ass4

September 22, 2021

[10]:
```
from sklearn.datasets import load_digits
import numpy
import pandas

digits = load_digits()

myData = pandas.DataFrame(data = digits.data)

display(myData)
```

```
       0    1     2     3     4     5    6    7    8    9   …   54   55  \
0     0.0  0.0   5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  …  0.0  0.0
1     0.0  0.0   0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  …  0.0  0.0
2     0.0  0.0   0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  …  5.0  0.0
3     0.0  0.0   7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  …  9.0  0.0
4     0.0  0.0   0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  …  0.0  0.0
…   …  …   …   …   …   …   …  …  …  …  …  …  …  …  …
1792  0.0  0.0   4.0  10.0  13.0   6.0  0.0  0.0  0.0  1.0  …  4.0  0.0
1793  0.0  0.0   6.0  16.0  13.0  11.0  1.0  0.0  0.0  0.0  …  1.0  0.0
1794  0.0  0.0   1.0  11.0  15.0   1.0  0.0  0.0  0.0  0.0  …  0.0  0.0
1795  0.0  0.0   2.0  10.0   7.0   0.0  0.0  0.0  0.0  0.0  …  2.0  0.0
1796  0.0  0.0  10.0  14.0   8.0   1.0  0.0  0.0  0.0  2.0  …  8.0  0.0

       56   57   58    59    60    61   62   63
0     0.0  0.0  6.0  13.0  10.0   0.0  0.0  0.0
1     0.0  0.0  0.0  11.0  16.0  10.0  0.0  0.0
2     0.0  0.0  0.0   3.0  11.0  16.0  9.0  0.0
3     0.0  0.0  7.0  13.0  13.0   9.0  0.0  0.0
4     0.0  0.0  0.0   2.0  16.0   4.0  0.0  0.0
…   …  …  …  …   …   …  …  …  …
1792  0.0  0.0  2.0  14.0  15.0   9.0  0.0  0.0
1793  0.0  0.0  6.0  16.0  14.0   6.0  0.0  0.0
1794  0.0  0.0  2.0   9.0  13.0   6.0  0.0  0.0
1795  0.0  0.0  5.0  12.0  16.0  12.0  0.0  0.0
1796  0.0  1.0  8.0  12.0  14.0  12.0  1.0  0.0

[1797 rows x 64 columns]
```

[11]:
```
# normalization using Z score
from sklearn.preprocessing import StandardScaler
X_scaled = StandardScaler().fit_transform(myData)

display(X_scaled)

# transpose of X scaled
display(X_scaled.T)
```

```
array([[ 0.        , -0.33501649, -0.04308102, …, -1.14664746,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, …,  0.54856067,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, …,  1.56568555,
         1.6951369 , -0.19600752],
       …,
       [ 0.        , -0.33501649, -0.88456568, …, -0.12952258,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -0.67419451, …,  0.8876023 ,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649,  1.00877481, …,  0.8876023 ,
        -0.26113572, -0.19600752]])
```

```
array([[ 0.        ,  0.        ,  0.        , …,  0.        ,
         0.        ,  0.        ],
       [-0.33501649, -0.33501649, -0.33501649, …, -0.33501649,
        -0.33501649, -0.33501649],
       [-0.04308102, -1.09493684, -1.09493684, …, -0.88456568,
        -0.67419451,  1.00877481],
       …,
       [-1.14664746,  0.54856067,  1.56568555, …, -0.12952258,
         0.8876023 ,  0.8876023 ],
       [-0.5056698 , -0.5056698 ,  1.6951369 , …, -0.5056698 ,
        -0.5056698 , -0.26113572],
       [-0.19600752, -0.19600752, -0.19600752, …, -0.19600752,
        -0.19600752, -0.19600752]])
```

[12]:
```
# find covariance matrix
features = X_scaled.T
cov_matrix = numpy.cov(features)

display(cov_matrix)
```

```
array([[ 0.        ,  0.        ,  0.        , …,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  1.00055679,  0.55692803, …, -0.02988686,
         0.02656195, -0.04391324],
       [ 0.        ,  0.55692803,  1.00055679, …, -0.04120565,
```

```
        0.07263924,  0.08256908],
       …,
       [ 0.        , -0.02988686, -0.04120565, …,  1.00055679,
        0.64868875,  0.26213704],
       [ 0.        ,  0.02656195,  0.07263924, …,  0.64868875,
        1.00055679,  0.62077355],
       [ 0.        , -0.04391324,  0.08256908, …,  0.26213704,
        0.62077355,  1.00055679]])
```

[13]:
```
# find eigen values and eigen vectors
values, vectors = numpy.linalg.eig(cov_matrix)

display(values)

display(vectors)
```

```
array([7.34477606, 5.83549054, 5.15396118, 3.96623597, 2.9663452 ,
       2.57204442, 2.40600941, 2.06867355, 1.82993314, 1.78951739,
       1.69784616, 1.57287889, 1.38870781, 1.35933609, 1.32152536,
       1.16829176, 1.08368678, 0.99977862, 0.97438293, 0.90891242,
       0.82271926, 0.77631014, 0.71155675, 0.64552365, 0.59527399,
       0.5765018 , 0.52673155, 0.5106363 , 0.48686381, 0.45560107,
       0.44285155, 0.42230086, 0.3991063 , 0.39110111, 0.36094517,
       0.34860306, 0.3195963 , 0.29406627, 0.27692285, 0.05037444,
       0.06328961, 0.258273  , 0.24783029, 0.2423566 , 0.07635394,
       0.08246812, 0.09018543, 0.09840876, 0.10250434, 0.11188655,
       0.11932898, 0.12426371, 0.13321081, 0.14311427, 0.217582  ,
       0.15818474, 0.16875236, 0.20799593, 0.17612894, 0.2000909 ,
       0.18983516, 0.        , 0.        , 0.        ])
```

```
array([[ 0.        ,  0.        ,  0.        , …,  1.        ,
         0.        ,  0.        ],
       [ 0.18223392, -0.04702701,  0.02358821, …,  0.        ,
         0.        ,  0.        ],
       [ 0.285868  , -0.0595648 , -0.05679875, …,  0.        ,
         0.        ,  0.        ],
       …,
       [ 0.103198  ,  0.24261778, -0.02227952, …,  0.        ,
         0.        ,  0.        ],
       [ 0.1198106 ,  0.16508926,  0.10036559, …,  0.        ,
         0.        ,  0.        ],
       [ 0.07149362,  0.07132924,  0.09244589, …,  0.        ,
         0.        ,  0.        ]])
```

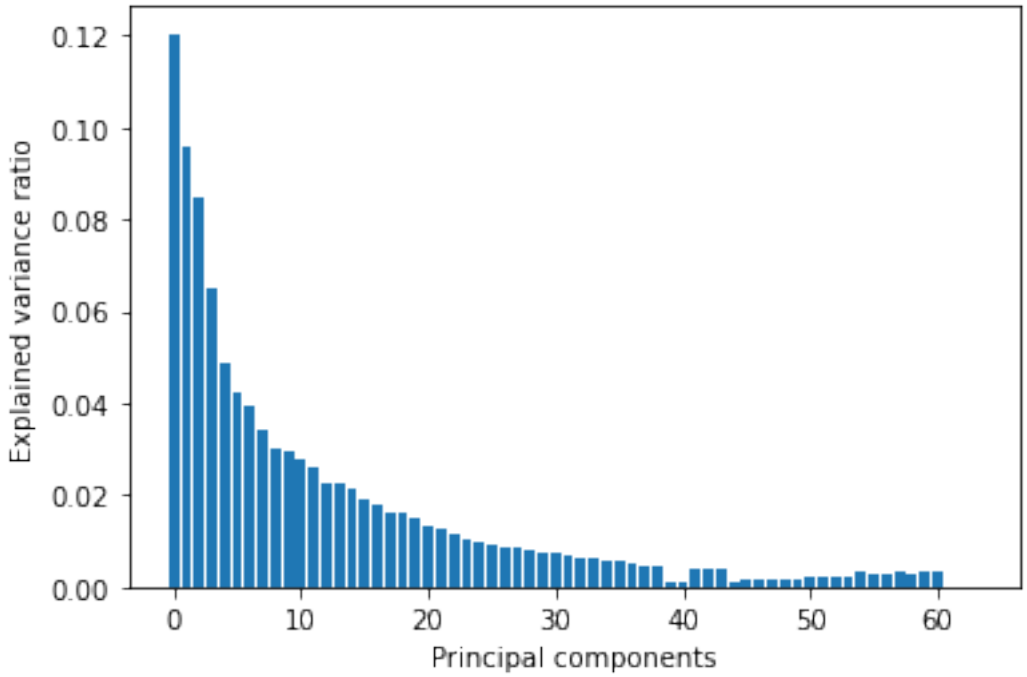
[14]:
```
# find explained variance to check which top k features to choose

from matplotlib import pyplot as plt
```

```
explained_variances = []
for i in range(len(values)):
    explained_variances.append(values[i] / numpy.sum(values))

display(explained_variances)

# plt.figure(figsize=(6, 4))
plt.bar(range(len(explained_variances)),␣
 ↪explained_variances,align='center',label='individual explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
```

```
[0.12033916097734892,
 0.09561054403097873,
 0.08444414892624551,
 0.06498407907524166,
 0.048601548759664055,
 0.042141198692719414,
 0.03942082803567392,
 0.03389380924638329,
 0.029982210116252288,
 0.029320025512522174,
 0.027818054635503367,
 0.02577055092581992,
 0.02275303315764242,
 0.02227179739514352,
 0.021652294318492464,
 0.019141666064421355,
 0.017755470851681932,
 0.01638069274284425,
 0.01596460168862353,
 0.01489191187087822,
 0.013479695658179346,
 0.012719313702347556,
 0.011658373505919532,
 0.010576465985363203,
 0.009753159471981106,
 0.009445589897319973,
 0.008630138269707224,
 0.008366428536685115,
 0.007976932484112409,
 0.00746471370926061,
 0.007255821513702756,
 0.006919112454811813,
 0.006539085355726171,
 0.006407925738459863,
 0.005913841117223419,
```

```
0.005711624052235243,
0.005236368034166354,
0.0048180758644514364,
0.004537192598584495,
0.0008253509448180381,
0.0010369573015571757,
0.004231627532327798,
0.004060530699790386,
0.003970848082758281,
0.001251007424973021,
0.0013511841133708583,
0.0014776269410608806,
0.0016123606225672954,
0.0016794638749558467,
0.0018331849919718227,
0.0019551242601981867,
0.0020359763452537645,
0.0021825685771200837,
0.0023448300553563523,
0.0035649330314261665,
0.002591749408814644,
0.0027648926352354672,
0.0034078718147029998,
0.0028857529410893402,
0.003278353352879543,
0.0031103200734535733,
0.0,
0.0,
0.0]
```

[14]: `Text(0.5, 0, 'Principal components')`



[15]:
```
# as you can see in the above graph - top ten features have most of the␣
↪significance


# making reduced data set

# projected Value is X normalized matrix . eigen vector

projectedList = []
colsNamesList = []

projected = X_scaled.dot(vectors.T[0])
reduced = pandas.DataFrame(projected, columns = ['PC1'])

for i in range(1 , 10):
    reduced["PC{}".format(i+1)] = X_scaled.dot(vectors.T[i])

reduced['Y'] = digits.target

display(reduced)
```

```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0 -1.914214  0.954502 -3.946035 -2.028723 -0.267173  0.530327  1.415321
1 -0.588980 -0.924636  3.924755  1.779850 -0.993430 -0.675652 -1.878565
2 -1.302039  0.317189  3.023333  2.043376 -2.081155  0.935121  1.296200
```

```
3     3.020770  0.868772 -0.801744  2.187039 -0.556813  0.727124 -0.959766
4    -4.528949  1.093480  0.973121  1.419510 -1.715106  1.431592 -1.073649
…         …         …         …         …         …         …         …
1792 -0.104331 -0.255024 -3.765861  1.947006 -0.190094 -0.555760 -0.531222
1793 -2.423234  1.429611 -3.045245 -2.632089 -0.822902  0.004061  1.106872
1794 -1.022596  0.147911  2.469974  0.620307 -0.972043 -0.007377 -0.381115
1795 -1.076055  0.380906 -2.455487  1.312013  0.253533 -0.638322 -1.034470
1796  1.257702  2.227591  0.283628  0.127073 -1.570173  2.342953  0.383075

           PC8       PC9       PC10  Y
0     1.496062  0.124914  0.822246  0
1     0.556336  1.079877 -0.087451  1
2     1.156160  0.785606  1.099206  2
3    -1.382638  0.259075 -0.744555  3
4    -0.968240 -1.660216 -1.174593  4
…         …         …         … ..
1792  0.476475  1.152430  0.473054  9
1793  2.330903  0.569455  1.654173  0
1794  0.529064  2.054709  2.036838  8
1795  0.763325  1.077474  0.334527  9
1796 -1.200566  0.816831  1.825428  8

[1797 rows x 11 columns]
```

[16]:
```
# using in built PCA function
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_new = pca.fit_transform(X_scaled)

display(X_new)
```

```
array([[ 1.91430678, -0.95400148, -3.94678904, …,  1.48116561,
         0.09798218, -0.8195825 ],
       [ 0.58895417,  0.92442305,  3.92490783, …,  0.5535355 ,
         1.0885217 ,  0.09290643],
       [ 1.30195331, -0.3174669 ,  3.02404268, …,  1.16454507,
         0.82086686, -1.08485351],
       …,
       [ 1.02255294, -0.14805297,  2.47011423, …,  0.52195177,
         2.11291453, -2.03480023],
       [ 1.0761348 , -0.38054162, -2.4563717 , …,  0.75339184,
         1.05133302, -0.32799044],
       [-1.25780692, -2.228135  ,  0.28467675, …, -1.17198335,
         0.85344087, -1.80414529]])
```

[ ]:
```

```