20230412_Wirtschaftsinformatik_MV2

April 12, 2023

[40]:
```
import numpy as np
```

[41]:
```
# mittelwert
```

[42]:
```
geschwindigkeit = [99,86,87,88,111,86,103,87,94,78,77,85,86]
```

[43]:
```
# (99+86+87+88+111+86+103+87+94+78+77+85+86) / 13 = 89.77
```

[44]:
```
mittelwert = np.mean(geschwindigkeit)
```

[45]:
```
print(mittelwert)
```

```
89.76923076923077
```

[46]:
```
# standard abweichung
```

[47]:
```
std = np.std(geschwindigkeit)
```

[48]:
```
print(std)
```

```
9.258292301032677
```

[69]:
```
print('Die Standard Abweichung STD ist ', std, '.')
```

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

[49]:
```
# percentile
```

[50]:
```
# percentile 60 bedeutet der Wert, welche 60% der Daten "links" lässt
```

[51]:
```
percentile_60 = np.percentile(geschwindigkeit, 60)
```

[52]:
```
print(percentile_60)
```

```
87.2
```

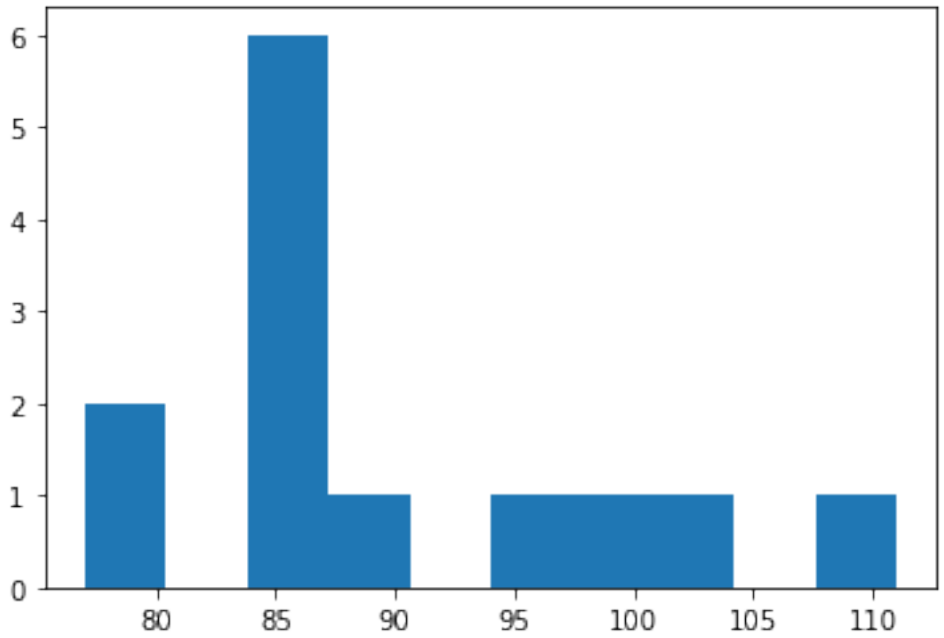
[53]:
```
# 60% der Daten sind kleiner als 87.2
```

[54]:
```
# daten distribution (Verteilung)
```

[55]:
```
# Verteilungen werden idR mit Hilfe von einem Histogram dargestellt
```

[56]:
```
import matplotlib.pyplot as plt
```

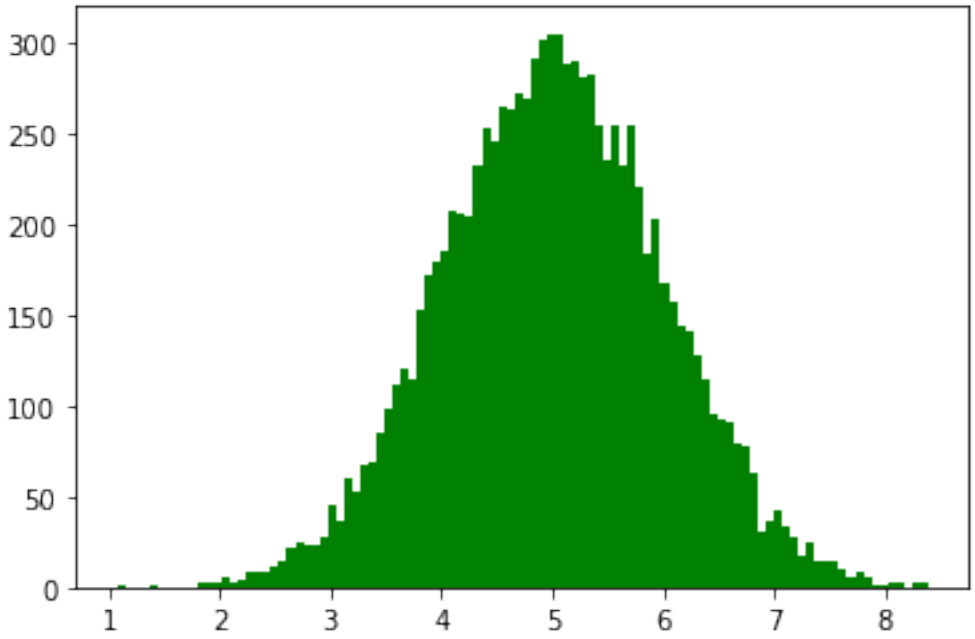
[71]:
```
plt.hist(geschwindigkeit)
plt.show()
```



[78]:
```
# beispiel

x = np.random.normal(5, 1, 10000)

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



[79]:
```
import pandas as pd
url = 'https://raw.githubusercontent.com/ProfH4/Lecture_Notes/main/
  ↪2023%20Sommersemester/410636%20Wirtschaftsinformatik/20230412_data.csv'
df = pd.read_csv(url,index_col=0,parse_dates=[0])
```

[81]:
```
print(df.head())
```

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

[82]:
```
# normierung der Daten

# z = (x-mittelwert)/standard_abweichung

import pandas as pd
```

[83]:
```
!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: numpy>=1.14.6 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.23.2)
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: scipy>=1.1.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.9.1)
Requirement already satisfied: joblib>=0.11 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn)
(1.1.0)
```

[84]:
```
import sklearn
```

[86]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
df_nummerisch = df[['Weight', 'Volume', 'CO2']]
scaled_df = scale.fit_transform(df_nummerisch)

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]
```

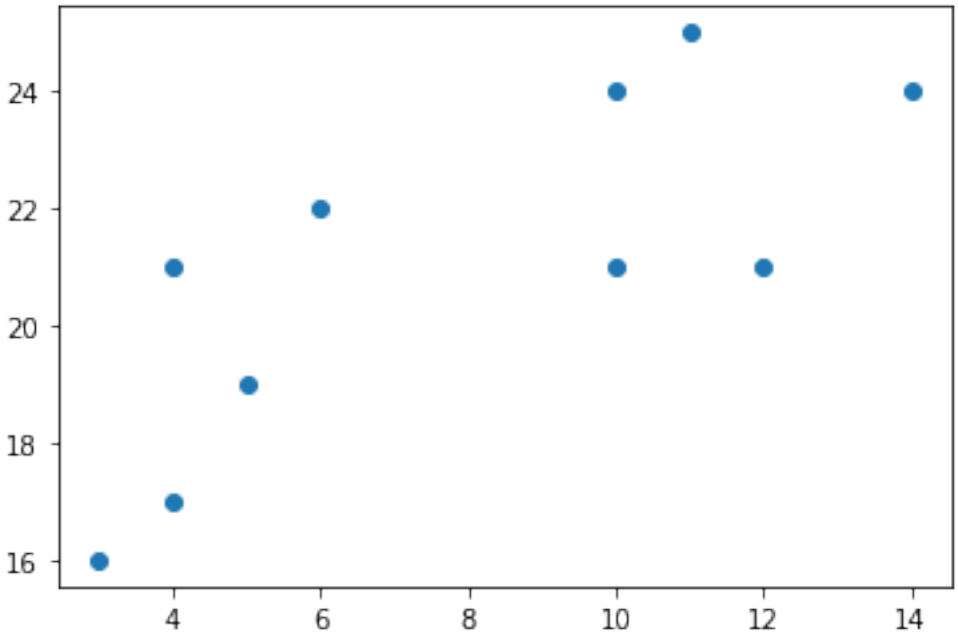
```
[ 0.72348212 -0.28970299 -0.00377913]
[ 0.8281997   1.01396046  0.26831836]
[ 1.81254495  1.01396046  1.62880584]
[ 0.96642691 -0.0289703   0.9485621 ]
[ 1.72877089  1.01396046  1.62880584]
[ 1.30990057  1.27469315  1.76485459]
[ 1.90050772  1.01396046  2.03695208]
[-0.23991961 -0.0289703   0.26831836]
[ 0.40932938 -0.0289703   0.81251335]
[ 0.47215993 -0.0289703   0.9485621 ]
[ 0.4302729   2.31762392  2.44509833]]
```

[87]:
```
# Gruppen finden in Daten - Clustering
```

[89]:
```
# K-Means Clustering (K sind die Anzahl Gruppen oder Zentren)
```

[91]:
```
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()
```



[92]:
```
data = list(zip(Gehalt, Ausgaben_Webseite))
```

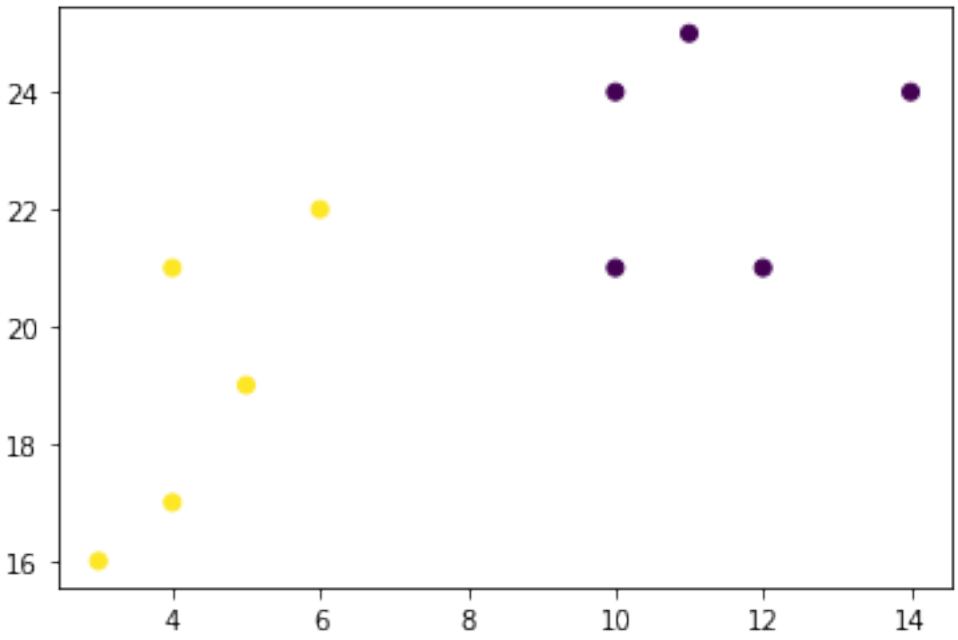
[93]:
```
data
```

[93]:
```
[(4, 21),
 (5, 19),
 (10, 24),
 (4, 17),
 (3, 16),
 (11, 25),
 (14, 24),
 (6, 22),
 (10, 21),
 (12, 21)]
```

[94]:
```
from sklearn.cluster import KMeans

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

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



[95]:
```
# Übung für die Prüfungsvorbereitung:
# KMeans Cluster umsetzen für dataset Autos (siehe oben)
# für Weight und Volume.
```

[ ]:
```

```