**from** **time** **import** time

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn.model\_selection** **import** [train\_test\_split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split)

**from** **sklearn.model\_selection** **import** [RandomizedSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV)

**from** **sklearn.datasets** **import** [fetch\_lfw\_people](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html#sklearn.datasets.fetch_lfw_people)

**from** **sklearn.metrics** **import** [classification\_report](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report)

**from** **sklearn.metrics** **import** ConfusionMatrixDisplay

**from** **sklearn.preprocessing** **import** [StandardScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler)

**from** **sklearn.decomposition** **import** [PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA)

**from** **sklearn.svm** **import** [SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)

**from** **sklearn.utils.fixes** **import** loguniform

**from** **time** **import** [time](https://docs.python.org/3/library/time.html#time.time)

**import** **matplotlib.pyplot** **as** **plt**

Download the data, if not already on disk and load it as numpy arrays

lfw\_people = [fetch\_lfw\_people](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html#sklearn.datasets.fetch_lfw_people)(min\_faces\_per\_person=70, resize=0.4)

*# introspect the images arrays to find the shapes (for plotting)*

n\_samples, h, w = lfw\_people.images.shape

*# for machine learning we use the 2 data directly (as relative pixel*

*# positions info is ignored by this model)*

X = lfw\_people.data

n\_features = X.shape[1]

*# the label to predict is the id of the person*

y = lfw\_people.target

target\_names = lfw\_people.target\_names

n\_classes = target\_names.shape[0]

print("Total dataset size:")

print("n\_samples: *%d*" % n\_samples)

print("n\_features: *%d*" % n\_features)

print("n\_classes: *%d*" % n\_classes)

Split into a training set and a test and keep 25% of the data for testing.

X\_train, X\_test, y\_train, y\_test = [train\_test\_split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split)(

X, y, test\_size=0.25, random\_state=42

)

scaler = [StandardScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler)()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled dataset): unsupervised feature extraction / dimensionality reduction

n\_components = 150

print(

"Extracting the top *%d* eigenfaces from *%d* faces" % (n\_components, X\_train.shape[0])

)

t0 = [time](https://docs.python.org/3/library/time.html#time.time)()

pca = [PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA)(n\_components=n\_components, svd\_solver="randomized", whiten=**True**).fit(X\_train)

print("done in *%0.3f*s" % ([time](https://docs.python.org/3/library/time.html#time.time)() - t0))

eigenfaces = pca.components\_.reshape((n\_components, h, w))

print("Projecting the input data on the eigenfaces orthonormal basis")

t0 = [time](https://docs.python.org/3/library/time.html#time.time)()

X\_train\_pca = pca.transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

print("done in *%0.3f*s" % ([time](https://docs.python.org/3/library/time.html#time.time)() - t0))

Train a SVM classification model

print("Fitting the classifier to the training set")

t0 = [time](https://docs.python.org/3/library/time.html#time.time)()

param\_grid = {

"C": loguniform(1e3, 1e5),

"gamma": loguniform(1e-4, 1e-1),

}

clf = [RandomizedSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV)(

[SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)(kernel="rbf", class\_weight="balanced"), param\_grid, n\_iter=10

)

clf = clf.fit(X\_train\_pca, y\_train)

print("done in *%0.3f*s" % ([time](https://docs.python.org/3/library/time.html#time.time)() - t0))

print("Best estimator found by grid search:")

print(clf.best\_estimator\_)

Quantitative evaluation of the model quality on the test set

print("Predicting people's names on the test set")

t0 = [time](https://docs.python.org/3/library/time.html#time.time)()

y\_pred = clf.predict(X\_test\_pca)

print("done in *%0.3f*s" % ([time](https://docs.python.org/3/library/time.html#time.time)() - t0))

print([classification\_report](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report)(y\_test, y\_pred, target\_names=target\_names))

[ConfusionMatrixDisplay.from\_estimator](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics.ConfusionMatrixDisplay.from_estimator)(

clf, X\_test\_pca, y\_test, display\_labels=target\_names, xticks\_rotation="vertical"

)

[plt.tight\_layout](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.tight_layout.html#matplotlib.pyplot.tight_layout)()

[plt.show](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html#matplotlib.pyplot.show)()