20230413_Wirtschaftsinformatik_MPW2

April 13, 2023

[1]:
```
import numpy as np
```

[2]:
```
# mittelwert
```

[3]:
```
geschwindigkeit = [99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86]
```

[4]:
```
mittelwert = np.mean(geschwindigkeit)
```

[5]:
```
print(mittelwert)
```

```
89.76923076923077
```

[6]:
```
# standard abweichung
```

[8]:
```
std = np.std(geschwindigkeit)
```

[9]:
```
print(std)
```

```
9.258292301032677
```

[11]:
```
print('Die Standard Abweichung STD ist ', std, '.')
```

```
Die Standard Abweichung STD ist  9.258292301032677 .
```

[12]:
```
# Percentil
```

[13]:
```
# Percentile 60 bedeutet der Wert, welche 60% der Daten "links" lässt.
```

[14]:
```
percentile_60 = np.percentile(geschwindigkeit, 60)
```

[15]:
```
print(percentile_60)
```

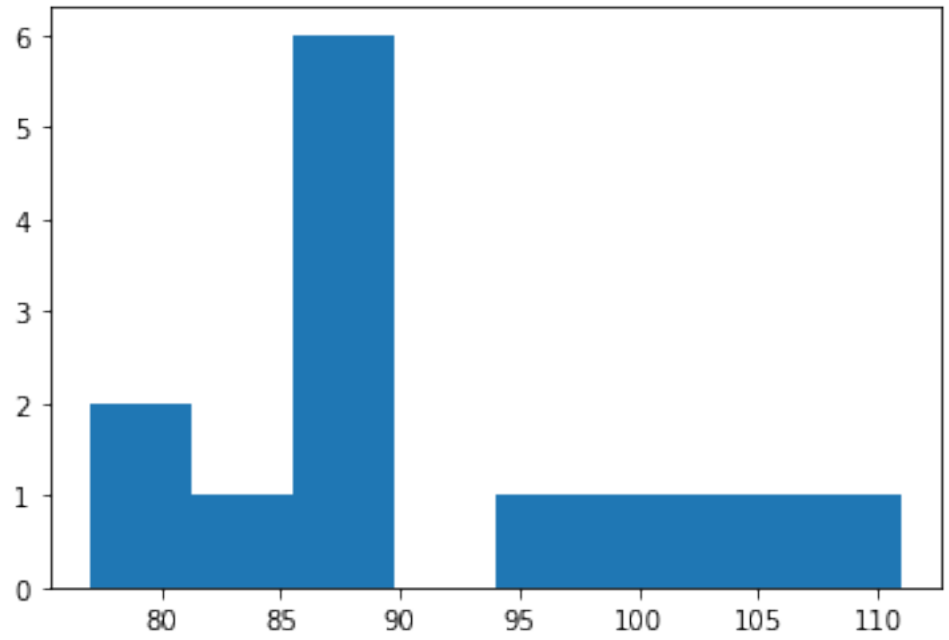
```
87.2
```

[16]:
```
# 60% der Daten kleiner sind als 87.2
```

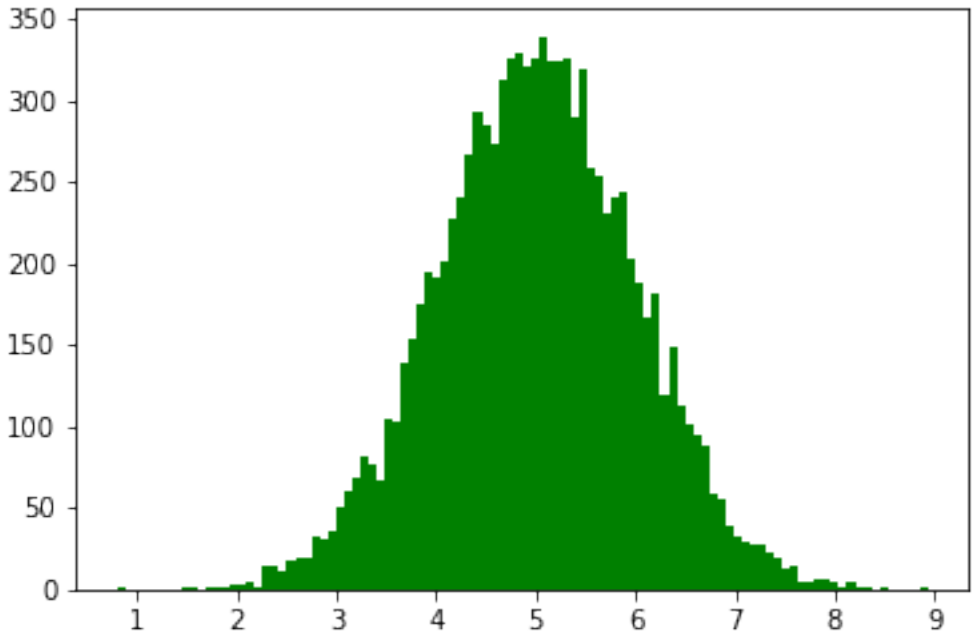
[17]:
```
# daten distribution (Verteilung)
```

[18]:
```
import matplotlib.pyplot as plt
```

[21]:
```
plt.hist(geschwindigkeit)
plt.show()
```



[22]:
```
# normaldistribution (Prüfung 2022)

x = np.random.normal(5, 1, 10000)
# 10000 Zufallsgenerierte Daten Normalverteilt mit Mittelwert 5 und Std Abw 1

plt.hist(x, bins= 100, color = 'g')
plt.show()
```



[24]:
```
import pandas as pd

df = pd.read_csv ('/Users/h4/desktop/20230413_data.csv', delimiter = ',')
```

[25]:
```
df
```

[25]:

	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
5	VW	Up!	1000	929	105
6	Skoda	Fabia	1400	1109	90
7	Mercedes	A-Class	1500	1365	92
8	Ford	Fiesta	1500	1112	98
9	Audi	A1	1600	1150	99
10	Hyundai	I20	1100	980	99
11	Suzuki	Swift	1300	990	101
12	Ford	Fiesta	1000	1112	99
13	Honda	Civic	1600	1252	94
14	Hundai	I30	1600	1326	97
15	Opel	Astra	1600	1330	97
16	BMW	1	1600	1365	99
17	Mazda	3	2200	1280	104
18	Skoda	Rapid	1600	1119	104

```
19        Ford       Focus    2000    1328  105
20        Ford      Mondeo    1600    1584   94
21        Opel    Insignia    2000    1428   99
22    Mercedes     C-Class    2100    1365   99
23       Skoda     Octavia    1600    1415   99
24       Volvo         S60    2000    1415   99
25    Mercedes         CLA    1500    1465  102
26        Audi          A4    2000    1490  104
27        Audi          A6    2000    1725  114
28       Volvo         V70    1600    1523  109
29         BMW           5    2000    1705  114
30    Mercedes     E-Class    2100    1605  115
31       Volvo        XC70    2000    1746  117
32        Ford       B-Max    1600    1235  104
33         BMW         216    1600    1390  108
34        Opel      Zafira    1600    1405  109
35    Mercedes         SLK    2500    1395  120
```

[ ]:
```
url = 'https://github.com/ProfH4/Lecture_Notes/blob/main/2023%20Sommersemester/
↪480636%20Wirtschaftsinformatik/20230413_data.csv'

df_2 = pd.read_csv(url, index_col = 0, parse_dates=[0])
```

[27]:
```
print(df)
```

```
           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
5           VW         Up!    1000     929  105
6        Skoda       Fabia    1400    1109   90
7     Mercedes     A-Class    1500    1365   92
8         Ford      Fiesta    1500    1112   98
9         Audi          A1    1600    1150   99
10     Hyundai         I20    1100     980   99
11      Suzuki       Swift    1300     990  101
12        Ford      Fiesta    1000    1112   99
13       Honda       Civic    1600    1252   94
14      Hundai         I30    1600    1326   97
15        Opel       Astra    1600    1330   97
16         BMW           1    1600    1365   99
17       Mazda           3    2200    1280  104
18       Skoda       Rapid    1600    1119  104
19        Ford       Focus    2000    1328  105
20        Ford      Mondeo    1600    1584   94
```

```
21        Opel    Insignia    2000    1428   99
22    Mercedes     C-Class    2100    1365   99
23       Skoda     Octavia    1600    1415   99
24       Volvo         S60    2000    1415   99
25    Mercedes         CLA    1500    1465  102
26        Audi          A4    2000    1490  104
27        Audi          A6    2000    1725  114
28       Volvo         V70    1600    1523  109
29         BMW           5    2000    1705  114
30    Mercedes     E-Class    2100    1605  115
31       Volvo        XC70    2000    1746  117
32        Ford       B-Max    1600    1235  104
33         BMW         216    1600    1390  108
34        Opel      Zafira    1600    1405  109
35    Mercedes         SLK    2500    1395  120
```

[28]:
```
import pandas as pd
```

[29]:
```
# normierung von Daten
```

[30]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in
/Users/h4/anaconda3/lib/python3.9/site-packages (0.0)
Requirement already satisfied: scikit-learn in
/Users/h4/anaconda3/lib/python3.9/site-packages (from sklearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(2.2.0)
Requirement already satisfied: joblib>=0.11 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.1.0)
Requirement already satisfied: scipy>=1.1.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.9.1)
Requirement already satisfied: numpy>=1.14.6 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.23.2)
```

[31]:
```
import sklearn
```

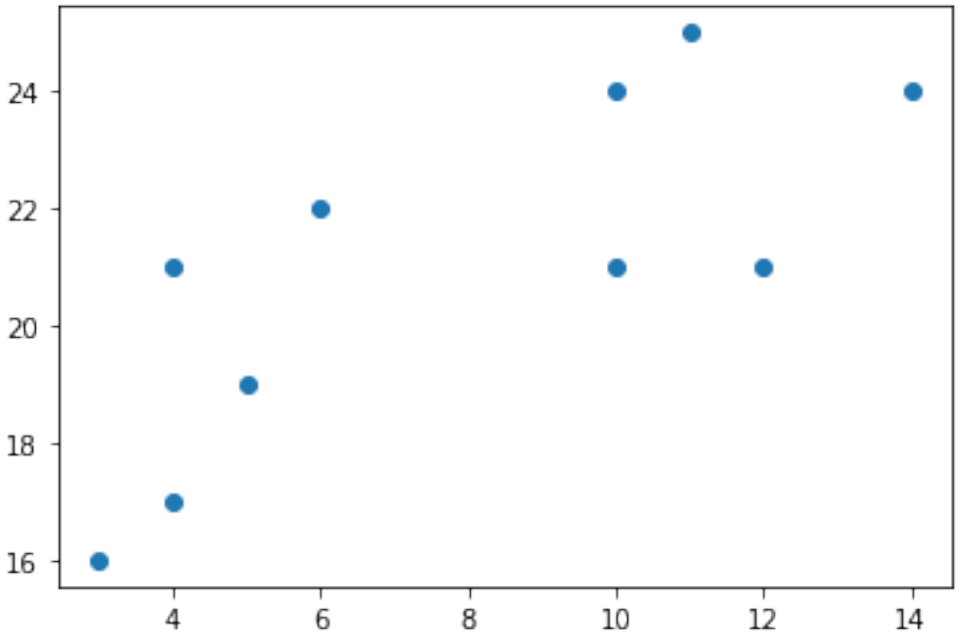
[33]:
```
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

df_nummerisch = df[['Weight', 'Volume', 'CO2']] # nur die nummerischen Daten␣
 ↪aus dem Dataframe df geholt
```

```
scaled_df = scale.fit_transform(df_nummerisch) # hier haben eine Normierung␣
 ↪gemacht

# z = (x-mittelwert)/std_abw

print(scaled_df)
```

```
[[-2.10389253 -1.59336644 -0.41192538]
 [-0.55407235 -1.07190106 -0.95612037]
 [-1.52166278 -1.59336644 -0.95612037]
 [-1.78973979 -1.85409913 -1.63636411]
 [-0.63784641 -0.28970299  0.40436711]
 [-1.52166278 -1.59336644  0.40436711]
 [-0.76769621 -0.55043568 -1.63636411]
 [ 0.3046118  -0.28970299 -1.36426661]
 [-0.7551301  -0.28970299 -0.54797412]
 [-0.59595938 -0.0289703  -0.41192538]
 [-1.30803892 -1.33263375 -0.41192538]
 [-1.26615189 -0.81116837 -0.13982788]
 [-0.7551301  -1.59336644 -0.41192538]
 [-0.16871166 -0.0289703  -1.09216911]
 [ 0.14125238 -0.0289703  -0.68402287]
 [ 0.15800719 -0.0289703  -0.68402287]
 [ 0.3046118  -0.0289703  -0.41192538]
 [-0.05142797  1.53542584  0.26831836]
 [-0.72580918 -0.0289703   0.26831836]
 [ 0.14962979  1.01396046  0.40436711]
 [ 1.2219378  -0.0289703  -1.09216911]
 [ 0.5685001   1.01396046 -0.41192538]
 [ 0.3046118   1.27469315 -0.41192538]
 [ 0.51404696 -0.0289703  -0.41192538]
 [ 0.51404696  1.01396046 -0.41192538]
 [ 0.72348212 -0.28970299 -0.00377913]
 [ 0.8281997   1.01396046  0.26831836]
 [ 1.81254495  1.01396046  1.62880584]
 [ 0.96642691 -0.0289703   0.9485621 ]
 [ 1.72877089  1.01396046  1.62880584]
 [ 1.30990057  1.27469315  1.76485459]
 [ 1.90050772  1.01396046  2.03695208]
 [-0.23991961 -0.0289703   0.26831836]
 [ 0.40932938 -0.0289703   0.81251335]
 [ 0.47215993 -0.0289703   0.9485621 ]
 [ 0.4302729   2.31762392  2.44509833]]
```

[34]:
```
# Gruppen in Daten finden - Clustering
# K-Means Clustering (K sind die Anzahl Gruppen)
```

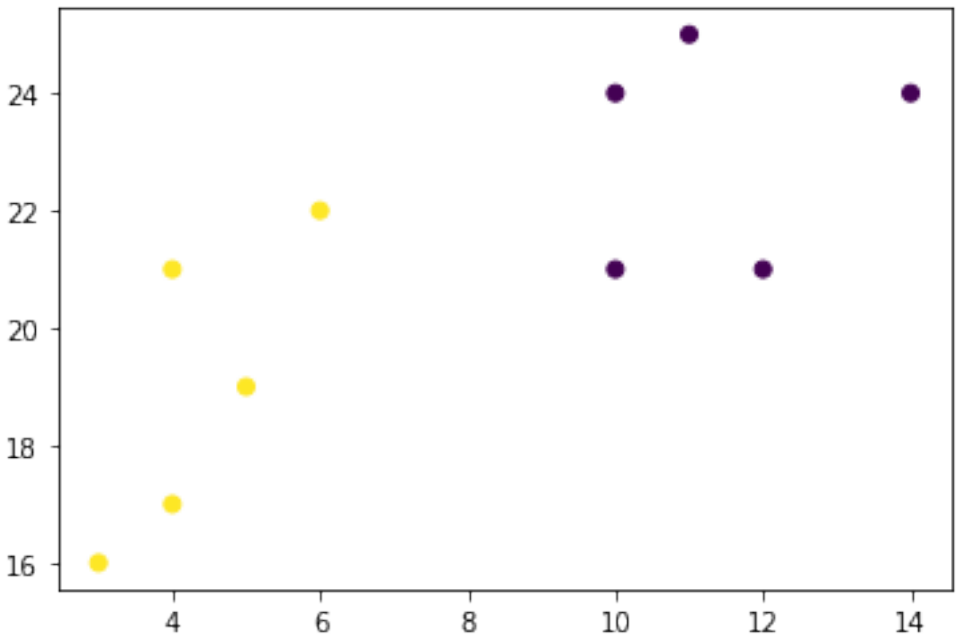
[35]:
```
Gehalt = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
Ausgaben_Webseite = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

plt.scatter(Gehalt, Ausgaben_Webseite)
plt.show()
```



[37]:
```
data = list(zip(Gehalt, Ausgaben_Webseite))
print(data)
```

```
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10,
21), (12, 21)]
```

[39]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 2)
kmeans.fit(data)

plt.scatter(Gehalt, Ausgaben_Webseite, c = kmeans.labels_)

plt.show()
```



[40]:
```
# Übung für die Prüfungsvorbereitung:
# KMeans Cluster mit 3 Zentren umsetzen für Dataset Auto (siehe oben) für␣
 ↪Weight und CO2.

# Hinweis: setzt voraus, dass eine Standardisierung stattfinden muss.
```

[ ]:
```

```