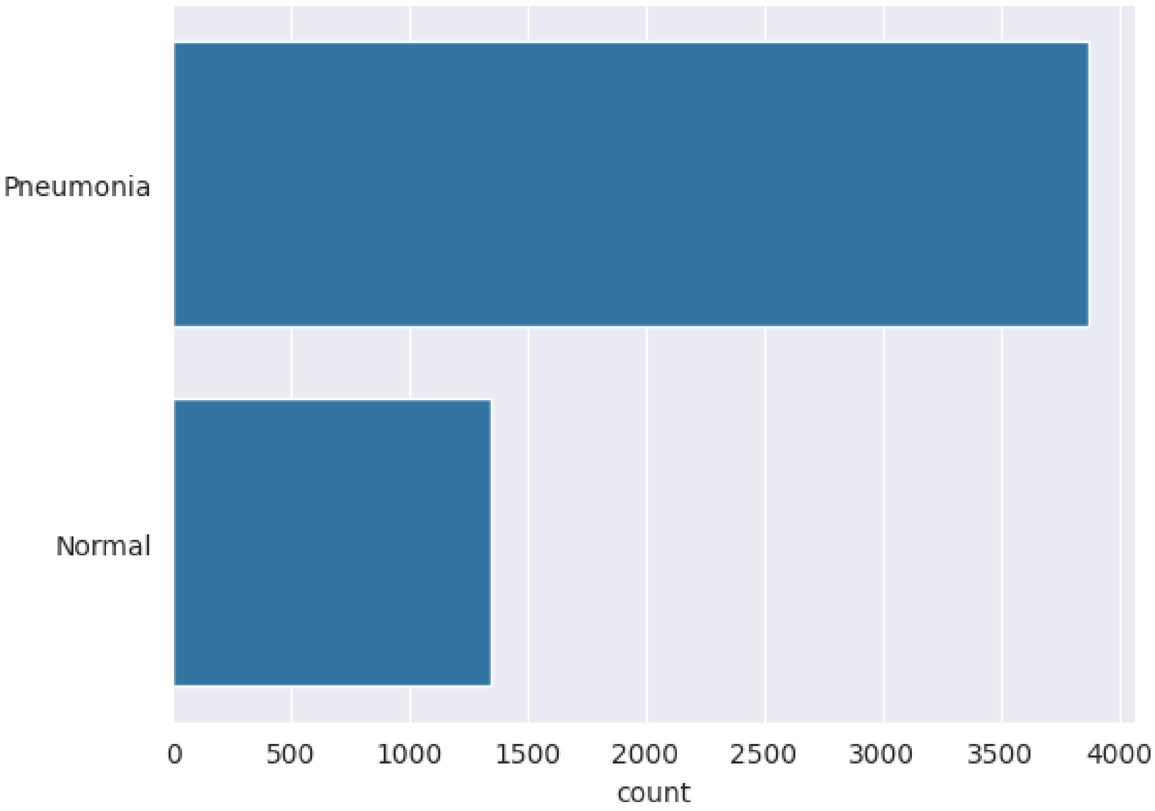
```python
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os

labels = ['PNEUMONIA', 'NORMAL']
img_size = 150
def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_path = os.path.join(path, img)
                img_arr = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

                # Check if image was successfully loaded
                if img_arr is None:
                    print(f"Warning: Could not read image {img_path}.
Skipping.")
                    continue # Skip to the next image

                resized_arr = cv2.resize(img_arr, (img_size,
img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(f"Error processing image {img_path}: {e}")
    return np.array(data, dtype=object)

import kagglehub
import os

# Download latest version and get the base path
path = kagglehub.dataset_download("paultimothymooney/chest-xray-
pneumonia")

print("Path to dataset files:", path)

# Construct the correct paths using the downloaded dataset's base path
train_dir = os.path.join(path, 'chest_xray', 'chest_xray', 'train')
test_dir = os.path.join(path, 'chest_xray', 'chest_xray', 'test')
val_dir = os.path.join(path, 'chest_xray', 'chest_xray', 'val')

train = get_training_data(train_dir)
test = get_training_data(test_dir)
val = get_training_data(val_dir)
```

```
Using Colab cache for faster access to the 'chest-xray-pneumonia'
dataset.
Path to dataset files: /kaggle/input/chest-xray-pneumonia
Warning: Could not read image
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/train/PNEUMON
IA/.DS_Store. Skipping.
Warning: Could not read image
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/train/NORMAL/
.DS_Store. Skipping.
Warning: Could not read image
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/val/PNEUMONIA
/.DS_Store. Skipping.
Warning: Could not read image
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/val/NORMAL/.D
S_Store. Skipping.
```

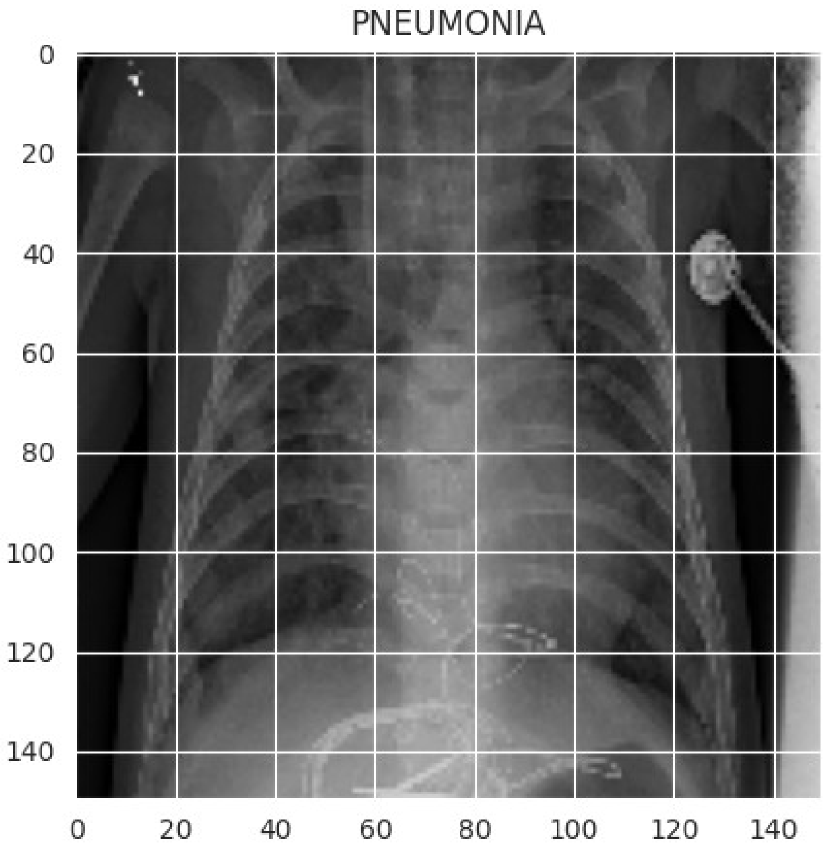
```
l = []
for i in train:
    if(i[1] == 0):
        l.append("Pneumonia")
    else:
        l.append("Normal")
sns.set_style('darkgrid')
sns.countplot(l)
```

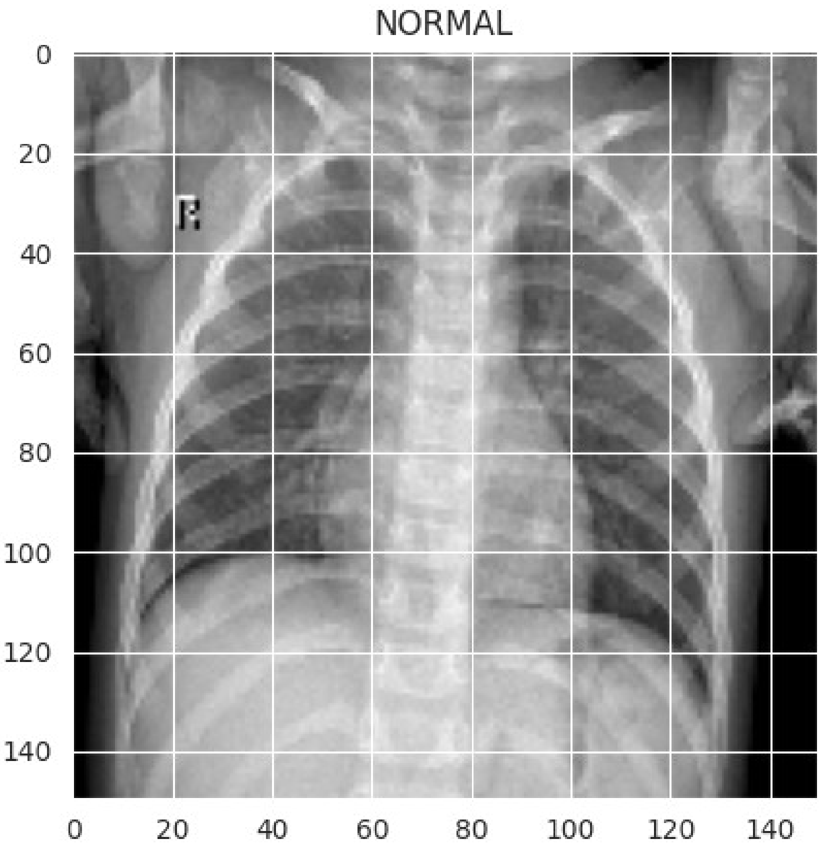
```
<Axes: xlabel='count'>
```



```
plt.figure(figsize = (5,5))
plt.imshow(train[0][0], cmap='gray')
plt.title(labels[train[0][1]])

plt.figure(figsize = (5,5))
plt.imshow(train[-1][0], cmap='gray')
plt.title(labels[train[-1][1]])
```

```
Text(0.5, 1.0, 'NORMAL')
```


PNEUMONIA
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140



```
x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
```

```
x_test = np.array(x_test) / 255

# resize data for deep learning
x_train = x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 1)
y_test = np.array(y_test)

# With data augmentation to prevent overfitting and handling the
imbalance in dataset

datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the
dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std
of the dataset
        samplewise_std_normalization=False,  # divide each input by
its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range
(degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally
(fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically
(fraction of total height)
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images


datagen.fit(x_train)

#Training the Model

model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' ,
activation = 'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' ,
activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' ,
activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' ,
activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' ,
activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' ,
metrics = ['accuracy'])
model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/
convolutional/base_conv.py:113: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ conv2d (Conv2D)                 │ (None, 150, 150, 32)   │
320 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ batch_normalization             │ (None, 150, 150, 32)   │
128 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 75, 75, 32)     │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
```

```
─────┤
│ conv2d_1 (Conv2D)               │ (None, 75, 75, 64)     │
18,496 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dropout (Dropout)               │ (None, 75, 75, 64)     │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ batch_normalization_1           │ (None, 75, 75, 64)     │
256 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 38, 38, 64)     │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ conv2d_2 (Conv2D)               │ (None, 38, 38, 64)     │
36,928 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ batch_normalization_2           │ (None, 38, 38, 64)     │
256 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 19, 19, 64)     │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ conv2d_3 (Conv2D)               │ (None, 19, 19, 128)    │
73,856 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dropout_1 (Dropout)             │ (None, 19, 19, 128)    │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ batch_normalization_3           │ (None, 19, 19, 128)    │
512 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ max_pooling2d_3 (MaxPooling2D)  │ (None, 10, 10, 128)    │
0 │
```

```
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ conv2d_4 (Conv2D)               │ (None, 10, 10, 256)    │
295,168 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dropout_2 (Dropout)             │ (None, 10, 10, 256)    │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ batch_normalization_4           │ (None, 10, 10, 256)    │
1,024 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ max_pooling2d_4 (MaxPooling2D)  │ (None, 5, 5, 256)      │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ flatten (Flatten)               │ (None, 6400)           │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense (Dense)                   │ (None, 128)            │
819,328 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dropout_3 (Dropout)             │ (None, 128)            │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_1 (Dense)                 │ (None, 1)              │
129 │
└─────────────────────────────────┴────────────────────────┴──────────
─────┘

 Total params: 1,246,401 (4.75 MB)

 Trainable params: 1,245,313 (4.75 MB)

 Non-trainable params: 1,088 (4.25 KB)

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
patience = 2, verbose=1,factor=0.3, min_lr=0.000001)

history = model.fit(datagen.flow(x_train,y_train, batch_size =
32) ,epochs = 12 , validation_data = datagen.flow(x_val,
y_val) ,callbacks = [learning_rate_reduction])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

Epoch 1/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 512s 3s/step - accuracy: 0.7783 - loss:
1.5024 - val_accuracy: 0.5000 - val_loss: 17.0006 - learning_rate:
0.0010
Epoch 2/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 477s 3s/step - accuracy: 0.8708 - loss:
0.3009 - val_accuracy: 0.5000 - val_loss: 34.3684 - learning_rate:
0.0010
Epoch 3/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - accuracy: 0.9116 - loss:
0.2360
Epoch 3: ReduceLROnPlateau reducing learning rate to
0.0003000000142492354.
163/163 ━━━━━━━━━━━━━━━━━━━━ 448s 3s/step - accuracy: 0.9116 - loss:
0.2359 - val_accuracy: 0.5000 - val_loss: 29.7506 - learning_rate:
0.0010
Epoch 4/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 446s 3s/step - accuracy: 0.9380 - loss:
0.1627 - val_accuracy: 0.5000 - val_loss: 30.1362 - learning_rate:
3.0000e-04
Epoch 5/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 459s 3s/step - accuracy: 0.9538 - loss:
0.1336 - val_accuracy: 0.7500 - val_loss: 0.4058 - learning_rate:
3.0000e-04
Epoch 6/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 459s 3s/step - accuracy: 0.9584 - loss:
0.1259 - val_accuracy: 0.5000 - val_loss: 1.5086 - learning_rate:
3.0000e-04
Epoch 7/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - accuracy: 0.9540 - loss:
0.1214
Epoch 7: ReduceLROnPlateau reducing learning rate to
9.000000427477062e-05.
163/163 ━━━━━━━━━━━━━━━━━━━━ 445s 3s/step - accuracy: 0.9540 - loss:
0.1214 - val_accuracy: 0.5625 - val_loss: 1.8846 - learning_rate:
3.0000e-04
Epoch 8/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 447s 3s/step - accuracy: 0.9609 - loss:
0.1247 - val_accuracy: 0.6875 - val_loss: 0.6540 - learning_rate:
9.0000e-05
Epoch 9/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - accuracy: 0.9686 - loss:
```

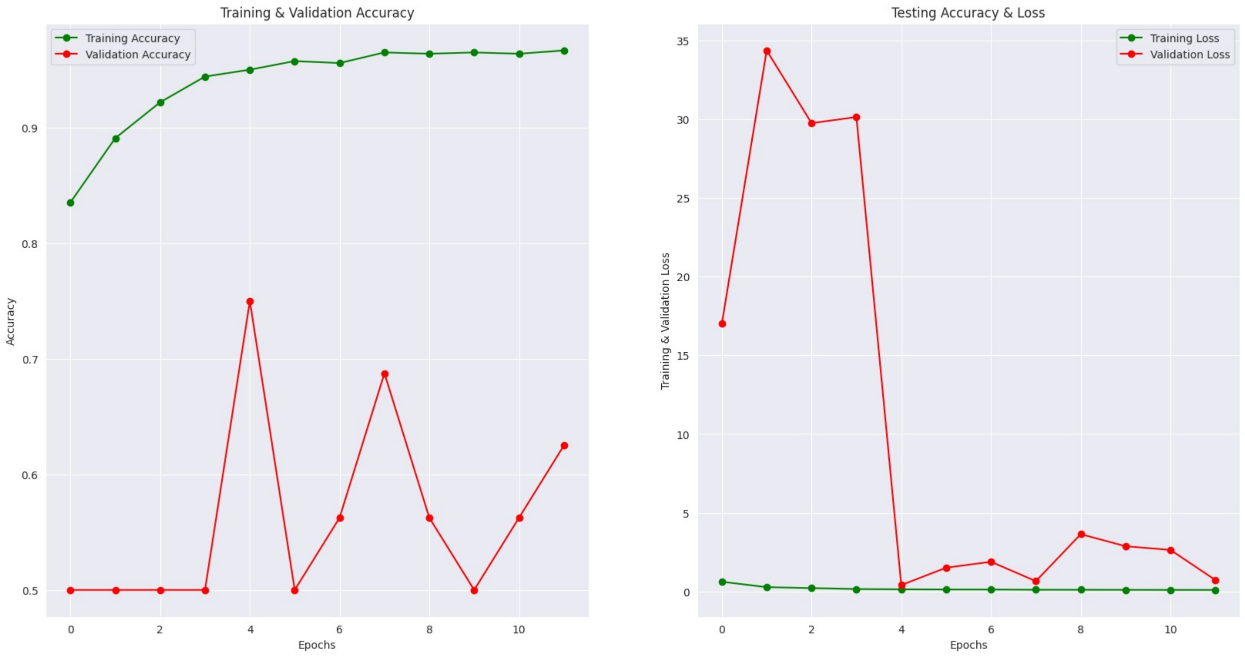
```
0.0944
Epoch 9: ReduceLROnPlateau reducing learning rate to
2.700000040931627e-05.
163/163 ━━━━━━━━━━━━━━━━━━━━ 439s 3s/step - accuracy: 0.9686 - loss:
0.0945 - val_accuracy: 0.5625 - val_loss: 3.6390 - learning_rate:
9.0000e-05
Epoch 10/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 431s 3s/step - accuracy: 0.9647 - loss:
0.1026 - val_accuracy: 0.5000 - val_loss: 2.8720 - learning_rate:
2.7000e-05
Epoch 11/12
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - accuracy: 0.9613 - loss:
0.0998
Epoch 11: ReduceLROnPlateau reducing learning rate to
8.100000013655517e-06.
163/163 ━━━━━━━━━━━━━━━━━━━━ 435s 3s/step - accuracy: 0.9614 - loss:
0.0998 - val_accuracy: 0.5625 - val_loss: 2.6391 - learning_rate:
2.7000e-05
Epoch 12/12
148/163 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.9692 - loss:
0.0948
```

```python
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)
[1]*100 , "%")
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 10s 477ms/step - accuracy: 0.9003 - loss:
0.2692
Loss of the model is -  0.2879559099674225
20/20 ━━━━━━━━━━━━━━━━━━━━ 16s 815ms/step - accuracy: 0.9003 - loss:
0.2692
Accuracy of the model is -  90.86538553237915 %
```

```python
epochs = [i for i in range(12)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
```

```
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()
```



```
predictions = (model.predict(x_test) > 0.5).astype("int32")
predictions = predictions.reshape(1,-1)[0]
predictions[:15]
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 10s 495ms/step
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0], dtype=int32)
```

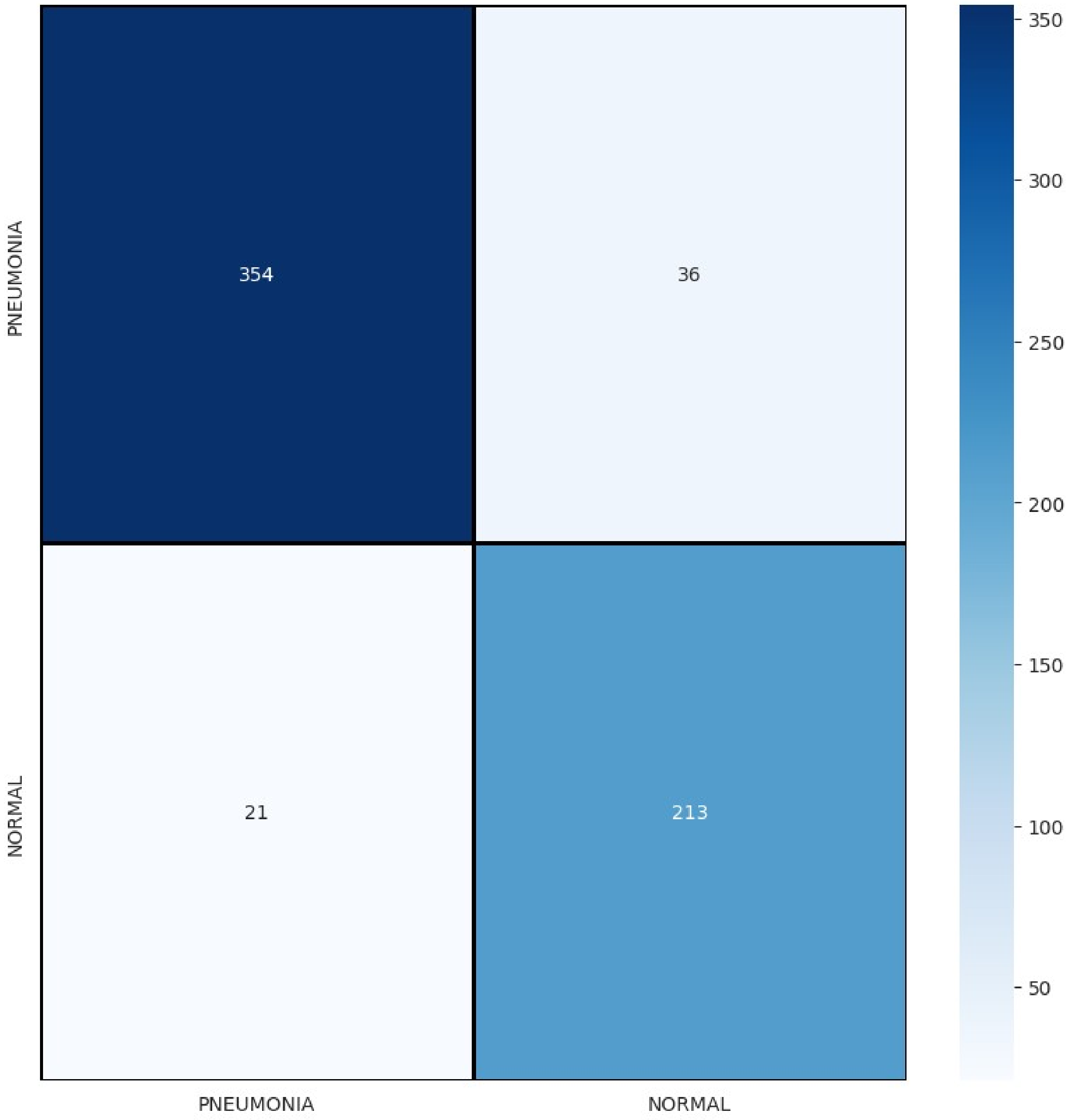
```
print(classification_report(y_test, predictions, target_names =
['Pneumonia (Class 0)','Normal (Class 1)']))
```

```
                     precision    recall  f1-score   support

Pneumonia (Class 0)       0.94      0.91      0.93       390
   Normal (Class 1)       0.86      0.91      0.88       234

           accuracy                           0.91       624
          macro avg       0.90      0.91      0.90       624
       weighted avg       0.91      0.91      0.91       624
```

```
cm = confusion_matrix(y_test,predictions)
cm
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
```

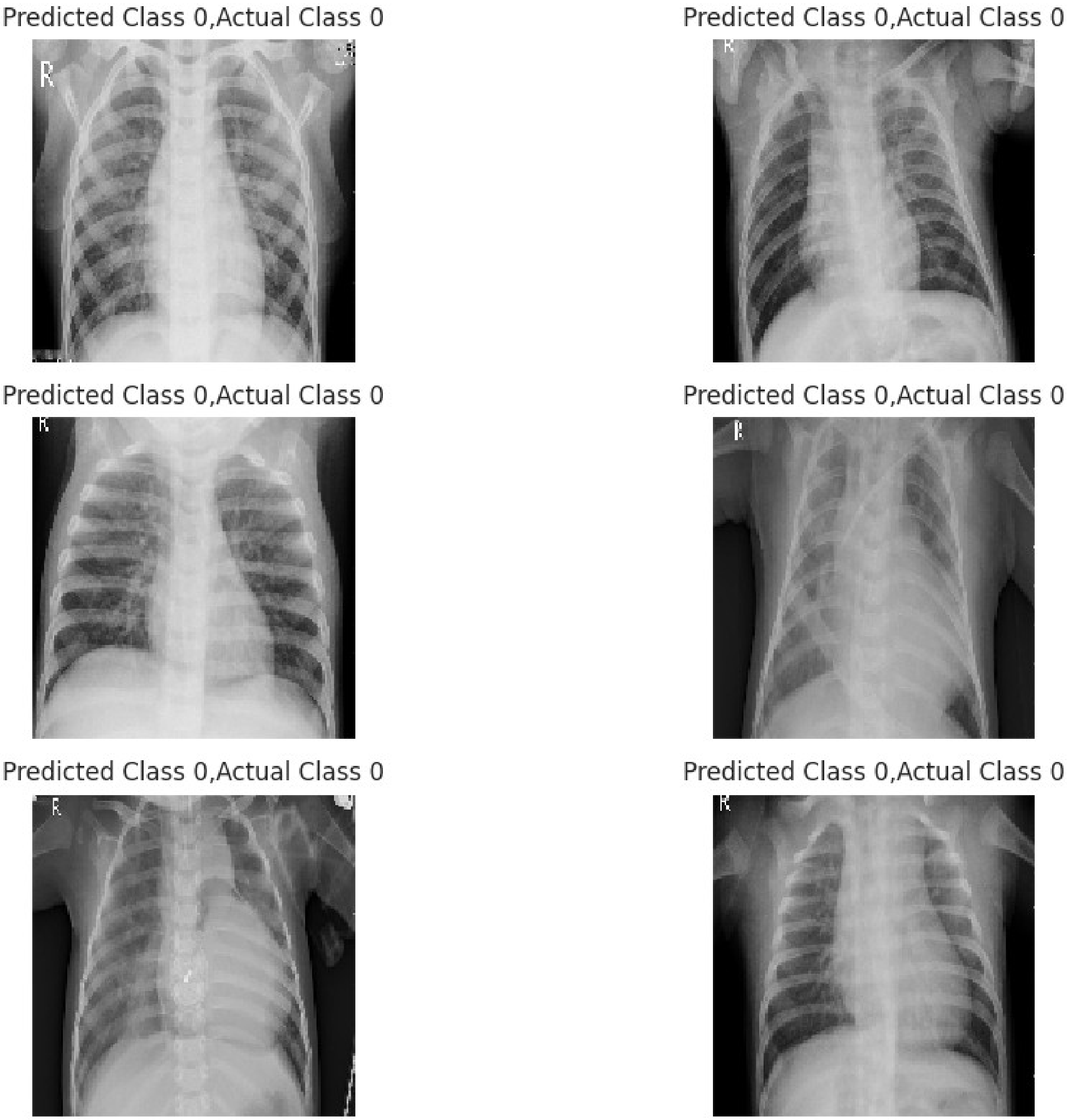
```
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 ,
annot = True, fmt='',xticklabels = labels,yticklabels = labels)
```

```
<Axes: >
```



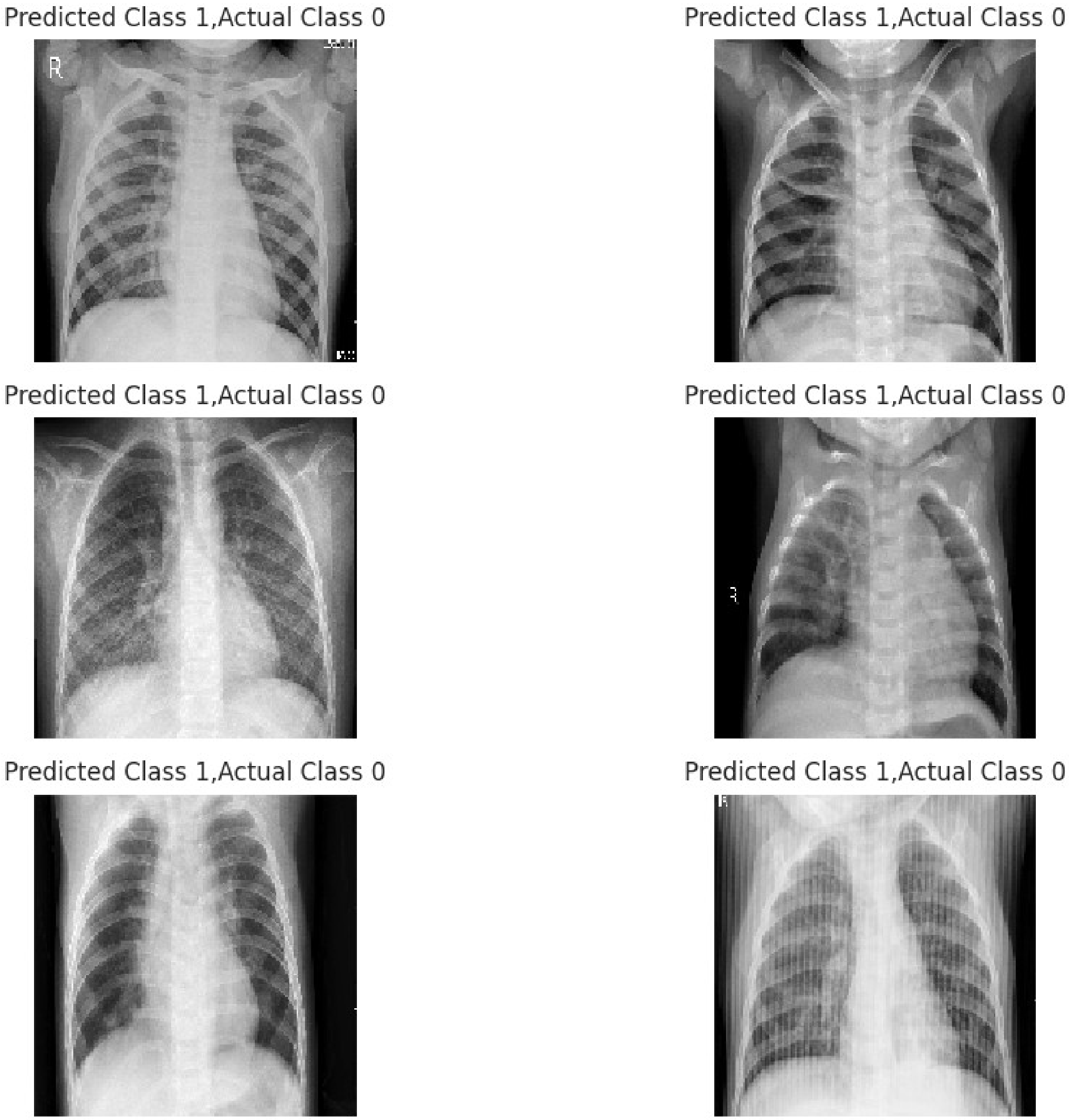
```
correct = np.nonzero(predictions == y_test)[0]
incorrect = np.nonzero(predictions != y_test)[0]
```

```
plt.figure(figsize=(12, 8)) # Set a larger figure size
i = 0
for c in correct[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(img_size, img_size), cmap="gray",
interpolation='none')
    plt.title("Predicted Class {},Actual Class
{}".format(predictions[c], y_test[c]))
    i += 1
plt.tight_layout() # Call once after all subplots are created
plt.show()
```


Predicted Class 0,Actual Class 0
Predicted Class 0,Actual Class 0
Predicted Class 0,Actual Class 0
Predicted Class 0,Actual Class 0
Predicted Class 0,Actual Class 0
Predicted Class 0,Actual Class 0

```
plt.figure(figsize=(12, 8)) # Set a larger figure size
i = 0
for c in incorrect[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(img_size, img_size), cmap="gray",
interpolation='none')
    plt.title("Predicted Class {},Actual Class
{}".format(predictions[c], y_test[c]))
```

```
    i += 1
plt.tight_layout() # Call once after all subplots are created
plt.show()
```



Predicted Class 1,Actual Class 0

Predicted Class 1,Actual Class 0

Predicted Class 1,Actual Class 0

Predicted Class 1,Actual Class 0

Predicted Class 1,Actual Class 0

Predicted Class 1,Actual Class 0