Untitled

April 11, 2024

0.1 basics statistik

[1]:
```
# Geschwindigkeiten einer Liste Autos
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
```

[2]:
```
import numpy
```

[3]:
```
x = numpy.mean(speed)
print(x)
```

```
89.76923076923077
```

[4]:
```
x = numpy.median(speed)
print(x)
```

```
87.0
```

[5]:
```
x = numpy.std(speed)
print(x)
```

```
9.258292301032677
```

0.2 erzeuge custom datasets

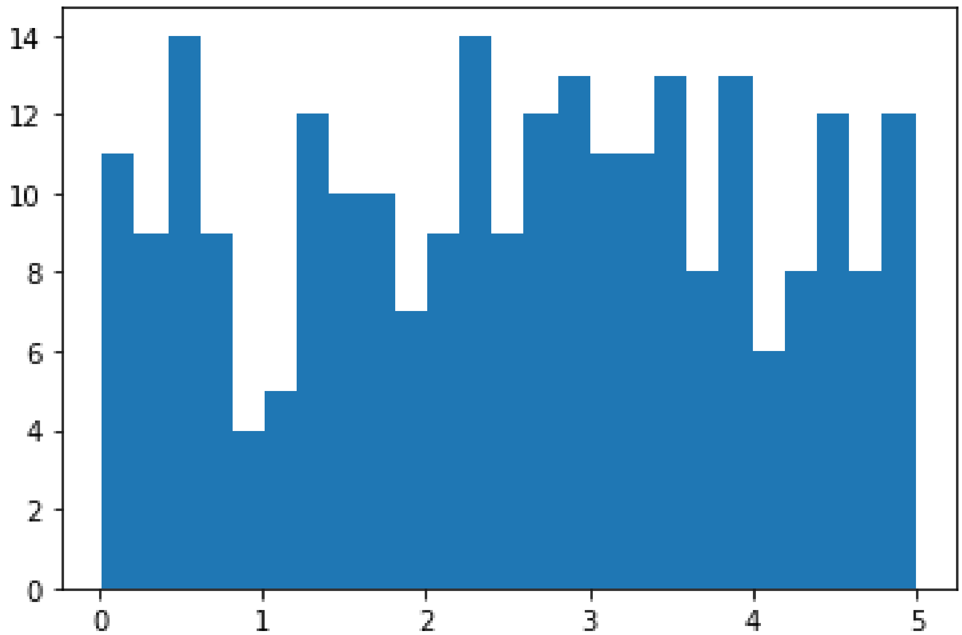
[14]:
```
x = numpy.random.uniform(0,5,250) #uniformverteilung
```

[10]:
```
print(x)
```

```
[0.06032085 3.41320662 4.50593933 2.94637761 0.1410724  0.03800528
 0.61358343 3.04186485 4.51038572 0.98294072 4.58864088 3.90476907
 4.41092348 0.38499656 1.78781045 2.80756224 4.84983127 2.92598717
 4.65906916 2.11561923 2.01634911 0.85464482 3.13599283 3.02666343
 1.81486456 3.863421   3.31720273 0.12463773 1.45120032 3.98421884
 3.06859232 0.37501715 2.75653545 0.72639817 2.78597801 3.10733904
 0.31204693 2.55718282 1.53773108 4.99270402 3.21495013 0.6232707
 3.31362495 2.63479695 4.3938338  0.47324732 3.30203586 3.1500965
 4.37854142 4.74025522 2.65357211 2.25992061 1.92873815 4.88152483
```

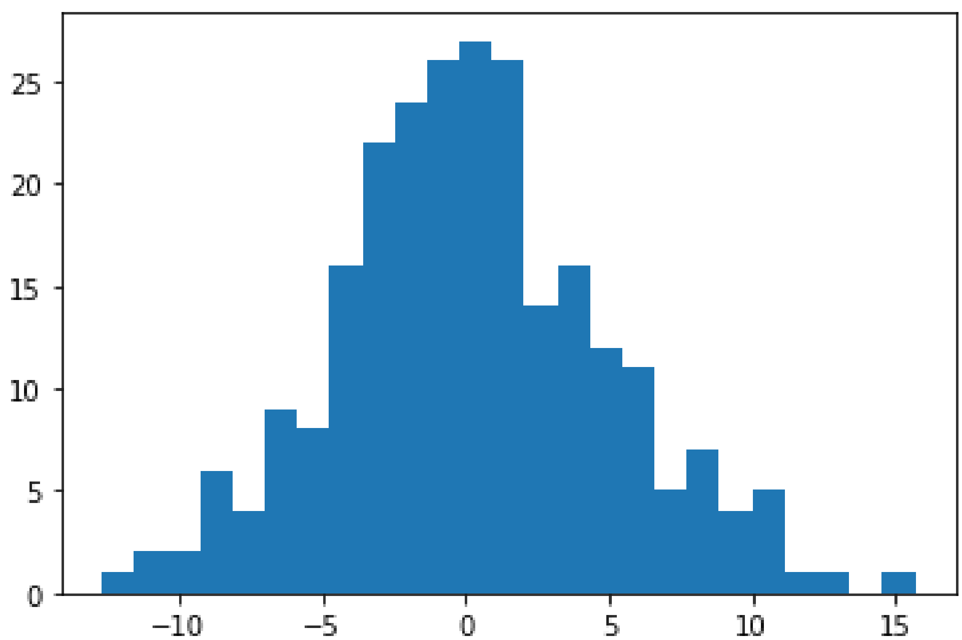
```
 2.15434613 4.38998886 0.30736193 2.3632409  4.04011302 4.11126082
 1.60374169 2.25418248 2.51992721 1.39224673 2.1363664  2.80107246
 2.42204608 2.11361577 3.64254711 1.26549142 1.41635017 2.96530357
 1.29425713 0.42862183 4.86972183 1.71812478 3.91118365 4.85783466
 0.54021423 4.78779081 3.0101047  1.35116234 0.43672425 3.22778641
 2.87084451 4.44010869 2.63860174 2.57718409 2.70430964 1.72733085
 4.30738876 3.56836191 0.55241982 2.2541157  1.5061269  3.57981334
 3.17810377 1.79708331 2.25928868 4.9927406  1.88717372 1.93011469
 2.38665356 3.94196158 3.6756224  2.98491367 4.88482235 1.23242381
 2.92971707 1.1780485  0.68346324 3.54768516 3.80014073 2.7203012
 4.70417976 2.88833576 4.61005805 2.98429002 0.01971999 0.18833381
 3.48840662 0.65998992 3.05333378 4.74968633 0.04010037 2.66009305
 1.56150523 2.40952926 2.93374553 0.51797123 4.57704366 4.81856347
 2.99439886 2.17979125 0.638195   3.89921225 3.08957522 0.51724147
 1.9259026  1.67232498 4.56993758 0.50004429 3.53970306 4.28094638
 3.83724574 0.90323611 0.81105364 4.15982666 0.27524222 1.32087187
 1.6925015  2.33706245 3.80631326 3.65370665 0.01805888 3.5808555
 0.58504058 1.66838627 1.40917914 4.57388805 3.81163106 2.51983384
 2.20914894 4.58949015 3.60304497 2.88494161 3.53935739 0.56043038
 3.9370061  1.63979289 2.32126025 2.83257741 0.02808154 2.23171813
 0.03305873 0.31463212 0.61397136 1.80897995 1.59131484 1.76998698
 4.31864891 3.9111814  1.77800511 3.58599208 0.63408624 4.5813537
 2.62056688 4.49768174 1.26283547 4.62449891 0.32565243 4.90413336
 2.39251465 0.76415258 3.30577558 4.14173351 3.60417479 3.31834342
 2.61329669 1.09748908 0.52201105 3.65577271 3.39536917 0.95301561
 3.63299523 4.33256744 1.5509261  3.87425944 3.22123705 1.18336251
 4.16734426 3.57210594 4.76150162 0.25141063 3.71440155 3.30466563
 0.2763223  1.33533215 3.59761502 3.46921096 1.24769155 2.45639717
 4.84758404 2.61370374 1.21273137 3.44693113 0.75688368 0.18610356
 1.08688544 1.45222248 3.39073385 4.12572989 2.36349516 2.57995574
 2.18933009 1.41458555 2.30596671 2.38552105 1.01483024 0.45526327
 1.21258965 4.26153392 1.99191466 3.09577282 4.81372709 2.06631544
 2.46672297 4.83009026 4.52474431 2.03129534]
```

[13]:
```
import matplotlib.pyplot as plt
plt.hist(x,25)
plt.show()
```

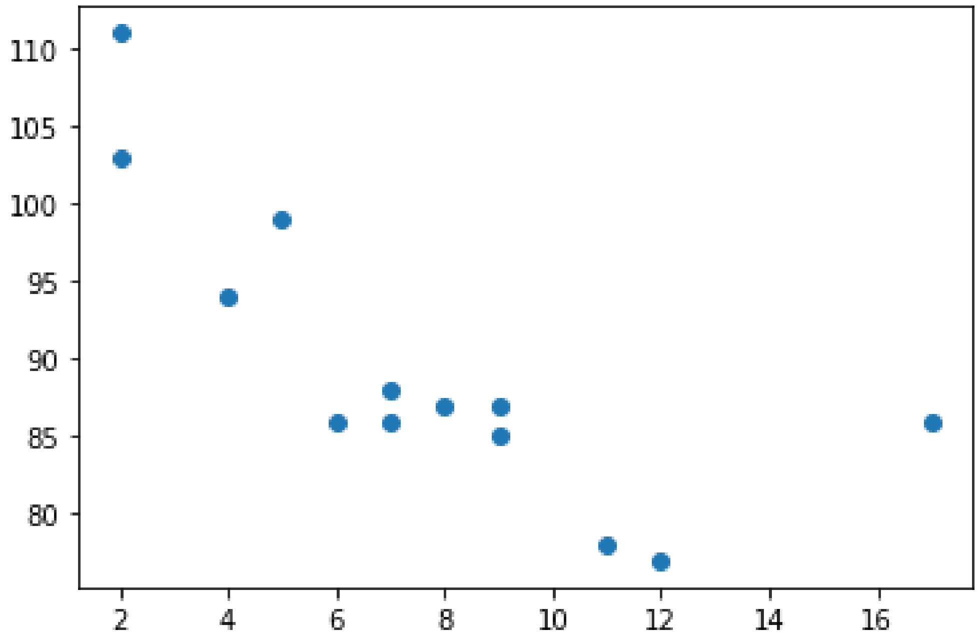


[15]:
```
x = numpy.random.normal(0,5,250) #normalverteilung
```

[16]:
```
import matplotlib.pyplot as plt
plt.hist(x,25)
plt.show()
```



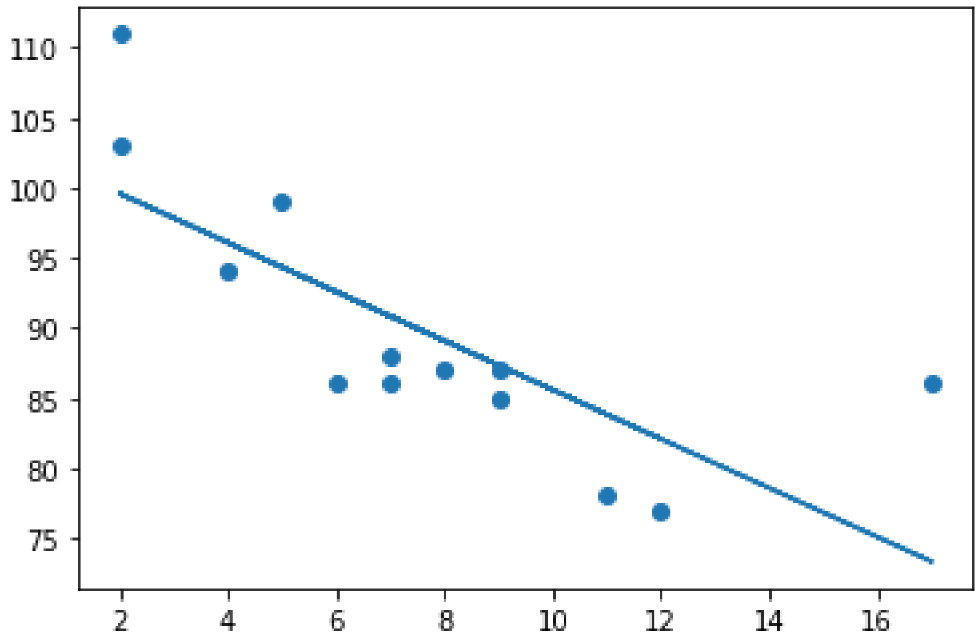
[17]:
```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x,y)
plt.show()
```



0.3 lineare regression

[18]:
```
from scipy import stats
```

[21]:
```
slope, intercept, r, p, std_err = stats.linregress(x,y)

def myfunc(x):
    return slope *x + intercept

mymodel = list(map(myfunc,x))

plt.scatter(x,y)
plt.plot(x,mymodel)
plt.show()
```

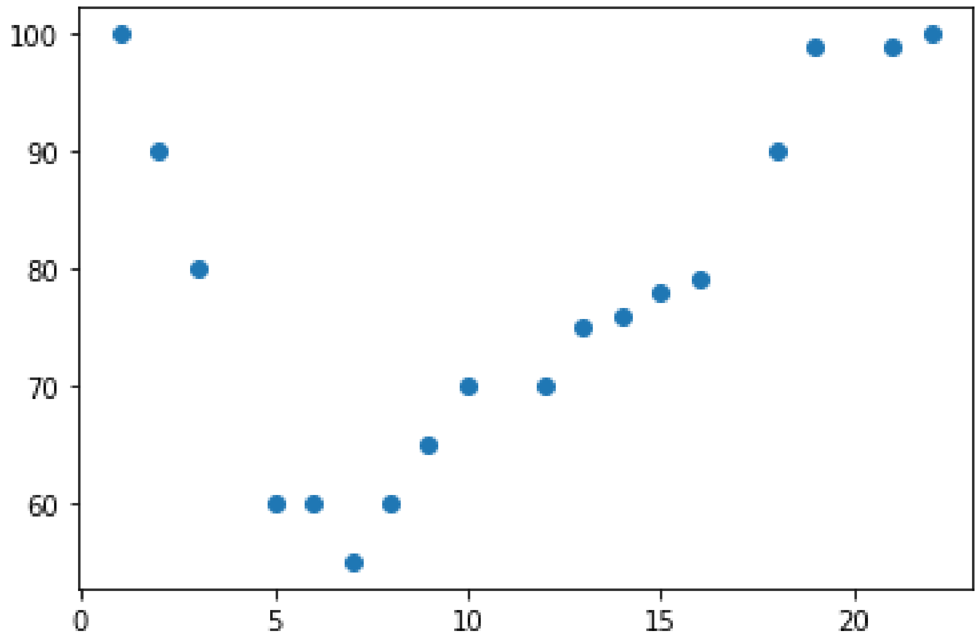


[22]:
```
speed = myfunc(10)
print(speed)
```

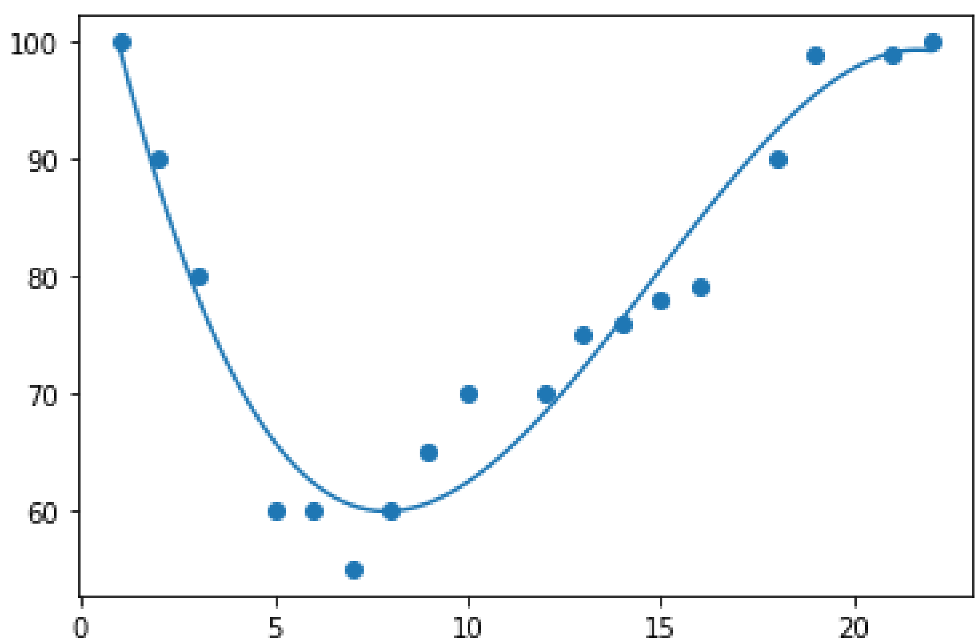
85.59308314937454

0.4 polynomische regression

[23]:
```
import matplotlib.pyplot as plt
x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
plt.scatter(x,y)
plt.show()
```



[29]:
```
mymodel = numpy.poly1d(numpy.polyfit(x,y,3))

myline = numpy.linspace(1,22,100)

plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```



0.5 echten Datensatz

[30]:
```
import pandas as pd
df = pd.read_csv('/Users/h4/desktop/cars.csv')
```

[31]:
```
df.head()
```

[31]:

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105

[32]:
```
X = df[['Weight', 'Volume']]
y = df['CO2']
```

[33]:
```
from sklearn import linear_model
```

[35]:
```
# !pip install sklearn (falls library nicht vorhanden)
```

[36]:
```
regression = linear_model.LinearRegression()
regression.fit(X,y)
```

[36]:
```
LinearRegression()
```

[37]:
```
predictedCO2 = regression.predict([[2300, 1300]])
```

```
/Users/h4/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450:
UserWarning: X does not have valid feature names, but LinearRegression was
fitted with feature names
  warnings.warn(
```

[38]:
```
predictedCO2
```

[38]:
```
array([107.2087328])
```

0.6 clustering

[40]:
```
wine = pd.read_csv('/Users/h4/desktop/wine.csv')
```

[41]:
```
wine.head()
```

```
[41]:    Cultivar  Alcohol  Malic acid   Ash  Alcalinity of ash   Magnesium  \
      0         1    14.23        1.71  2.43               15.6        127
      1         1    13.20        1.78  2.14               11.2        100
      2         1    13.16        2.36  2.67               18.6        101
      3         1    14.37        1.95  2.50               16.8        113
      4         1    13.24        2.59  2.87               21.0        118

         Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
      0           2.80        3.06                  0.28             2.29
      1           2.65        2.76                  0.26             1.28
      2           2.80        3.24                  0.30             2.81
      3           3.85        3.49                  0.24             2.18
      4           2.80        2.69                  0.39             1.82

         Color intensity   Hue  OD280/OD315 of diluted wines  Proline
      0             5.64  1.04                          3.92     1065
      1             4.38  1.05                          3.40     1050
      2             5.68  1.03                          3.17     1185
      3             7.80  0.86                          3.45     1480
      4             4.32  1.04                          2.93      735
```

```
[42]: # drop cultivar, da nicht notwendig für die clusterung

      wine = wine.drop('Cultivar', axis=1)
```

```
[43]: wine.head()
```

```
[43]:    Alcohol  Malic acid   Ash  Alcalinity of ash   Magnesium  Total phenols  \
      0    14.23        1.71  2.43               15.6        127           2.80
      1    13.20        1.78  2.14               11.2        100           2.65
      2    13.16        2.36  2.67               18.6        101           2.80
      3    14.37        1.95  2.50               16.8        113           3.85
      4    13.24        2.59  2.87               21.0        118           2.80

         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

         OD280/OD315 of diluted wines  Proline
      0                          3.92     1065
      1                          3.40     1050
      2                          3.17     1185
      3                          3.45     1480
      4                          2.93      735
```

[44]:
```python
from sklearn.cluster import KMeans
```

[46]:
```python
kmeans = KMeans(n_clusters = 3, random_state=42).fit(wine.values)
```

[47]:
```python
kmeans_3 = pd.DataFrame(kmeans.labels_, columns = ['cluster'])
print(kmeans_3)
```

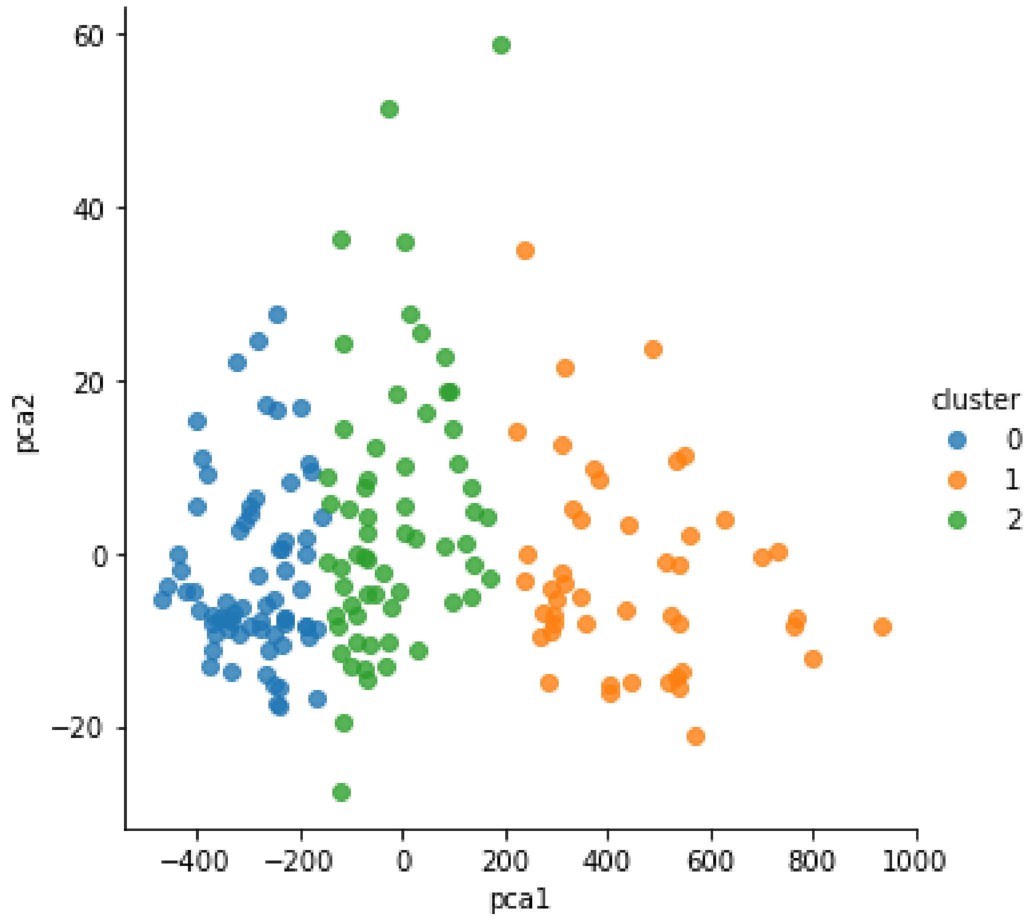
```
     cluster
0          1
1          1
2          1
3          1
4          2
..       …
173        2
174        2
175        2
176        2
177        0

[178 rows x 1 columns]
```

0.7 hauptkomponentenanalyse (principal component analysis)

[48]:
```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2).fit(wine)
```

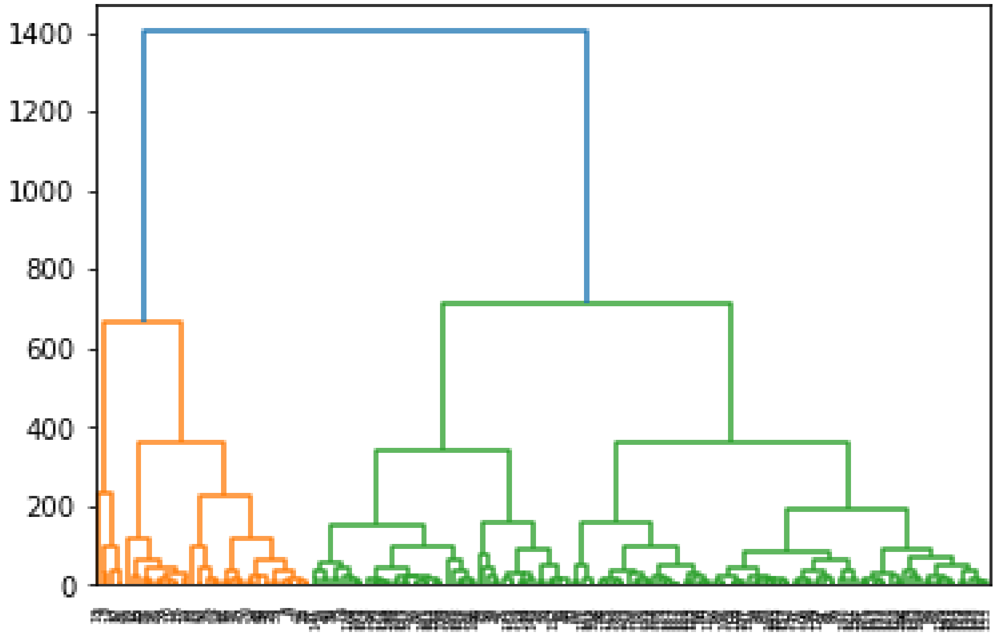
[50]:
```python
pca_trans = pca.transform(wine)
pca_trans_df = pd.DataFrame(pca_trans, columns = ['pca1', 'pca2'])
kmeans_3 = pd.concat([kmeans_3, pca_trans_df], axis= 1)
```

[51]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

fig = sns.lmplot(x='pca1', y='pca2', data=kmeans_3, hue='cluster',␣
 ↪fit_reg=False)
plt.show()
```



0.8 hierarchical clustering

[53]:
```
from scipy.cluster import hierarchy
wine_complete = hierarchy.complete(wine)
fig = plt.figure()
dn = hierarchy.dendrogram(wine_complete)
plt.show()
```

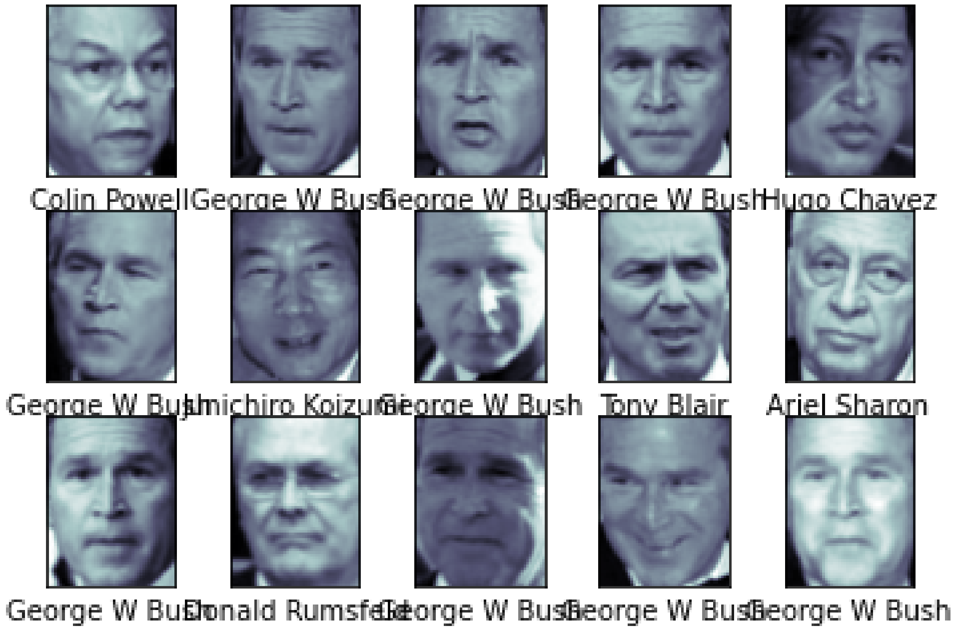


0.9 face recognition mit support vector machines

[54]:
```
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)
print(faces.target_names)
print(faces.images.shape)
```

```
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony Blair']
(1348, 62, 47)
```

[56]:
```
import matplotlib.pyplot as plt
fig, ax =plt.subplots(3,5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i],cmap='bone')
    axi.set(xticks=[], yticks=[],
        xlabel=faces.target_names[faces.target[i]])
```



[58]:
```python
from sklearn.svm import SVC
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.pipeline import make_pipeline

pca = RandomizedPCA(n_components = 90, whiten = True, random_state = 42)
svc = SVC(kernel='rbf', class_weight='balanced')
model = make_pipeline(pca,svc)
```

[59]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(faces.data, faces.target, random_state=42)
```

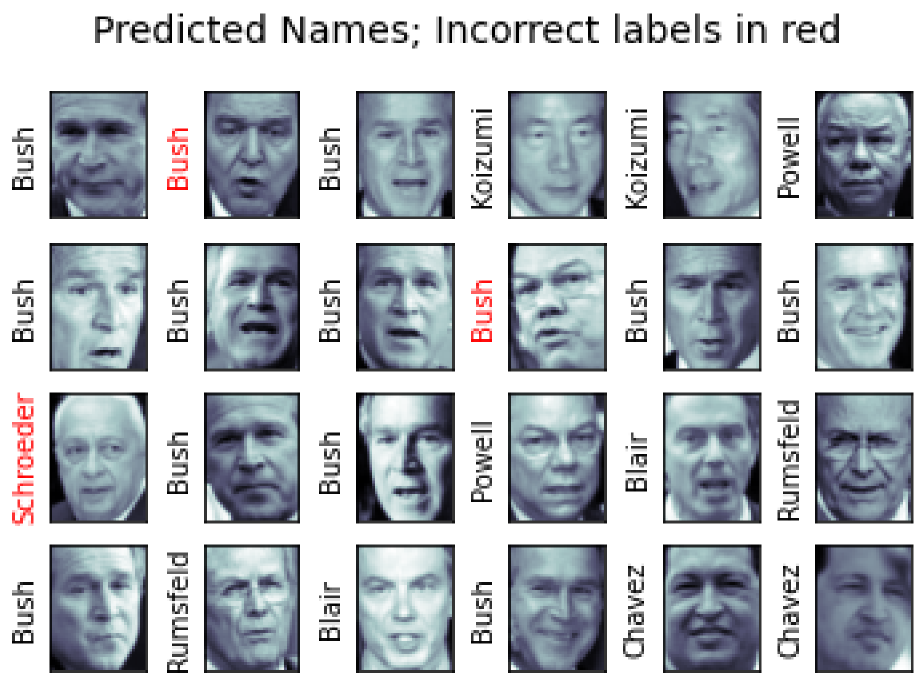
[61]:
```python
# Zeile nicht prüfungsrelevant

from sklearn.model_selection import GridSearchCV
param_grid = {'svc__C': [1, 5, 10, 50], 'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(model, param_grid)
%time grid.fit(X_train, y_train)
print(grid.best_params_)
```

```
CPU times: user 31min 45s, sys: 25min, total: 56min 46s
Wall time: 6min 35s
{'svc__C': 5, 'svc__gamma': 0.005}
```

[62]:
```
model = grid.best_estimator_
yfit = model.predict(X_test)
```

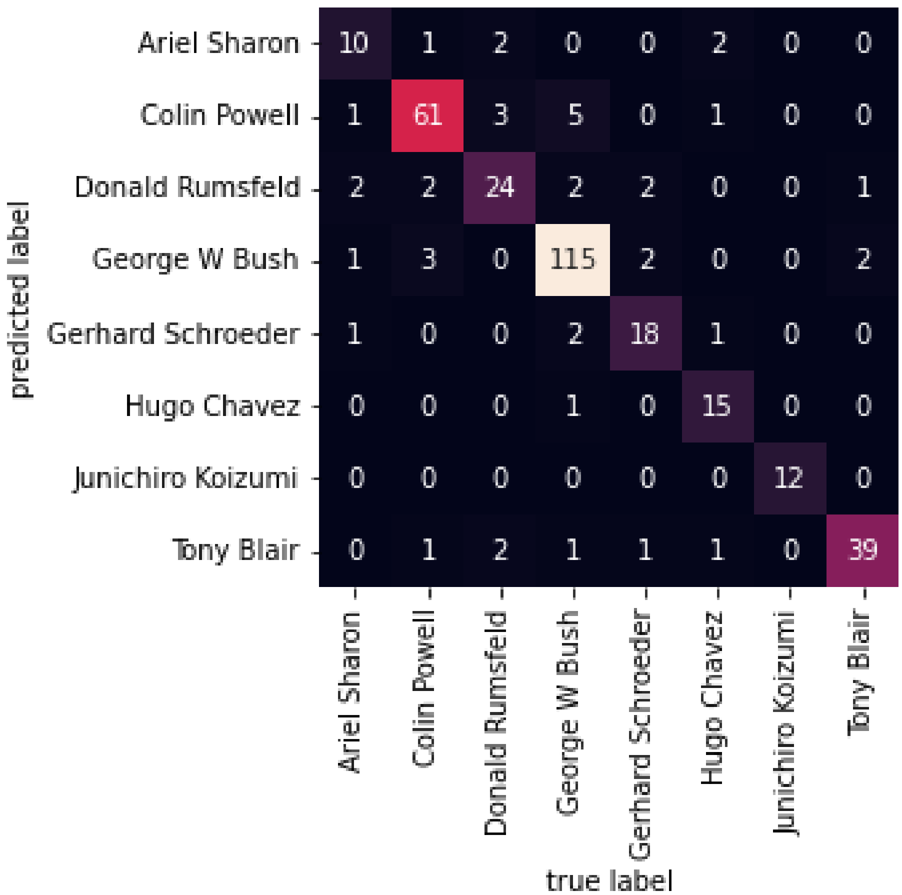
[66]:
```
fig, ax = plt.subplots(4,6)
for i, axi in enumerate(ax.flat):
    axi.imshow(X_test[i].reshape(62,47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
                   color='black' if yfit[i]==y_test[i] else 'red')
fig.suptitle('Predicted Names; Incorrect labels in red', size=14)
```

[66]: Text(0.5, 0.98, 'Predicted Names; Incorrect labels in red')



[67]:
```
# confusion matrix

from sklearn.metrics import confusion_matrix

mat = confusion_matrix(y_test, yfit)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,␣
 ↪xticklabels=faces.target_names,
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
```

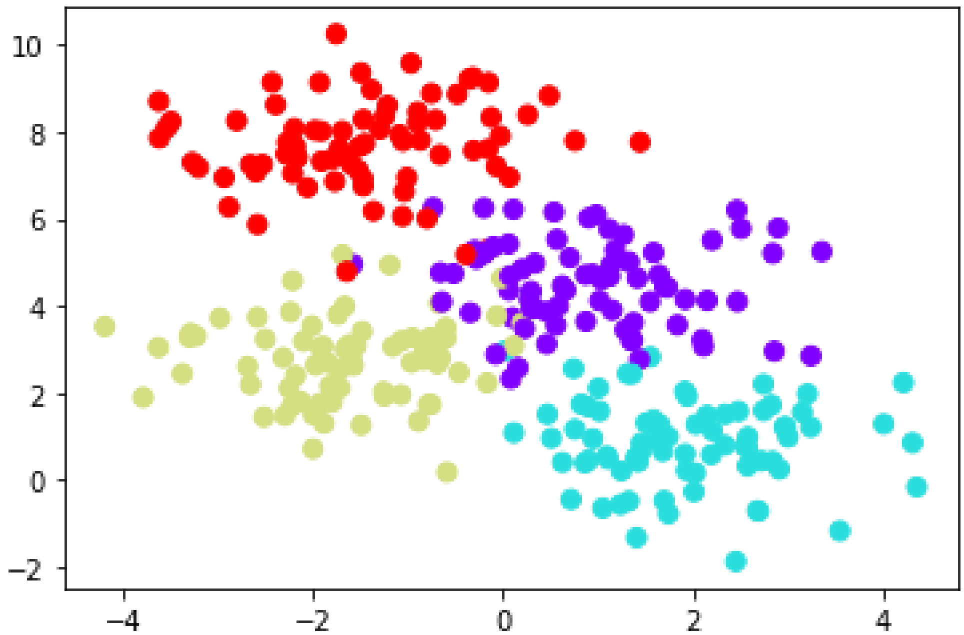
[67]: Text(91.68, 0.5, 'predicted label')



0.10 decission tree

[70]:
```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=300, centers=4, random_state=0, cluster_std=1)
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='rainbow')
```

[70]: <matplotlib.collections.PathCollection at 0x32fcafa90>



[71]:

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier().fit(X,y)
```

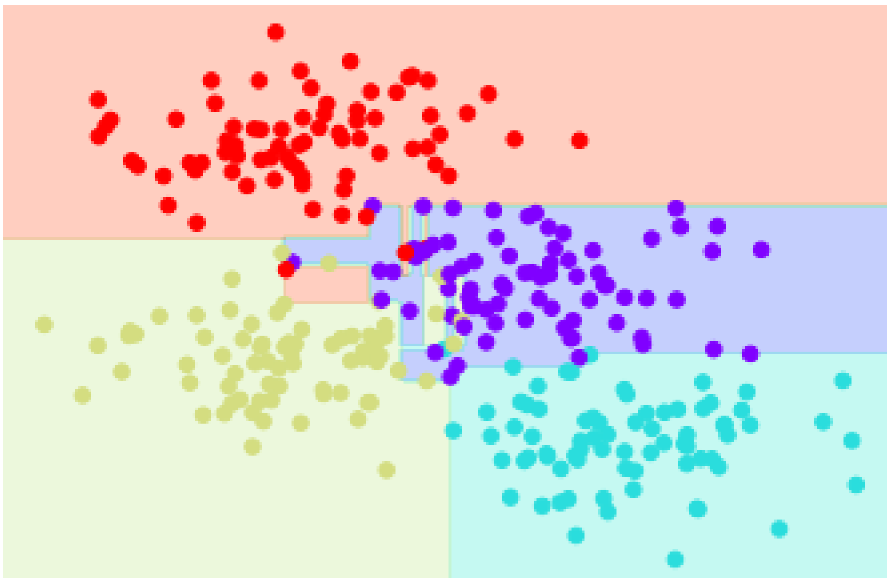
[76]:

```
#Zeile ist nicht prüfungsrelevant

import numpy as np

def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax = ax or plt.gca()
# Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
        clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    # fit the estimator
    model.fit(X, y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                         np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    # Create a color plot with the results
    n_classes = len(np.unique(y))
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                    levels=np.arange(n_classes + 1) - 0.5,
                    cmap=cmap, clim=(y.min(), y.max()),
```

```
                    zorder=1)
    ax.set(xlim=xlim, ylim=ylim)
```

[77]:
```
visualize_classifier(DecisionTreeClassifier(), X,y)
```

```
/var/folders/nw/k_k0_cbj7vl_npdmryvhl53c0000gn/T/ipykernel_46393/2167791633.py:2
1: UserWarning: The following kwargs were not used by contour: 'clim'
  contours = ax.contourf(xx, yy, Z, alpha=0.3,
```

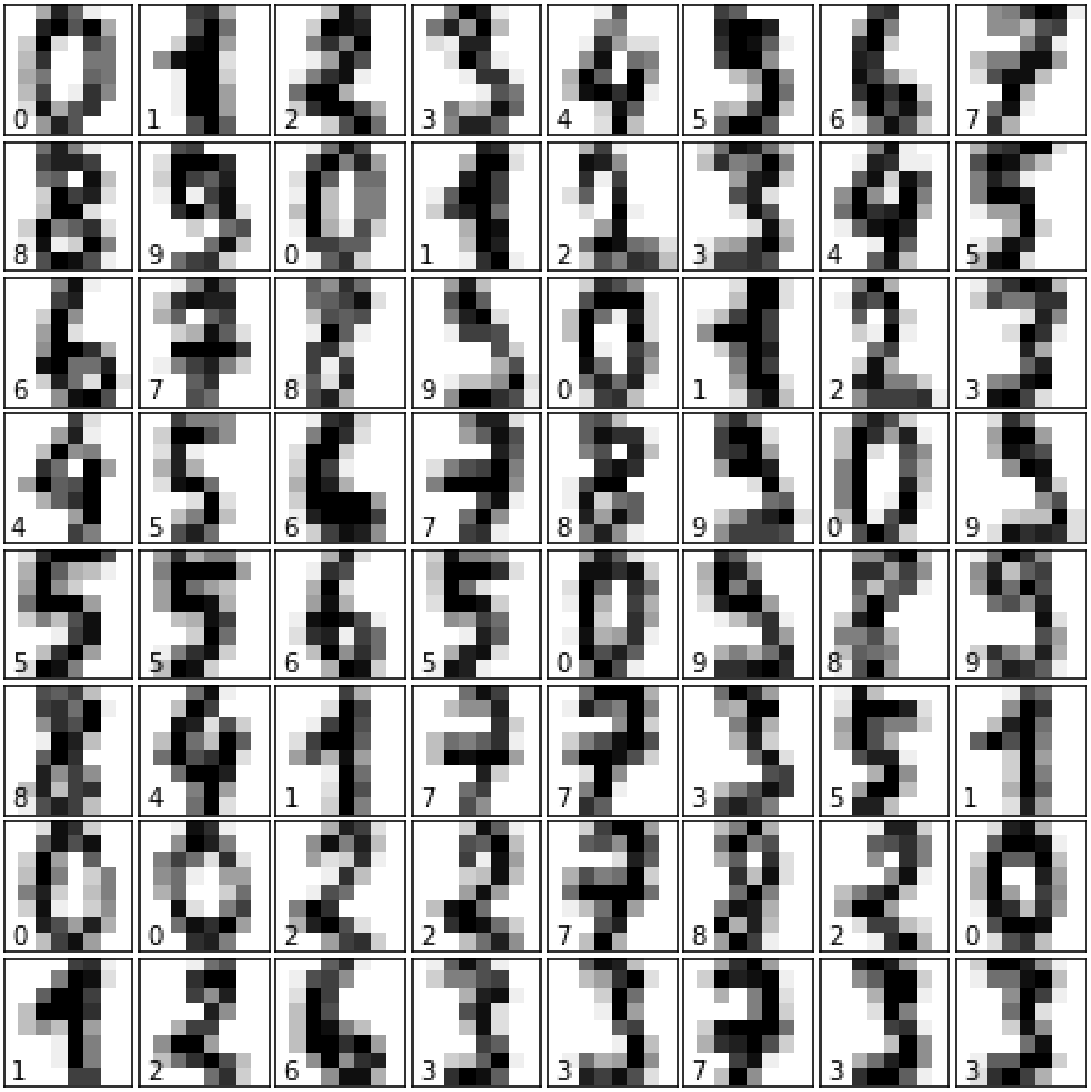


0.11 random forest für Bilder Klassifizierung

[78]:
```
from sklearn.datasets import load_digits
digits = load_digits()
digits.keys()
```

[78]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images',
'DESCR'])
```

[80]:
```
# Ziele nicht Prüfungsrelevant

fig = plt.figure(figsize=(6, 6))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
for i in range(64):
    ax = fig.add_subplot(8, 8, i + 1, xticks=[], yticks=[])
    ax.imshow(digits.images[i], cmap=plt.cm.binary, interpolation='nearest')
    ax.text(0, 7, str(digits.target[i]))
```



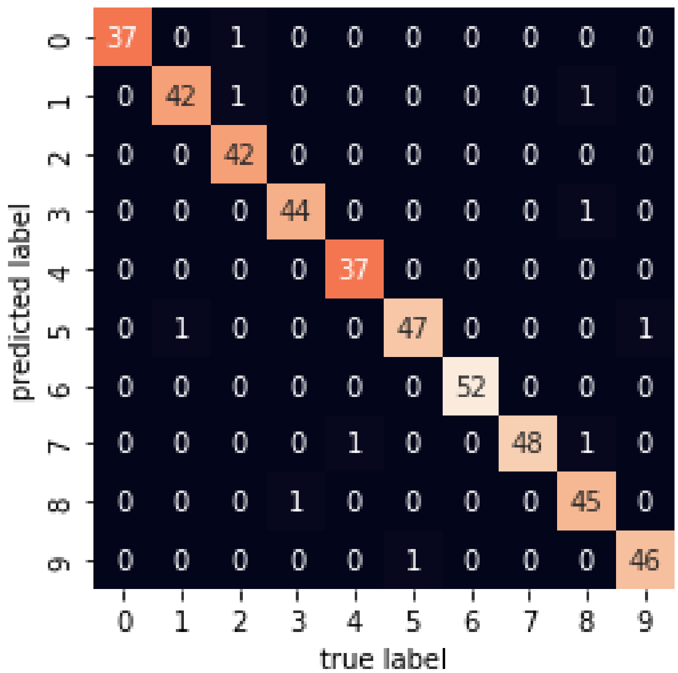
[82]:
```
from sklearn.model_selection import train_test_split
```

[83]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(digits.data, digits.target,␣
    ↪random_state=0)
```

[86]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=1000)
model.fit(Xtrain, ytrain)
ypred = model.predict(Xtest)
```

[87]:
```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
```

```python
plt.xlabel('true label')
plt.ylabel('predicted label');
```



[88]:

```python
##########
```

[ ]:

```python
# Maschine 1.

# Erzeugen Sie bitte einen Dataset mit 10 Dimensionen (10 KPIs).
# Die Dimensionen sollten 10 Variabeln von einer Maschine darstellen.
# Die Variabeln sind mit einer Weibul Verteilung ausgelegt mit Formparameter = 4,8 bei 3 Varibeln, 3,6 bei den Rest.
# Skalenparameter für Alle Variabeln 1.5.
# Pro KPI gibt es 2400 Datensätze.

# Bitte Erstellen einen Netzwerk mit den 10 dazu gehörigen Sensoren mit einem CC=2, und ein APL=2*ln(10).

# Bitte berechnen Sie den Laplacian vom Netzwerk. Dieser Lplacian Matrix sind die X und Y Dimensionen vom Zieldatset.
# Die Z-Dimension wird durch die KPIs darsgestellt, jeweils auf dem i=i Diagonale vom Laplacian.

# Somit ist unser Zieldataset (10,10, 2400).
```

```
# Maschine 2.
# wie Maschine 1 aber:
# Formparameter = 1.2 für 8 Sensoren und 2.3 für den Rest.
# CC=4, APL=ln(10)

## Sobald die Datensätze definiert wurden, bitte nutzen Sie PCA und K-Means␣
↪Cluster um beide Maschinen in einem 2 dimensionalen Raum darzustellen.
## Die Clusterung sollte erfolgen nach dem Formparameter der Weibul Verteilung.
```