Untitled1

April 12, 2022

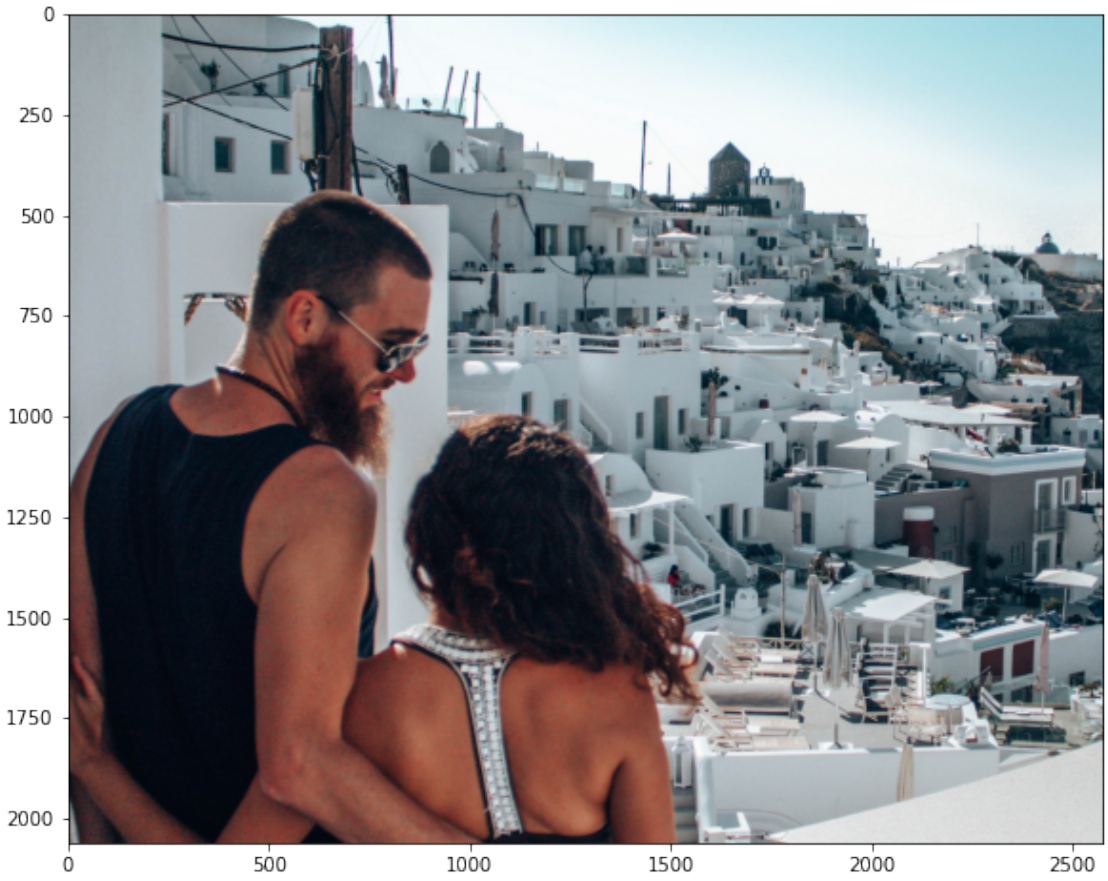
[2]:
```
# Kovarianzmatrix ist bekannt
# Eigenwerte und Eigenvektoren ist bekannt.
# Die Eigenvektoren der Kovarianzmatrix nennen wir "Hauptkomponenten"
# in englisch Principal Components. (PC)

# Diese geben die Hauptrichtungen in dem sich die Variabilitaet der Daten␣
 ↪befindet.

# Die PC Analyse hat viele Anwendungen in der Datenverarbeitung bzw. in der␣
 ↪Bildverarbeitung
```

[3]:
```
# Beispiel 1. Bildverarbeitung PCA
```

[5]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread

image_raw = imread('/Users/h4/Desktop/IMG_2197.jpeg') # aufladen vom bild
```

[6]:
```
# display vom bild

plt.figure(figsize=[12,8])
plt.imshow(image_raw)
```

[6]:
```
<matplotlib.image.AxesImage at 0x7ff68d71a070>
```



[7]:
```
"Das Bild ist Farbbild, d.h. es hat Daten in 3 Kanälen: ROT, GRÜN, BLAU RGB."
# Das farbige Bild ist also eine Matrix!
```

[7]: 'Das Bild ist Farbbild, d.h. es hat Daten in 3 Kanälen: ROT, GRÜN, BLAU RGB.'

[8]:
```
#Graue Skalierung
```

[10]:
```
image_sum = image_raw.sum(axis=2)
image_bw = image_sum/image_sum.max()

plt.figure(figsize=[12,8])
plt.imshow(image_bw, cmap=plt.cm.gray)
```

[10]: <matplotlib.image.AxesImage at 0x7ff680888cd0>

[11]:
```
# Hauptkomponenten vom Bild
```

[12]:
```
# Diese beschreiben 100% der Variabilität der Daten
```

[14]:
```
from sklearn.decomposition import PCA, IncrementalPCA
pca=PCA() # initialisierung vom PCA
pca.fit(image_bw) # hier erfolgt die PCA Analyse
```

[14]:
```
PCA()
```

[17]:
```
# kumulative Variabilität wird erstellt

var_cum= np.cumsum(pca.explained_variance_ratio_)*100
```

[21]:
```
# wie viele PCs erklären 95% der Variabilität von einem Bild

k = np.argmax(var_cum>95)

print('Anzahl Komponenten die 95% der Variabilität erklären:' + str(k))
```

```
Anzahl Komponenten die 95% der Variabilität erklären:37
```

[19]:
```
# image compression

ipca = IncrementalPCA(n_components=k)
image_recon = ipca.inverse_transform(ipca.fit_transform(image_bw))

plt.figure(figsize=[12,8])
plt.imshow(image_recon, cmap=plt.cm.gray)
```

[19]:
```
<matplotlib.image.AxesImage at 0x7ff6743501c0>
```

[22]:
```
# Beispiel 2.
```

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_wine # dataset wird hochgeladen
```

```
from sklearn.preprocessing import StandardScaler # normierung z=(x-mittelwert)/
 ↪StdAbweichung
from sklearn.decomposition import PCA
```

[24]:
```
wine = load_wine() # laden der Daten
```

[25]:
```
df = pd.DataFrame(wine.data, columns=wine.feature_names)
```

[26]:
```
print(df.iloc[:,0:4].head())
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash
0    14.23        1.71  2.43               15.6
1    13.20        1.78  2.14               11.2
2    13.16        2.36  2.67               18.6
3    14.37        1.95  2.50               16.8
4    13.24        2.59  2.87               21.0
```

[27]:
```
df
```

[27]:
```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0      14.23        1.71  2.43               15.6      127.0           2.80
1      13.20        1.78  2.14               11.2      100.0           2.65
2      13.16        2.36  2.67               18.6      101.0           2.80
3      14.37        1.95  2.50               16.8      113.0           3.85
4      13.24        2.59  2.87               21.0      118.0           2.80
..       …          …   …                  …         …              …
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          …                    …                …                …    …
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
176        0.68                  0.53             1.46             9.30  0.60
177        0.76                  0.56             1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
```

```
1                             3.40   1050.0
2                             3.17   1185.0
3                             3.45   1480.0
4                             2.93    735.0
..                             …       …
173                           1.74    740.0
174                           1.56    750.0
175                           1.56    835.0
176                           1.62    840.0
177                           1.60    560.0

[178 rows x 13 columns]
```

[30]:
```
# statistische untersuchung der Daten

print(df.describe())
```

```
          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000
mean    13.000618    2.336348    2.366517          19.494944   99.741573
std      0.811827    1.117146    0.274344           3.339564   14.282484
min     11.030000    0.740000    1.360000          10.600000   70.000000
25%     12.362500    1.602500    2.210000          17.200000   88.000000
50%     13.050000    1.865000    2.360000          19.500000   98.000000
75%     13.677500    3.082500    2.557500          21.500000  107.000000
max     14.830000    5.800000    3.230000          30.000000  162.000000

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000
mean        2.295112    2.029270              0.361854         1.590899
std         0.625851    0.998859              0.124453         0.572359
min         0.980000    0.340000              0.130000         0.410000
25%         1.742500    1.205000              0.270000         1.250000
50%         2.355000    2.135000              0.340000         1.555000
75%         2.800000    2.875000              0.437500         1.950000
max         3.880000    5.080000              0.660000         3.580000

       color_intensity         hue  od280/od315_of_diluted_wines      proline
count       178.000000  178.000000                    178.000000   178.000000
mean          5.058090    0.957449                      2.611685   746.893258
std           2.318286    0.228572                      0.709990   314.907474
min           1.280000    0.480000                      1.270000   278.000000
25%           3.220000    0.782500                      1.937500   500.500000
50%           4.690000    0.965000                      2.780000   673.500000
75%           6.200000    1.120000                      3.170000   985.000000
max          13.000000    1.710000                      4.000000  1680.000000
```

[35]:
```
# die Daten werden normiert Normal mit Mittelwert 0 und Std Abweichung 1
df = StandardScaler().fit_transform(df)
df
```

[35]:
```
array([[ 1.51861254, -0.5622498 ,  0.23205254, …,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, …,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, …,  0.31830389,
         0.78858745,  1.39514818],
       …,
       [ 0.33275817,  1.74474449, -0.38935541, …, -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, …, -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, …, -1.52437837,
        -1.42894777, -0.59516041]])
```

[36]:
```
df =pd.DataFrame(df,columns=wine.feature_names)
print(df.describe())
```

```
            alcohol    malic_acid           ash  alcalinity_of_ash  \
count  1.780000e+02  1.780000e+02  1.780000e+02       1.780000e+02
mean   7.983626e-17  1.197544e-16 -3.118604e-17       3.991813e-17
std    1.002821e+00  1.002821e+00  1.002821e+00       1.002821e+00
min   -2.434235e+00 -1.432983e+00 -3.679162e+00      -2.671018e+00
25%   -7.882448e-01 -6.587486e-01 -5.721225e-01      -6.891372e-01
50%    6.099988e-02 -4.231120e-01 -2.382132e-02       1.518295e-03
75%    8.361286e-01  6.697929e-01  6.981085e-01       6.020883e-01
max    2.259772e+00  3.109192e+00  3.156325e+00       3.154511e+00

          magnesium  total_phenols    flavanoids  nonflavanoid_phenols  \
count  1.780000e+02   1.780000e+02  1.780000e+02          1.780000e+02
mean  -7.983626e-17   1.596725e-16  3.991813e-17         -7.983626e-17
std    1.002821e+00   1.002821e+00  1.002821e+00          1.002821e+00
min   -2.088255e+00  -2.107246e+00 -1.695971e+00         -1.868234e+00
25%   -8.244151e-01  -8.854682e-01 -8.275393e-01         -7.401412e-01
50%   -1.222817e-01   9.595986e-02  1.061497e-01         -1.760948e-01
75%    5.096384e-01   8.089974e-01  8.490851e-01          6.095413e-01
max    4.371372e+00   2.539515e+00  3.062832e+00          2.402403e+00

       proanthocyanins  color_intensity           hue  \
count     1.780000e+02     1.780000e+02  1.780000e+02
mean     -1.197544e-16     5.987720e-17  2.794269e-16
std       1.002821e+00     1.002821e+00  1.002821e+00
min      -2.069034e+00    -1.634288e+00 -2.094732e+00
25%      -5.972835e-01    -7.951025e-01 -7.675624e-01
```

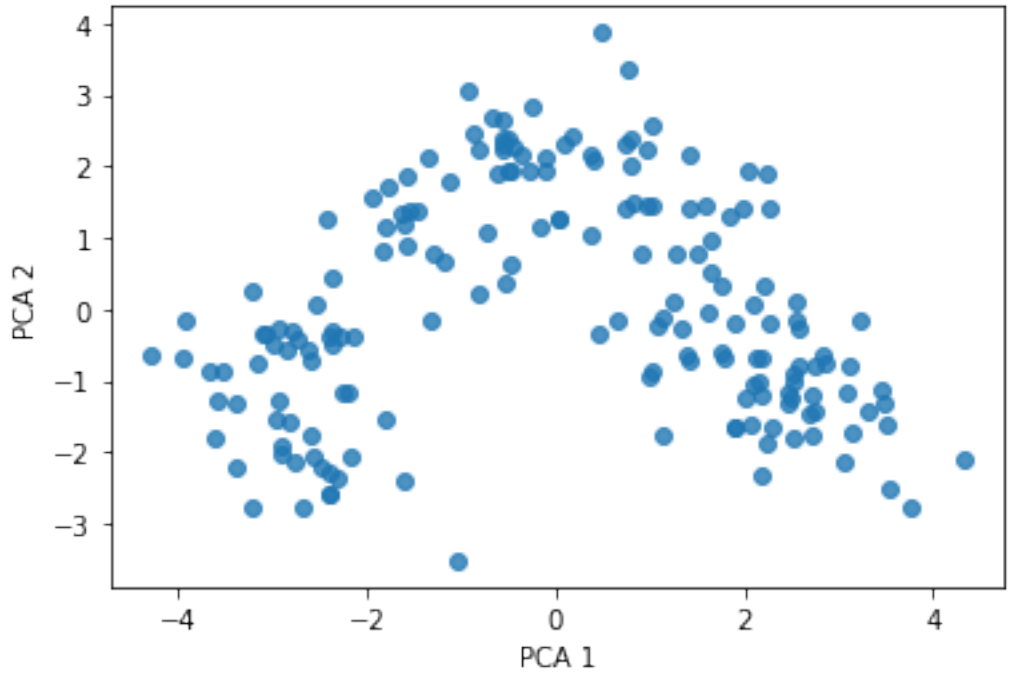
```
50%      -6.289785e-02    -1.592246e-01  3.312687e-02
75%       6.291754e-01     4.939560e-01  7.131644e-01
max       3.485073e+00     3.435432e+00  3.301694e+00

       od280/od315_of_diluted_wines       proline
count                    178.000000  1.780000e+02
mean                       0.000000 -3.991813e-17
std                        1.002821  1.002821e+00
min                       -1.895054 -1.493188e+00
25%                       -0.952248 -7.846378e-01
50%                        0.237735 -2.337204e-01
75%                        0.788587  7.582494e-01
max                        1.960915  2.971473e+00
```

[37]:
```
# die Daten werden mit 2 Hauptkomponenten beschrieben
```

[39]:
```
pca = PCA(n_components=2)
pca_model = pca.fit(df)
df_trans=pd.DataFrame(pca_model.transform(df), columns=['pca1', 'pca2'])

print(df_trans)
```

```
         pca1      pca2
0    3.316751 -1.443463
1    2.209465  0.333393
2    2.516740 -1.031151
3    3.757066 -2.756372
4    1.008908 -0.869831
..        …       …
173 -3.370524 -2.216289
174 -2.601956 -1.757229
175 -2.677839 -2.760899
176 -2.387017 -2.297347
177 -3.208758 -2.768920

[178 rows x 2 columns]
```

[40]:
```
# graphische Darstellung der Daten in 2 Dimensionen (Hauptkomponenten)

plt.scatter(df_trans['pca1'], df_trans['pca2'], alpha=0.8)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



[41]:
```
# erklärte Variabilität durch die PCs

print(pca_model.explained_variance_ratio_)
```

```
[0.36198848 0.1920749 ]
```

[42]:
```
"Der erste PC1 beschreibt 36,2% der Variabilität und der zweite PC2 beschreibt␣
↪19,2% der Variabilität."
```

[42]:
```
'Der erste PC1 beschreibt 36,2% der Variabilität und der zweite PC2 beschreibt
19,2% der Variabilität.'
```

[43]:
```
# Übung

# Ein User sollte eingeben können wie viele Hauptkomponenten er oder sie
# nutzen möchte um die Daten zu erklären.
# Bitte lade also die Daten auf Zeile24 hoch
# und erzeuge einen Inputbefehl, damit der Nutzer die erwünschte Anzahl an PCs␣
↪eingeben kann.
# Anschliessend bitte ermittle wie viel Variabilität dadurch beschrieben wird.
# Bitte drucke dies in eine Tabelle aus (Analog Zeile 39)
```

[ ]:
```

```