from sklearn.datasets import load\_digits

import pandas as pd

dataset = load\_digits()

dataset.keys()

Output

dict\_keys(['data', 'target', 'frame', 'feature\_names', 'target\_names', 'images', 'DESCR'])

dataset.data[0].reshape(8,8)

output

array([[ 0., 0., 5., 13., 9., 1., 0., 0.],

[ 0., 0., 13., 15., 10., 15., 5., 0.],

[ 0., 3., 15., 2., 0., 11., 8., 0.],

[ 0., 4., 12., 0., 0., 8., 8., 0.],

[ 0., 5., 8., 0., 0., 9., 8., 0.],

[ 0., 4., 11., 0., 1., 12., 7., 0.],

[ 0., 2., 14., 5., 10., 12., 0., 0.],

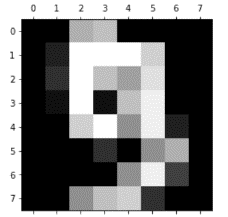
[ 0., 0., 6., 13., 10., 0., 0., 0.]])

from matplotlib import pyplot as plt

%matplotlib inline

plt.gray()

plt.matshow(dataset.data[9].reshape(8,8))



dataset.target[9]

Output

9

df = pd.DataFrame(dataset.data, columns=dataset.feature\_names)

X = df

y = dataset.target

**from** sklearn.preprocessing **import** StandardScaler

scaler **=** StandardScaler()

X\_scaled **=** scaler**.**fit\_transform(X)

**from** sklearn.linear\_model **import** LogisticRegression

model **=** LogisticRegression()

model**.**fit(X\_train, y\_train)

model**.**score(X\_test, y\_test)

Output

0.92

USAREMOS PCA PARA RECUDIR FEATURES

len(df.columns)

Output

64

**Use components such that 95% of variance is retained**

from sklearn.decomposition import PCA

pca = PCA(0.95)

X\_pca = pca.fit\_transform(X)

pca.explained\_variance\_ratio\_

pca.n\_components\_

Output

29

X\_train\_pca, X\_test\_pca, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.2, random\_state=30)

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression(max\_iter=1000)

model.fit(X\_train\_pca, y\_train)

model.score(X\_test\_pca, y\_test)

**PCA created 29 components out of 64 original columns**

X\_pca

X\_train\_pca, X\_test\_pca, y\_train, y\_test **=** train\_test\_split(X\_pca, y, test\_size**=**0.2, random\_state**=**30)

**from** sklearn.linear\_model **import** LogisticRegression

model **=** LogisticRegression(max\_iter**=**1000)

model**.**fit(X\_train\_pca, y\_train)

model**.**score(X\_test\_pca, y\_test)

Out[53]:

0.96944

**Let's now select only two components**

pca **=** PCA(n\_components**=**2)

X\_pca **=** pca**.**fit\_transform(X)

X\_pca**.**shape

Out[95]:

(1797, 2)

X\_pca

Out[96]:

array([[ -1.25946639, 21.27487891],

[ 7.95760922, -20.76869518],

[ 6.99192341, -9.95598163],

...,

[ 10.80128435, -6.96025523],

[ -4.87210315, 12.42395926],

[ -0.34438701, 6.36554335]])

pca**.**explained\_variance\_ratio\_

Out[97]:

array([0.14890594, 0.13618771])

**You can see that both combined retains 0.14+0.13=0.27 or 27% of important feature information**

X\_train\_pca, X\_test\_pca, y\_train, y\_test **=** train\_test\_split(X\_pca, y, test\_size**=**0.2, random\_state**=**30)

model **=** LogisticRegression(max\_iter**=**1000)

model**.**fit(X\_train\_pca, y\_train)

model**.**score(X\_test\_pca, y\_test)

\*Out[98]:

0.6083333333333333

We get less accuancy (~60%) as using only 2 components did not retain much of the feature information. However in real life you will find many cases where using 2 or few PCA components can still give you a pretty good accuracy