20220405_Wirtschaftsinformatik_FAT2

April 5, 2022

[16]:
```
# Beispiel 1.
```

[1]:
```
# Kovarianzmatrix
```

[2]:
```
DLZ = [84, 82, 81, 89, 73, 94, 92, 70, 88, 95]
Output = [85, 82, 72, 77, 75, 89, 95, 84, 77, 94]
Q = [97, 94, 93, 95, 88, 82, 78, 84, 69, 78]
```

[3]:
```
# Kovarianzmatrix

import numpy as np

data = np.array([DLZ, Output, Q])
```

[4]:
```
data
```

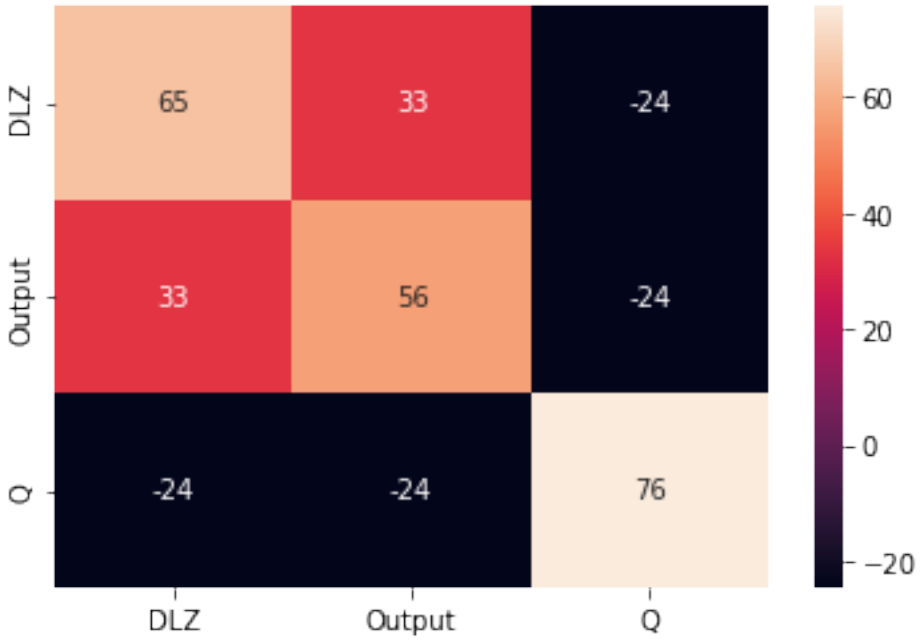
[4]:
```
array([[84, 82, 81, 89, 73, 94, 92, 70, 88, 95],
       [85, 82, 72, 77, 75, 89, 95, 84, 77, 94],
       [97, 94, 93, 95, 88, 82, 78, 84, 69, 78]])
```

[12]:
```
cov=np.cov(data, bias=True)
np.cov(data, bias=True)
```

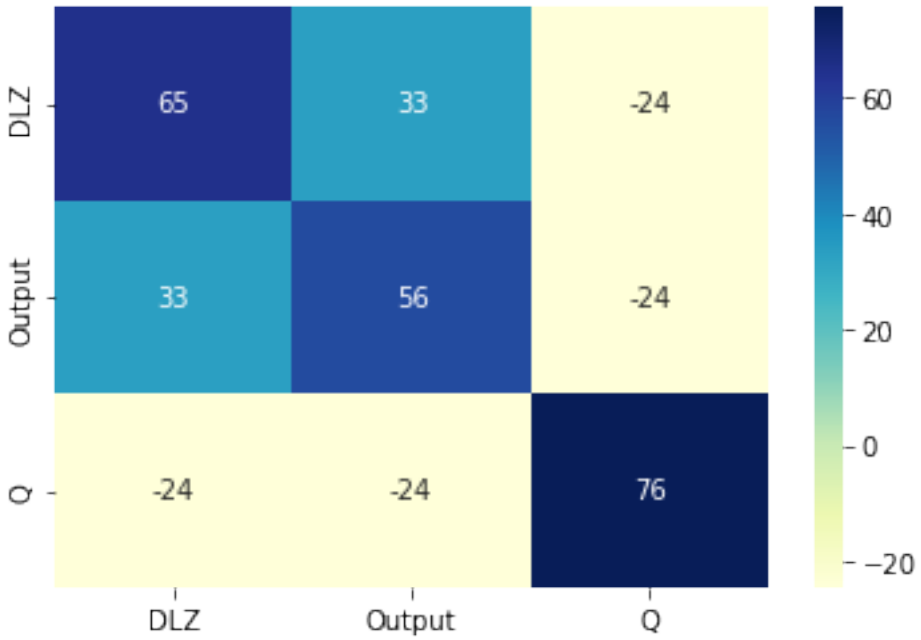
[12]:
```
array([[ 64.96,  33.2 , -24.44],
       [ 33.2 ,  56.4 , -24.1 ],
       [-24.44, -24.1 ,  75.56]])
```

[13]:
```
# graphische Darstellung der Kovarianzmatrix (Heatmap)
```

[14]:
```
labs = ['DLZ', 'Output', 'Q'] #labels

import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cov, annot=True, xticklabels=labs, yticklabels=labs)
plt.show()
```



[15]:
```
sns.heatmap(cov, annot=True, xticklabels=labs, yticklabels=labs, cmap='YlGnBu')
plt.show()
```

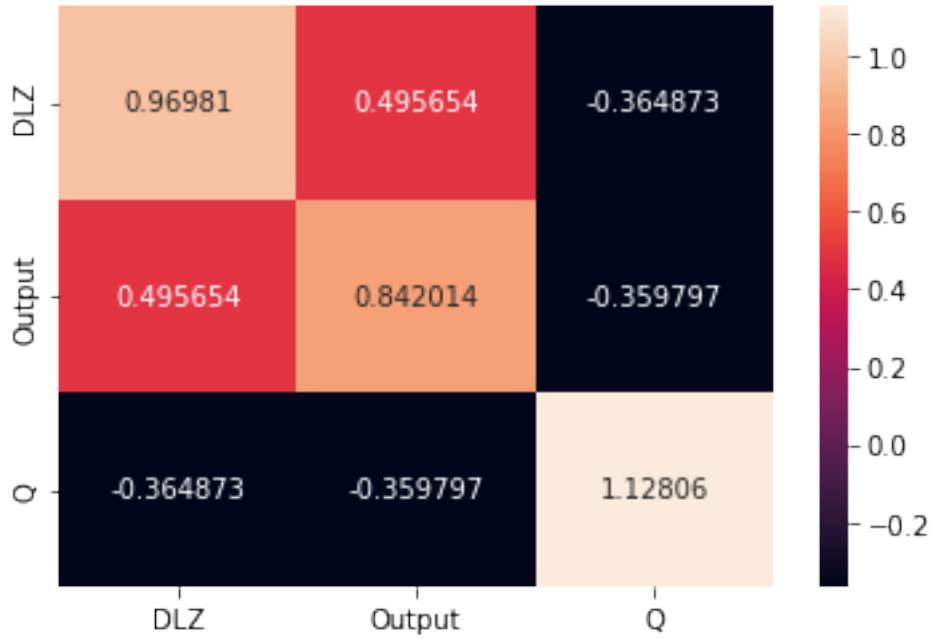


[17]:
```
# Beispiel 2.
```

[18]:
```
#Kovarianzmatrix mit Normierung
```

[19]:
```
data_norm = (data-np.mean(data))/np.std(data) #z=(x-mu)/sigma
```

[20]:
```
cov_norm = np.cov(data_norm, bias=True)

sns.heatmap(cov_norm, annot=True, fmt='g', xticklabels=labs, yticklabels=labs)
plt.show()
```



[21]:
```
#Beispiel 3.
```

[22]:
```
#Eigenvektoren (eigenwerte) der Kovarianzmatrix
```

[23]:
```
import scipy.linalg as la
```

[24]:
```
results = la.eig(cov_norm)
results
```

[24]:
```
(array([1.79718289+0.j, 0.40430681+0.j, 0.73839484+0.j]),
 array([[-0.58787377, -0.63449329,  0.50181939],
        [-0.53396503,  0.77034941,  0.34848691],
        [ 0.60768888,  0.06308769,  0.79166544]]))
```

[25]:
```
# Beispiel 4 ;-)
```

[26]:
```
# Normierung nach Datenreihen
```

[27]:
```
"""Bitte wiederholen Sie das Beispiel Nr. 2
mit einer Normierung der Mittelwerte nach Spalten.
Das heisst nicht mit dem gesamten Mittelwert und Std Abweichung
sondern mit dem Mittelwert der jeweiligen Datensatz."""
```

[27]:
```
'Bitte wiederholen Sie das Beispiel Nr. 2\nmit einer Normierung der Mittelwerte
nach Spalten.\nDas heisst nicht mit dem gesamten Mittelwert und Std
Abweichung\nsondern mit dem Mittelwert der jeweiligen Datensatz.'
```

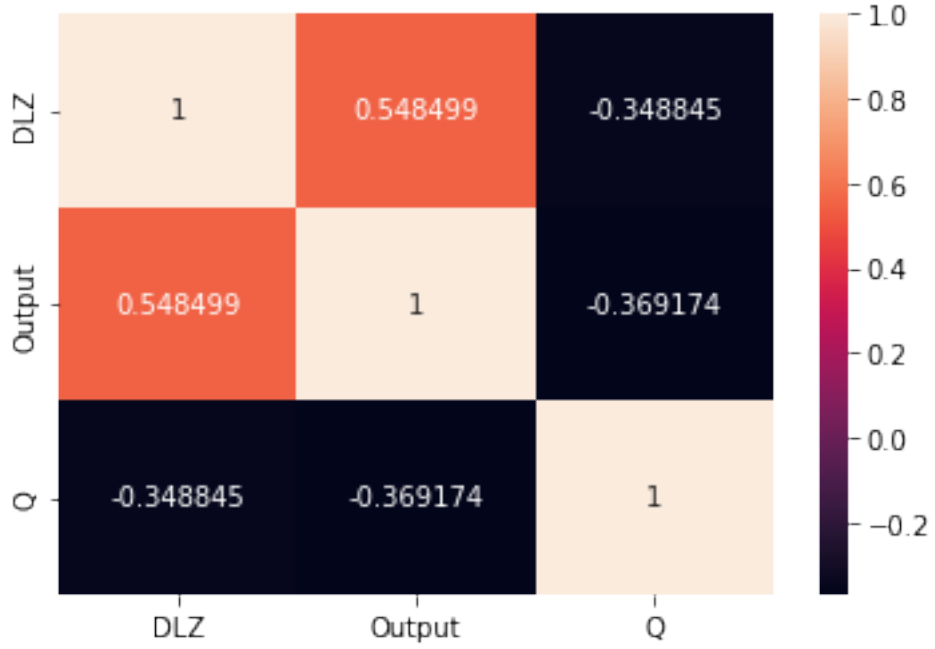
[28]:
```
DLZ_norm = (DLZ-np.mean(DLZ))/np.std(DLZ)
Output_norm = (Output-np.mean(Output))/np.std(Output)
Q_norm = (Q-np.mean(Q))/np.std(Q)
```

[29]:
```
data_norm2 = np.array([DLZ_norm, Output_norm, Q_norm])
```

[30]:
```
np.cov(data_norm2, bias=True)
```

[30]:
```
array([[ 1.        ,  0.5484986 , -0.34884477],
       [ 0.5484986 ,  1.        , -0.36917435],
       [-0.34884477, -0.36917435,  1.        ]])
```

[31]:
```
cov_norm2 = np.cov(data_norm2, bias=True)
labs = ['DLZ', 'Output', 'Q'] # labels
sns.heatmap(cov_norm2, annot=True, fmt='g', xticklabels=labs, yticklabels=labs)
plt.show()
```



[36]:
```
#Beispiel 5.
```

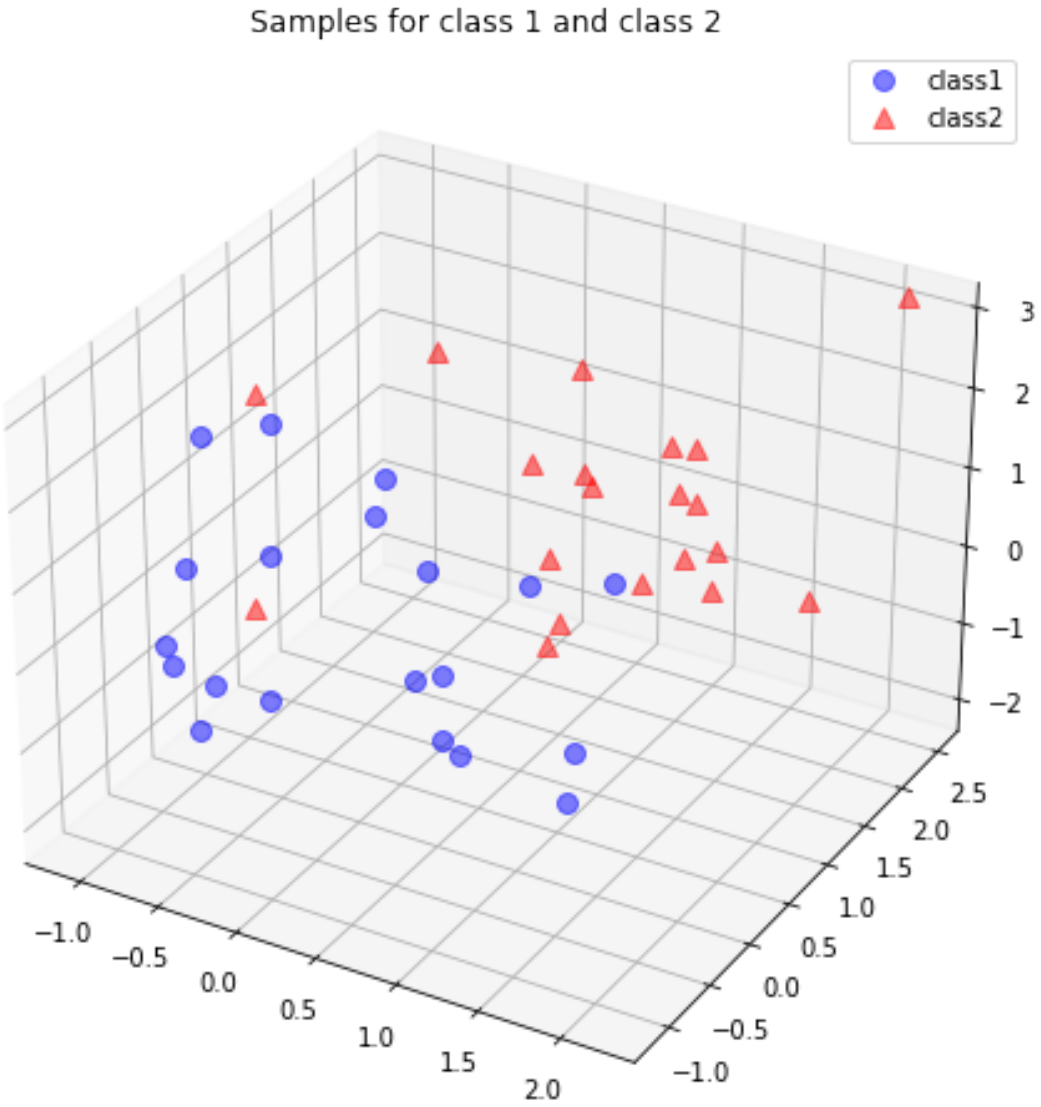
[37]:
```
# Graphische Darstellung. Eigenwerte und Eigenvektoren der Kovarianzmatrix
# Hauptkomponenten.
```

[38]:
```
# nicht Pruefungsrelevant

# Daten Generierung

np.random.seed(1)

mu_vec1 = np.array([0,0,0])
cov_mat1 = np.array([[1,0,0],[0,1,0],[0,0,1]])
class1_sample = np.random.multivariate_normal(mu_vec1, cov_mat1, 20).T
assert class1_sample.shape == (3,20), "The matrix has not the dimensions 3x20"

mu_vec2 = np.array([1,1,1])
cov_mat2 = np.array([[1,0,0],[0,1,0],[0,0,1]])
class2_sample = np.random.multivariate_normal(mu_vec2, cov_mat2, 20).T
assert class2_sample.shape == (3,20), "The matrix has not the dimensions 3x20"
```

[39]:
```
# nicht Pruefungsrelevant

# graphische Darstellung

%pylab inline
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import proj3d

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')

plt.rcParams['legend.fontsize'] = 10
ax.plot(class1_sample[0,:], class1_sample[1,:], class1_sample[2,:
 ↪],'o',markersize=8, color='blue', alpha=0.5, label='class1')
ax.plot(class2_sample[0,:], class2_sample[1,:], class2_sample[2,:
 ↪],'^',markersize=8, alpha=0.5, color='red', label='class2')
plt.title('Samples for class 1 and class 2')
ax.legend(loc='upper right')
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
/Users/h4/opt/anaconda3/lib/python3.8/site-
```

```
packages/IPython/core/magics/pylab.py:159: UserWarning: pylab import has
clobbered these variables: ['cov']
`%matplotlib` prevents importing * from pylab and numpy
  warn("pylab import has clobbered these variables: %s"  % clobbered +
```



[41]:
```
# Hauptkomponenten der Daten

all_samples = np.concatenate((class1_sample, class2_sample), axis=1)
assert all_samples.shape == (3,40)
```

[42]:
```
# Mittelwertvektor
```

```
mean_x = np.mean(all_samples[0,:])
mean_y = np.mean(all_samples[1,:])
mean_z = np.mean(all_samples[2,:])
mean_vector = np.array([[mean_x],[mean_y],[mean_z]])
print('Mean Vector:\n', mean_vector)
```

```
Mean Vector:
 [[0.41667492]
 [0.69848315]
 [0.49242335]]
```

[43]:
```
# Kovarianzmatrix

cov_mat = np.cov([all_samples[0,:],all_samples[1,:],all_samples[2,:]])
print('Covariance Matrix:\n', cov_mat)
```

```
Covariance Matrix:
 [[0.9868668  0.26943262 0.2855759 ]
 [0.26943262 0.92914135 0.30682016]
 [0.2855759  0.30682016 1.27528118]]
```

[44]:
```
# Eigenvektoren & Eigenwerte der Kovarianzmatrix

eig_val_cov, eig_vec_cov = np.linalg.eig(cov_mat)
```

[50]:
```
eig_val_cov # Eigenwerte
```

[50]:
```
array([1.67100943, 0.83832597, 0.68195393])
```

[51]:
```
eig_vec_cov # Eigenvektoren
```

[51]:
```
array([[-0.49210223, -0.64670286,  0.58276136],
       [-0.47927902, -0.35756937, -0.8015209 ],
       [-0.72672348,  0.67373552,  0.13399043]])
```

[ ]:
```

```