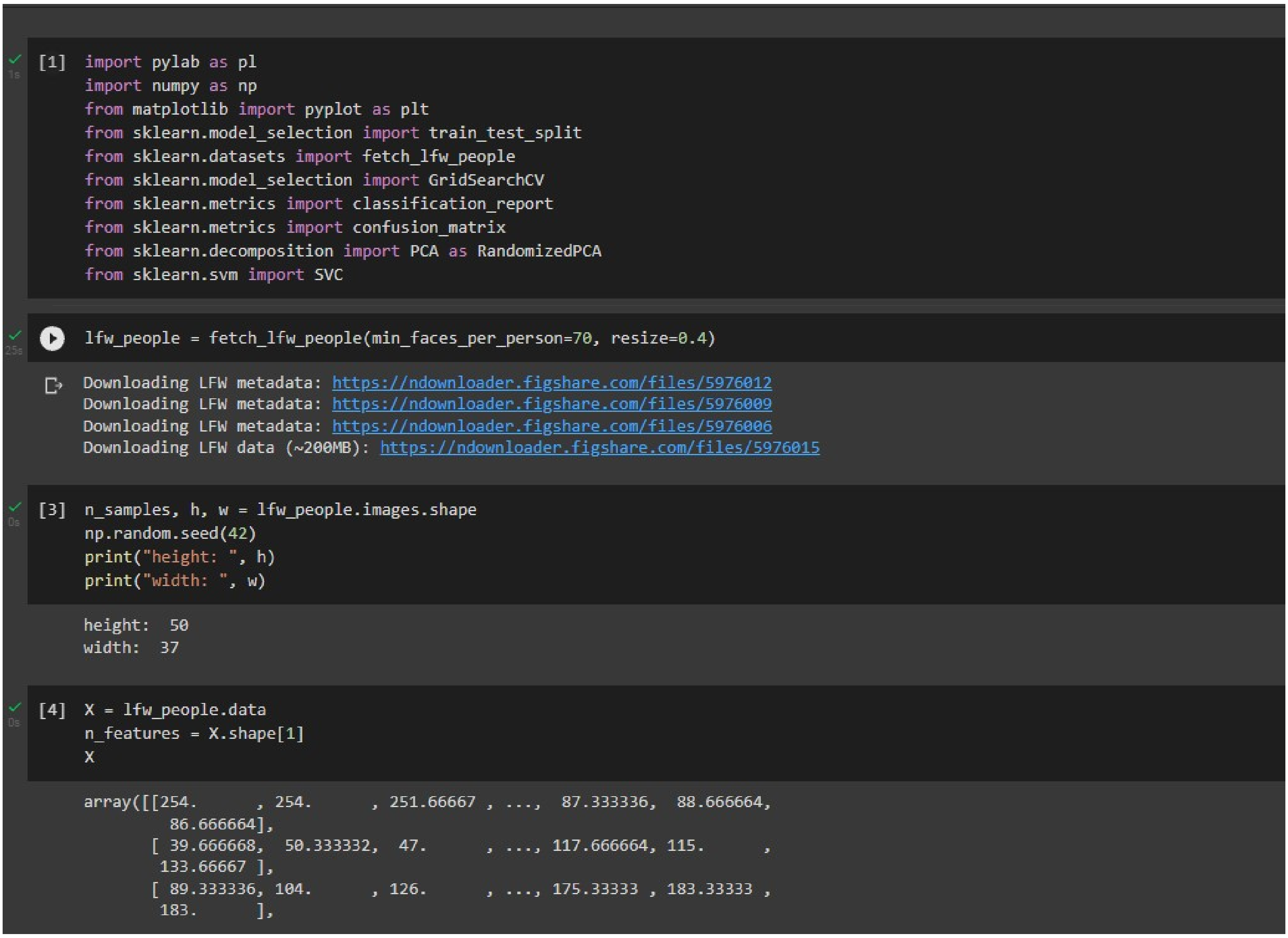
Machine learning model for face recognition by applying PCA.

Code and Screenshot

```
import pylab as pl
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.svm import SVC
```

```
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)


n_samples, h, w = lfw_people.images.shape

np.random.seed(42)

print("height: ", h)

print("width: ", w)
```

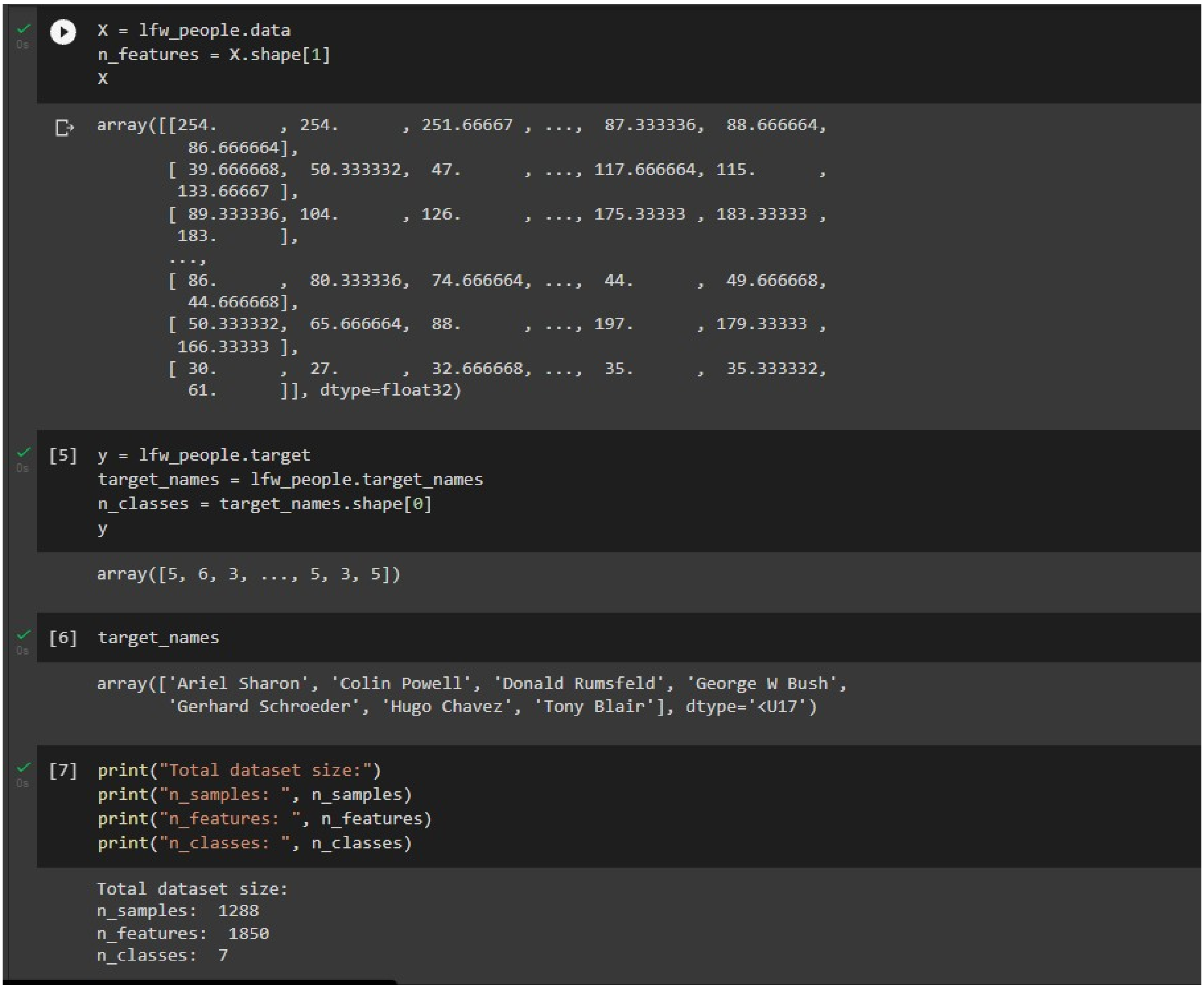


```
[1]  import pylab as pl
     import numpy as np
     from matplotlib import pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.datasets import fetch_lfw_people
     from sklearn.model_selection import GridSearchCV
     from sklearn.metrics import classification_report
     from sklearn.metrics import confusion_matrix
     from sklearn.decomposition import PCA as RandomizedPCA
     from sklearn.svm import SVC
```

```
     lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
     Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

```
[3]  n_samples, h, w = lfw_people.images.shape
     np.random.seed(42)
     print("height: ", h)
     print("width: ", w)

     height:  50
     width:  37
```

```
[4]  X = lfw_people.data
     n_features = X.shape[1]
     X

     array([[254.      , 254.      , 251.66667 , ...,  87.333336,  88.666664,
              86.666664],
            [ 39.666668,  50.333332,  47.      , ..., 117.666664, 115.      ,
             133.66667 ],
            [ 89.333336, 104.      , 126.      , ..., 175.33333 , 183.33333 ,
             183.      ],
X = lfw_people.data

n_features = X.shape[1]

X


y = lfw_people.target

target_names = lfw_people.target_names

n_classes = target_names.shape[0]

y


target_names


print("Total dataset size:")

print("n_samples: ", n_samples)

print("n_features: ", n_features)

print("n_classes: ", n_classes)
```



```
X = lfw_people.data
n_features = X.shape[1]
X
```

```
array([[254.      , 254.      , 251.66667 , ...,  87.333336,  88.666664,
         86.666664],
       [ 39.666668,  50.333332,  47.      , ..., 117.666664, 115.      ,
        133.66667 ],
       [ 89.333336, 104.      , 126.      , ..., 175.33333 , 183.33333 ,
        183.      ],
       ...,
       [ 86.      ,  80.333336,  74.666664, ...,  44.      ,  49.666668,
         44.666668],
       [ 50.333332,  65.666664,  88.      , ..., 197.      , 179.33333 ,
        166.33333 ],
       [ 30.      ,  27.      ,  32.666668, ...,  35.      ,  35.333332,
         61.      ]], dtype=float32)
```

```
[5] y = lfw_people.target
    target_names = lfw_people.target_names
    n_classes = target_names.shape[0]
    y
```

```
array([5, 6, 3, ..., 5, 3, 5])
```

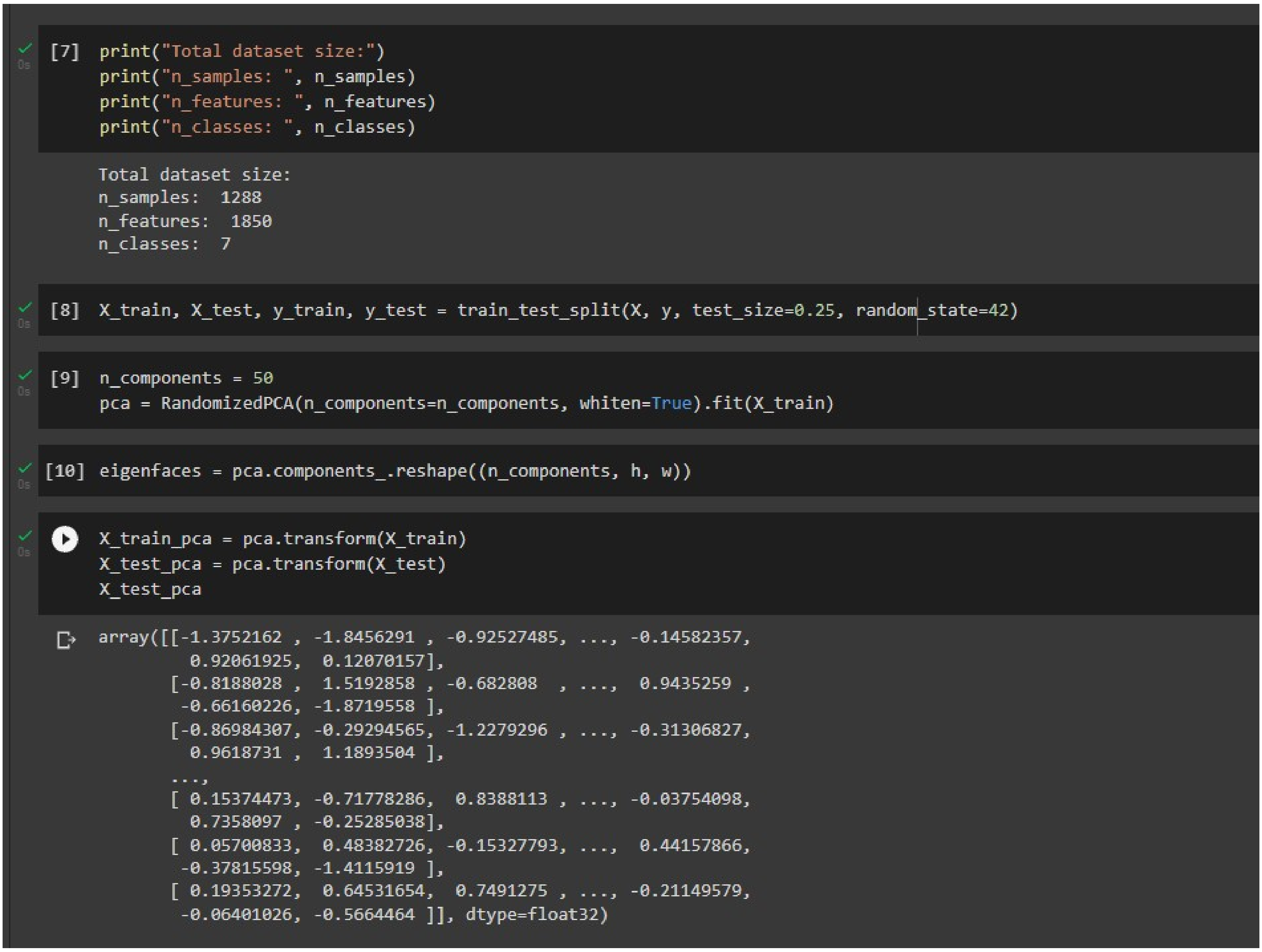
```
[6] target_names
```

```
array(['Ariel Sharon', 'Colin Powell', 'Donald Rumsfeld', 'George W Bush',
       'Gerhard Schroeder', 'Hugo Chavez', 'Tony Blair'], dtype='<U17')
```

```
[7] print("Total dataset size:")
    print("n_samples: ", n_samples)
    print("n_features: ", n_features)
    print("n_classes: ", n_classes)
```

```
Total dataset size:
n_samples:  1288
n_features:  1850
n_classes:  7
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

n_components = 50

pca = RandomizedPCA(n_components=n_components, whiten=True).fit(X_train)

eigenfaces = pca.components_.reshape((n_components, h, w))

X_train_pca = pca.transform(X_train)

X_test_pca = pca.transform(X_test)

X_test_pca
```



```
[7] print("Total dataset size:")
    print("n_samples: ", n_samples)
    print("n_features: ", n_features)
    print("n_classes: ", n_classes)

    Total dataset size:
    n_samples:  1288
    n_features:  1850
    n_classes:  7
```

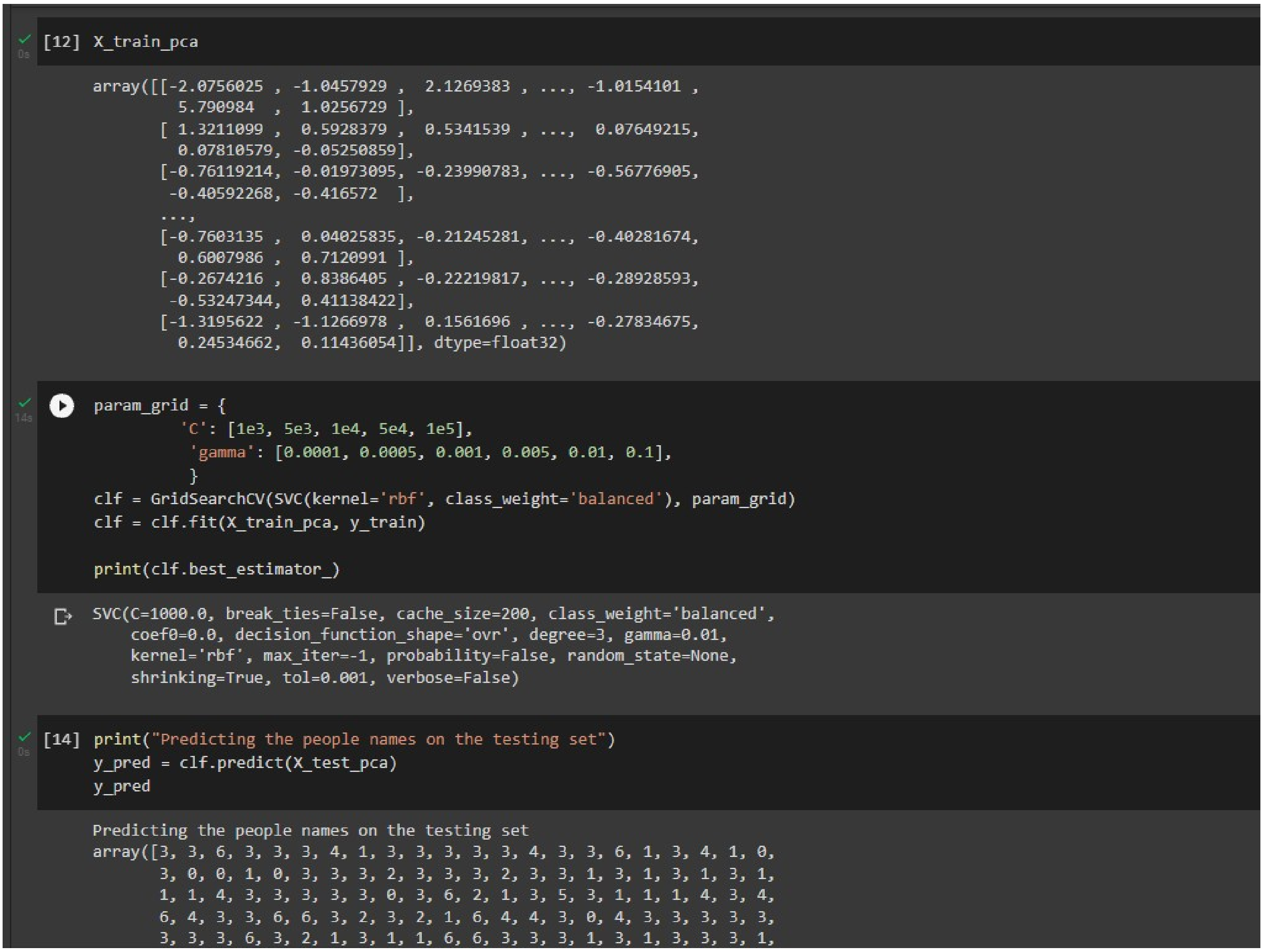
```
[8] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
[9] n_components = 50
    pca = RandomizedPCA(n_components=n_components, whiten=True).fit(X_train)
```

```
[10] eigenfaces = pca.components_.reshape((n_components, h, w))
```

```
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
X_test_pca

array([[-1.3752162 , -1.8456291 , -0.92527485, ..., -0.14582357,
         0.92061925,  0.12070157],
       [-0.8188028 ,  1.5192858 , -0.682808  , ...,  0.9435259 ,
        -0.66160226, -1.8719558 ],
       [-0.86984307, -0.29294565, -1.2279296 , ..., -0.31306827,
         0.9618731 ,  1.1893504 ],
       ...,
       [ 0.15374473, -0.71778286,  0.8388113 , ..., -0.03754098,
         0.7358097 , -0.25285038],
       [ 0.05700833,  0.48382726, -0.15327793, ...,  0.44157866,
        -0.37815598, -1.4115919 ],
       [ 0.19353272,  0.64531654,  0.7491275 , ..., -0.21149579,
        -0.06401026, -0.5664464 ]], dtype=float32)
```

```
X_train_pca


param_grid = {

        'C': [1e3, 5e3, 1e4, 5e4, 1e5],

         'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1],

         }

clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)

clf = clf.fit(X_train_pca, y_train)

print(clf.best_estimator_)
```



```
[12] X_train_pca
```

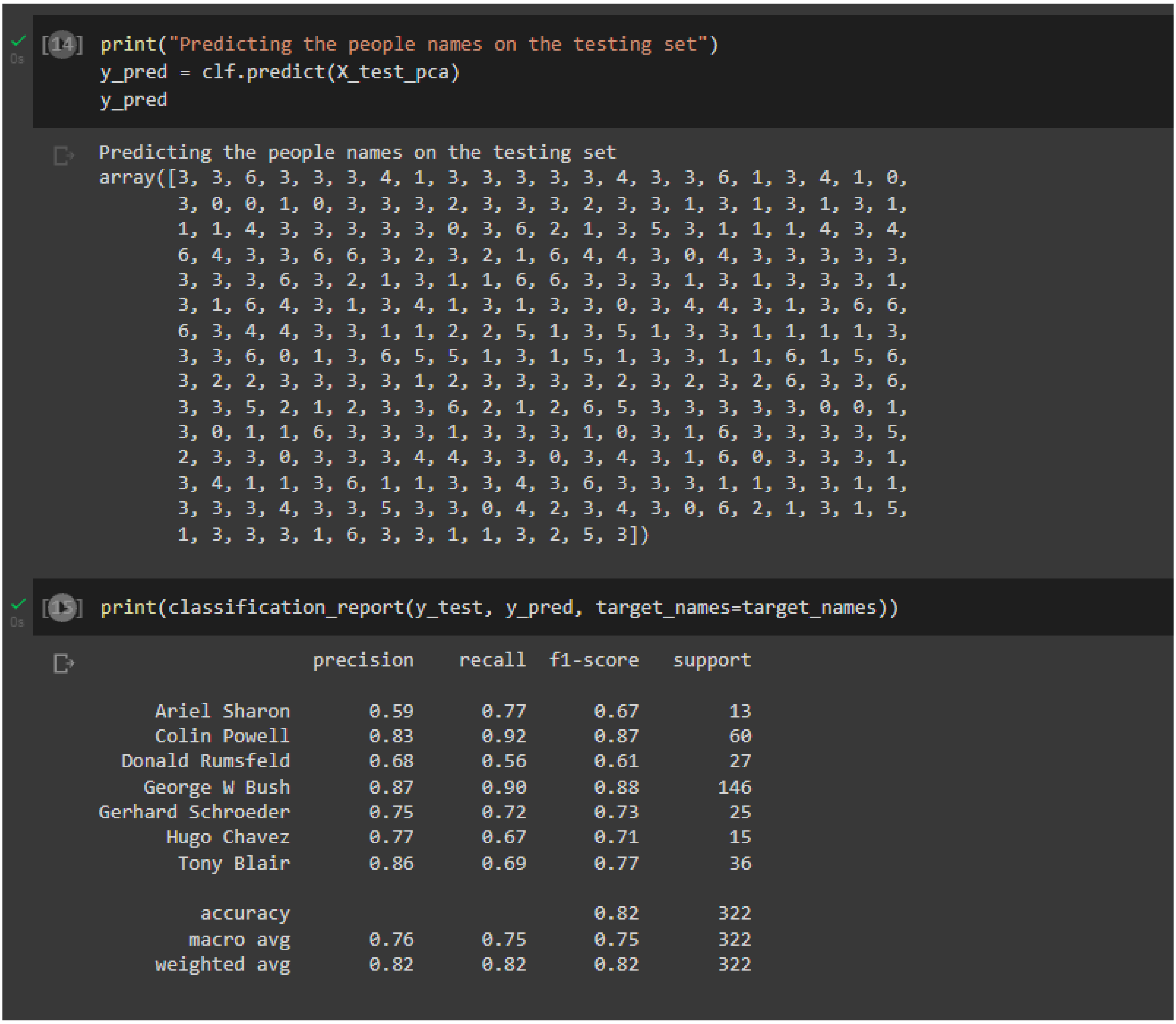
```
array([[-2.0756025 , -1.0457929 ,  2.1269383 , ..., -1.0154101 ,
         5.790984  ,  1.0256729 ],
       [ 1.3211099 ,  0.5928379 ,  0.5341539 , ...,  0.07649215,
         0.07810579, -0.05250859],
       [-0.76119214, -0.01973095, -0.23990783, ..., -0.56776905,
        -0.40592268, -0.416572  ],
       ...,
       [-0.7603135 ,  0.04025835, -0.21245281, ..., -0.40281674,
         0.6007986 ,  0.7120991 ],
       [-0.2674216 ,  0.8386405 , -0.22219817, ..., -0.28928593,
        -0.53247344,  0.41138422],
       [-1.3195622 , -1.1266978 ,  0.1561696 , ..., -0.27834675,
         0.24534662,  0.11436054]], dtype=float32)
```

```
param_grid = {
         'C': [1e3, 5e3, 1e4, 5e4, 1e5],
          'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1],
          }
clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
clf = clf.fit(X_train_pca, y_train)

print(clf.best_estimator_)
```

```
SVC(C=1000.0, break_ties=False, cache_size=200, class_weight='balanced',
    coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.01,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
[14] print("Predicting the people names on the testing set")
     y_pred = clf.predict(X_test_pca)
     y_pred
```

```
Predicting the people names on the testing set
array([3, 3, 6, 3, 3, 3, 4, 1, 3, 3, 3, 3, 3, 4, 3, 3, 6, 1, 3, 4, 1, 0,
       3, 0, 0, 1, 0, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 1, 3, 1, 3, 1, 3, 1,
       1, 1, 4, 3, 3, 3, 3, 3, 0, 3, 6, 2, 1, 3, 5, 3, 1, 1, 1, 4, 3, 4,
       6, 4, 3, 3, 6, 6, 3, 2, 3, 2, 1, 6, 4, 4, 3, 0, 4, 3, 3, 3, 3, 3,
       3, 3, 3, 6, 3, 2, 1, 3, 1, 1, 6, 6, 3, 3, 3, 1, 3, 1, 3, 3, 3, 1,
print("Predicting the people names on the testing set")

y_pred = clf.predict(X_test_pca)

y_pred


print(classification_report(y_test, y_pred, target_names=target_names))
```



```
[14] print("Predicting the people names on the testing set")
     y_pred = clf.predict(X_test_pca)
     y_pred
```

```
Predicting the people names on the testing set
array([3, 3, 6, 3, 3, 3, 4, 1, 3, 3, 3, 3, 3, 4, 3, 3, 6, 1, 3, 4, 1, 0,
       3, 0, 0, 1, 0, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 1, 3, 1, 3, 1, 3, 1,
       1, 1, 4, 3, 3, 3, 3, 3, 0, 3, 6, 2, 1, 3, 5, 3, 1, 1, 1, 4, 3, 4,
       6, 4, 3, 3, 6, 6, 3, 2, 3, 2, 1, 6, 4, 4, 3, 0, 4, 3, 3, 3, 3, 3,
       3, 3, 3, 6, 3, 2, 1, 3, 1, 1, 6, 6, 3, 3, 3, 1, 3, 1, 3, 3, 3, 1,
       3, 1, 6, 4, 3, 1, 3, 4, 1, 3, 1, 3, 3, 0, 3, 4, 4, 3, 1, 3, 6, 6,
       6, 3, 4, 4, 3, 3, 1, 1, 2, 2, 5, 1, 3, 5, 1, 3, 3, 1, 1, 1, 1, 3,
       3, 3, 6, 0, 1, 3, 6, 5, 5, 1, 3, 1, 5, 1, 3, 3, 1, 1, 6, 1, 5, 6,
       3, 2, 2, 3, 3, 3, 3, 1, 2, 3, 3, 3, 3, 2, 3, 2, 3, 2, 6, 3, 3, 6,
       3, 3, 5, 2, 1, 2, 3, 3, 6, 2, 1, 2, 6, 5, 3, 3, 3, 3, 3, 0, 0, 1,
       3, 0, 1, 1, 6, 3, 3, 3, 1, 3, 3, 3, 1, 0, 3, 1, 6, 3, 3, 3, 3, 5,
       2, 3, 3, 0, 3, 3, 3, 4, 4, 3, 3, 0, 3, 4, 3, 1, 6, 0, 3, 3, 3, 1,
       3, 4, 1, 1, 3, 6, 1, 1, 3, 3, 4, 3, 6, 3, 3, 3, 1, 1, 3, 3, 1, 1,
       3, 3, 3, 4, 3, 3, 5, 3, 3, 0, 4, 2, 3, 4, 3, 0, 6, 2, 1, 3, 1, 5,
       1, 3, 3, 3, 1, 6, 3, 3, 1, 1, 3, 2, 5, 3])
```

```
[15] print(classification_report(y_test, y_pred, target_names=target_names))
```

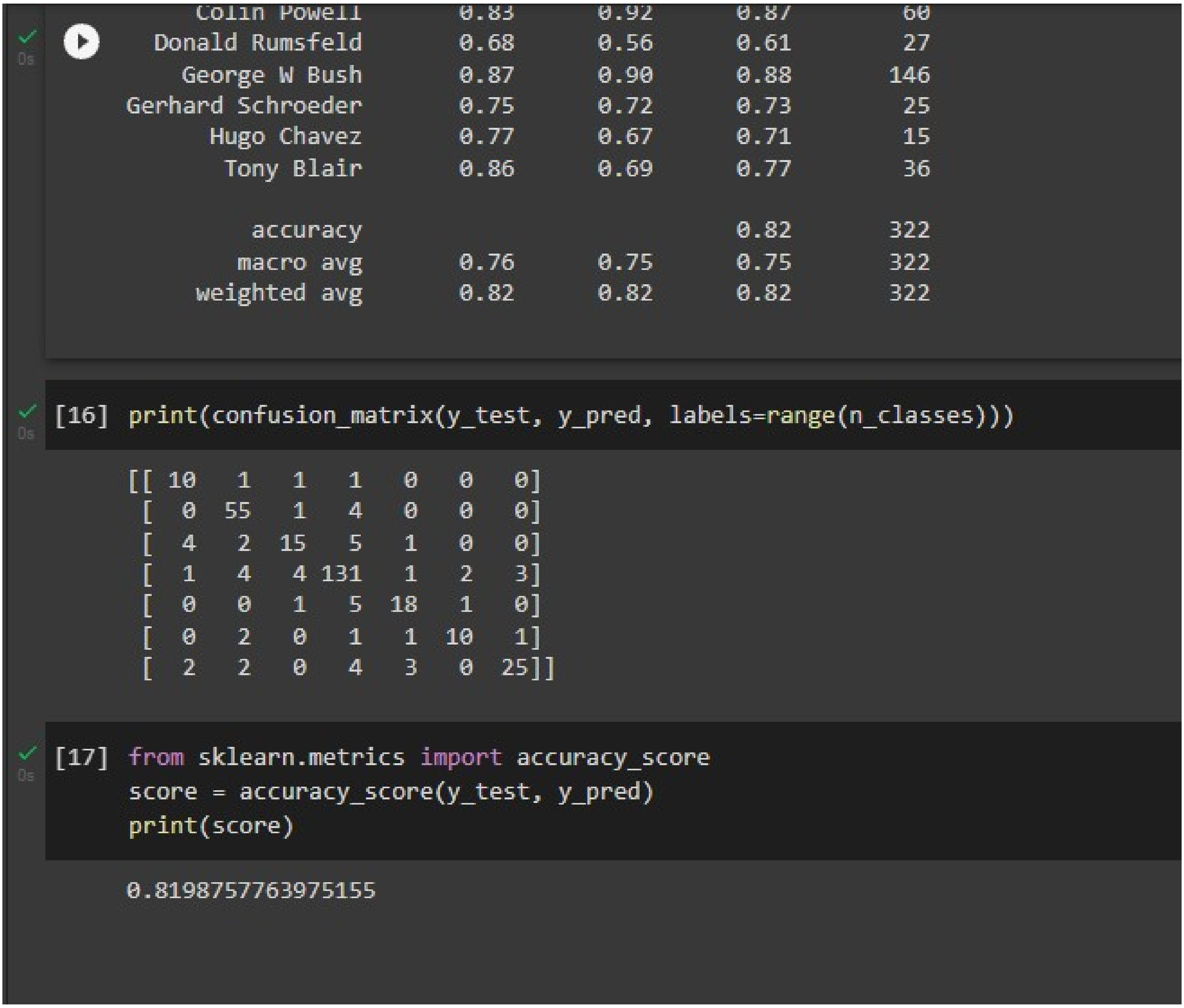
```
                   precision    recall  f1-score   support

     Ariel Sharon       0.59      0.77      0.67        13
     Colin Powell       0.83      0.92      0.87        60
  Donald Rumsfeld       0.68      0.56      0.61        27
    George W Bush       0.87      0.90      0.88       146
Gerhard Schroeder       0.75      0.72      0.73        25
      Hugo Chavez       0.77      0.67      0.71        15
       Tony Blair       0.86      0.69      0.77        36

         accuracy                           0.82       322
        macro avg       0.76      0.75      0.75       322
     weighted avg       0.82      0.82      0.82       322
```

print(confusion_matrix(y_test, y_pred, labels=range(n_classes)))

from sklearn.metrics import accuracy_score

score = accuracy_score(y_test, y_pred)

print(score)



```
     Colin Powell       0.83      0.92      0.87        60
  Donald Rumsfeld       0.68      0.56      0.61        27
    George W Bush       0.87      0.90      0.88       146
Gerhard Schroeder       0.75      0.72      0.73        25
      Hugo Chavez       0.77      0.67      0.71        15
       Tony Blair       0.86      0.69      0.77        36

         accuracy                           0.82       322
        macro avg       0.76      0.75      0.75       322
     weighted avg       0.82      0.82      0.82       322
```

```
[16] print(confusion_matrix(y_test, y_pred, labels=range(n_classes)))
```

```
[[ 10   1   1   1   0   0   0]
 [  0  55   1   4   0   0   0]
 [  4   2  15   5   1   0   0]
 [  1   4   4 131   1   2   3]
 [  0   0   1   5  18   1   0]
 [  0   2   0   1   1  10   1]
 [  2   2   0   4   3   0  25]]
```

```
[17] from sklearn.metrics import accuracy_score
     score = accuracy_score(y_test, y_pred)
     print(score)
```

```
0.8198757763975155
```