In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

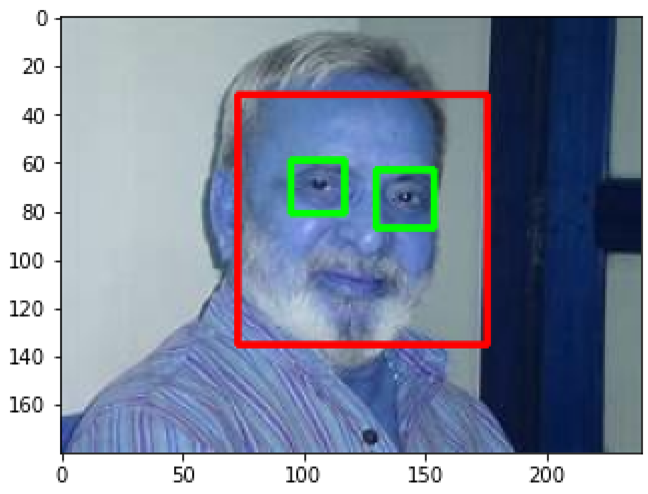
In [52]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from time import sleep
%matplotlib inline
```

In [53]:
```
img = cv2.imread( "C://Users//harsha k g//OneDrive//Desktop//8th sem n
plt.imshow(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray,cmap='gray')
#faces = detector.detectMultiScale(gray, 1.2, 5)

```

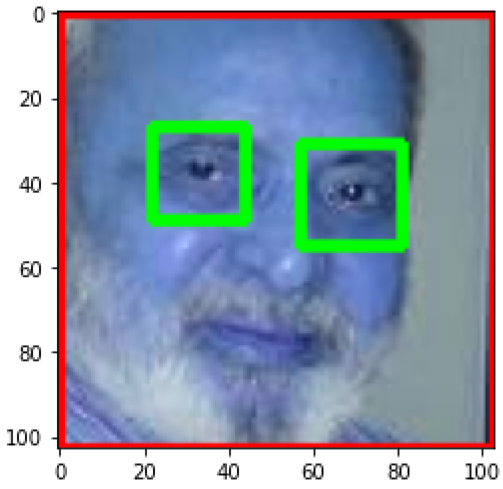
Out[53]:
```
<matplotlib.image.AxesImage at 0x250fe738ca0>
```

In [54]:

```
#face_cascade=cv2.CascadeClassifier("C://Users//harsha k g//OneDrive//
face_cascade= cv2.CascadeClassifier("C://Users//harsha k g//OneDrive//
#face_cascade= cv2.CascadeClassifier("C://Users//harsha k g//OneDrive/
eye_cascade= cv2.CascadeClassifier("C://Users//harsha k g//OneDrive//D
#eye_cascade= cv2.CascadeClassifier("C://Users//harsha k g//OneDrive//
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
faces
(x,y,w,h)=faces[0]
x,y,w,h
face_img=cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
#plt.imshow(face_img)
cv2.destroyAllWindows()
for (x,y,w,h) in faces:
    face_img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = face_img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)


plt.figure()
plt.imshow(face_img, cmap='gray')
plt.show()




#roi_gray = gray[y:y+h, x:x+w]
#roi_color = img[y:y+h, x:x+w]
#eyes = eye_cascade.detectMultiScale(roi_gray)
#for (ex,ey,ew,eh) in eyes:
 #   cv2.rectangle(roi_color, (ex, ey), (ex+ew,ey+eh), (0,255,0), 2)
#plt.figure()
#plt.imshow(face_img,cmap='gray')
#plt.show()
```



In [55]:
```
%matplotlib inline
plt.imshow(roi_color,cmap='gray')
cropped_img = np.array(roi_color)
```



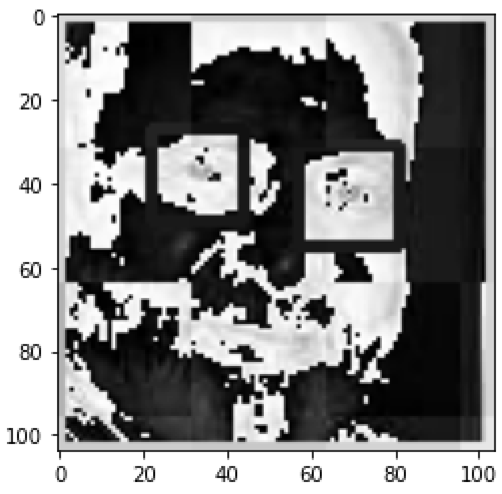
In [71]:
```
#nothing to do here
cropped_file_path = cropped_folder + "//" + cropped_file_name
print(cropped_file_path)
print(cropped_folder)
print(cropped_file_name)
print(path_to_cr_data + celebrity_name)
print(path_to_cr_data)
print(celebrity_name)
```

```
C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets cl
assification//dataset//croppeddataset\URA//dataset\URA6.png
C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets cl
assification//dataset//croppeddataset\URA
dataset\URA6.png
C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets cl
assification//dataset//croppeddataset\URA
C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets cl
assification//dataset//cropped
dataset\URA
```

In [56]:
```
import numpy as np
import pywt
import cv2

def w2d(img, mode='haar', level=1):
    imArray = img
    #Datatype conversions
    #convert to grayscale
    imArray = cv2.cvtColor( imArray,cv2.COLOR_RGB2GRAY )
    #convert to float
    imArray =  np.float32(imArray)
    imArray /= 255;
    # compute coefficients
    coeffs=pywt.wavedec2(imArray, mode, level=level)

    #Process Coefficients
    coeffs_H=list(coeffs)
    coeffs_H[0] *= 0;

    # reconstruction
    imArray_H=pywt.waverec2(coeffs_H, mode);
    imArray_H *= 255;
    imArray_H =  np.uint8(imArray_H)

    return imArray_H
```

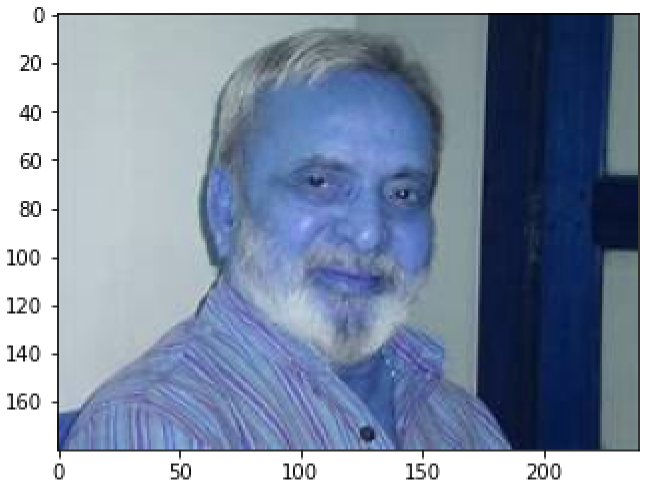
In [57]:
```
im_har = w2d(cropped_img,'db1',5)
plt.imshow(im_har, cmap='gray')
```

Out[57]:
```
<matplotlib.image.AxesImage at 0x250fe8c38e0>
```



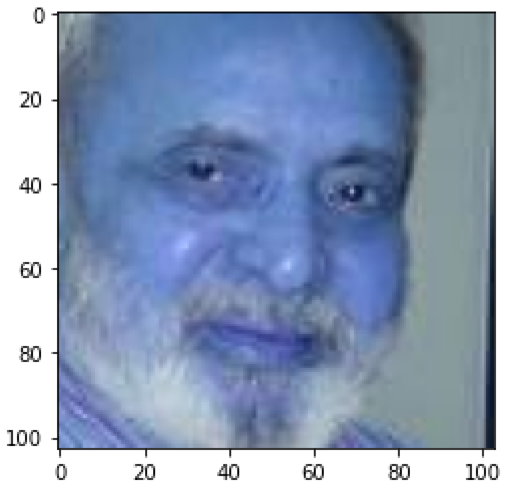
In [58]:
```
def get_cropped_image_if_2_eyes(image_path):
    img =cv2.imread(image_path)
    #print(img)
    #print(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        if len(eyes) >= 2:
            return roi_color
original_image = cv2.imread("C:/Users/harsha k g/OneDrive/Desktop/8th
plt.imshow(original_image)
#cropped_image = get_cropped_image_if_2_eyes("C:/Users/harsha k g/OneDı
#plt.imshow(cropped_image)
```

Out[58]:
```
<matplotlib.image.AxesImage at 0x250fe6c9e20>
```



In [59]:
```
cropped_image = get_cropped_image_if_2_eyes("C:/Users/harsha k g/OneDr
plt.imshow(cropped_image)
#path_to_data = ("C://Users//harsha k g//OneDrive//Desktop//8th sem no
#path_to_cr_data = ("C://Users//harsha k g//OneDrive//Desktop//8th sem
```

Out[59]:
```
<matplotlib.image.AxesImage at 0x250fe688730>
```



In [70]:
```
#path_to_data = "./dataset/"
#path_to_cr_data = "./dataset/cropped/"
path_to_data = ("C://Users//harsha k g//OneDrive//Desktop//8th sem note
path_to_cr_data = ("C://Users//harsha k g//OneDrive//Desktop//8th sem n
print(path_to_cr_data)
```

```
C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets cl
assification//dataset//cropped
```

In [61]:
```
import os
img_dirs = []
for entry in os.scandir(path_to_data):
    if entry.is_dir():
        img_dirs.append(entry.path)
```

In [62]:
```
img_dirs
```

Out[62]:
```
['C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset\\Girish karnaad',
 'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset\\Kuvempu',
 'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset\\Nisar Ahmed',
 'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset\\URA']
```

In [63]:
```
import shutil
if os.path.exists(path_to_cr_data):
     shutil.rmtree(path_to_cr_data)
os.mkdir(path_to_cr_data)
```

In [88]:
```python
cropped_image_dirs = []
celebrity_file_names_dict = {}
for img_dir in img_dirs:
    count = 1
    celebrity_name = img_dir.split('\\')[-1]
    print( celebrity_name)
    #print(img_dir.split)
    celebrity_file_names_dict[celebrity_name] = []
    for entry in os.scandir(img_dir):
        roi_color = get_cropped_image_if_2_eyes(entry.path)
        if roi_color is not None:
            cropped_folder = path_to_cr_data + celebrity_name
            if not os.path.exists(cropped_folder):
                os.makedirs(cropped_folder)
                cropped_image_dirs.append(cropped_folder)
                print("Generating cropped images in folder: ",cropped_
            cropped_file_name = celebrity_name + str(count) + ".png"
            cropped_file_path = cropped_folder + "//" + cropped_file_n
            cv2.imwrite(cropped_file_path, roi_color)
            celebrity_file_names_dict[celebrity_name].append(cropped_f
            count += 1
```

```
Girish karnaad
Generating cropped images in folder:  C://Users//harsha k g//OneDrive//De
sktop//8th sem notes//Project_poets classification//dataset//croppedGiris
h karnaad
Kuvempu
Generating cropped images in folder:  C://Users//harsha k g//OneDrive//De
sktop//8th sem notes//Project_poets classification//dataset//croppedKuvem
pu
Nisar Ahmed
Generating cropped images in folder:  C://Users//harsha k g//OneDrive//De
sktop//8th sem notes//Project_poets classification//dataset//croppedNisar
Ahmed
URA
Generating cropped images in folder:  C://Users//harsha k g//OneDrive//De
sktop//8th sem notes//Project_poets classification//dataset//croppedURA
```

In [89]:
```python
celebrity_file_names_dict = {}
for img_dir in cropped_image_dirs:
    celebrity_name = img_dir.split('/')[-1]
    file_list = []
    for entry in os.scandir(img_dir):
        file_list.append(entry.path)
    celebrity_file_names_dict[celebrity_name] = file_list
celebrity_file_names_dict
```

Out[89]:
```
{'croppedGirish karnaad': ['C://Users//harsha k g//OneDrive//Desktop//8th
sem notes//Project_poets classification//dataset//croppedGirish karnaad
\\Girish karnaad1.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedGirish karnaad\\Girish karnaad2.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedGirish karnaad\\Girish karnaad3.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedGirish karnaad\\Girish karnaad4.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedGirish karnaad\\Girish karnaad5.png'],
 'croppedKuvempu': ['C://Users//harsha k g//OneDrive//Desktop//8th sem no
tes//Project_poets classification//dataset//croppedKuvempu\\Kuvempu1.pn
g',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedKuvempu\\Kuvempu2.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedKuvempu\\Kuvempu3.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedKuvempu\\Kuvempu4.png'],
 'croppedNisar Ahmed': ['C://Users//harsha k g//OneDrive//Desktop//8th se
m notes//Project_poets classification//dataset//croppedNisar Ahmed\\Nisar
Ahmed1.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedNisar Ahmed\\Nisar Ahmed2.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedNisar Ahmed\\Nisar Ahmed3.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedNisar Ahmed\\Nisar Ahmed4.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedNisar Ahmed\\Nisar Ahmed5.png'],
 'croppedURA': ['C://Users//harsha k g//OneDrive//Desktop//8th sem note
s//Project_poets classification//dataset//croppedURA\\URA1.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedURA\\URA2.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedURA\\URA3.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedURA\\URA4.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedURA\\URA5.png',
  'C://Users//harsha k g//OneDrive//Desktop//8th sem notes//Project_poets
classification//dataset//croppedURA\\URA6.png']}
```

In [90]:
```
class_dict = {}
count = 0
for celebrity_name in celebrity_file_names_dict.keys():
    class_dict[celebrity_name] = count
    count = count + 1
class_dict
```

Out[90]:
```
{'croppedGirish karnaad': 0,
 'croppedKuvempu': 1,
 'croppedNisar Ahmed': 2,
 'croppedURA': 3}
```

In [92]:
```
X, y = [], []
for celebrity_name, training_files in celebrity_file_names_dict.items(
    for training_image in training_files:
        img = cv2.imread(training_image)
        scalled_raw_img = cv2.resize(img, (32, 32))
        img_har = w2d(img,'db1',5)
        scalled_img_har = cv2.resize(img_har, (32, 32))
        combined_img = np.vstack((scalled_raw_img.reshape(32*32*3,1),s
        X.append(combined_img)
        y.append(class_dict[celebrity_name])
```

In [110]:
```

X = np.array(X).reshape(len(X),4096).astype(float)
X.shape


```

Out[110]:
```
(20, 4096)
```

In [111]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'r
pipe.fit(X_train, y_train)
pipe.score(X_test, y_test)
```

Out[111]:
```
0.4
```

In [112]:
```
print(classification_report(y_test, pipe.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.33      1.00      0.50         1
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         1
           3       0.50      0.50      0.50         2

    accuracy                           0.40         5
   macro avg       0.21      0.38      0.25         5
weighted avg       0.27      0.40      0.30         5


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classifica
tion.py:1245: UndefinedMetricWarning: Precision and F-score are ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classifica
tion.py:1245: UndefinedMetricWarning: Precision and F-score are ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `z
```

In [115]:
```
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto',probability=True),
        'params' : {
            'svc__C': [1,10,100,1000],
            'svc__kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'randomforestclassifier__n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='au
        'params': {
            'logisticregression__C': [1,5,10]
        }
    }
}

```

In [146]:
```
scores = []
best_estimators = {}
import pandas as pd
for algo, mp in model_params.items():
    pipe = make_pipeline(StandardScaler(), mp['model'])
    clf =  GridSearchCV(pipe, mp['params'], cv=4, return_train_score=F
    clf.fit(X_train, y_train)
    scores.append({
        'model': algo,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

    best_estimators[algo] = clf.best_estimator_

df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
#df = pd.DataFrame(scores,columns=['model'])
df
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_spli
t.py:666: UserWarning: The least populated class in y has only 3 members,
which is less than n_splits=4.
  warnings.warn(("The least populated class in y has only %d"
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_spli
t.py:666: UserWarning: The least populated class in y has only 3 members,
which is less than n_splits=4.
  warnings.warn(("The least populated class in y has only %d"
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_spli
t.py:666: UserWarning: The least populated class in y has only 3 members,
which is less than n_splits=4.
  warnings.warn(("The least populated class in y has only %d"
```

Out[146]:

	model	best_score	best_params
0	svm	0.875000	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.520833	{'randomforestclassifier__n_estimators': 1}
2	logistic_regression	0.812500	{'logisticregression__C': 1}

In [156]:
```
best_estimators
```

Out[156]:
```
{'svm': Pipeline(steps=[('standardscaler', StandardScaler()),
                 ('svc',
                  SVC(C=1, gamma='auto', kernel='linear', probability=Tru
e))]),
 'random_forest': Pipeline(steps=[('standardscaler', StandardScaler()),
                 ('randomforestclassifier',
                  RandomForestClassifier(n_estimators=1))]),
 'logistic_regression': Pipeline(steps=[('standardscaler', StandardScaler
()),
                 ('logisticregression',
                  LogisticRegression(C=1, solver='liblinear'))])}
```

In [157]:
```
best_estimators['logistic_regression'].score(X_test,y_test)
```

Out[157]:
```
0.2
```

In [158]:
```
best_estimators['random_forest'].score(X_test,y_test)
```

Out[158]:
```
0.4
```

In [159]:
```
best_estimators['svm'].score(X_test,y_test)
```

Out[159]:
```
0.4
```

In [161]:
```
best_clf = best_estimators['svm']
best_clf
```

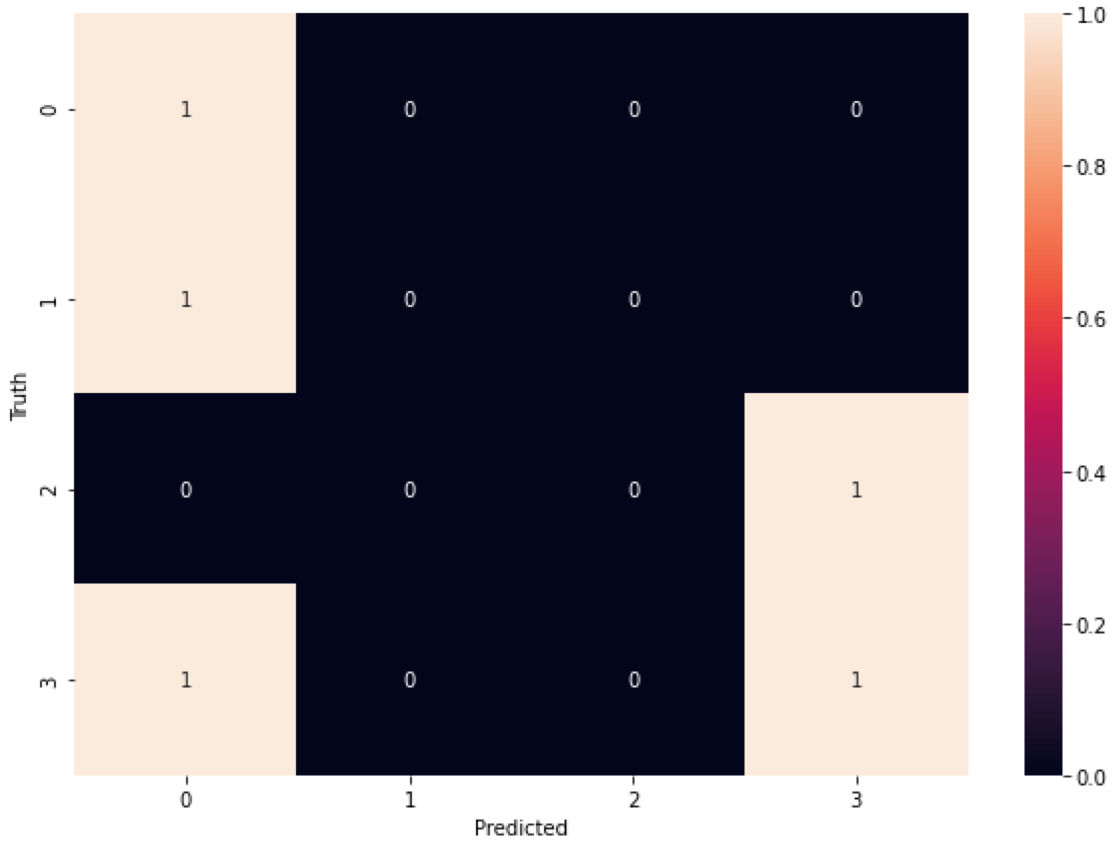
Out[161]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc',
                 SVC(C=1, gamma='auto', kernel='linear', probability=Tru
e))])
```

In [162]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, best_clf.predict(X_test))
cm
```

Out[162]:
```
array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [0, 0, 0, 1],
       [1, 0, 0, 1]], dtype=int64)
```

In [163]:
```
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[163]:
```
Text(69.0, 0.5, 'Truth')
```



In [ ]:
```
 
```

In [ ]:
```
 
```