**FACE RECOGNITION WITH SCIKIT-LEARN IMAGES**

**SOURCE**: We will use the Labeled Faces in the Wild dataset, which consists of several thousand collated photos of various public figures

**DATA FOR THE PROJECT**

The Data is from Scikit-Learn datasets for fetch\_lfw\_people

Now let check how we can use supervised learning algorithm to solve Facial recognition problems for scikit-learn datasets for the fetch\_lfw\_people

**SUPPORT VECTOR MACHINE CLASSIFICATION**

## **What is the Support Vector Machine?**

“Support Vector Machine” (SVM) is a supervised [machine learning algorithm](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=understandingsupportvectormachinearticle) that can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well, so we will use Support Vector Machine to solve the problem.

LET’S PLOT A FEW OF THESE FACES TO SEE WHAT WE’RE WORKING WITH



LET’S TAKE A LOOK AT A FEW OF THE TEST IMAGES ALONG WITH THEIR PREDICTED VALUES



We can get a better sense of our estimator’s performance using the classification report, which lists recovery statistics label by label:

precision recall f1-score support

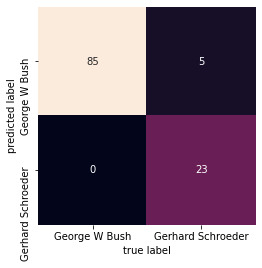
George W Bush 0.94 1.00 0.97 85

Gerhard Schroeder 1.00 0.82 0.90 28

accuracy 0.96 113

macro avg 0.97 0.91 0.94 113

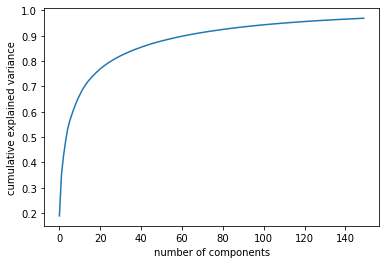
weighted avg 0.96 0.96 0.95 113

We might also display the confusion matrix between these classes:

In this case, it can be interesting to visualize the images associated with the first several principal components (these components are technically known as “eigenvectors,” so these types of images are often called “eigenfaces”).



Let’s take a look at the cumulative variance of these components to see how much of the data information the projection is preserving



To make this more concrete, we can compare the input images with the images reconstructed from

# these 150 components

# Compute the components and projected faces



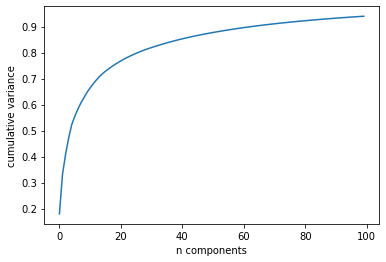
let’s apply Isomap on some faces data. We will use the Labeled Faces in the Wild dataset

Let’s quickly visualize several of these images to see what we’re working with



We would like to plot a low-dimensional embedding of the 2,914-dimensional data to learn the

# fundamental relationships between the images. One useful way to start is to compute a PCA, and

# examine the explained variance ratio, which will give us an idea of how many linear features are required to describe the data

To get a better idea of what the projection tells us, let’s define a function that will output image thumbnails at the locations of the projections:

