20220113_Informationsmanagement_FAT1

January 13, 2022

[1]:
```
# Maschinelles lernen
```

[2]:
```
speed = [99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86]
```

[3]:
```
# Mittelwert

import numpy as np

x = np.mean(speed)

print(x)
```

```
89.76923076923077
```

[5]:
```
# Median
# Der Medianwert ist der Wert in der Mitte, nach dem Sie alle Werte sortiert␣
 ↪haben.

# Median ist auch Perzentil 50

x = np.median(speed)

print(x)
```

```
87.0
```

[6]:
```
# Mode

# Ist der der am häufigsten vorkommt.

from scipy import stats

x = stats.mode(speed)

print(x)
```

```
ModeResult(mode=array([86]), count=array([3]))
```

[7]:
```
# Standardabweichung.

# Die StdAbw ist ist eine Zahl, die beschreibt, wie gestreut die Werte sind.
# Eine niedrige SdtAbw bedeutet, dass die meisten Zahlen nahe am Mittelwert␣
 ↪liegen.

# stdabweichung= sqrt(((xi-mean(x))^2)/(n-1))

x = np.std(speed)

print(x)
```

```
9.258292301032677
```

[9]:
```
# Perzentile

# Perzentile werden in Statistiken verwendet, um eine Zahl zu geben,
# die den Wert beschreibt, unter dem ein bestimmter Prozentsatz der Werte␣
 ↪kleiner ist.

x = np.percentile(speed, 75)

print(x)

# Dieser Wert lässt, nach einer Sortierung, 75% der Werte links und 25% der␣
 ↪Werte rechts von ihm.
```

```
94.0
```

[10]:
```
# Percentile 90

x = np.percentile(speed, 90)
print(x)
```

```
102.2
```

[12]:
```
x = np.percentile(speed, 50)
print(x)

# das ist die Mode
```

```
87.0
```

[13]:
```
# Datenverteilungen

# Um grosse Datensaetze zum Testen zu erstellen, verwenden wir das Python-Modul␣
 ↪Numpy, das
```

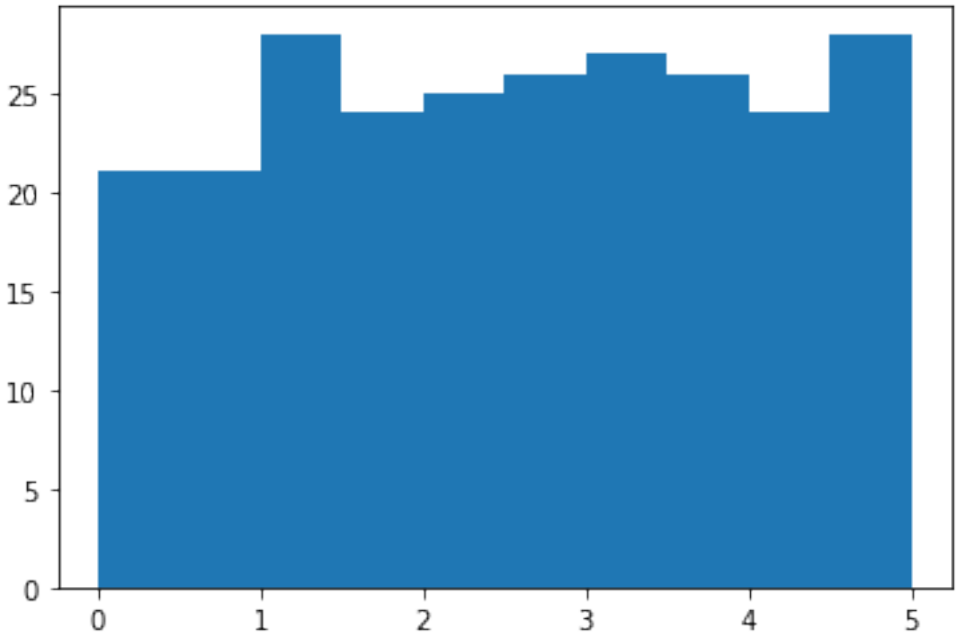
```
# eine Reihe von Methoden enthält, um zufällige Datensäetze beliebiger Groesse
 ↪zu erstellen.
```

[14]:
```
# Beispiel: Erstellen Sie ein Array mit 250 zufälligen Gleitkommazahlen zw 0
 ↪und 5

import numpy as np

x = np.random.uniform(0, 5, 250)

print(x)
```

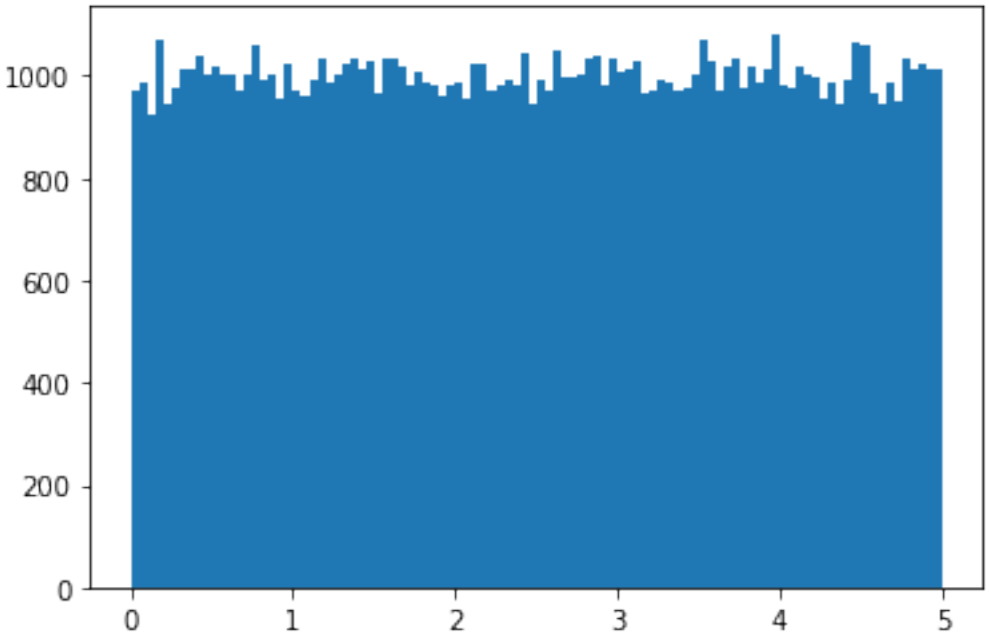
```
[1.12204751e+00 4.89606781e-02 8.60036522e-01 9.08071153e-01
 3.98148429e+00 2.09772812e+00 4.92558003e+00 1.33938737e+00
 2.29076644e+00 4.95953011e+00 7.49093930e-01 2.54552395e+00
 3.53524647e+00 4.08031781e+00 4.66443374e+00 4.58210485e+00
 4.76059955e+00 9.40525349e-01 1.10147377e+00 1.32500052e+00
 4.41977718e+00 2.86558833e+00 2.53717411e+00 3.08924442e+00
 3.02470990e+00 4.35644708e+00 1.01640455e+00 3.64710974e+00
 6.87010961e-01 6.43186081e-01 3.48846684e+00 4.81135014e+00
 4.33319615e+00 2.91045787e+00 2.19866331e+00 8.76887772e-02
 2.33843284e+00 4.95693629e+00 2.71258884e+00 4.36249991e+00
 4.60688545e+00 3.25963044e+00 2.88233973e+00 3.50834939e+00
 1.74419626e+00 2.80494271e+00 3.24801658e+00 2.89979585e+00
 7.06901966e-01 3.63338149e+00 1.28398744e+00 4.16617586e+00
 2.86960093e+00 1.95073150e+00 4.65029861e+00 6.82834742e-01
 2.45377175e-01 2.53695442e+00 7.41725386e-01 3.96805761e+00
 2.28421207e+00 3.74402163e+00 7.58660950e-01 1.58731266e-01
 4.13301120e+00 4.04457844e+00 3.24850521e+00 7.86416289e-01
 1.99498042e+00 3.01036177e+00 3.12327983e+00 8.86303839e-02
 3.39892876e+00 6.25679876e-01 1.93721980e+00 1.61191790e+00
 2.70868585e-01 8.83627896e-01 2.84092591e+00 3.15005173e+00
 1.52728488e+00 1.31053829e+00 2.87398322e+00 4.57842306e+00
 7.65124772e-01 1.50420605e+00 3.07317014e+00 3.11651127e+00
 4.95591664e+00 4.31900146e+00 2.27710841e+00 7.58065668e-01
 3.06183685e+00 3.72480856e-01 8.94301529e-01 1.29754819e+00
 2.86009803e+00 4.62435607e+00 3.73452807e+00 1.20707621e+00
 4.37295350e+00 1.08262468e+00 3.00131979e+00 1.17144443e+00
 1.37550845e+00 1.31262388e+00 1.83257912e+00 4.42412457e+00
 3.34475946e+00 2.43584754e+00 2.74194240e+00 2.55643093e-01
 3.96599737e+00 1.79473103e+00 4.12542284e+00 3.07747824e+00
 3.92789337e+00 1.17164453e+00 1.38857719e+00 2.82073598e+00
 2.14517286e+00 2.62467566e+00 1.65993836e+00 2.16257927e+00
 3.00399403e+00 4.93258742e+00 3.88841491e+00 1.55171798e+00
 1.15277184e+00 9.23435498e-01 3.57896734e+00 1.54942793e+00
 4.82427198e+00 8.04102624e-01 8.75054133e-01 4.39819146e+00
 4.10734720e+00 3.36589475e+00 1.31429264e+00 3.98767566e+00
```

```
4.93452496e+00 1.49440104e+00 9.39475266e-01 3.55373327e+00
3.08811776e+00 2.95475217e+00 2.92722062e+00 1.15106851e+00
3.87566003e+00 3.88039779e+00 1.08279512e+00 2.00514236e-01
3.56797428e+00 1.31466750e+00 2.15363049e+00 2.47555188e+00
3.36789147e+00 6.14096649e-01 2.86964645e+00 4.97510602e-01
1.45710918e-01 4.15675099e+00 2.53609080e+00 4.96615350e+00
2.60368253e+00 6.46270177e-01 4.19373785e+00 1.80958198e+00
1.32533541e+00 4.31101177e+00 8.59454799e-01 2.71502620e+00
1.44132015e+00 4.09366908e+00 1.85666495e+00 3.55687157e+00
4.26709235e+00 3.63871867e+00 4.52008159e-02 2.80036124e+00
8.23910625e-01 1.92700126e+00 3.88793576e+00 7.79336486e-02
3.63818237e+00 8.84733504e-02 4.59143909e+00 3.79322017e-01
2.33331183e+00 4.10118170e+00 4.15710398e+00 4.15091510e+00
2.41960846e+00 4.86396124e+00 8.27233073e-01 3.54402353e+00
7.82885196e-01 1.95046505e+00 1.27458470e-02 1.83805223e+00
4.44256521e+00 8.69347495e-02 1.11442405e+00 1.15321726e+00
2.88240366e+00 1.20580307e+00 4.38272589e+00 4.73769301e+00
7.78626236e-01 3.79097884e+00 1.56434601e+00 1.06180810e-04
3.24660000e+00 7.35573341e-01 4.73359521e+00 4.73081251e+00
4.29095696e+00 9.41975078e-01 1.96358749e+00 1.73742045e+00
1.81994715e+00 4.33493233e+00 3.18830668e+00 2.31851779e+00
3.87891728e+00 2.82341322e+00 1.41894935e+00 1.68014688e+00
4.76487578e+00 3.86591417e+00 3.49803343e+00 1.52909826e+00
3.74841794e+00 3.21702434e+00 3.91588009e+00 1.69534130e-01
1.68570231e+00 4.38654692e+00 2.42605811e+00 1.26536691e+00
3.31003319e+00 4.68103086e+00 1.07251588e+00 4.21366504e+00
4.33726851e+00 4.79381662e+00 1.50033319e+00 4.11444808e+00
1.24108843e+00 8.50565250e-02]
```

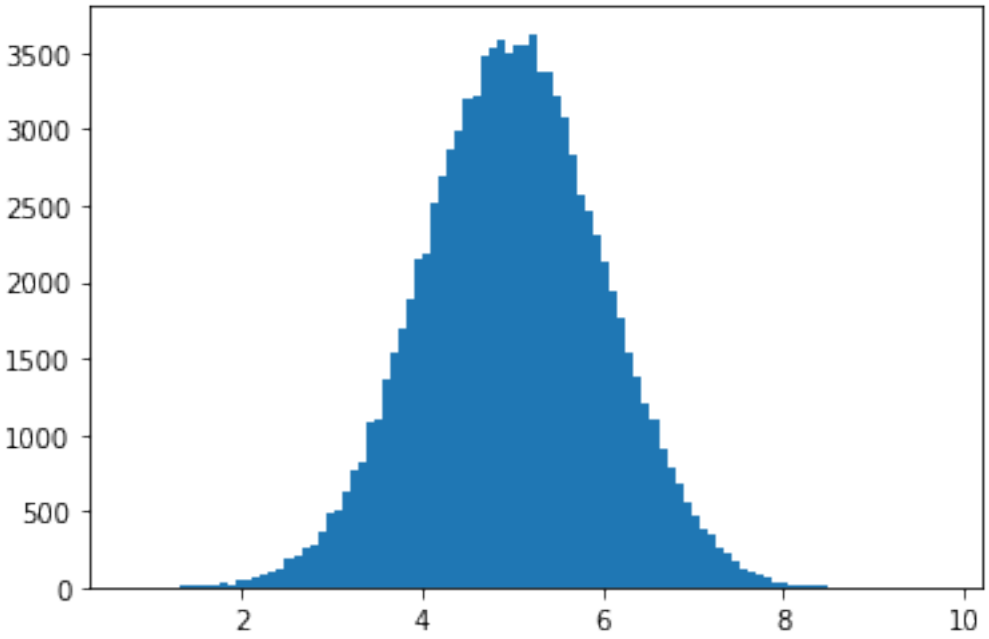
[15]:
```python
# Hitograme werden idR benutzt um Datenzaetze zu visualisieren.

import numpy as np
import matplotlib.pyplot as plt

x = np.random.uniform(0, 5, 250)

plt.hist(x,10)
plt.show()
```



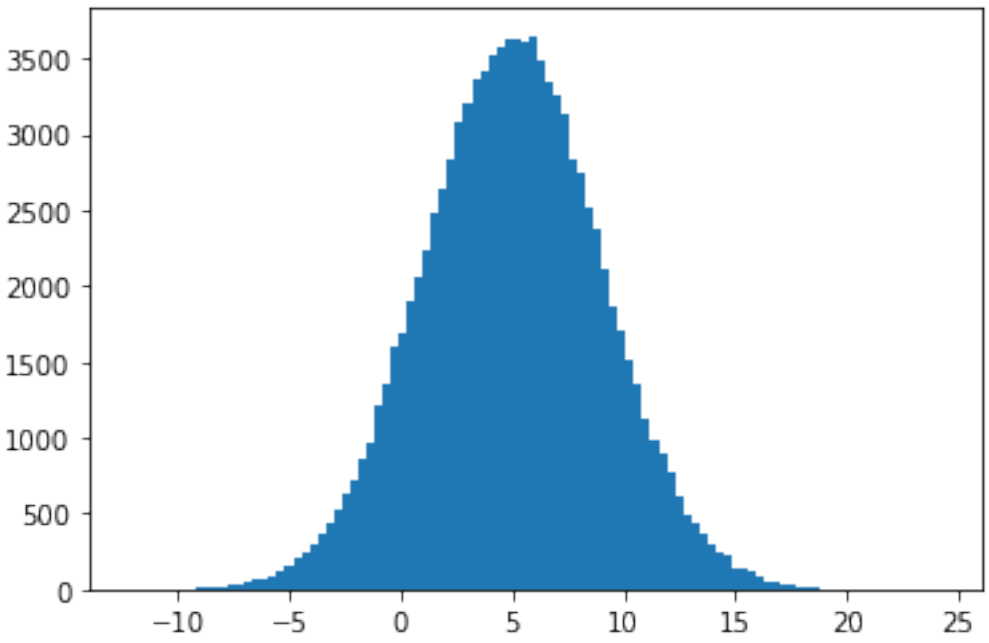
[16]:
```
# Big--Data Verteilungen erzeugen

# Ein Array mit 250 Werten wird nicht als sehr gross angesehen.
# Beispiel: Erstellen Sie ein Array mit 100000 Zufallszahlen zw. 0 und 5 und␣
 ↪zeigen Sie mit einem Histogram
# mit 100 Balken die Daten an:

x = np.random.uniform(0, 5, 100000)

plt.hist(x, 100)
plt.show()
```



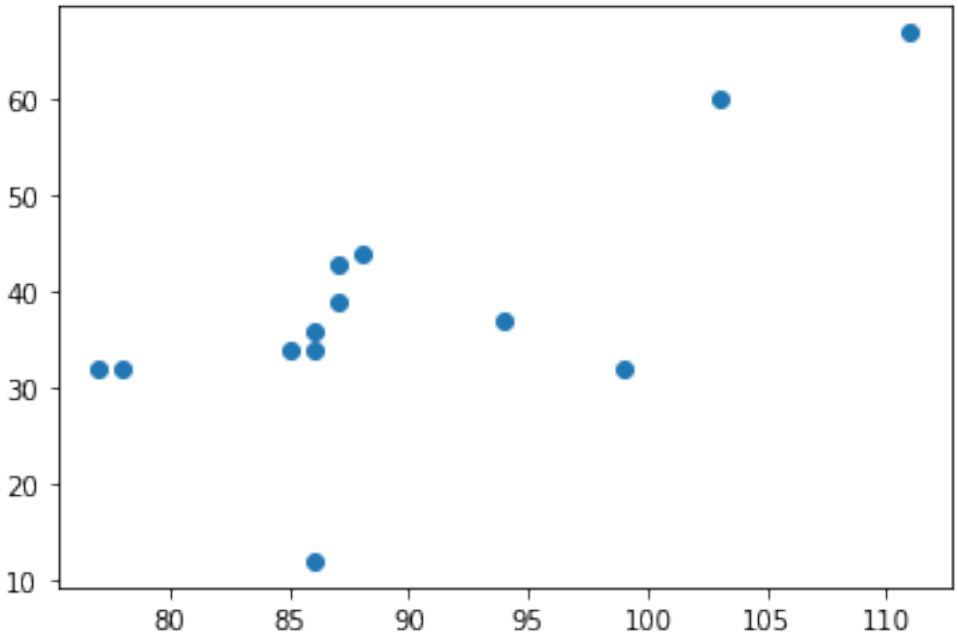
[17]:
```
# Normale (Gauss) Datenverteilung

# In der Wahrscheinlichkeitstheorie wird diese Art der Datenverteilung
# als "Normalverteilung" bezeichnet.

# Beispiel: Erstellen Sie eine normalverteilte Gaussverteilung mit 100000␣
 ↪Datenzaetze, Mittelwert auf 5 und
# Std Abw 1.

import numpy as np
import matplotlib.pyplot as plt

x = np.random.normal(5, 1, 100000)

plt.hist(x, 100)
plt.show()
```



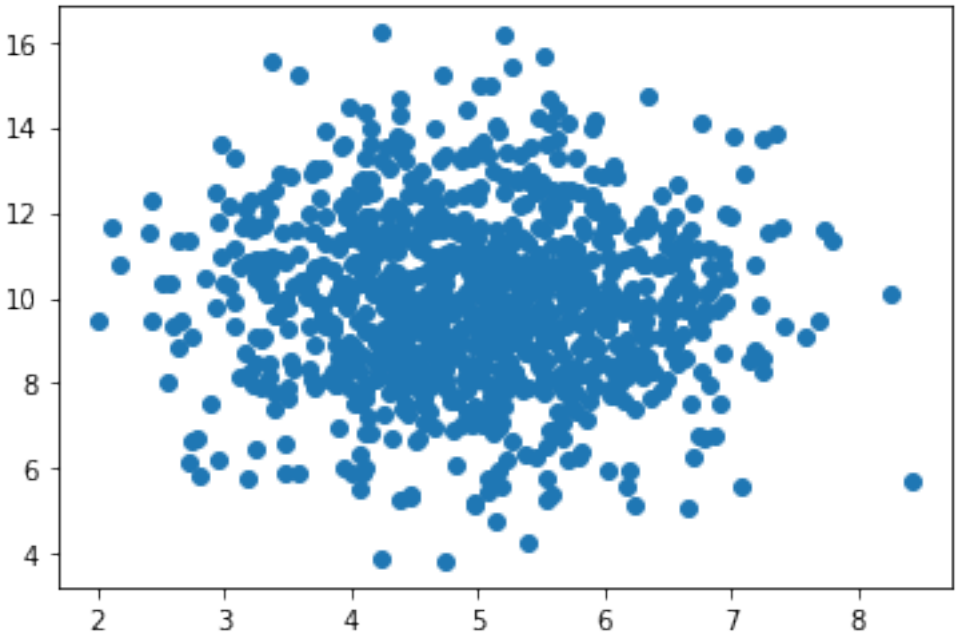
[18]:
```
# Beispiel: Erstellen Sie eine normalverteilte Gaussverteilung mit 100000␣
 ↪Datenzaetze, Mittelwert auf 5 und
# Std Abw 4.

import numpy as np
import matplotlib.pyplot as plt

x = np.random.normal(5, 4, 100000)

plt.hist(x, 100)
plt.show()
```



[21]:
```
# Streudiagramme (Scatterplots)

verbrauch = [32, 12, 43, 44, 67, 34, 60, 39, 37, 32, 32, 34, 36]

import matplotlib.pyplot as plt

plt.scatter(speed, verbrauch)
plt.show()
```

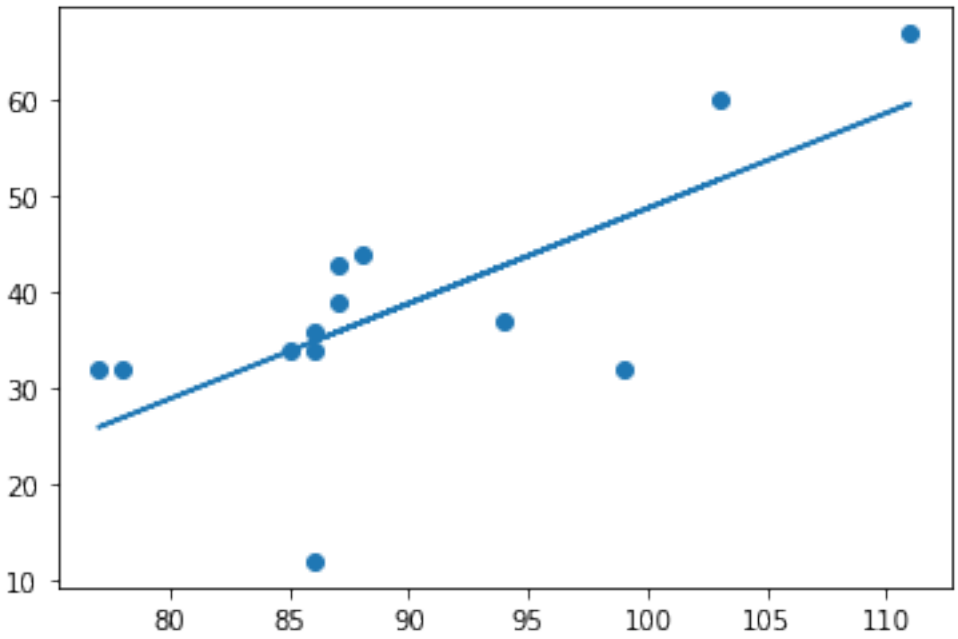


[22]:
```
# Zufällige Datenverteilung

# Beispiel: wir erzeugen eine 2 Dimensionale zufällige Normalverteilung von␣
 ↪1000 Datensätzen.

import numpy as np
import matplotlib.pyplot as plt

x = np.random.normal(5,1,1000)
y = np.random.normal(10,2,1000)

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



[27]:
```
# Lineare Regression

# Der Begriff Regression wird verwendet, wenn wir versuchen, die Beziehung zw.␣
 ↪Variablen zu finden.

# Im ML und statische Modellirung wird diese Beziehung verwendet,
# um das Ergebnis zuküfitger Ereignisse vorherzusagen.

# Lineare Regression verwendet die Beziehung zw. den Datenpunkten um eine␣
 ↪gerade Linie durch alle zu ziehen.

# Streudiagramme (Scatterplots)

import matplotlib.pyplot as plt
from scipy import stats

speed = [99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86]
verbrauch = [32, 12, 43, 44, 67, 34, 60, 39, 37, 32, 32, 34, 36]

# führen wir eine Methode aus, die einige wichtige Schlüsselwerte der lineararen␣
 ↪Regression zurueckgibt:
# "Slope" = Neigung der Linie
# "Intercept" = der Punkt an dem die Linie mit der y Achse schneidet.

slope, intercept, r, p, std_err = stats.linregress(speed,verbrauch)
```

```
# Erstellen wir eine Funktion, die die slope und intercept Werte verwendet, um␣
 ↪einen neuen Wert zurueckzugeben.
# Dieser neue Wert stellt dar, wo auf der y Achse der entsprechende x Wert␣
 ↪platziert wird.

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

# Führen wir jeden Wert des x Arrays durch die Funktion. Dies führt zu einem␣
 ↪neuen Array mit neuen Werten füir die y-Achse.

mymodel= list(map(myfunc, speed))


plt.scatter(speed, verbrauch)
plt.plot(speed, mymodel)
plt.show()
```

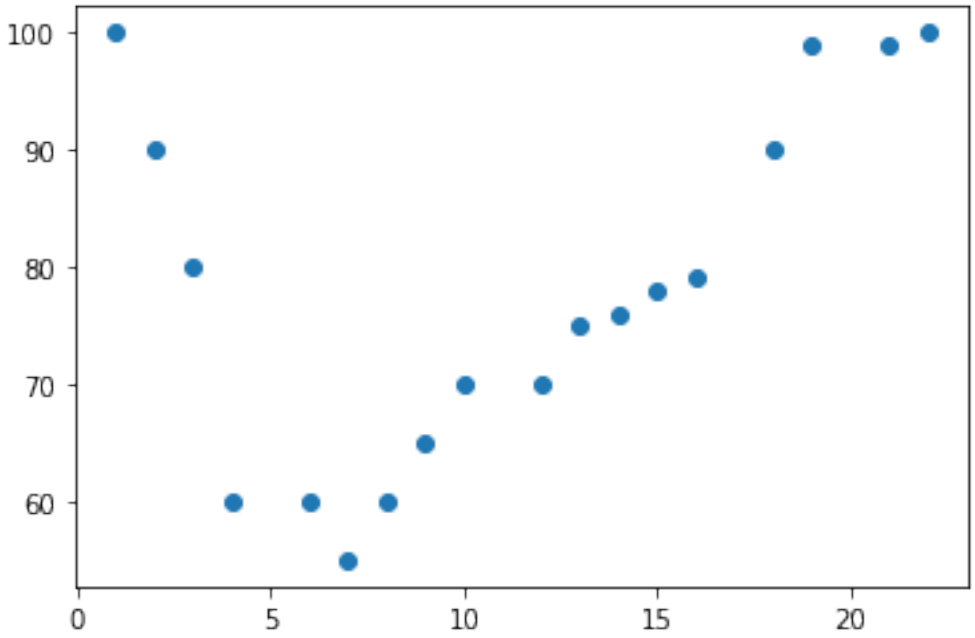


[28]:
```
myfunc(95)
```

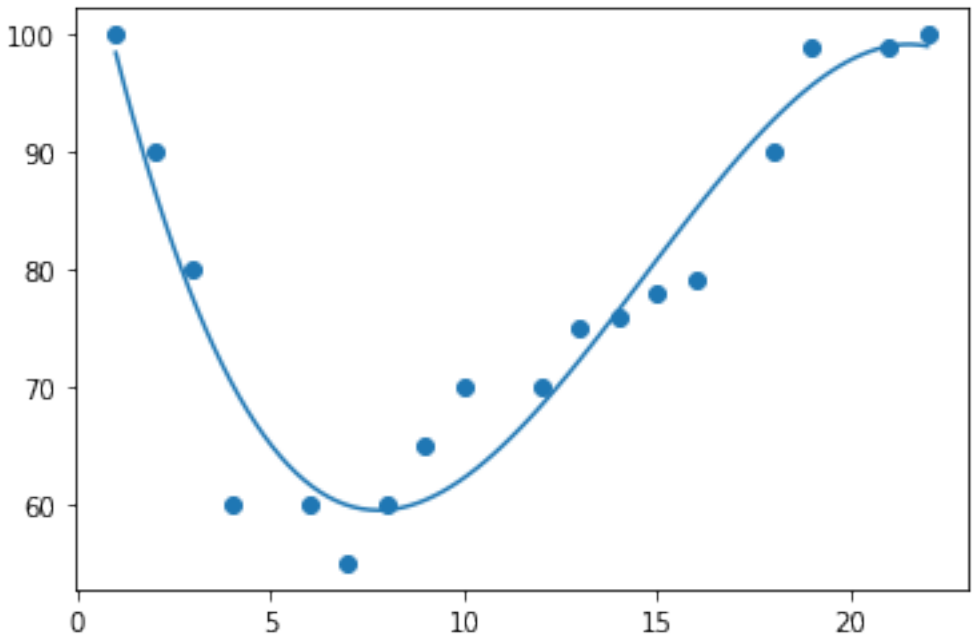
[28]:
```
43.811127985641306
```

[29]:
```
# Polynomische Regression
```

```
x = [1,2,3,4,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
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()
```



[38]:
```
# Numpy hat eine Methode mit der wir ein Polynommodell erstellen können:

mymodel = np.poly1d(np.polyfit(x,y,3))

# Geben wir an, wie die Linie angezeigt wird, wir beginnen bei Position 1, und␣
 ↪enden bei Position 22:

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

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


100
90
80
70
60
0
5
10
15
20

[ ]: