# Imported Libraries

In [1]:
```
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout, BatchNormalization, LSTM, Input, Re
shape
from tensorflow.keras.applications import VGG19
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.optimizers import RMSprop
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import cv2
import os
```

# Image Dataset Import

In [2]:
```
labels = ['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
img_size = 224
def get_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_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB format
                crop_image= img_arr[0:1728,430:2190]
                resized_arr = cv2.resize(crop_image, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

In [3]:
```
#function call to get_data function that takes file path of the dataset.
data= get_data('dataset/dataset_all_equal_size_image/')
```

```
<ipython-input-2-b08f5e223f84>:17: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which
is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do
this, you must specify 'dtype=object' when creating the ndarray
  return np.array(data)
```

In [4]:
```
data.shape
```

Out[4]:
```
(600, 2)
```

In [5]:
```
type(data)
```

Out[5]:
```
numpy.ndarray
```

# Dividing Data Ndarray into Normal, Cataract, Glaucoma and Retina diseases.

In [6]:
```
normal= data[0:300]
normal.shape
```

Out[6]:
```
(300, 2)
```

In [7]:
```
cataract=data[300:400]
cataract.shape
```

Out[7]:
```
(100, 2)
```

In [8]:
```
glaucoma= data[400:500]
glaucoma.shape
```

Out[8]:
```
(100, 2)
```

In [9]:
```
retina_disease= data[500:600]
retina_disease.shape
```

Out[9]: (100, 2)

In [10]:
```
random.seed(10)
np.random.shuffle(normal)
np.random.shuffle(cataract)
np.random.shuffle(glaucoma)
np.random.shuffle(retina_disease)
```

# Performing Normalization and Resize operation

In [11]:
```
def normalize(x_train,x_val,x_test):

    x_train = np.array(x_train) / 255
    x_train.reshape(-1, img_size, img_size, 1)

    x_test= np.array(x_test) / 255
    x_test.reshape(-1, img_size, img_size, 1)


    x_val= np.array(x_val) / 255
    x_val.reshape(-1, img_size, img_size, 1)


    return (x_train,x_val,x_test)
```

# Separating the Images and Labels into Respective Variables

In [12]:
```
def image_label_split(train,validation,test):

    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 validation:
      x_val.append(feature)
      y_val.append(label)


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


    y_train = np.array(y_train)
    y_val = np.array(y_val)
    y_test= np.array(y_test)

    return (x_train,y_train,x_val,y_val,x_test,y_test)
```

# VGG19-LSTM MODEL

In [13]:
```
def model_build_compile(k):
    baseModel = VGG19(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
    for layer in baseModel.layers:
            layer.trainable = False

    x = baseModel.output

        # LSTM layer
    x = Reshape((49,512))(x)
    x = ((LSTM(512, activation="relu", return_sequences=True, trainable=False)))(x)
    x = BatchNormalization()(x)
    #

        # FC layer
    x = Flatten(name="flatten")(x)

        # fc1 layer
    x = Dense(units=4096, activation='relu')(x)
    x = BatchNormalization()(x)
    #

        # fc2 layer
    x = Dense(units=4096, activation='relu')(x)
    x = BatchNormalization()(x)
    #

        # Output layer
    output = Dense(units=4, activation='softmax')(x)

    model = Model(inputs=baseModel.input, outputs=output)
    opt = RMSprop(learning_rate=0.01, clipvalue=100)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=["accuracy"])
    k=k+1
    print("model building and compiling for fold",k)

    return model
```

## Model prediction for Test Images and Computation of Sensitivity and Specificity

In [14]:
```
def test_pred(x_val,y_val,k):
    predictions = model.predict(x_val)
    predictions = np.argmax(predictions, axis = -1)

    print('----------------Test accuracy for',k+1,'fold----------------')
    #Confusion matrix, Accuracy, sensitivity and specificity
    cm1 = confusion_matrix(y_val,predictions)
    print('Confusion Matrix : \n', cm1)


    #####from confusion matrix calculate accuracy


    sensitivity_1_normal = (cm1[0,0])/(cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[0,3])
    #print('Sensitivity_1_normal          : ', sensitivity_1_normal )

    sensitivity_2_cataract = (cm1[1,1])/(cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[1,3])
    #print('Sensitivity_2_cataract        : ', sensitivity_2_cataract )


    sensitivity_3_glaucoma = (cm1[2,2])/(cm1[2,0]+cm1[2,1]+cm1[2,2]+cm1[2,3])
    #print('Sensitivity_3_glaucoma        : ', sensitivity_3_glaucoma )

    sensitivity_4_retina_disease = (cm1[3,3])/(cm1[3,0]+cm1[3,1]+cm1[3,2]+cm1[3,3])
    #print('Sensitivity_4_retina_disease  : ', sensitivity_4_retina_disease )

    specificity_1_normal = (cm1[1,1]+cm1[1,2]+cm1[1,3]+cm1[2,1]+cm1[2,2]+cm1[2,3]+cm1[3,1]+cm1[3,2]+cm1[3,3])/(cm1[1,0
]+cm1[2,0]+cm1[3,0]+cm1[1,1]+cm1[1,2]+cm1[1,3]+cm1[2,1]+cm1[2,2]+cm1[2,3]+cm1[3,1]+cm1[3,2]+cm1[3,3])
    #print('Specificity : ', specificity_1_normal)

    specificity_2_cataract = (cm1[0,0]+cm1[0,2]+cm1[0,3]+cm1[2,0]+cm1[2,2]+cm1[2,3]+cm1[3,0]+cm1[3,2]+cm1[3,3])/(cm1[0
,1]+cm1[2,1]+cm1[3,1]+cm1[0,0]+cm1[0,2]+cm1[0,3]+cm1[2,0]+cm1[2,2]+cm1[2,3]+cm1[3,0]+cm1[3,2]+cm1[3,3])
    #print('Specificity : ', specificity_2_cataract)

    specificity_3_glaucoma = (cm1[0,0]+cm1[0,1]+cm1[0,3]+cm1[1,0]+cm1[1,1]+cm1[1,3]+cm1[3,0]+cm1[3,1]+cm1[3,3])/(cm1[0
,2]+cm1[1,2]+cm1[3,2]+cm1[0,0]+cm1[0,1]+cm1[0,3]+cm1[1,0]+cm1[1,1]+cm1[1,3]+cm1[3,0]+cm1[3,1]+cm1[3,3])
    #print('Specificity : ', specificity_3_glaucoma)

    specificity_4_retina_disease= (cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[2,0]+cm1[2,1]+cm1[2,2])/(
cm1[0,3]+cm1[1,3]+cm1[2,3]+cm1[0,0]+cm1[0,1]+cm1[0,2]+cm1[1,0]+cm1[1,1]+cm1[1,2]+cm1[2,0]+cm1[2,1]+cm1[2,2])
    #print('Specificity : ', specificity_4_retina_disease)

    Sensitivity= (sensitivity_1_normal + sensitivity_2_cataract + sensitivity_3_glaucoma + sensitivity_4_retina_diseas
e)/4
    #print(Sensitivity)

    Specificity= (specificity_1_normal + specificity_2_cataract + specificity_3_glaucoma + specificity_4_retina_diseas
e)/4
    #print(Specificity)

    total1=sum(sum(cm1))
    test_accuracy=(cm1[0,0]+cm1[1,1]+cm1[2,2]+cm1[3,3])/total1

    print ('Accuracy    : ', test_accuracy)
    print ('Specificity : ', Specificity)
    print ('Sensitivity : ', Sensitivity)
    print('-----------------------------End of',k+1,'Fold--------------------------')
    return test_accuracy,Specificity,Sensitivity,cm1
```

In [15]:
```
CM= []
test_accuracy=[]
test_sensitivity=[]
test_specificity=[]
train_acc = []
val_acc = []
train_loss = []
val_loss = []
```

# VGG19-LSTM 5 Fold Cross Validation

In [16]:
```
for k in range (5):  # for loop to run 5 folds
    n_normal=30      # specifying the number of images for normal class in test phase,calulated as per 10% of total no
rmal class images 300.
    n_rest=10        # specifying the number of images for disease classes in test phase,calulated as per 10% of total
normal class images 100.

    # Adding the images in normal validation set by using k*n_normal to (k+1)*n_normal as index values for normal data
set divided in cell 6.
    test_normal= normal[k*n_normal:(k+1)*n_normal]
    print('------------------------Start of',k+1,'Fold------------------------')
    print('test images for normal class from',k*n_normal,(k+1)*n_normal)

    # Adding the images in cataract validation set by using k*n_rest to (k+1)*n_rest as index values for cataract data
set divided in cell 7.
    test_cataract= cataract[k*n_rest:(k+1)*n_rest]
    print('test images for cataract class from',k*n_rest,(k+1)*n_rest)

    # Adding the images in gluacoma validation set by using k*n_rest to (k+1)*n_rest as index values for gluacoma data
set divided in cell 8.
    test_glaucoma= glaucoma[k*n_rest:(k+1)*n_rest]
    print('test images for glaucoma class from',k*n_rest,(k+1)*n_rest)

    # Adding the images in retina disease validation set by using k*n_rest to (k+1)*n_rest as index values for retina
 disease dataset divided in cell 9.
    test_retina= retina_disease[k*n_rest:(k+1)*n_rest]
    print('test images for retina disease class from',k*n_rest,(k+1)*n_rest)

    # Now for train and validation set of Normal images first adding 0 to k*n_normal images and then adding all the im
ages from (k+1)*n_normal till last image.

    train_validation_normal= normal[:k*n_normal]
    train_validation_normal= np.append(train_validation_normal,normal[(k+1)*n_normal:],axis=0)
    print('train_validation images for normal class from 0 to',k*n_normal,'and',(k+1)*n_normal,'to 300')

    # Now for train and validation set of cataract images first adding 0 to k*n_rest images and then adding all the im
ages from (k+1)*n_rest till last image.

    train_validation_cataract= cataract[:k*n_rest]
    train_validation_cataract= np.append(train_validation_cataract,cataract[(k+1)*n_rest:],axis=0)
    print('train_validation images for cataract class from 0 to',k*n_rest,'and',(k+1)*n_rest,'to 100')

    # Now for train and validation set of glaucoma images first adding 0 to k*n_rest images and then adding all the im
ages from (k+1)*n_rest till last image.
    train_validation_glaucoma= glaucoma[:k*n_rest]
    train_validation_glaucoma= np.append(train_validation_glaucoma,glaucoma[(k+1)*n_rest:],axis=0)
    print('train_validation  images for glaucoma class from 0',k*n_rest,'and',(k+1)*n_rest,'to 100')

    # Now for train and validation set of retina disease images first adding 0 to k*n_rest images and then adding all
 the images from (k+1)*n_rest till last image.
    train_validation_retina= retina_disease[:k*n_rest]
    train_validation_retina= np.append(train_validation_retina,retina_disease[(k+1)*n_rest:],axis=0)
    print('train_validation images for retina disease class from 0 to',k*n_rest,'and',(k+1)*n_rest,'to 100')


    # Splitting the train validation datasets in 80:20 ratio which would eventually give us 70% images in train and 2
0% images in validation and 10% in test.
    normal_train, normal_validation                 = train_test_split(train_validation_normal, test_size=0.20, random
_state=14,shuffle=True)
    cataract_train, cataract_validation             = train_test_split(train_validation_cataract, test_size=0.20, rand
om_state=14,shuffle=True)
    glaucoma_train, glaucoma_validation             = train_test_split(train_validation_glaucoma, test_size=0.20, rand
om_state=14,shuffle=True)
    retina_disease_train, retina_disease_validation = train_test_split(train_validation_retina, test_size=0.20, random
_state=14,shuffle=True)

    # Appending all train set images for all classes
    train= np.append(normal_train,cataract_train,axis=0)
    train= np.append(train,glaucoma_train,axis=0)
    train= np.append(train,retina_disease_train,axis=0)

    # Appending all validation set images for all classes
    validation= np.append(normal_validation,cataract_validation,axis=0)
    validation= np.append(validation,glaucoma_validation,axis=0)
    validation= np.append(validation,retina_disease_validation,axis=0)

    # Appending all test set images for all classes
    test= np.append(test_normal,test_cataract,axis=0)
    test= np.append(test,test_glaucoma,axis=0)
    test= np.append(test,test_retina,axis=0)

    # Shuffling the train validation and test set as they are added sequentially.
    random.seed(6)
    np.random.shuffle(train)
    np.random.shuffle(validation)
    np.random.shuffle(test)

    # Passing the train validation test as argument for image_label_split function that return features and labels sep
arated.
```

```
    x_train,y_train,x_val,y_val,x_test,y_test = image_label_split(train,validation,test)

    # Passing the x_Train x_val and x_test as a argument for normalize function that returns the normalized and reshap
ed sets.
    x_train,x_val,x_test = normalize(x_train,x_val,x_test)

    # model building and model compile is done using a model_build_compile().

    model = model_build_compile(k)
    history = model.fit(x_train,y_train,epochs =50, validation_data = (x_val,y_val))


    train_acc = np.append(train_acc,history.history['accuracy'])
    val_acc = np.append(val_acc,history.history['val_accuracy'])

    train_loss = np.append(train_loss,history.history['loss'])
    val_loss = np.append(val_loss,history.history['val_loss'])

    x,y,z,c = test_pred(x_test,y_test,k)


    CM.append([c])
    test_accuracy.append(x)
    test_specificity.append(y)
    test_sensitivity.append(z)
```

```
------------------------Start of 1 Fold------------------------
test images for normal class from 0 30
test images for cataract class from 0 10
test images for glaucoma class from 0 10
test images for retina disease class from 0 10
train_validation images for normal class from 0 to 0 and 30 to 300
train_validation images for cataract class from 0 to 0 and 10 to 100
train_validation  images for glaucoma class from 0 0 and 10 to 100
train_validation images for retina disease class from 0 to 0 and 10 to 100
model building and compiling for fold 1
Epoch 1/50
14/14 [==============================] - 95s 7s/step - loss: 17.1688 - accuracy: 0.4329 - val_loss: 196.7006 - val_ac
curacy: 0.1667
Epoch 2/50
14/14 [==============================] - 104s 7s/step - loss: 10.2623 - accuracy: 0.5324 - val_loss: 182.0252 - val_a
ccuracy: 0.1667
Epoch 3/50
14/14 [==============================] - 106s 8s/step - loss: 6.5122 - accuracy: 0.5486 - val_loss: 80.9171 - val_acc
uracy: 0.1667
Epoch 4/50
14/14 [==============================] - 111s 8s/step - loss: 4.8127 - accuracy: 0.5463 - val_loss: 89.6496 - val_acc
uracy: 0.1667
Epoch 5/50
14/14 [==============================] - 108s 8s/step - loss: 3.0963 - accuracy: 0.5579 - val_loss: 92.5832 - val_acc
uracy: 0.1667
Epoch 6/50
14/14 [==============================] - 112s 8s/step - loss: 2.4622 - accuracy: 0.5671 - val_loss: 126.0165 - val_ac
curacy: 0.1667
Epoch 7/50
14/14 [==============================] - 115s 8s/step - loss: 1.9443 - accuracy: 0.6204 - val_loss: 95.8986 - val_acc
uracy: 0.1667
Epoch 8/50
14/14 [==============================] - 111s 8s/step - loss: 1.4325 - accuracy: 0.6505 - val_loss: 60.4648 - val_acc
uracy: 0.1667
Epoch 9/50
14/14 [==============================] - 123s 9s/step - loss: 1.9062 - accuracy: 0.6181 - val_loss: 45.0711 - val_acc
uracy: 0.1667
Epoch 10/50
14/14 [==============================] - 127s 9s/step - loss: 1.2777 - accuracy: 0.6574 - val_loss: 34.6552 - val_acc
uracy: 0.1667
Epoch 11/50
14/14 [==============================] - 137s 10s/step - loss: 1.0739 - accuracy: 0.6551 - val_loss: 23.9938 - val_ac
curacy: 0.1667
Epoch 12/50
14/14 [==============================] - 126s 9s/step - loss: 0.8713 - accuracy: 0.7037 - val_loss: 28.1554 - val_acc
uracy: 0.1667
Epoch 13/50
14/14 [==============================] - 134s 10s/step - loss: 1.0494 - accuracy: 0.6875 - val_loss: 30.5307 - val_ac
curacy: 0.1667
Epoch 14/50
14/14 [==============================] - 131s 9s/step - loss: 1.0944 - accuracy: 0.7176 - val_loss: 41.3850 - val_acc
uracy: 0.1667
Epoch 15/50
14/14 [==============================] - 135s 10s/step - loss: 1.0734 - accuracy: 0.7315 - val_loss: 48.4695 - val_ac
curacy: 0.1667
Epoch 16/50
14/14 [==============================] - 128s 9s/step - loss: 0.8891 - accuracy: 0.7616 - val_loss: 42.0234 - val_acc
uracy: 0.1667
Epoch 17/50
14/14 [==============================] - 128s 9s/step - loss: 0.7176 - accuracy: 0.7685 - val_loss: 74.6133 - val_acc
uracy: 0.1667
Epoch 18/50
14/14 [==============================] - 128s 9s/step - loss: 0.6128 - accuracy: 0.7917 - val_loss: 47.9438 - val_acc
uracy: 0.1667
Epoch 19/50
14/14 [==============================] - 128s 9s/step - loss: 0.5688 - accuracy: 0.8264 - val_loss: 38.3707 - val_acc
uracy: 0.1667
Epoch 20/50
14/14 [==============================] - 128s 9s/step - loss: 0.7104 - accuracy: 0.7870 - val_loss: 33.0233 - val_acc
uracy: 0.1667
Epoch 21/50
14/14 [==============================] - 128s 9s/step - loss: 0.6159 - accuracy: 0.7847 - val_loss: 29.1670 - val_acc
uracy: 0.1667
Epoch 22/50
14/14 [==============================] - 129s 9s/step - loss: 0.3171 - accuracy: 0.8843 - val_loss: 20.9521 - val_acc
uracy: 0.1667
Epoch 23/50
14/14 [==============================] - 128s 9s/step - loss: 0.6538 - accuracy: 0.7986 - val_loss: 21.1757 - val_acc
uracy: 0.1667
Epoch 24/50
14/14 [==============================] - 129s 9s/step - loss: 0.5560 - accuracy: 0.8148 - val_loss: 29.3038 - val_acc
uracy: 0.1667
Epoch 25/50
14/14 [==============================] - 130s 9s/step - loss: 0.5522 - accuracy: 0.8634 - val_loss: 28.3086 - val_acc
uracy: 0.1667
Epoch 26/50
14/14 [==============================] - 129s 9s/step - loss: 0.3393 - accuracy: 0.8889 - val_loss: 7.7337 - val_accu
racy: 0.1667
```

```
Epoch 27/50
14/14 [==============================] - 130s 9s/step - loss: 0.4488 - accuracy: 0.8380 - val_loss: 25.2924 - val_acc
uracy: 0.1667
Epoch 28/50
14/14 [==============================] - 129s 9s/step - loss: 0.3826 - accuracy: 0.8588 - val_loss: 291.1323 - val_ac
curacy: 0.1667
Epoch 29/50
14/14 [==============================] - 129s 9s/step - loss: 0.4341 - accuracy: 0.8426 - val_loss: 20.8130 - val_acc
uracy: 0.1667
Epoch 30/50
14/14 [==============================] - 129s 9s/step - loss: 0.3195 - accuracy: 0.8935 - val_loss: 24.3082 - val_acc
uracy: 0.1667
Epoch 31/50
14/14 [==============================] - 129s 9s/step - loss: 0.2103 - accuracy: 0.9352 - val_loss: 26.8555 - val_acc
uracy: 0.1667
Epoch 32/50
14/14 [==============================] - 129s 9s/step - loss: 0.2979 - accuracy: 0.9074 - val_loss: 32.6124 - val_acc
uracy: 0.1667
Epoch 33/50
14/14 [==============================] - 129s 9s/step - loss: 0.3904 - accuracy: 0.8866 - val_loss: 27.5971 - val_acc
uracy: 0.1667
Epoch 34/50
14/14 [==============================] - 129s 9s/step - loss: 0.3697 - accuracy: 0.8773 - val_loss: 15.7412 - val_acc
uracy: 0.2037
Epoch 35/50
14/14 [==============================] - 128s 9s/step - loss: 0.1819 - accuracy: 0.9236 - val_loss: 16.8729 - val_acc
uracy: 0.2037
Epoch 36/50
14/14 [==============================] - 129s 9s/step - loss: 0.2518 - accuracy: 0.9259 - val_loss: 17.8562 - val_acc
uracy: 0.1667
Epoch 37/50
14/14 [==============================] - 128s 9s/step - loss: 0.4690 - accuracy: 0.8796 - val_loss: 13.6484 - val_acc
uracy: 0.1667
Epoch 38/50
14/14 [==============================] - 129s 9s/step - loss: 0.2475 - accuracy: 0.9167 - val_loss: 19.7469 - val_acc
uracy: 0.1667
Epoch 39/50
14/14 [==============================] - 129s 9s/step - loss: 0.1568 - accuracy: 0.9444 - val_loss: 8.9758 - val_accu
racy: 0.2315
Epoch 40/50
14/14 [==============================] - 129s 9s/step - loss: 0.2722 - accuracy: 0.9213 - val_loss: 18.1675 - val_acc
uracy: 0.2130
Epoch 41/50
14/14 [==============================] - 129s 9s/step - loss: 0.1806 - accuracy: 0.9375 - val_loss: 5.3432 - val_accu
racy: 0.2037
Epoch 42/50
14/14 [==============================] - 129s 9s/step - loss: 0.2604 - accuracy: 0.9213 - val_loss: 23.0669 - val_acc
uracy: 0.1667
Epoch 43/50
14/14 [==============================] - 130s 9s/step - loss: 0.1508 - accuracy: 0.9421 - val_loss: 17.0223 - val_acc
uracy: 0.2222
Epoch 44/50
14/14 [==============================] - 129s 9s/step - loss: 0.1405 - accuracy: 0.9491 - val_loss: 16.5403 - val_acc
uracy: 0.1852
Epoch 45/50
14/14 [==============================] - 129s 9s/step - loss: 0.2358 - accuracy: 0.9144 - val_loss: 10.8756 - val_acc
uracy: 0.2130
Epoch 46/50
14/14 [==============================] - 128s 9s/step - loss: 0.2321 - accuracy: 0.9398 - val_loss: 9.7610 - val_accu
racy: 0.2222
Epoch 47/50
14/14 [==============================] - 129s 9s/step - loss: 0.1479 - accuracy: 0.9537 - val_loss: 8.7932 - val_accu
racy: 0.3056
Epoch 48/50
14/14 [==============================] - 129s 9s/step - loss: 0.2223 - accuracy: 0.9468 - val_loss: 5.4136 - val_accu
racy: 0.4444
Epoch 49/50
14/14 [==============================] - 129s 9s/step - loss: 0.1005 - accuracy: 0.9699 - val_loss: 4.8416 - val_accu
racy: 0.5463
Epoch 50/50
14/14 [==============================] - 129s 9s/step - loss: 0.2247 - accuracy: 0.9259 - val_loss: 4.8594 - val_accu
racy: 0.4444
----------------Test accuracy for 1 fold----------------
Confusion Matrix :
 [[18  0  0 12]
 [ 9  0  0  1]
 [ 7  0  2  1]
 [ 6  0  0  4]]
Accuracy    :  0.4
Specificity :  0.7006302521008404
Sensitivity :  0.30000000000000004
-----------------------------End of 1 Fold-------------------------
------------------------Start of 2 Fold------------------------
test images for normal class from 30 60
test images for cataract class from 10 20
test images for glaucoma class from 10 20
test images for retina disease class from 10 20
train_validation images for normal class from 0 to 30 and 60 to 300
```

```
train_validation images for cataract class from 0 to 10 and 20 to 100
train_validation  images for glaucoma class from 0 10 and 20 to 100
train_validation images for retina disease class from 0 to 10 and 20 to 100
model building and compiling for fold 2
Epoch 1/50
14/14 [==============================] - 131s 9s/step - loss: 22.7639 - accuracy: 0.3542 - val_loss: 73.1095 - val_ac
curacy: 0.1667
Epoch 2/50
14/14 [==============================] - 130s 9s/step - loss: 8.1997 - accuracy: 0.4931 - val_loss: 62.4114 - val_acc
uracy: 0.5000
Epoch 3/50
14/14 [==============================] - 129s 9s/step - loss: 7.6824 - accuracy: 0.5278 - val_loss: 40.7816 - val_acc
uracy: 0.5000
Epoch 4/50
14/14 [==============================] - 130s 9s/step - loss: 6.0112 - accuracy: 0.5370 - val_loss: 72.6413 - val_acc
uracy: 0.5000
Epoch 5/50
14/14 [==============================] - 130s 9s/step - loss: 3.1000 - accuracy: 0.5764 - val_loss: 29.9370 - val_acc
uracy: 0.5000
Epoch 6/50
14/14 [==============================] - 129s 9s/step - loss: 2.9786 - accuracy: 0.5764 - val_loss: 106.9048 - val_ac
curacy: 0.1667
Epoch 7/50
14/14 [==============================] - 129s 9s/step - loss: 2.0087 - accuracy: 0.5949 - val_loss: 85.5513 - val_acc
uracy: 0.1667
Epoch 8/50
14/14 [==============================] - 128s 9s/step - loss: 1.5976 - accuracy: 0.6134 - val_loss: 44.1482 - val_acc
uracy: 0.1667
Epoch 9/50
14/14 [==============================] - 130s 9s/step - loss: 1.5211 - accuracy: 0.6065 - val_loss: 71.5443 - val_acc
uracy: 0.1667
Epoch 10/50
14/14 [==============================] - 129s 9s/step - loss: 0.8962 - accuracy: 0.6644 - val_loss: 53.1208 - val_acc
uracy: 0.1667
Epoch 11/50
14/14 [==============================] - 129s 9s/step - loss: 0.9243 - accuracy: 0.6667 - val_loss: 24.7045 - val_acc
uracy: 0.5000
Epoch 12/50
14/14 [==============================] - 129s 9s/step - loss: 1.0120 - accuracy: 0.6759 - val_loss: 30.8359 - val_acc
uracy: 0.1667
Epoch 13/50
14/14 [==============================] - 129s 9s/step - loss: 0.9481 - accuracy: 0.7014 - val_loss: 20.3741 - val_acc
uracy: 0.5000
Epoch 14/50
14/14 [==============================] - 129s 9s/step - loss: 0.8221 - accuracy: 0.7222 - val_loss: 18.3269 - val_acc
uracy: 0.2500
Epoch 15/50
14/14 [==============================] - 129s 9s/step - loss: 1.0514 - accuracy: 0.7384 - val_loss: 17.8524 - val_acc
uracy: 0.4907
Epoch 16/50
14/14 [==============================] - 129s 9s/step - loss: 0.7396 - accuracy: 0.7407 - val_loss: 17.9020 - val_acc
uracy: 0.4815
Epoch 17/50
14/14 [==============================] - 129s 9s/step - loss: 0.6754 - accuracy: 0.7315 - val_loss: 10.0179 - val_acc
uracy: 0.3611
Epoch 18/50
14/14 [==============================] - 130s 9s/step - loss: 0.5176 - accuracy: 0.8310 - val_loss: 22.5914 - val_acc
uracy: 0.5000
Epoch 19/50
14/14 [==============================] - 129s 9s/step - loss: 0.5162 - accuracy: 0.8310 - val_loss: 13.0883 - val_acc
uracy: 0.3519
Epoch 20/50
14/14 [==============================] - 129s 9s/step - loss: 0.5550 - accuracy: 0.8056 - val_loss: 12.7450 - val_acc
uracy: 0.4352
Epoch 21/50
14/14 [==============================] - 129s 9s/step - loss: 0.4527 - accuracy: 0.8403 - val_loss: 11.2413 - val_acc
uracy: 0.2593
Epoch 22/50
14/14 [==============================] - 129s 9s/step - loss: 0.4674 - accuracy: 0.8125 - val_loss: 11.9682 - val_acc
uracy: 0.2500
Epoch 23/50
14/14 [==============================] - 130s 9s/step - loss: 0.4932 - accuracy: 0.8218 - val_loss: 19.0884 - val_acc
uracy: 0.1667
Epoch 24/50
14/14 [==============================] - 129s 9s/step - loss: 0.2938 - accuracy: 0.8866 - val_loss: 22.3165 - val_acc
uracy: 0.1667
Epoch 25/50
14/14 [==============================] - 129s 9s/step - loss: 0.3891 - accuracy: 0.8704 - val_loss: 22.8577 - val_acc
uracy: 0.1667
Epoch 26/50
14/14 [==============================] - 129s 9s/step - loss: 0.4058 - accuracy: 0.8657 - val_loss: 10.7747 - val_acc
uracy: 0.1944
Epoch 27/50
14/14 [==============================] - 129s 9s/step - loss: 0.2404 - accuracy: 0.9097 - val_loss: 17.9907 - val_acc
uracy: 0.1667
Epoch 28/50
14/14 [==============================] - 130s 9s/step - loss: 0.2563 - accuracy: 0.9144 - val_loss: 25.2315 - val_acc
uracy: 0.1667
```

```
Epoch 29/50
14/14 [==============================] - 129s 9s/step - loss: 0.2678 - accuracy: 0.9120 - val_loss: 22.2630 - val_acc
uracy: 0.1667
Epoch 30/50
14/14 [==============================] - 129s 9s/step - loss: 0.3754 - accuracy: 0.8843 - val_loss: 21.2296 - val_acc
uracy: 0.1667
Epoch 31/50
14/14 [==============================] - 130s 9s/step - loss: 0.2379 - accuracy: 0.9167 - val_loss: 30.0420 - val_acc
uracy: 0.1667
Epoch 32/50
14/14 [==============================] - 132s 10s/step - loss: 0.4721 - accuracy: 0.8796 - val_loss: 12.4837 - val_ac
curacy: 0.1667
Epoch 33/50
14/14 [==============================] - 131s 9s/step - loss: 0.3102 - accuracy: 0.8912 - val_loss: 15.3751 - val_acc
uracy: 0.1852
Epoch 34/50
14/14 [==============================] - 129s 9s/step - loss: 0.1754 - accuracy: 0.9421 - val_loss: 14.2184 - val_acc
uracy: 0.1759
Epoch 35/50
14/14 [==============================] - 132s 10s/step - loss: 0.2523 - accuracy: 0.9375 - val_loss: 13.1062 - val_ac
curacy: 0.2130
Epoch 36/50
14/14 [==============================] - 138s 10s/step - loss: 0.2307 - accuracy: 0.9282 - val_loss: 11.5629 - val_ac
curacy: 0.2222
Epoch 37/50
14/14 [==============================] - 136s 10s/step - loss: 0.1026 - accuracy: 0.9583 - val_loss: 11.2741 - val_ac
curacy: 0.2593
Epoch 38/50
14/14 [==============================] - 129s 9s/step - loss: 0.1554 - accuracy: 0.9537 - val_loss: 15.8083 - val_acc
uracy: 0.2315
Epoch 39/50
14/14 [==============================] - 129s 9s/step - loss: 0.1609 - accuracy: 0.9444 - val_loss: 6.9012 - val_accu
racy: 0.2963
Epoch 40/50
14/14 [==============================] - 128s 9s/step - loss: 0.1829 - accuracy: 0.9398 - val_loss: 8.2940 - val_accu
racy: 0.3889
Epoch 41/50
14/14 [==============================] - 129s 9s/step - loss: 0.3198 - accuracy: 0.9051 - val_loss: 6.9268 - val_accu
racy: 0.2870
Epoch 42/50
14/14 [==============================] - 129s 9s/step - loss: 0.1830 - accuracy: 0.9514 - val_loss: 5.0011 - val_accu
racy: 0.3426
Epoch 43/50
14/14 [==============================] - 129s 9s/step - loss: 0.1737 - accuracy: 0.9514 - val_loss: 5.6181 - val_accu
racy: 0.5000
Epoch 44/50
14/14 [==============================] - 129s 9s/step - loss: 0.1494 - accuracy: 0.9537 - val_loss: 6.4810 - val_accu
racy: 0.5556
Epoch 45/50
14/14 [==============================] - 128s 9s/step - loss: 0.1837 - accuracy: 0.9398 - val_loss: 6.4928 - val_accu
racy: 0.3981
Epoch 46/50
14/14 [==============================] - 130s 9s/step - loss: 0.2638 - accuracy: 0.9213 - val_loss: 7.1057 - val_accu
racy: 0.3519
Epoch 47/50
14/14 [==============================] - 128s 9s/step - loss: 0.0830 - accuracy: 0.9769 - val_loss: 7.1976 - val_accu
racy: 0.4444
Epoch 48/50
14/14 [==============================] - 128s 9s/step - loss: 0.0483 - accuracy: 0.9884 - val_loss: 5.9421 - val_accu
racy: 0.5000
Epoch 49/50
14/14 [==============================] - 129s 9s/step - loss: 0.1384 - accuracy: 0.9491 - val_loss: 10.6615 - val_acc
uracy: 0.5741
Epoch 50/50
14/14 [==============================] - 128s 9s/step - loss: 0.0790 - accuracy: 0.9699 - val_loss: 5.4603 - val_accu
racy: 0.5463
----------------Test accuracy for 2 fold----------------
Confusion Matrix :
 [[25  0  1  4]
 [ 2  7  0  1]
 [ 6  0  1  3]
 [ 6  0  0  4]]
Accuracy    :  0.6166666666666667
Specificity :  0.8098473708229805
Sensitivity :  0.5083333333333333
----------------------------End of 2 Fold-------------------------
------------------------Start of 3 Fold------------------------
test images for normal class from 60 90
test images for cataract class from 20 30
test images for glaucoma class from 20 30
test images for retina disease class from 20 30
train_validation images for normal class from 0 to 60 and 90 to 300
train_validation images for cataract class from 0 to 20 and 30 to 100
train_validation  images for glaucoma class from 0 20 and 30 to 100
train_validation images for retina disease class from 0 to 20 and 30 to 100
model building and compiling for fold 3
Epoch 1/50
14/14 [==============================] - 132s 9s/step - loss: 20.3265 - accuracy: 0.4167 - val_loss: 186.2779 - val_a
```

```
ccuracy: 0.1667
Epoch 2/50
14/14 [==============================] - 129s 9s/step - loss: 9.1986 - accuracy: 0.5255 - val_loss: 29.4180 - val_acc
uracy: 0.4537
Epoch 3/50
14/14 [==============================] - 130s 9s/step - loss: 7.4582 - accuracy: 0.5324 - val_loss: 88.0118 - val_acc
uracy: 0.1667
Epoch 4/50
14/14 [==============================] - 129s 9s/step - loss: 4.3997 - accuracy: 0.5694 - val_loss: 90.7220 - val_acc
uracy: 0.5000
Epoch 5/50
14/14 [==============================] - 130s 9s/step - loss: 4.0392 - accuracy: 0.5162 - val_loss: 68.4370 - val_acc
uracy: 0.5000
Epoch 6/50
14/14 [==============================] - 131s 9s/step - loss: 2.5803 - accuracy: 0.5602 - val_loss: 69.9856 - val_acc
uracy: 0.1667
Epoch 7/50
14/14 [==============================] - 129s 9s/step - loss: 1.6130 - accuracy: 0.6111 - val_loss: 155.0898 - val_ac
curacy: 0.1667
Epoch 8/50
14/14 [==============================] - 129s 9s/step - loss: 1.7526 - accuracy: 0.5856 - val_loss: 45.9068 - val_acc
uracy: 0.1667
Epoch 9/50
14/14 [==============================] - 129s 9s/step - loss: 1.1657 - accuracy: 0.6319 - val_loss: 65.8980 - val_acc
uracy: 0.1667
Epoch 10/50
14/14 [==============================] - 131s 9s/step - loss: 1.1736 - accuracy: 0.6690 - val_loss: 43.4120 - val_acc
uracy: 0.5000
Epoch 11/50
14/14 [==============================] - 128s 9s/step - loss: 1.0301 - accuracy: 0.6829 - val_loss: 53.1353 - val_acc
uracy: 0.1667
Epoch 12/50
14/14 [==============================] - 129s 9s/step - loss: 0.8919 - accuracy: 0.6759 - val_loss: 51.3644 - val_acc
uracy: 0.1667
Epoch 13/50
14/14 [==============================] - 129s 9s/step - loss: 0.8432 - accuracy: 0.7153 - val_loss: 49.7942 - val_acc
uracy: 0.1667
Epoch 14/50
14/14 [==============================] - 129s 9s/step - loss: 0.7262 - accuracy: 0.7685 - val_loss: 21.8442 - val_acc
uracy: 0.1944
Epoch 15/50
14/14 [==============================] - 130s 9s/step - loss: 0.8151 - accuracy: 0.7523 - val_loss: 26.8705 - val_acc
uracy: 0.1667
Epoch 16/50
14/14 [==============================] - 129s 9s/step - loss: 0.6369 - accuracy: 0.7662 - val_loss: 15.5927 - val_acc
uracy: 0.2315
Epoch 17/50
14/14 [==============================] - 130s 9s/step - loss: 0.6728 - accuracy: 0.7685 - val_loss: 35.5542 - val_acc
uracy: 0.1667
Epoch 18/50
14/14 [==============================] - 129s 9s/step - loss: 0.6492 - accuracy: 0.7801 - val_loss: 18.1785 - val_acc
uracy: 0.1759
Epoch 19/50
14/14 [==============================] - 129s 9s/step - loss: 0.5586 - accuracy: 0.8218 - val_loss: 21.4348 - val_acc
uracy: 0.1667
Epoch 20/50
14/14 [==============================] - 129s 9s/step - loss: 0.4737 - accuracy: 0.8426 - val_loss: 25.0764 - val_acc
uracy: 0.1667
Epoch 21/50
14/14 [==============================] - 129s 9s/step - loss: 0.5186 - accuracy: 0.8287 - val_loss: 26.3748 - val_acc
uracy: 0.1944
Epoch 22/50
14/14 [==============================] - 129s 9s/step - loss: 0.3535 - accuracy: 0.8750 - val_loss: 21.4976 - val_acc
uracy: 0.1759
Epoch 23/50
14/14 [==============================] - 129s 9s/step - loss: 0.4351 - accuracy: 0.8611 - val_loss: 16.9738 - val_acc
uracy: 0.2222
Epoch 24/50
14/14 [==============================] - 130s 9s/step - loss: 0.3842 - accuracy: 0.8704 - val_loss: 19.8801 - val_acc
uracy: 0.1667
Epoch 25/50
14/14 [==============================] - 129s 9s/step - loss: 0.3492 - accuracy: 0.8889 - val_loss: 27.9871 - val_acc
uracy: 0.1667
Epoch 26/50
14/14 [==============================] - 129s 9s/step - loss: 0.2648 - accuracy: 0.9097 - val_loss: 23.4832 - val_acc
uracy: 0.1667
Epoch 27/50
14/14 [==============================] - 129s 9s/step - loss: 0.3765 - accuracy: 0.8704 - val_loss: 19.6827 - val_acc
uracy: 0.1667
Epoch 28/50
14/14 [==============================] - 130s 9s/step - loss: 0.3372 - accuracy: 0.8750 - val_loss: 16.4502 - val_acc
uracy: 0.1759
Epoch 29/50
14/14 [==============================] - 130s 9s/step - loss: 0.3474 - accuracy: 0.8750 - val_loss: 14.3086 - val_acc
uracy: 0.2130
Epoch 30/50
14/14 [==============================] - 129s 9s/step - loss: 0.3082 - accuracy: 0.8889 - val_loss: 22.0569 - val_acc
uracy: 0.1667
```

```
Epoch 31/50
14/14 [==============================] - 129s 9s/step - loss: 0.2001 - accuracy: 0.9398 - val_loss: 15.7909 - val_acc
uracy: 0.1759
Epoch 32/50
14/14 [==============================] - 129s 9s/step - loss: 0.1853 - accuracy: 0.9352 - val_loss: 18.1949 - val_acc
uracy: 0.1667
Epoch 33/50
14/14 [==============================] - 130s 9s/step - loss: 0.2479 - accuracy: 0.9120 - val_loss: 17.0745 - val_acc
uracy: 0.1667
Epoch 34/50
14/14 [==============================] - 129s 9s/step - loss: 0.2318 - accuracy: 0.9236 - val_loss: 20.1353 - val_acc
uracy: 0.1667
Epoch 35/50
14/14 [==============================] - 129s 9s/step - loss: 0.2159 - accuracy: 0.9282 - val_loss: 12.8572 - val_acc
uracy: 0.2222
Epoch 36/50
14/14 [==============================] - 131s 9s/step - loss: 0.1974 - accuracy: 0.9236 - val_loss: 20.5982 - val_acc
uracy: 0.2315
Epoch 37/50
14/14 [==============================] - 130s 9s/step - loss: 0.1871 - accuracy: 0.9282 - val_loss: 15.4470 - val_acc
uracy: 0.1852
Epoch 38/50
14/14 [==============================] - 130s 9s/step - loss: 0.4398 - accuracy: 0.9051 - val_loss: 21.5899 - val_acc
uracy: 0.1759
Epoch 39/50
14/14 [==============================] - 132s 10s/step - loss: 0.1318 - accuracy: 0.9630 - val_loss: 13.6851 - val_ac
curacy: 0.2222
Epoch 40/50
14/14 [==============================] - 131s 9s/step - loss: 0.1362 - accuracy: 0.9468 - val_loss: 17.8450 - val_acc
uracy: 0.1944
Epoch 41/50
14/14 [==============================] - 130s 9s/step - loss: 0.1584 - accuracy: 0.9514 - val_loss: 18.9317 - val_acc
uracy: 0.1759
Epoch 42/50
14/14 [==============================] - 130s 9s/step - loss: 0.1540 - accuracy: 0.9421 - val_loss: 16.6727 - val_acc
uracy: 0.2037
Epoch 43/50
14/14 [==============================] - 130s 9s/step - loss: 0.0842 - accuracy: 0.9653 - val_loss: 12.6529 - val_acc
uracy: 0.2870
Epoch 44/50
14/14 [==============================] - 129s 9s/step - loss: 0.2735 - accuracy: 0.9259 - val_loss: 16.8285 - val_acc
uracy: 0.2685
Epoch 45/50
14/14 [==============================] - 129s 9s/step - loss: 0.2554 - accuracy: 0.9329 - val_loss: 16.9756 - val_acc
uracy: 0.2130
Epoch 46/50
14/14 [==============================] - 129s 9s/step - loss: 0.2009 - accuracy: 0.9398 - val_loss: 12.2837 - val_acc
uracy: 0.2685
Epoch 47/50
14/14 [==============================] - 129s 9s/step - loss: 0.0333 - accuracy: 0.9861 - val_loss: 11.3765 - val_acc
uracy: 0.2130
Epoch 48/50
14/14 [==============================] - 129s 9s/step - loss: 0.1526 - accuracy: 0.9560 - val_loss: 5.0745 - val_accu
racy: 0.3889
Epoch 49/50
14/14 [==============================] - 129s 9s/step - loss: 0.3412 - accuracy: 0.9329 - val_loss: 4.6086 - val_accu
racy: 0.4259
Epoch 50/50
14/14 [==============================] - 129s 9s/step - loss: 0.0426 - accuracy: 0.9884 - val_loss: 4.8796 - val_accu
racy: 0.4167
----------------Test accuracy for 3 fold----------------
Confusion Matrix :
 [[17  0  3 10]
 [ 0  7  1  2]
 [ 1  0  6  3]
 [ 2  1  2  5]]
Accuracy    :  0.5833333333333334
Specificity :  0.8294745484400656
Sensitivity :  0.5916666666666667
-----------------------------End of 3 Fold--------------------------
------------------------Start of 4 Fold------------------------
test images for normal class from 90 120
test images for cataract class from 30 40
test images for glaucoma class from 30 40
test images for retina disease class from 30 40
train_validation images for normal class from 0 to 90 and 120 to 300
train_validation images for cataract class from 0 to 30 and 40 to 100
train_validation  images for glaucoma class from 0 30 and 40 to 100
train_validation images for retina disease class from 0 to 30 and 40 to 100
model building and compiling for fold 4
Epoch 1/50
14/14 [==============================] - 132s 9s/step - loss: 19.9487 - accuracy: 0.3773 - val_loss: 56.0515 - val_ac
curacy: 0.1667
Epoch 2/50
14/14 [==============================] - 130s 9s/step - loss: 10.4136 - accuracy: 0.5069 - val_loss: 59.6108 - val_ac
curacy: 0.5000
Epoch 3/50
14/14 [==============================] - 130s 9s/step - loss: 7.2192 - accuracy: 0.5486 - val_loss: 42.8328 - val_acc
```

```
uracy: 0.2222
Epoch 4/50
14/14 [==============================] - 129s 9s/step - loss: 4.5782 - accuracy: 0.5139 - val_loss: 36.5421 - val_acc
uracy: 0.1019
Epoch 5/50
14/14 [==============================] - 129s 9s/step - loss: 3.4752 - accuracy: 0.5394 - val_loss: 64.9410 - val_acc
uracy: 0.1667
Epoch 6/50
14/14 [==============================] - 130s 9s/step - loss: 2.2201 - accuracy: 0.5579 - val_loss: 112.9586 - val_ac
curacy: 0.1667
Epoch 7/50
14/14 [==============================] - 129s 9s/step - loss: 2.1753 - accuracy: 0.5995 - val_loss: 36.8194 - val_acc
uracy: 0.3056
Epoch 8/50
14/14 [==============================] - 129s 9s/step - loss: 1.8793 - accuracy: 0.5486 - val_loss: 67.7545 - val_acc
uracy: 0.1667
Epoch 9/50
14/14 [==============================] - 129s 9s/step - loss: 1.3433 - accuracy: 0.6134 - val_loss: 53.6693 - val_acc
uracy: 0.1667
Epoch 10/50
14/14 [==============================] - 129s 9s/step - loss: 1.5150 - accuracy: 0.6343 - val_loss: 57.6387 - val_acc
uracy: 0.1667
Epoch 11/50
14/14 [==============================] - 129s 9s/step - loss: 1.3714 - accuracy: 0.6250 - val_loss: 45.3015 - val_acc
uracy: 0.1667
Epoch 12/50
14/14 [==============================] - 129s 9s/step - loss: 0.8553 - accuracy: 0.7037 - val_loss: 86.3025 - val_acc
uracy: 0.1667
Epoch 13/50
14/14 [==============================] - 130s 9s/step - loss: 0.8192 - accuracy: 0.7245 - val_loss: 62.3690 - val_acc
uracy: 0.1667
Epoch 14/50
14/14 [==============================] - 130s 9s/step - loss: 0.8323 - accuracy: 0.7454 - val_loss: 54.7759 - val_acc
uracy: 0.1667
Epoch 15/50
14/14 [==============================] - 130s 9s/step - loss: 0.8244 - accuracy: 0.7384 - val_loss: 62.8009 - val_acc
uracy: 0.1667
Epoch 16/50
14/14 [==============================] - 130s 9s/step - loss: 0.6679 - accuracy: 0.7616 - val_loss: 29.6254 - val_acc
uracy: 0.1667
Epoch 17/50
14/14 [==============================] - 129s 9s/step - loss: 0.7446 - accuracy: 0.7407 - val_loss: 31.9314 - val_acc
uracy: 0.1667
Epoch 18/50
14/14 [==============================] - 130s 9s/step - loss: 0.4738 - accuracy: 0.8287 - val_loss: 69.3954 - val_acc
uracy: 0.1667
Epoch 19/50
14/14 [==============================] - 129s 9s/step - loss: 0.5804 - accuracy: 0.7917 - val_loss: 26.7138 - val_acc
uracy: 0.1667
Epoch 20/50
14/14 [==============================] - 129s 9s/step - loss: 0.5550 - accuracy: 0.8171 - val_loss: 38.4196 - val_acc
uracy: 0.1667
Epoch 21/50
14/14 [==============================] - 129s 9s/step - loss: 0.4879 - accuracy: 0.8333 - val_loss: 19.8174 - val_acc
uracy: 0.1667
Epoch 22/50
14/14 [==============================] - 128s 9s/step - loss: 0.5065 - accuracy: 0.8218 - val_loss: 5.6819 - val_accu
racy: 0.4259
Epoch 23/50
14/14 [==============================] - 128s 9s/step - loss: 0.4480 - accuracy: 0.8611 - val_loss: 29.3656 - val_acc
uracy: 0.1667
Epoch 24/50
14/14 [==============================] - 129s 9s/step - loss: 0.4954 - accuracy: 0.8519 - val_loss: 40.5523 - val_acc
uracy: 0.1667
Epoch 25/50
14/14 [==============================] - 128s 9s/step - loss: 0.4893 - accuracy: 0.8426 - val_loss: 22.5466 - val_acc
uracy: 0.1944
Epoch 26/50
14/14 [==============================] - 121s 9s/step - loss: 0.3728 - accuracy: 0.8634 - val_loss: 16.6383 - val_acc
uracy: 0.1574
Epoch 27/50
14/14 [==============================] - 127s 9s/step - loss: 0.4282 - accuracy: 0.8565 - val_loss: 29.3892 - val_acc
uracy: 0.1667
Epoch 28/50
14/14 [==============================] - 125s 9s/step - loss: 0.4240 - accuracy: 0.8704 - val_loss: 54.1846 - val_acc
uracy: 0.1667
Epoch 29/50
14/14 [==============================] - 123s 9s/step - loss: 0.4484 - accuracy: 0.8611 - val_loss: 36.6850 - val_acc
uracy: 0.1667
Epoch 30/50
14/14 [==============================] - 121s 9s/step - loss: 0.4546 - accuracy: 0.8796 - val_loss: 30.8557 - val_acc
uracy: 0.2685
Epoch 31/50
14/14 [==============================] - 124s 9s/step - loss: 0.3456 - accuracy: 0.9144 - val_loss: 25.6985 - val_acc
uracy: 0.1852
Epoch 32/50
14/14 [==============================] - 118s 9s/step - loss: 0.2448 - accuracy: 0.9329 - val_loss: 26.5540 - val_acc
uracy: 0.2130
```

```
Epoch 33/50
14/14 [==============================] - 119s 9s/step - loss: 0.2004 - accuracy: 0.9375 - val_loss: 27.8413 - val_acc
uracy: 0.1944
Epoch 34/50
14/14 [==============================] - 119s 9s/step - loss: 0.2852 - accuracy: 0.9097 - val_loss: 21.8860 - val_acc
uracy: 0.1667
Epoch 35/50
14/14 [==============================] - 124s 9s/step - loss: 0.2545 - accuracy: 0.9259 - val_loss: 24.4507 - val_acc
uracy: 0.1667
Epoch 36/50
14/14 [==============================] - 115s 8s/step - loss: 0.2670 - accuracy: 0.9282 - val_loss: 17.4135 - val_acc
uracy: 0.1852
Epoch 37/50
14/14 [==============================] - 118s 8s/step - loss: 0.1891 - accuracy: 0.9306 - val_loss: 19.0491 - val_acc
uracy: 0.1667
Epoch 38/50
14/14 [==============================] - 119s 9s/step - loss: 0.2905 - accuracy: 0.9282 - val_loss: 18.0407 - val_acc
uracy: 0.2315
Epoch 39/50
14/14 [==============================] - 114s 8s/step - loss: 0.3009 - accuracy: 0.9259 - val_loss: 31.8809 - val_acc
uracy: 0.1667
Epoch 40/50
14/14 [==============================] - 123s 9s/step - loss: 0.2640 - accuracy: 0.9282 - val_loss: 24.9125 - val_acc
uracy: 0.1944
Epoch 41/50
14/14 [==============================] - 121s 9s/step - loss: 0.1348 - accuracy: 0.9537 - val_loss: 22.3907 - val_acc
uracy: 0.1667
Epoch 42/50
14/14 [==============================] - 128s 9s/step - loss: 0.1414 - accuracy: 0.9491 - val_loss: 17.0094 - val_acc
uracy: 0.1852
Epoch 43/50
14/14 [==============================] - 125s 9s/step - loss: 0.0883 - accuracy: 0.9699 - val_loss: 25.2902 - val_acc
uracy: 0.1759
Epoch 44/50
14/14 [==============================] - 128s 9s/step - loss: 0.2116 - accuracy: 0.9444 - val_loss: 21.3764 - val_acc
uracy: 0.2315
Epoch 45/50
14/14 [==============================] - 130s 9s/step - loss: 0.2573 - accuracy: 0.9259 - val_loss: 23.1816 - val_acc
uracy: 0.2130
Epoch 46/50
14/14 [==============================] - 129s 9s/step - loss: 0.2972 - accuracy: 0.9259 - val_loss: 17.4839 - val_acc
uracy: 0.2222
Epoch 47/50
14/14 [==============================] - 129s 9s/step - loss: 0.0513 - accuracy: 0.9769 - val_loss: 10.4678 - val_acc
uracy: 0.2407
Epoch 48/50
14/14 [==============================] - 129s 9s/step - loss: 0.0887 - accuracy: 0.9606 - val_loss: 6.0852 - val_accu
racy: 0.4259
Epoch 49/50
14/14 [==============================] - 130s 9s/step - loss: 0.1453 - accuracy: 0.9560 - val_loss: 18.9466 - val_acc
uracy: 0.5000
Epoch 50/50
14/14 [==============================] - 129s 9s/step - loss: 0.0769 - accuracy: 0.9722 - val_loss: 4.9469 - val_accu
racy: 0.5093
----------------Test accuracy for 4 fold----------------
Confusion Matrix :
 [[20  0  0 10]
 [ 1  5  1  3]
 [ 4  2  2  2]
 [ 3  1  0  6]]
Accuracy    :  0.55
Specificity :  0.7834701420890937
Sensitivity :  0.4916666666666666
-----------------------------End of 4 Fold-------------------------
------------------------Start of 5 Fold------------------------
test images for normal class from 120 150
test images for cataract class from 40 50
test images for glaucoma class from 40 50
test images for retina disease class from 40 50
train_validation images for normal class from 0 to 120 and 150 to 300
train_validation images for cataract class from 0 to 40 and 50 to 100
train_validation  images for glaucoma class from 0 40 and 50 to 100
train_validation images for retina disease class from 0 to 40 and 50 to 100
model building and compiling for fold 5
Epoch 1/50
14/14 [==============================] - 131s 9s/step - loss: 19.1024 - accuracy: 0.4421 - val_loss: 76.3194 - val_ac
curacy: 0.2407
Epoch 2/50
14/14 [==============================] - 129s 9s/step - loss: 12.1731 - accuracy: 0.4676 - val_loss: 44.1315 - val_ac
curacy: 0.1759
Epoch 3/50
14/14 [==============================] - 129s 9s/step - loss: 7.1245 - accuracy: 0.5602 - val_loss: 48.7694 - val_acc
uracy: 0.5000
Epoch 4/50
14/14 [==============================] - 130s 9s/step - loss: 5.5250 - accuracy: 0.5370 - val_loss: 81.8588 - val_acc
uracy: 0.1667
Epoch 5/50
14/14 [==============================] - 129s 9s/step - loss: 4.2050 - accuracy: 0.5255 - val_loss: 45.8915 - val_acc
```

```
uracy: 0.1667
Epoch 6/50
14/14 [==============================] - 130s 9s/step - loss: 2.5114 - accuracy: 0.5741 - val_loss: 46.3477 - val_acc
uracy: 0.1667
Epoch 7/50
14/14 [==============================] - 129s 9s/step - loss: 1.6744 - accuracy: 0.5463 - val_loss: 55.2291 - val_acc
uracy: 0.1667
Epoch 8/50
14/14 [==============================] - 130s 9s/step - loss: 1.3771 - accuracy: 0.6204 - val_loss: 64.9893 - val_acc
uracy: 0.1667
Epoch 9/50
14/14 [==============================] - 129s 9s/step - loss: 1.1414 - accuracy: 0.6574 - val_loss: 28.9510 - val_acc
uracy: 0.1667
Epoch 10/50
14/14 [==============================] - 129s 9s/step - loss: 1.5625 - accuracy: 0.5718 - val_loss: 75.2214 - val_acc
uracy: 0.1667
Epoch 11/50
14/14 [==============================] - 130s 9s/step - loss: 1.0045 - accuracy: 0.6852 - val_loss: 13.8591 - val_acc
uracy: 0.4074
Epoch 12/50
14/14 [==============================] - 129s 9s/step - loss: 0.9084 - accuracy: 0.6898 - val_loss: 12.6140 - val_acc
uracy: 0.1667
Epoch 13/50
14/14 [==============================] - 130s 9s/step - loss: 0.7633 - accuracy: 0.7338 - val_loss: 16.2906 - val_acc
uracy: 0.1667
Epoch 14/50
14/14 [==============================] - 129s 9s/step - loss: 0.8698 - accuracy: 0.6875 - val_loss: 20.3907 - val_acc
uracy: 0.1667
Epoch 15/50
14/14 [==============================] - 129s 9s/step - loss: 0.6855 - accuracy: 0.7454 - val_loss: 25.1888 - val_acc
uracy: 0.1667
Epoch 16/50
14/14 [==============================] - 129s 9s/step - loss: 0.8417 - accuracy: 0.7222 - val_loss: 7.3610 - val_accu
racy: 0.5093
Epoch 17/50
14/14 [==============================] - 129s 9s/step - loss: 0.5583 - accuracy: 0.7801 - val_loss: 3.8231 - val_accu
racy: 0.4907
Epoch 18/50
14/14 [==============================] - 131s 9s/step - loss: 0.6172 - accuracy: 0.7870 - val_loss: 9.8201 - val_accu
racy: 0.1667
Epoch 19/50
14/14 [==============================] - 129s 9s/step - loss: 0.5726 - accuracy: 0.8009 - val_loss: 4.8010 - val_accu
racy: 0.1944
Epoch 20/50
14/14 [==============================] - 130s 9s/step - loss: 0.5304 - accuracy: 0.8264 - val_loss: 3.2000 - val_accu
racy: 0.2500
Epoch 21/50
14/14 [==============================] - 129s 9s/step - loss: 0.4171 - accuracy: 0.8657 - val_loss: 5.2008 - val_accu
racy: 0.2685
Epoch 22/50
14/14 [==============================] - 131s 9s/step - loss: 0.4355 - accuracy: 0.8310 - val_loss: 2.2960 - val_accu
racy: 0.4259
Epoch 23/50
14/14 [==============================] - 129s 9s/step - loss: 0.3642 - accuracy: 0.8773 - val_loss: 1.8079 - val_accu
racy: 0.5463
Epoch 24/50
14/14 [==============================] - 129s 9s/step - loss: 0.4080 - accuracy: 0.8519 - val_loss: 3.5566 - val_accu
racy: 0.2963
Epoch 25/50
14/14 [==============================] - 131s 9s/step - loss: 0.3371 - accuracy: 0.8681 - val_loss: 3.0495 - val_accu
racy: 0.2685
Epoch 26/50
14/14 [==============================] - 130s 9s/step - loss: 0.2719 - accuracy: 0.9097 - val_loss: 3.7285 - val_accu
racy: 0.2778
Epoch 27/50
14/14 [==============================] - 130s 9s/step - loss: 0.3893 - accuracy: 0.8611 - val_loss: 5.2684 - val_accu
racy: 0.2963
Epoch 28/50
14/14 [==============================] - 129s 9s/step - loss: 0.3539 - accuracy: 0.8819 - val_loss: 7.5092 - val_accu
racy: 0.1944
Epoch 29/50
14/14 [==============================] - 129s 9s/step - loss: 0.3090 - accuracy: 0.9051 - val_loss: 6.8276 - val_accu
racy: 0.2778
Epoch 30/50
14/14 [==============================] - 132s 10s/step - loss: 0.3826 - accuracy: 0.8727 - val_loss: 5.3081 - val_acc
uracy: 0.2685
Epoch 31/50
14/14 [==============================] - 130s 9s/step - loss: 0.3620 - accuracy: 0.8657 - val_loss: 8.1349 - val_accu
racy: 0.2315
Epoch 32/50
14/14 [==============================] - 129s 9s/step - loss: 0.1663 - accuracy: 0.9375 - val_loss: 9.3898 - val_accu
racy: 0.1944
Epoch 33/50
14/14 [==============================] - 129s 9s/step - loss: 0.3719 - accuracy: 0.8981 - val_loss: 7.5733 - val_accu
racy: 0.3241
Epoch 34/50
14/14 [==============================] - 130s 9s/step - loss: 0.2424 - accuracy: 0.9074 - val_loss: 9.7610 - val_accu
racy: 0.2593
```

```
Epoch 35/50
14/14 [==============================] - 129s 9s/step - loss: 0.2590 - accuracy: 0.9444 - val_loss: 7.6043 - val_accu
racy: 0.2963
Epoch 36/50
14/14 [==============================] - 129s 9s/step - loss: 0.2786 - accuracy: 0.9074 - val_loss: 5.4886 - val_accu
racy: 0.2037
Epoch 37/50
14/14 [==============================] - 129s 9s/step - loss: 0.3365 - accuracy: 0.8981 - val_loss: 4.4866 - val_accu
racy: 0.3611
Epoch 38/50
14/14 [==============================] - 129s 9s/step - loss: 0.1451 - accuracy: 0.9514 - val_loss: 11.8666 - val_acc
uracy: 0.2130
Epoch 39/50
14/14 [==============================] - 130s 9s/step - loss: 0.2913 - accuracy: 0.9074 - val_loss: 4.9647 - val_accu
racy: 0.4352
Epoch 40/50
14/14 [==============================] - 132s 9s/step - loss: 0.2394 - accuracy: 0.9213 - val_loss: 3.2289 - val_accu
racy: 0.5185
Epoch 41/50
14/14 [==============================] - 131s 9s/step - loss: 0.2364 - accuracy: 0.9282 - val_loss: 2.5292 - val_accu
racy: 0.5278
Epoch 42/50
14/14 [==============================] - 129s 9s/step - loss: 0.1624 - accuracy: 0.9583 - val_loss: 6.8088 - val_accu
racy: 0.3241
Epoch 43/50
14/14 [==============================] - 129s 9s/step - loss: 0.1073 - accuracy: 0.9769 - val_loss: 4.0980 - val_accu
racy: 0.4074
Epoch 44/50
14/14 [==============================] - 130s 9s/step - loss: 0.1829 - accuracy: 0.9606 - val_loss: 4.6882 - val_accu
racy: 0.3981
Epoch 45/50
14/14 [==============================] - 131s 9s/step - loss: 0.3989 - accuracy: 0.8981 - val_loss: 6.3932 - val_accu
racy: 0.3796
Epoch 46/50
14/14 [==============================] - 129s 9s/step - loss: 0.1153 - accuracy: 0.9537 - val_loss: 4.2665 - val_accu
racy: 0.5370
Epoch 47/50
14/14 [==============================] - 129s 9s/step - loss: 0.2666 - accuracy: 0.9421 - val_loss: 7.3023 - val_accu
racy: 0.2963
Epoch 48/50
14/14 [==============================] - 130s 9s/step - loss: 0.1058 - accuracy: 0.9722 - val_loss: 9.7769 - val_accu
racy: 0.2963
Epoch 49/50
14/14 [==============================] - 129s 9s/step - loss: 0.0886 - accuracy: 0.9792 - val_loss: 5.6769 - val_accu
racy: 0.4352
Epoch 50/50
14/14 [==============================] - 130s 9s/step - loss: 0.0906 - accuracy: 0.9653 - val_loss: 6.5316 - val_accu
racy: 0.3611
WARNING:tensorflow:5 out of the last 9 calls to <function Model.make_predict_function.<locals>.predict_function at 0x
0000023CBA5DE5E0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be
due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python
objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has e
xperimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), plea
se refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/p
ython/tf/function for  more details.
----------------Test accuracy for 5 fold----------------
Confusion Matrix :
 [[11  0  0 19]
 [ 0  8  0  2]
 [ 1  0  6  3]
 [ 3  1  0  6]]
Accuracy    :  0.5166666666666667
Specificity :  0.82546768707483
Sensitivity :  0.5916666666666667
----------------------------End of 5 Fold-------------------------
```

## Test Evaluation Results

In [17]:
```
test_accuracy
```

Out[17]:
```
[0.4, 0.6166666666666667, 0.5833333333333334, 0.55, 0.5166666666666667]
```

In [18]:
```
mean_test_accuracy=np.mean(test_accuracy)
mean_test_accuracy
```

Out[18]:
```
0.5333333333333334
```

In [19]:
```
test_sensitivity
```

Out[19]:
```
[0.30000000000000004,
 0.5083333333333333,
 0.5916666666666667,
 0.4916666666666666,
 0.5916666666666667]
```

In [20]:
```
mean_test_sensitivity= np.mean(test_sensitivity)
mean_test_sensitivity
```
Out[20]:
```
0.4966666666666667
```

In [21]:
```
test_specificity
```
Out[21]:
```
[0.7006302521008404,
 0.8098473708229805,
 0.8294745484400656,
 0.7834701420890937,
 0.82546768707483]
```

In [22]:
```
mean_test_specificity= np.mean(test_specificity)
mean_test_specificity
```
Out[22]:
```
0.7897780001055621
```

## Training and Validation Evaluation Results

In [23]:
```
train_acc
```
Out[23]:
```
array([0.43287036, 0.5324074 , 0.5486111 , 0.5462963 , 0.55787039,
       0.56712961, 0.62037039, 0.65046299, 0.61805558, 0.6574074 ,
       0.6550926 , 0.7037037 , 0.6875    , 0.7175926 , 0.73148149,
       0.76157409, 0.76851851, 0.79166669, 0.8263889 , 0.78703701,
       0.78472221, 0.88425928, 0.7986111 , 0.81481481, 0.86342591,
       0.8888889 , 0.83796299, 0.8587963 , 0.8425926 , 0.89351851,
       0.93518519, 0.9074074 , 0.88657409, 0.87731481, 0.9236111 ,
       0.92592591, 0.87962961, 0.91666669, 0.94444442, 0.9212963 ,
       0.9375    , 0.9212963 , 0.94212961, 0.94907409, 0.91435188,
       0.93981481, 0.9537037 , 0.94675928, 0.9699074 , 0.92592591,
       0.35416666, 0.49305555, 0.52777779, 0.53703701, 0.5763889 ,
       0.5763889 , 0.5949074 , 0.61342591, 0.60648149, 0.66435188,
       0.66666669, 0.67592591, 0.7013889 , 0.72222221, 0.73842591,
       0.74074072, 0.73148149, 0.83101851, 0.83101851, 0.80555558,
       0.84027779, 0.8125    , 0.82175928, 0.88657409, 0.87037039,
       0.86574072, 0.90972221, 0.91435188, 0.91203701, 0.88425928,
       0.91666669, 0.87962961, 0.8912037 , 0.94212961, 0.9375    ,
       0.92824072, 0.95833331, 0.9537037 , 0.94444442, 0.93981481,
       0.9050926 , 0.9513889 , 0.9513889 , 0.9537037 , 0.93981481,
       0.9212963 , 0.97685188, 0.98842591, 0.94907409, 0.9699074 ,
       0.41666666, 0.52546299, 0.5324074 , 0.56944442, 0.5162037 ,
       0.56018519, 0.6111111 , 0.58564812, 0.63194442, 0.66898149,
       0.68287039, 0.67592591, 0.71527779, 0.76851851, 0.75231481,
       0.7662037 , 0.76851851, 0.7800926 , 0.82175928, 0.8425926 ,
       0.8287037 , 0.875     , 0.8611111 , 0.87037039, 0.8888889 ,
       0.90972221, 0.87037039, 0.875     , 0.875     , 0.8888889 ,
       0.93981481, 0.93518519, 0.91203701, 0.9236111 , 0.92824072,
       0.9236111 , 0.92824072, 0.9050926 , 0.96296299, 0.94675928,
       0.9513889 , 0.94212961, 0.96527779, 0.92592591, 0.93287039,
       0.93981481, 0.9861111 , 0.95601851, 0.93287039, 0.98842591,
       0.37731481, 0.50694442, 0.5486111 , 0.5138889 , 0.53935188,
       0.55787039, 0.59953701, 0.5486111 , 0.61342591, 0.63425928,
       0.625     , 0.7037037 , 0.72453701, 0.74537039, 0.73842591,
       0.76157409, 0.74074072, 0.8287037 , 0.79166669, 0.81712961,
       0.83333331, 0.82175928, 0.8611111 , 0.85185188, 0.8425926 ,
       0.86342591, 0.85648149, 0.87037039, 0.8611111 , 0.87962961,
       0.91435188, 0.93287039, 0.9375    , 0.90972221, 0.92592591,
       0.92824072, 0.93055558, 0.92824072, 0.92592591, 0.92824072,
       0.9537037 , 0.94907409, 0.9699074 , 0.94444442, 0.92592591,
       0.92592591, 0.97685188, 0.96064812, 0.95601851, 0.97222221,
       0.44212964, 0.4675926 , 0.56018519, 0.53703701, 0.52546299,
       0.57407409, 0.5462963 , 0.62037039, 0.6574074 , 0.57175928,
       0.68518519, 0.68981481, 0.7337963 , 0.6875    , 0.74537039,
       0.72222221, 0.7800926 , 0.78703701, 0.80092591, 0.8263889 ,
       0.86574072, 0.83101851, 0.87731481, 0.85185188, 0.86805558,
       0.90972221, 0.8611111 , 0.88194442, 0.9050926 , 0.87268519,
       0.86574072, 0.9375    , 0.89814812, 0.9074074 , 0.94444442,
       0.9074074 , 0.89814812, 0.9513889 , 0.9074074 , 0.9212963 ,
       0.92824072, 0.95833331, 0.97685188, 0.96064812, 0.89814812,
       0.9537037 , 0.94212961, 0.97222221, 0.97916669, 0.96527779])
```

In [24]:
```
mean_train_accuracy=np.mean(train_acc)
mean_train_accuracy
```
Out[24]:
```
0.8081111098527909
```

In [25]:
```
val_acc
```

Out[25]:
```
array([0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.2037037 , 0.2037037 ,
       0.16666667, 0.16666667, 0.16666667, 0.23148148, 0.21296297,
       0.2037037 , 0.16666667, 0.22222222, 0.18518518, 0.21296297,
       0.22222222, 0.30555555, 0.44444445, 0.5462963 , 0.44444445,
       0.16666667, 0.5       , 0.5       , 0.5       , 0.5       ,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.5       , 0.16666667, 0.5       , 0.25      , 0.49074075,
       0.48148149, 0.3611111 , 0.5       , 0.35185185, 0.43518519,
       0.25925925, 0.25      , 0.16666667, 0.16666667, 0.16666667,
       0.19444445, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.18518518, 0.17592593, 0.21296297,
       0.22222222, 0.25925925, 0.23148148, 0.2962963 , 0.3888889 ,
       0.28703704, 0.3425926 , 0.5       , 0.55555558, 0.39814815,
       0.35185185, 0.44444445, 0.5       , 0.57407409, 0.5462963 ,
       0.16666667, 0.4537037 , 0.16666667, 0.5       , 0.5       ,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.5       ,
       0.16666667, 0.16666667, 0.16666667, 0.19444445, 0.16666667,
       0.23148148, 0.16666667, 0.17592593, 0.16666667, 0.16666667,
       0.19444445, 0.17592593, 0.22222222, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.17592593, 0.21296297, 0.16666667,
       0.17592593, 0.16666667, 0.16666667, 0.16666667, 0.22222222,
       0.23148148, 0.18518518, 0.17592593, 0.22222222, 0.19444445,
       0.17592593, 0.2037037 , 0.28703704, 0.26851851, 0.21296297,
       0.26851851, 0.21296297, 0.3888889 , 0.42592594, 0.41666666,
       0.16666667, 0.5       , 0.22222222, 0.10185185, 0.16666667,
       0.16666667, 0.30555555, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.16666667, 0.42592594, 0.16666667, 0.16666667, 0.19444445,
       0.1574074 , 0.16666667, 0.16666667, 0.16666667, 0.26851851,
       0.18518518, 0.21296297, 0.19444445, 0.16666667, 0.16666667,
       0.18518518, 0.16666667, 0.23148148, 0.16666667, 0.19444445,
       0.16666667, 0.18518518, 0.17592593, 0.23148148, 0.21296297,
       0.22222222, 0.24074075, 0.42592594, 0.5       , 0.50925928,
       0.24074075, 0.17592593, 0.5       , 0.16666667, 0.16666667,
       0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.4074074 , 0.16666667, 0.16666667, 0.16666667, 0.16666667,
       0.50925928, 0.49074075, 0.16666667, 0.19444445, 0.25      ,
       0.26851851, 0.42592594, 0.5462963 , 0.2962963 , 0.26851851,
       0.27777779, 0.2962963 , 0.19444445, 0.27777779, 0.26851851,
       0.23148148, 0.19444445, 0.32407406, 0.25925925, 0.2962963 ,
       0.2037037 , 0.3611111 , 0.21296297, 0.43518519, 0.51851851,
       0.52777779, 0.32407406, 0.4074074 , 0.39814815, 0.37962964,
       0.53703701, 0.2962963 , 0.2962963 , 0.43518519, 0.3611111 ])
```

In [26]:
```
mean_val_accuracy=np.mean(val_acc)
mean_val_accuracy
```

Out[26]:
```
0.24918518805503845
```

In [27]:
```
train_loss
```

Out[27]:
```
array([17.16884613, 10.26230907,  6.51222944,  4.81271744,  3.09632754,
        2.4621551 ,  1.9442625 ,  1.43246734,  1.90623188,  1.27773678,
        1.07394147,  0.87130767,  1.04937458,  1.09441841,  1.07344973,
        0.8890729 ,  0.71756178,  0.61275244,  0.56879395,  0.71044892,
        0.61590344,  0.31707865,  0.65379256,  0.55595618,  0.55217272,
        0.33930972,  0.44880334,  0.38256741,  0.43406156,  0.31948191,
        0.21025078,  0.29787585,  0.390434  ,  0.3696712 ,  0.18193397,
        0.25177252,  0.46901476,  0.24745643,  0.15675326,  0.27219316,
        0.18059446,  0.26041675,  0.1508359 ,  0.14051768,  0.23576857,
        0.23207729,  0.14787097,  0.22228053,  0.10054346,  0.22472632,
       22.76387596,  8.1996994 ,  7.68244505,  6.01120377,  3.09999633,
        2.97862554,  2.00866818,  1.59761965,  1.52111614,  0.8961941 ,
        0.92433751,  1.01203024,  0.94809061,  0.82211614,  1.05137146,
        0.73955929,  0.67535377,  0.51757038,  0.51620591,  0.55500889,
        0.45272636,  0.46744645,  0.49319854,  0.29376107,  0.38909706,
        0.40577152,  0.24038051,  0.25625777,  0.26776031,  0.37543491,
        0.23785429,  0.47213891,  0.31022489,  0.17538825,  0.25233066,
        0.23071407,  0.10260782,  0.15540059,  0.16086183,  0.182896  ,
        0.31984267,  0.18301515,  0.17368403,  0.14937259,  0.18370038,
        0.26381287,  0.08296758,  0.0483087 ,  0.13836065,  0.07896914,
       20.32651711,  9.19863129,  7.45822573,  4.39966488,  4.03915071,
        2.58029175,  1.61296761,  1.75261068,  1.1657083 ,  1.17358613,
        1.03008831,  0.89191705,  0.84323716,  0.72622257,  0.81511754,
        0.63694912,  0.67281127,  0.64918107,  0.55855876,  0.47365788,
        0.51864117,  0.3535006 ,  0.43510062,  0.38417658,  0.34921509,
        0.26478714,  0.37654713,  0.33723545,  0.34737983,  0.30819696,
        0.20010702,  0.18529521,  0.24792963,  0.23180866,  0.2159476 ,
        0.19742577,  0.18708155,  0.4398087 ,  0.13175289,  0.13616197,
        0.15843354,  0.15398848,  0.08415846,  0.27346274,  0.25536424,
        0.20090877,  0.03330778,  0.15262681,  0.34118676,  0.04263935,
       19.94869614, 10.41363716,  7.21924639,  4.57822132,  3.47517109,
        2.22008538,  2.17525578,  1.87929678,  1.34327686,  1.51499128,
        1.37136137,  0.85528529,  0.81921756,  0.83230919,  0.82439351,
        0.66785359,  0.74464577,  0.47376588,  0.58036602,  0.55496383,
        0.48793986,  0.50651479,  0.44801405,  0.49543929,  0.48931751,
        0.3727937 ,  0.4282375 ,  0.42403385,  0.44839996,  0.45464054,
        0.34559458,  0.24484561,  0.20036827,  0.28518921,  0.25453803,
        0.26704907,  0.18908551,  0.29052088,  0.30089927,  0.26400042,
        0.13483807,  0.1414071 ,  0.08827762,  0.2115733 ,  0.25731349,
        0.29715797,  0.05126875,  0.08869969,  0.14533363,  0.07690968,
       19.10235596, 12.17309666,  7.12447786,  5.5249629 ,  4.20496845,
        2.5113554 ,  1.67438686,  1.37707686,  1.14140785,  1.56252313,
        1.00452757,  0.90842932,  0.7632907 ,  0.8697747 ,  0.68549514,
        0.8417443 ,  0.55832529,  0.61716419,  0.57261592,  0.53044814,
        0.417083  ,  0.43549296,  0.36424136,  0.40795749,  0.33712858,
        0.27185237,  0.38930452,  0.35393411,  0.30898187,  0.38259915,
        0.36204565,  0.16626227,  0.37194729,  0.24235004,  0.25903141,
        0.27855581,  0.33647382,  0.14507879,  0.29130992,  0.23941849,
        0.23640192,  0.16242382,  0.10734586,  0.18293571,  0.39888787,
        0.11533704,  0.26663193,  0.10575551,  0.08858291,  0.09059335])
```

In [28]:
```
mean_train_loss=np.mean(train_loss)
mean_train_loss
```

Out[28]:
```
1.4102391167730093
```

In [29]:
```
val_loss
```

Out[29]:
```
array([196.7006073 , 182.025177  ,  80.91713715,  89.64961243,
        92.58319855, 126.0164566 ,  95.89859772,  60.46477127,
        45.07113647,  34.65521622,  23.9937706 ,  28.15537453,
        30.53067589,  41.38500214,  48.4694519 ,  42.02339554,
        74.61327362,  47.94384766,  38.3707428 ,  33.02332306,
        29.16703224,  20.95206642,  21.17567635,  29.30384254,
        28.30862617,   7.73365974,  25.29241943, 291.1322937 ,
        20.81303596,  24.30821419,  26.85552597,  32.61240005,
        27.59710693,  15.74124813,  16.87287521,  17.85619354,
        13.64843845,  19.74687576,   8.97575283,  18.1675396 ,
         5.34323311,  23.0668602 ,  17.0223217 ,  16.54026222,
        10.8756361 ,   9.76104546,   8.79319191,   5.41357422,
         4.84164858,   4.85939884,  73.10952759,  62.4114418 ,
        40.78161621,  72.64131165,  29.93701553, 106.9048233 ,
        85.55125427,  44.14822388,  71.54429626,  53.12083435,
        24.70445251,  30.83588791,  20.37410927,  18.32690811,
        17.8523941 ,  17.9019928 ,  10.01793098,  22.59142685,
        13.08829594,  12.74496269,  11.24125481,  11.96821213,
        19.08836174,  22.31653595,  22.85772896,  10.77473164,
        17.99071312,  25.23149109,  22.2630043 ,  21.22958183,
        30.0420208 ,  12.4836607 ,  15.37508488,  14.21842766,
        13.10617924,  11.56291008,  11.27408981,  15.80828285,
         6.90118408,   8.29402065,   6.92682123,   5.00111103,
         5.61814833,   6.48097277,   6.49277973,   7.10568953,
         7.19764519,   5.9421134 ,  10.66146183,   5.46029663,
       186.27786255,  29.41801453,  88.0117569 ,  90.72203827,
        68.43704987,  69.98561096, 155.08978271,  45.90680695,
        65.89796448,  43.41199112,  53.13532639,  51.36444473,
        49.79423904,  21.84422302,  26.87053108,  15.59266758,
        35.55423355,  18.17847443,  21.43475723,  25.07643318,
        26.37480736,  21.49757576,  16.97380066,  19.88010979,
        27.98708153,  23.48323631,  19.68272591,  16.45022583,
        14.30860138,  22.05692101,  15.790905  ,  18.19487572,
        17.07453346,  20.13529015,  12.8572216 ,  20.59818268,
        15.44700527,  21.58992004,  13.68514633,  17.84495544,
        18.93167877,  16.67273712,  12.65290546,  16.82845688,
        16.97555351,  12.28371048,  11.37647343,   5.07447529,
         4.60864735,   4.87961626,  56.05149841,  59.61083603,
        42.83281326,  36.54211426,  64.94100189, 112.95860291,
        36.81937027,  67.75453186,  53.66930771,  57.63870621,
        45.30149841,  86.30254364,  62.36903763,  54.77590179,
        62.80085373,  29.62540054,  31.93139267,  69.39536285,
        26.71376801,  38.41957092,  19.81740379,   5.68188858,
        29.36557961,  40.55231094,  22.54656029,  16.63834381,
        29.38924217,  54.18463516,  36.6849823 ,  30.85567665,
        25.698452  ,  26.55399513,  27.84131813,  21.88601303,
        24.45067024,  17.41348839,  19.04910851,  18.04071426,
        31.88088799,  24.91248703,  22.39069366,  17.00941467,
        25.29018402,  21.37638283,  23.18158722,  17.48385429,
        10.46775627,   6.0851922 ,  18.94661522,   4.94686031,
        76.3193512 ,  44.13154984,  48.76940155,  81.85879517,
        45.89152145,  46.34767532,  55.22911072,  64.98926544,
        28.95098686,  75.2213974 ,  13.85912037,  12.614048  ,
        16.29063988,  20.3907032 ,  25.18883133,   7.36099911,
         3.82306862,   9.82007694,   4.80102348,   3.20002294,
         5.20075893,   2.29600906,   1.80786884,   3.55663347,
         3.04946542,   3.72845936,   5.2683835 ,   7.50917578,
         6.82759619,   5.30814743,   8.13490963,   9.3897934 ,
         7.57334614,   9.76101303,   7.60430241,   5.48855019,
         4.48660803,  11.86657619,   4.96469307,   3.2288847 ,
         2.52922964,   6.8087616 ,   4.09799242,   4.68821001,
         6.39323235,   4.26648331,   7.30228043,   9.77688503,
         5.67690468,   6.53164148])
```

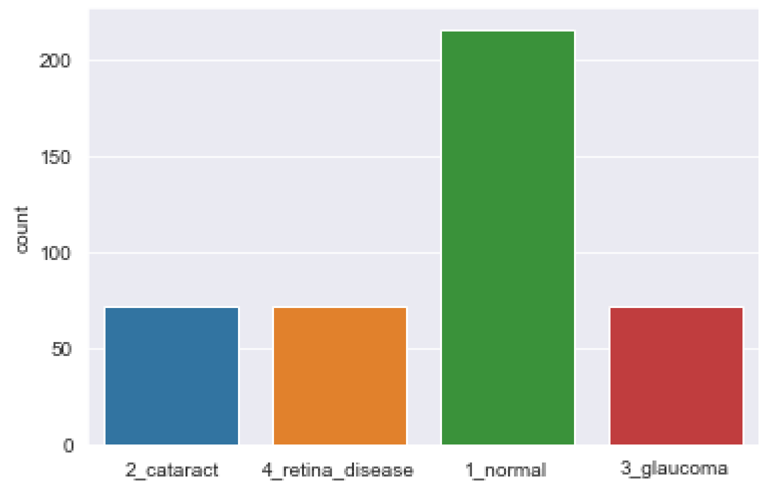
In [30]:
```
mean_val_loss=np.mean(val_loss)
mean_val_loss
```

Out[30]:
```
30.984921466350556
```

## Plot to Visualize the Number of Images in Each Label of Trainig Dataset

In [31]:
```python
l = []
for i in train:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l)
```

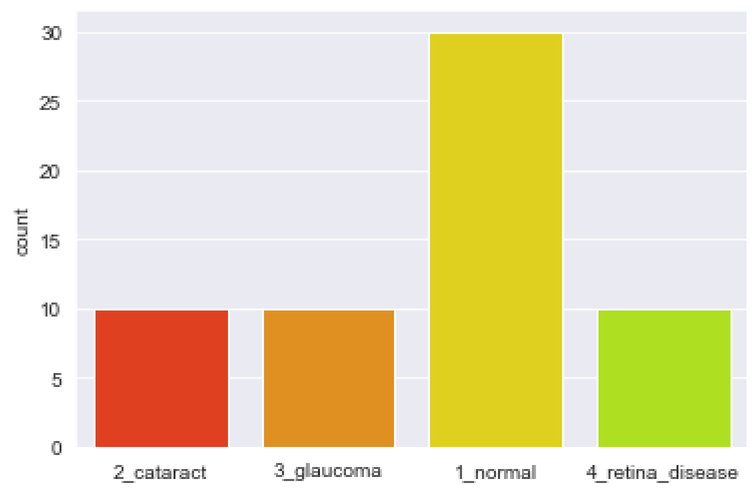
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x23caa311fa0>




## Plot to Visualize the Number of Images in Each Label of Test Dataset.

In [32]:
```python
l = []
for i in test:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l,palette='prism')
```

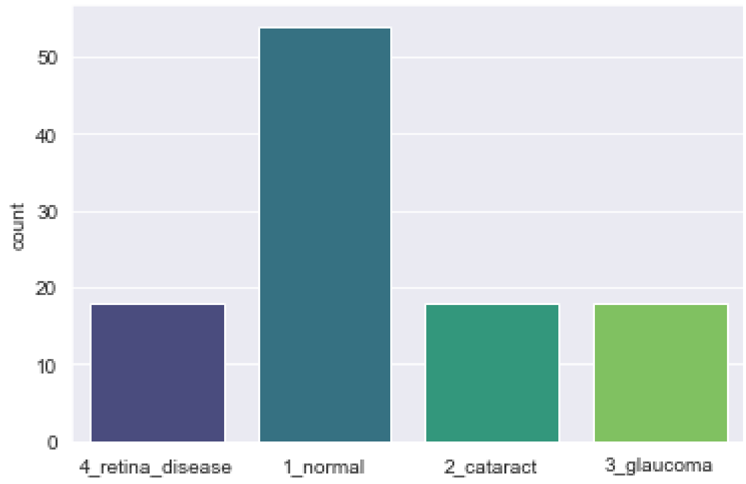
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x23cb01e6f70>




## Plot to Visualize the Number of Images in Each Label of Validation Dataset.

In [33]:
```python
l = []
for i in validation:
    if(i[1] == 0):
        l.append("1_normal")

    elif (i[1] == 1):
        l.append("2_cataract")

    elif (i[1] == 2):
        l.append("3_glaucoma")

    else :
        l.append("4_retina_disease")


sns.set_style('darkgrid')
sns.countplot(l,palette='viridis')
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x23c9e0092b0>



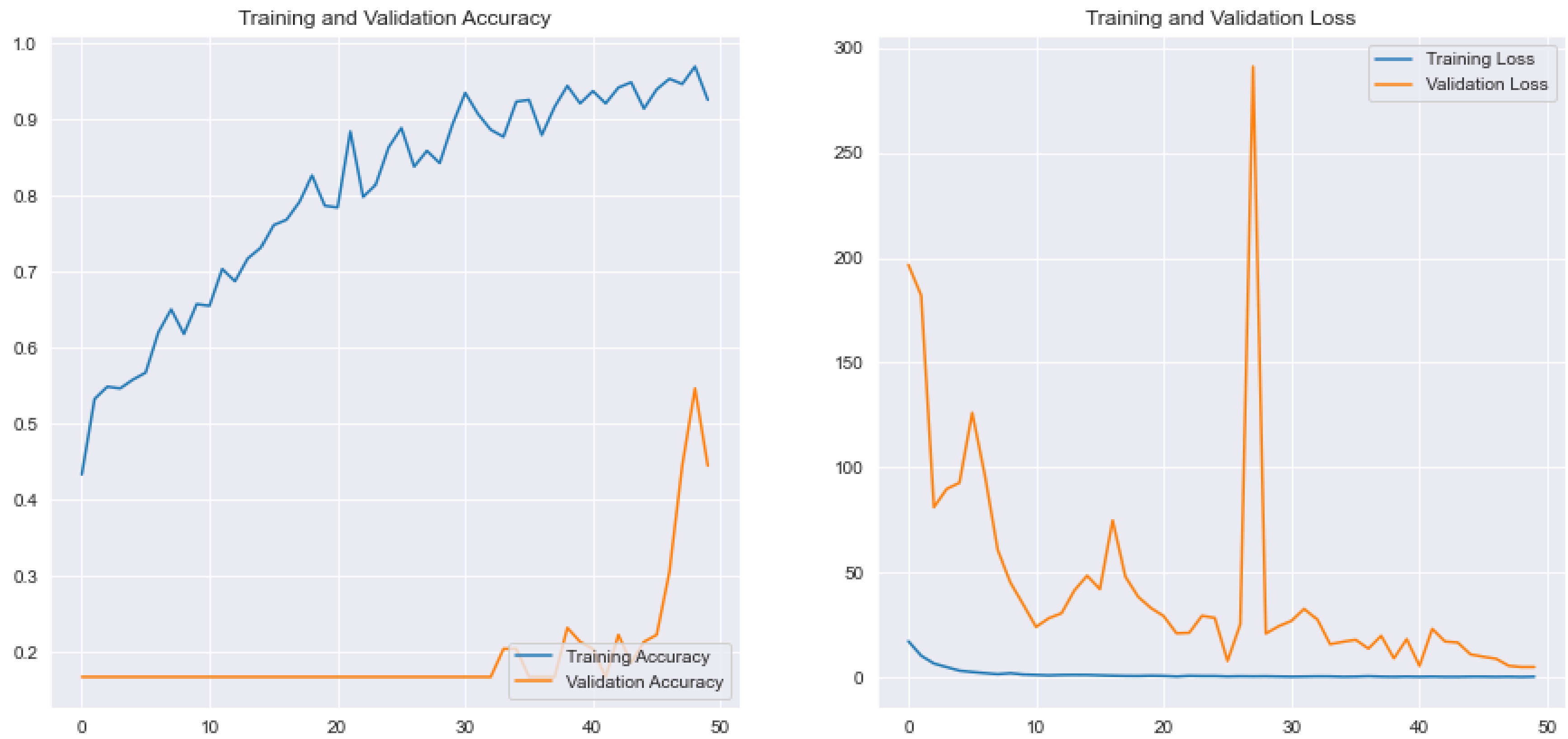

# Training,Validation Accuracy and Loss Plot for 50 Epochs

In [34]:
```python
def plot_print(i,j):
    epochs_range = range(50)

    plt.figure(figsize=(15, 15))
    plt.subplot(2, 2, 1)
    plt.plot(epochs_range, train_acc[i:j], label='Training Accuracy')
    plt.plot(epochs_range, val_acc[i:j], label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(2, 2, 2)
    plt.plot(epochs_range, train_loss[i:j], label='Training Loss')
    plt.plot(epochs_range, val_loss[i:j], label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    return plt.show()
```

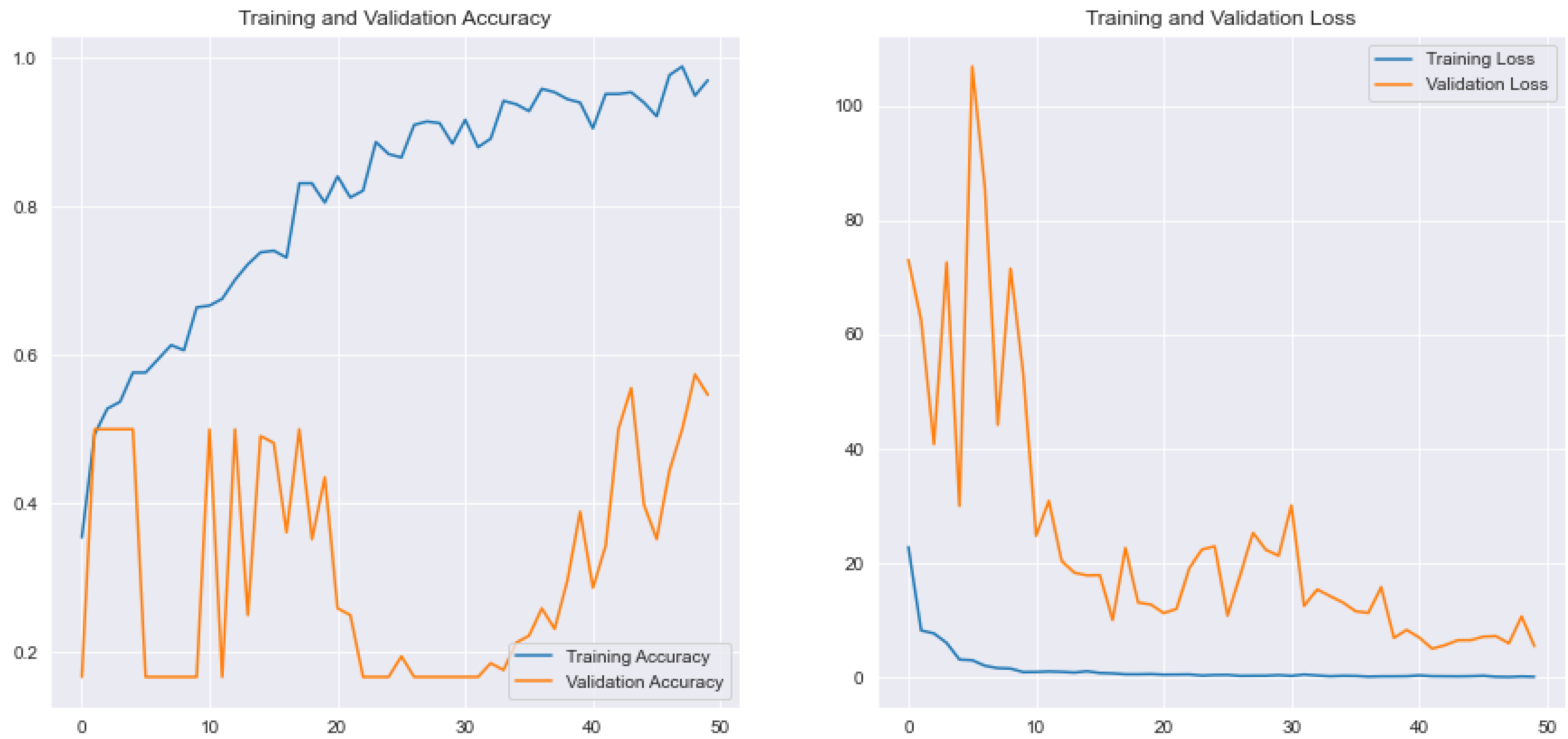
In [35]:

```
k=1
j=0
for i in range(0,250,50):
    j +=50
    print('Plot for ',k,'cross validation accuracy and loss for Training and Validation phase')
    k +=1
    plot_print(i,j)
```

Plot for  1 cross validation accuracy and loss for Training and Validation phase

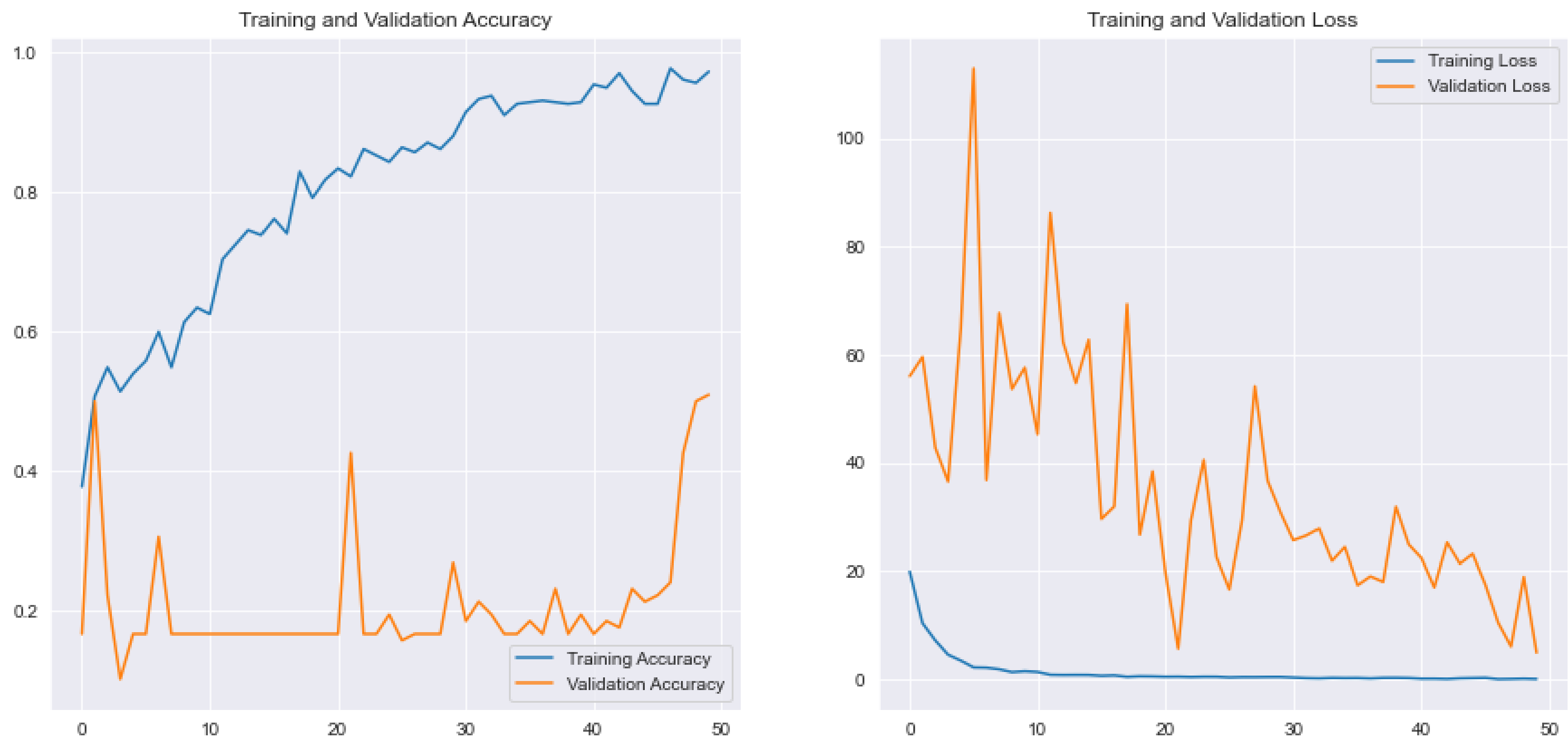



Plot for  2 cross validation accuracy and loss for Training and Validation phase




Plot for  3 cross validation accuracy and loss for Training and Validation phase


Plot for  4 cross validation accuracy and loss for Training and Validation phase




Plot for  5 cross validation accuracy and loss for Training and Validation phase


# Visualizing Confusion Matrix for Each Fold

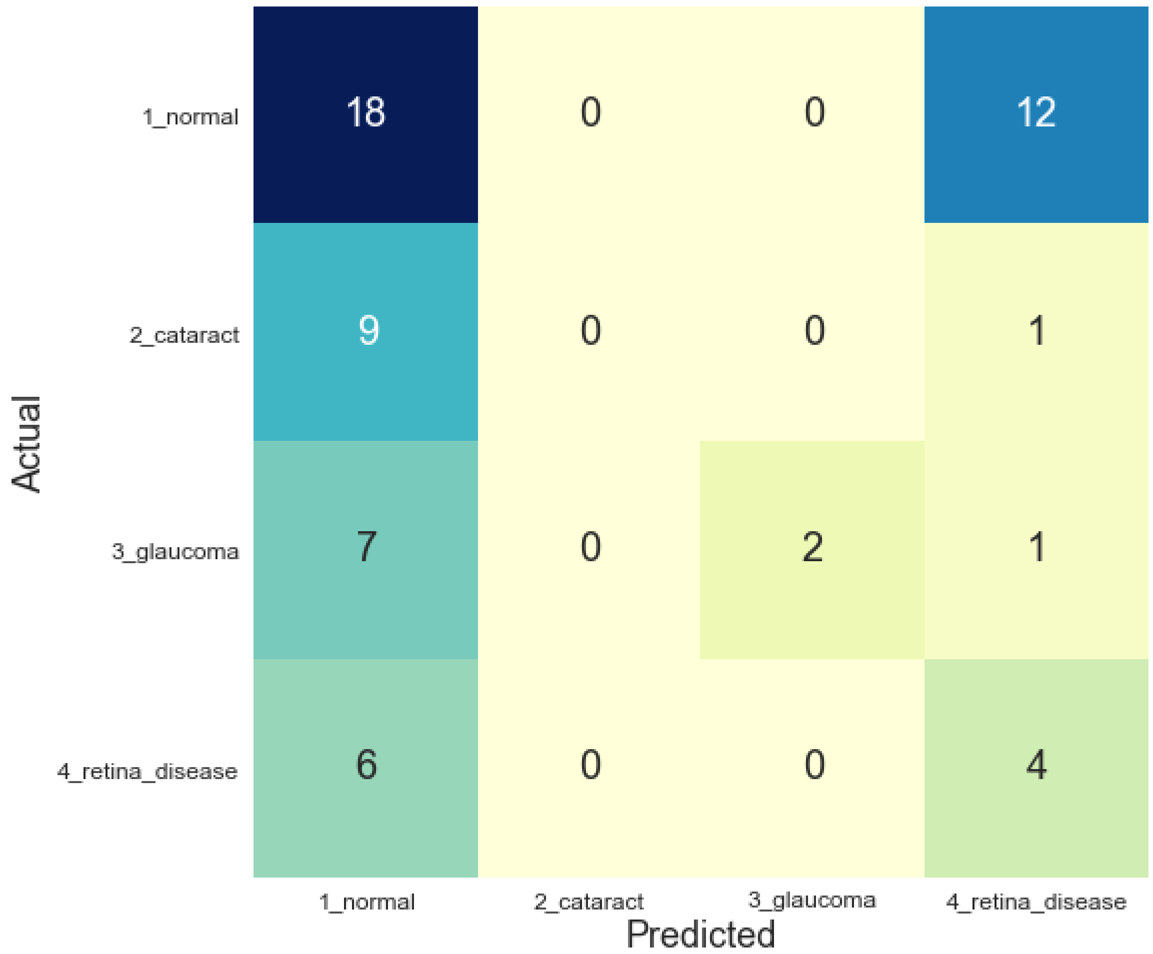
In [36]:
```
CM= np.array(CM)
CM.resize(5,4,4)
```

In [37]:
```
def confusionmatrix_vis(i):

        yticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
        xticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
        plt.figure(figsize=(8, 8))
        hm =sns.heatmap(CM[i], annot=True,annot_kws={"size": 20}, cbar=False,cmap="YlGnBu",yticklabels=yticklabels,xti
cklabels=xticklabels)

        hm.set_xticklabels(hm.get_xticklabels(), rotation=0, fontsize = 12, )
        hm.set_yticklabels(hm.get_yticklabels(), rotation=0, fontsize = 12)

        plt.ylabel("Actual", fontsize = 18)
        plt.xlabel("Predicted",fontsize = 18)

        return plt.show()
```

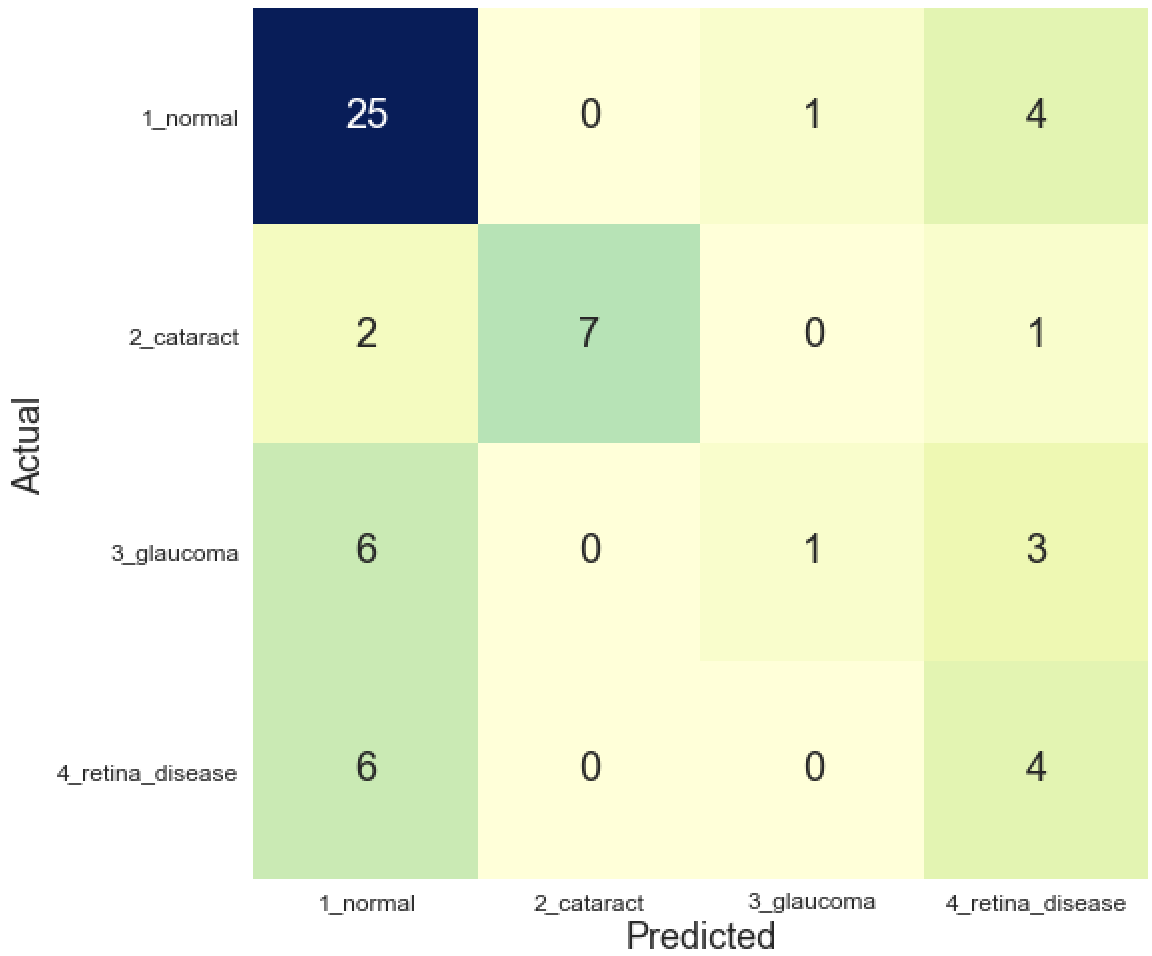
In [38]:

```
k=1
for i in range(5):
    print('Confusion Matrix for ',k,'Cross Validation Test phase')
    k +=1
    confusionmatrix_vis(i)
```

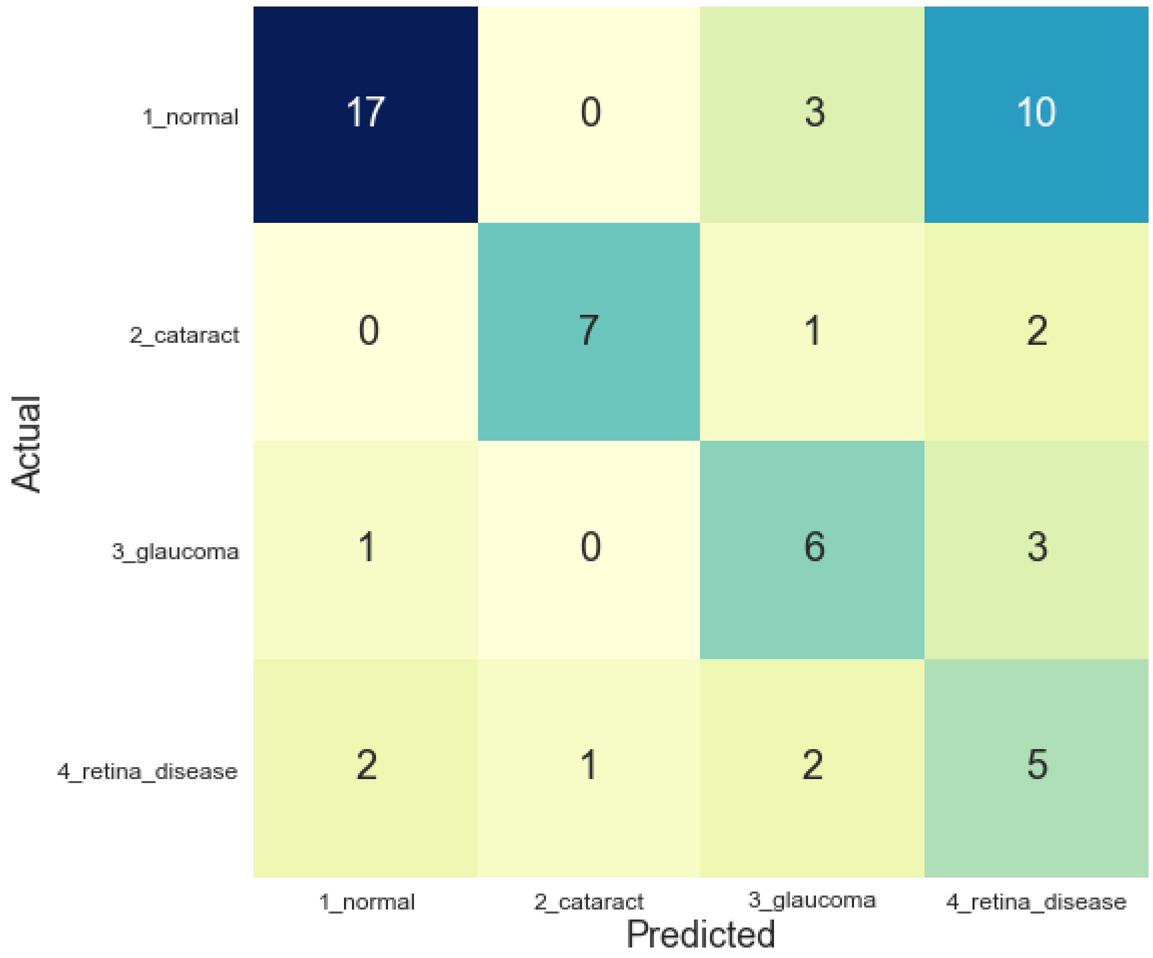
Confusion Matrix for  1 Cross Validation Test phase



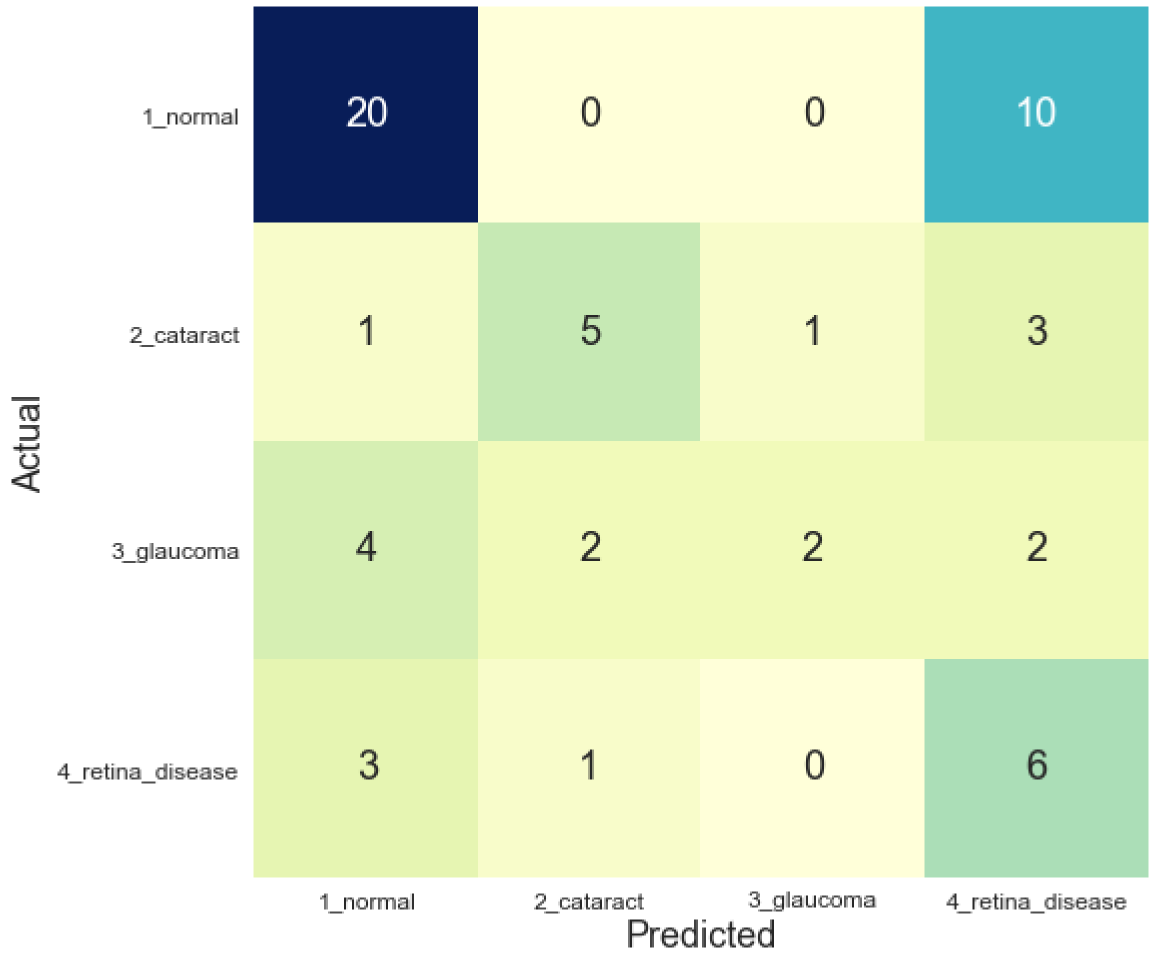

Confusion Matrix for  2 Cross Validation Test phase



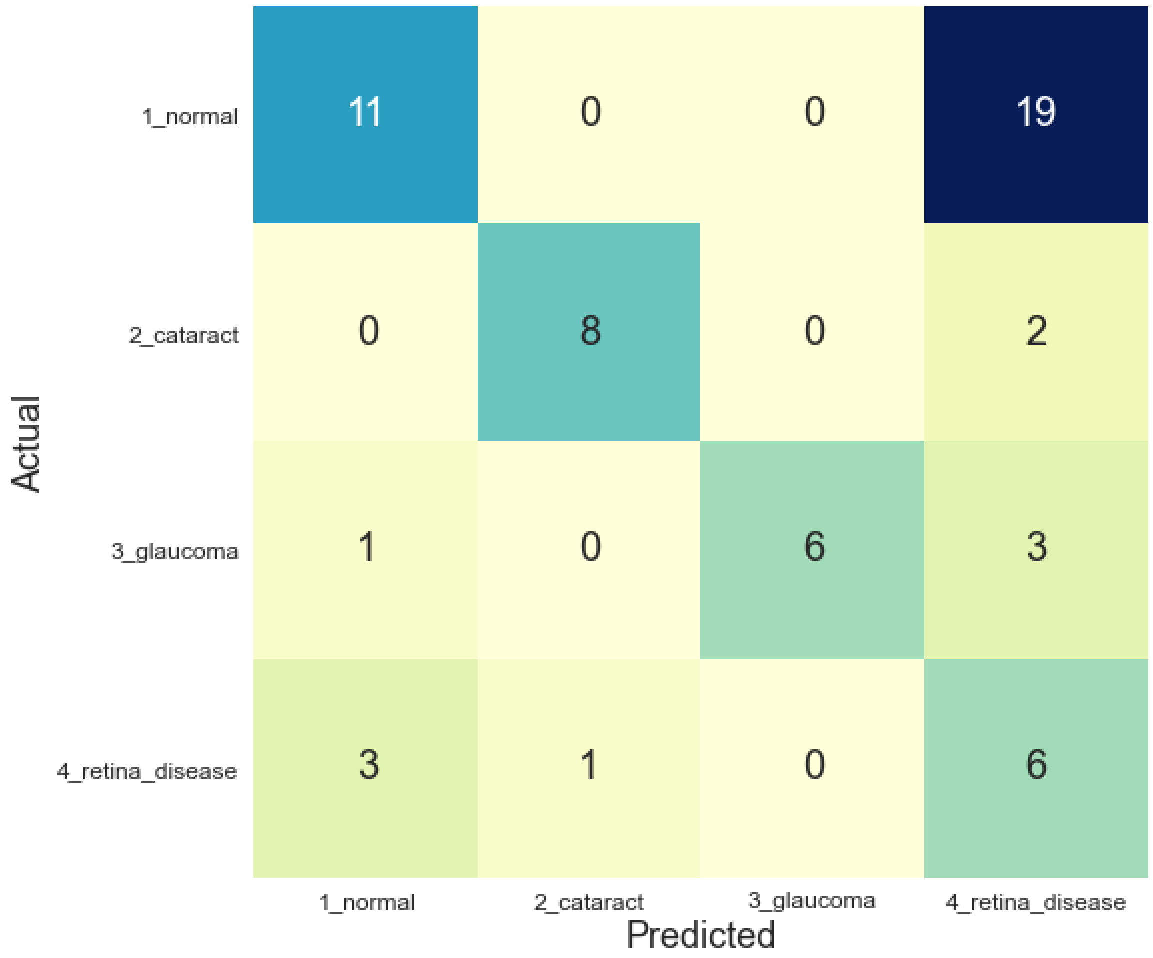

Confusion Matrix for  3 Cross Validation Test phase




```
Confusion Matrix for  4 Cross Validation Test phase
```




```
Confusion Matrix for  5 Cross Validation Test phase
```

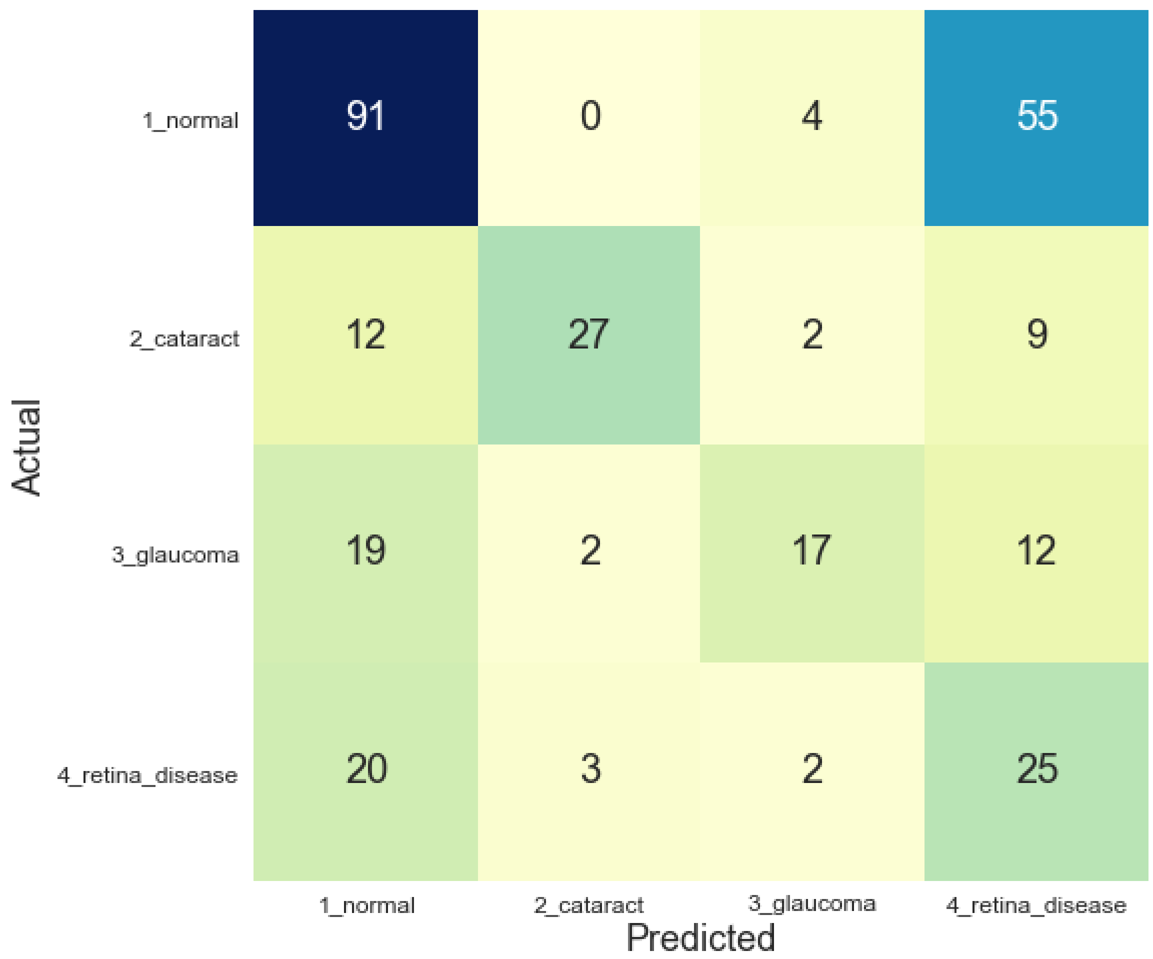



## Visualizing Summarized Confusion Matrix of all 5 folds

In [39]:
```
CM_sum = CM[0]+CM[1]+CM[2]+CM[3]+CM[4]
CM_sum
```

Out[39]:
```
array([[91,  0,  4, 55],
       [12, 27,  2,  9],
       [19,  2, 17, 12],
       [20,  3,  2, 25]], dtype=int64)
```

In [40]:
```
yticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
xticklabels=['1_normal', '2_cataract','3_glaucoma','4_retina_disease']
plt.figure(figsize=(8, 8))
hm =sns.heatmap(CM_sum, annot=True,annot_kws={"size": 20},fmt='g', cbar=False,cmap="YlGnBu",yticklabels=yticklabels,xt
icklabels=xticklabels)

hm.set_xticklabels(hm.get_xticklabels(), rotation=0, fontsize = 12, )
hm.set_yticklabels(hm.get_yticklabels(), rotation=0, fontsize = 12)

plt.ylabel("Actual", fontsize = 18)
plt.xlabel("Predicted",fontsize = 18)

plt.show()
```



## Reconfirming the values of Accuracy,Sensitivity and Specificity

In [41]:
```
    sensitivity_1_normal = (CM_sum[0,0])/(CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[0,3])
    #print('Sensitivity_1_normal          : ', sensitivity_1_normal )

    sensitivity_2_cataract = (CM_sum[1,1])/(CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3])
    #print('Sensitivity_2_cataract        : ', sensitivity_2_cataract )


    sensitivity_3_glaucoma = (CM_sum[2,2])/(CM_sum[2,0]+CM_sum[2,1]+CM_sum[2,2]+CM_sum[2,3])
    #print('Sensitivity_3_glaucoma        : ', sensitivity_3_glaucoma )

    sensitivity_4_retina_disease = (CM_sum[3,3])/(CM_sum[3,0]+CM_sum[3,1]+CM_sum[3,2]+CM_sum[3,3])
    #print('Sensitivity_4_retina_disease  : ', sensitivity_4_retina_disease )

    specificity_1_normal = (CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3]+CM_sum[2,1]+CM_sum[2,2]+CM_sum[2,3]+CM_sum[3,1]+CM_sum
[3,2]+CM_sum[3,3])/(CM_sum[1,0]+CM_sum[2,0]+CM_sum[3,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[1,3]+CM_sum[2,1]+CM_sum[2,2]+CM
_sum[2,3]+CM_sum[3,1]+CM_sum[3,2]+CM_sum[3,3])
    #print('Specificity : ', specificity_1_normal)

    specificity_2_cataract = (CM_sum[0,0]+CM_sum[0,2]+CM_sum[0,3]+CM_sum[2,0]+CM_sum[2,2]+CM_sum[2,3]+CM_sum[3,0]+CM_s
um[3,2]+CM_sum[3,3])/(CM_sum[0,1]+CM_sum[2,1]+CM_sum[3,1]+CM_sum[0,0]+CM_sum[0,2]+CM_sum[0,3]+CM_sum[2,0]+CM_sum[2,2]+
CM_sum[2,3]+CM_sum[3,0]+CM_sum[3,2]+CM_sum[3,3])
    #print('Specificity : ', specificity_2_cataract)

    specificity_3_glaucoma = (CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,3]+CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,3]+CM_sum[3,0]+CM_s
um[3,1]+CM_sum[3,3])/(CM_sum[0,2]+CM_sum[1,2]+CM_sum[3,2]+CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,3]+CM_sum[1,0]+CM_sum[1,1]+
CM_sum[1,3]+CM_sum[3,0]+CM_sum[3,1]+CM_sum[3,3])
    #print('Specificity : ', specificity_3_glaucoma)

    specificity_4_retina_disease= (CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[1,0]+CM_sum[1,1]+CM_sum[1,2]+CM_sum[2,0]
+CM_sum[2,1]+CM_sum[2,2])/(CM_sum[0,3]+CM_sum[1,3]+CM_sum[2,3]+CM_sum[0,0]+CM_sum[0,1]+CM_sum[0,2]+CM_sum[1,0]+CM_sum[
1,1]+CM_sum[1,2]+CM_sum[2,0]+CM_sum[2,1]+CM_sum[2,2])
    #print('Specificity : ', specificity_4_retina_disease)

    Sensitivity= (sensitivity_1_normal + sensitivity_2_cataract + sensitivity_3_glaucoma + sensitivity_4_retina_diseas
e)/4
    #print(Sensitivity)

    Specificity= (specificity_1_normal + specificity_2_cataract + specificity_3_glaucoma + specificity_4_retina_diseas
e)/4
    #print(Specificity)

    total1=sum(sum(CM_sum))
    test_accuracy=(CM_sum[0,0]+CM_sum[1,1]+CM_sum[2,2]+CM_sum[3,3])/total1

    print ('Accuracy    : ', test_accuracy)
    print ('Specificity : ', Specificity)
    print ('Sensitivity : ', Sensitivity)
```

```
Accuracy    :  0.5333333333333333
Specificity :  0.7813996025079956
Sensitivity :  0.4966666666666667
```

## Model Summary

In [42]:
```
model_build_compile(k)
```

```
model building and compiling for fold 7
```

Out[42]:
```
<tensorflow.python.keras.engine.functional.Functional at 0x23caad2e550>
```

In [43]:
```
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
reshape_4 (Reshape)          (None, 49, 512)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 49, 512)           2099200
_________________________________________________________________
batch_normalization_12 (Batc (None, 49, 512)           2048
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense_12 (Dense)             (None, 4096)              102764544
_________________________________________________________________
batch_normalization_13 (Batc (None, 4096)              16384
_________________________________________________________________
dense_13 (Dense)             (None, 4096)              16781312
_________________________________________________________________
batch_normalization_14 (Batc (None, 4096)              16384
_________________________________________________________________
dense_14 (Dense)             (None, 4)                 16388
=================================================================
Total params: 141,720,644
Trainable params: 119,579,652
Non-trainable params: 22,140,992
_________________________________________________________________
```