Ex2.boxplot

January 16, 2024

1 2nd exercise: Work with Boxplots for anomaly detection

- Course: AML
- Lecturer: Gernot Heisenberg
- Author of notebook: Finn Heydemann
- Date: 28.10.2023

GENERAL NOTE 1: Please make sure you are reading the entire notebook, since it contains a lot of information on your tasks (e.g. regarding the set of certain paramaters or a specific computational trick), and the written mark downs as well as comments contain a lot of information on how things work together as a whole.
GENERAL NOTE 2: * Please, when commenting source code, just use English language only. * When describing an observation please use English language, too * This applies to all exercises throughout this course.

1.0.1 DESCRIPTION:

This notebook allows you for using boxplots to detect anomalies. Try to interpret the boxplot.

1.0.2 TASKS:

The tasks that you need to work on within this notebook are always indicated below as bullet points. If a task is more challenging and consists of several steps, this is indicated as well. Make sure you have worked down the task list and commented your doings. This should be done by using markdown. Make sure you don't forget to specify your name and your matriculation number in the notebook.

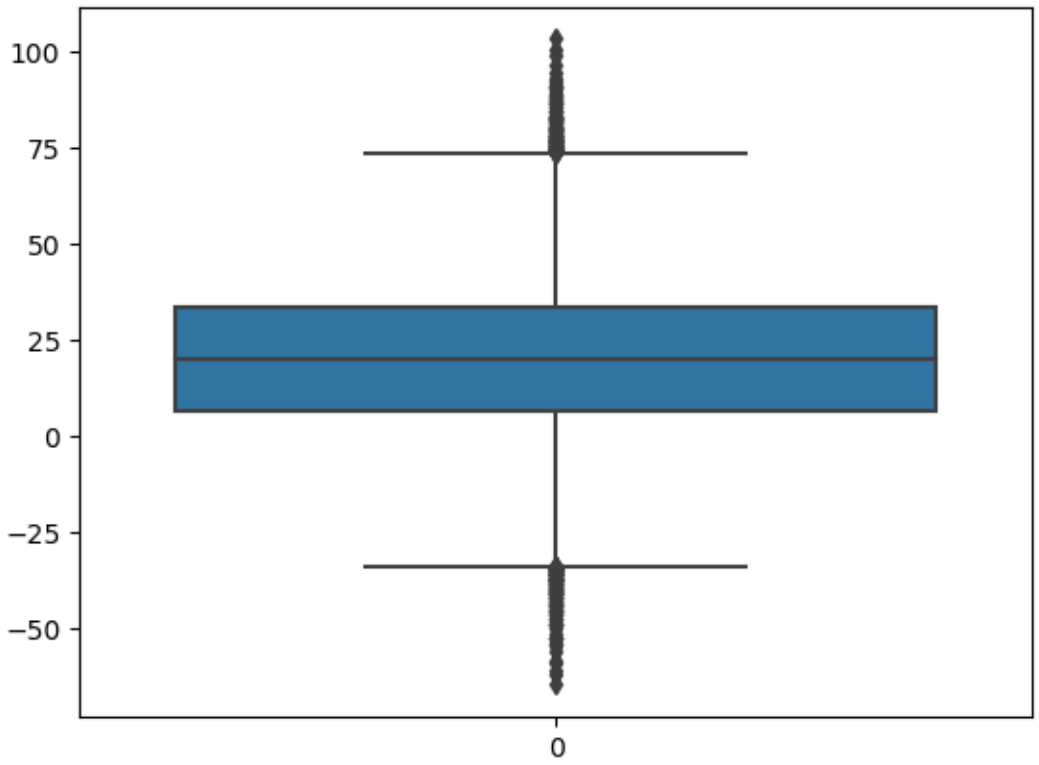
YOUR TASKS in this exercise are as follows: 1. import the notebook to Google Colab or use your local machine. 2. make sure you specified you name and your matriculation number in the header below my name and date. * set the date too and remove mine. 3. read the entire

notebook carefully * add comments whereever you feel it necessary for better understanding * run the notebook for the first time. 4. take the three data sets from exercize 1 and create a boxplot for each of them 5. interpret the visual results and describe them 6. how can you test your data for being normally distributed? ———————————————————————————————

[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import randn
np.random.seed(1)
```

[2]:
```
# multiply and add by random numbers to get some real values
# randn generates samples from the normal distribution (important - see below)
random_data = np.random.randn(50000)  * 20 + 20
```

[3]:
```
sns.boxplot(data=random_data)
plt.show()
```



[29]:
```
def get_outliers(data):
    q25, q75 = np.quantile(data, 0.25), np.quantile(data, 0.75)
    iqr = q75 - q25
    u_bound, l_bound = q75 + 1.5 * iqr, q25 - 1.5 * iqr
```

```
    return data[np.logical_or(data < l_bound, data > u_bound)]

get_outliers(random_data)
```

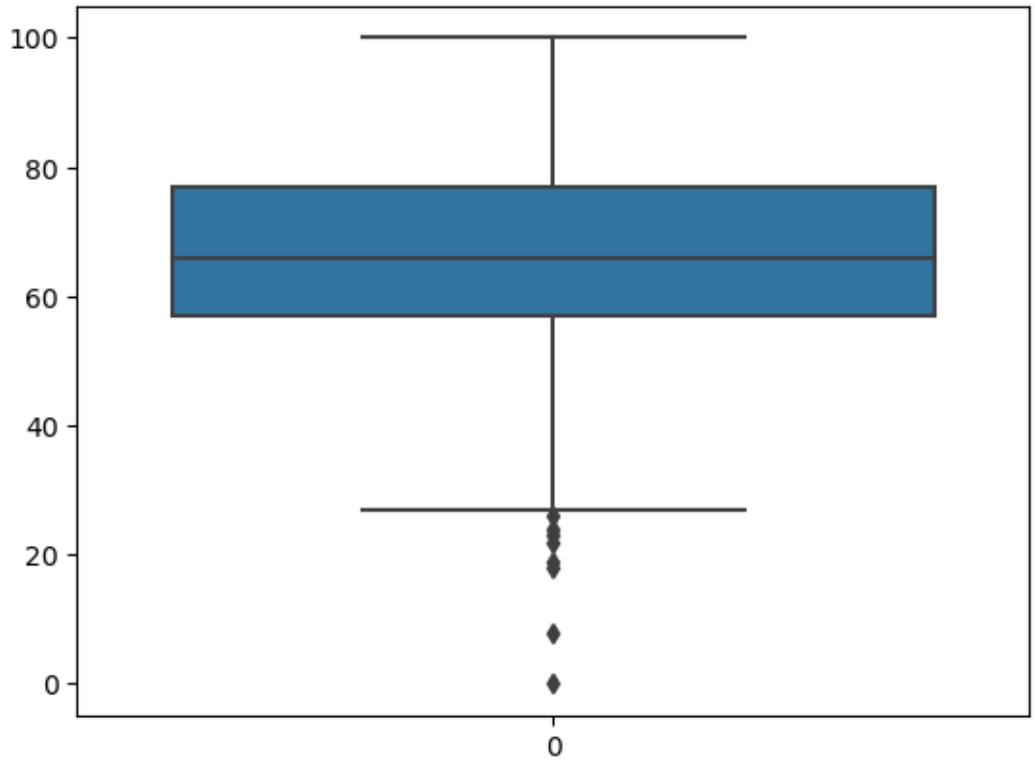
[29]:
```
array([-35.8617    ,  80.61714225, -35.65068935,  99.17205408,
       -35.81992813,  86.42157512, -33.96723483, -41.07528761,
        78.34617526,  76.54599588,  74.23899748, -43.067149  ,
       -40.3206397 ,  88.65326864, -35.82888076,  76.87877139,
       -41.28282712,  94.80497807,  74.58924271,  77.88772383,
        82.7009468 ,  84.76686394, -34.78283478, -37.44100501,
       -34.20787481,  75.74722896,  74.12249844,  77.87185104,
        74.84310423,  76.95564235,  75.54105986, -36.03124433,
        76.38174189, -45.0606847 ,  74.34772486, -34.39876182,
       -38.28997835,  77.08848163,  74.61690622,  82.64059456,
        80.25509136,  76.43577761, -36.89875188, -39.69672273,
       -43.81232694, -45.21230192, -33.97837966, -37.02139085,
       -34.16678629, -38.50610072,  88.08604551,  79.51578577,
       -34.21033755,  77.81616357,  82.36959182,  85.97081047,
        82.9796805 , -36.04264086,  78.5366674 , -34.46354787,
       -45.89716813, -34.07986795, -41.11266481, -35.21399504,
       100.53698089, -44.38264211,  77.15467882,  76.09101895,
        82.23783662, -37.73388132,  74.20425715,  92.26554014,
       -36.14464871,  87.96231313,  91.21746641, -53.12880199,
        75.97638813, -45.64157593,  80.09264253, -37.29267907,
        91.21225295, -35.98303297, -43.46923275, -34.08442185,
        82.13498657,  82.90737927, -39.86982454, -46.21685124,
       -37.15055461,  74.14251134, -41.95653104, -37.58465058,
       -36.7891546 , -45.60655194,  75.08540296,  77.38927174,
       -38.56479496,  76.871437  ,  76.78295427,  74.63852025,
        80.52436989, -35.59292026,  76.12359139, -39.67330888,
        74.09865159, -43.30420249,  79.51049288,  74.26327018,
        77.15907871, -42.25263324,  81.59215546,  74.12707741,
        77.04719248, -39.68480773,  75.29533039, 103.36235356,
        74.02110979,  96.68762042, -40.58687956, -48.7185162 ,
       -45.18412137,  81.54158108, -35.09207617,  74.80784467,
       -34.72710031, -37.36091861, -37.37803455, -37.08108277,
        74.5619158 ,  75.40605486, -39.31399767,  84.55872241,
        79.82213434,  77.93003549, -38.64509283, -40.76885172,
       -44.08128901,  76.86311843, -42.04329787,  87.71430312,
        76.78575604,  88.93912032,  74.72061321, -40.52113315,
       -49.02805813, -34.27289958,  86.72050503, -35.4277613 ,
       -44.45227837, -58.55028132,  76.51837469, -36.45328824,
        87.83885146, -41.81006578, -39.34934736,  75.11321611,
        74.45219054,  75.52708945,  75.2745851 , -42.38237155,
        91.19937116,  74.29202493,  75.88551312,  93.19531684,
       -36.10682579, -41.01728295,  82.94710559,  77.20168456,
       -36.92257349, -36.40929274, -37.4039325 ,  86.84102578,
```

```
 79.74073867, -47.35894572, -35.7353855 , -47.27471768,
 77.31043419,  73.82331644, -37.58905227, -44.0528396 ,
 77.68210664,  74.35283883,  81.39224117, -64.66329595,
-41.91924536,  76.66955617,  74.13141131,  77.33477115,
 86.98293402, -42.78324538, -43.44070545,  83.14254869,
-38.79374341,  76.87828875,  79.26711716,  78.70519455,
 80.29722046,  80.97664482, -34.05891632,  79.4440025 ,
 79.87234235, -34.78093295,  76.57433111, -37.25081912,
-49.1353069 , -39.30115858,  90.95359817,  85.75049141,
 77.99813217,  90.63353727, -52.25624571, -35.14250257,
 78.2720295 , -48.42727398,  73.99698456,  92.45148786,
-40.60136533,  77.52783527, -34.38975118, -34.28407191,
-34.66404324,  74.67513853,  79.31507012,  90.66197938,
-40.65200033,  78.05468898,  76.10665342, -39.7258654 ,
 85.47272253, -40.59878231, -37.6879442 , -34.50634921,
-53.95783717, -38.31172548, -35.15621329, -40.80036485,
 82.47403923, -38.52266653,  82.69560614, -37.56320898,
 73.90157419,  77.9433286 , -46.18148127,  86.99084876,
 79.03591658,  77.76137077, -35.31903536, -38.68031117,
 80.14572971,  82.24721889,  74.49607437,  74.99869464,
-45.83227596,  84.17626694, -36.25476452, -34.63844819,
-36.98596119, -38.7804372 ,  75.30982429, -39.45102341,
 84.41938097,  74.4208264 ,  74.26219913,  75.47032111,
-48.12080166, -47.66016306, -37.67850883,  83.03299707,
 78.18464751,  75.34171294, -36.98426248, -46.34572896,
-43.94127801,  76.23971068, -34.08308692, -44.88889348,
 75.36273255,  82.73490952,  83.57335544, -36.62496027,
 73.85844051, -33.93766999,  75.72339019, -39.66076026,
 80.0340636 ,  78.89243874, -39.88268155,  80.77274131,
-37.25289797,  73.89505015, -52.50413207,  75.01576701,
-36.86932899, -49.69745467,  77.11012485,  75.7783091 ,
 77.98008641,  80.22051939, -61.76977168, -35.99473753,
-41.57467214, -37.01407673, -37.31450019,  76.44535736,
-35.34890278,  74.55129507, -58.8628464 , -51.19738386,
 78.76972805,  75.60342332, -34.408634  , -56.11928396,
-36.16808082, -48.85042291, -38.6027919 , -52.64159592,
-34.58919861,  79.42685188,  74.4909021 , -40.85801593,
 90.00512514, -35.85217866, -44.15080376,  81.07575084,
 74.43680601,  88.42082611, -48.97515467, -41.87410602,
 81.91272478,  84.69303765, -40.93984642, -35.22769246,
-52.61561504, -34.72573462, -49.39834453,  84.48319182,
-36.15276168,  79.96406554,  78.51670098,  80.41393381,
-34.54292521,  89.04319371, -35.02476011,  83.11545349,
 75.84397277, -35.34727853,  86.3948526 ,  74.15549321,
 75.43073823, -54.34260433, -53.88408955, -60.83360878,
 75.7614544 , -38.16842492, -45.50295147, -38.81306884,
-35.08762697,  74.04412014, -42.8411709 ])
```

[24]:
```
import pandas as pd

data = pd.read_csv("data/exercise_1/StudentsPerformance.csv")["math score"]
sns.boxplot(data)
```

[24]:
```
<Axes: >
```



[31]:
```
get_outliers(data)
```

[31]:
```
17     18
59      0
145    22
338    24
466    26
787    19
842    23
980     8
Name: math score, dtype: int64
```

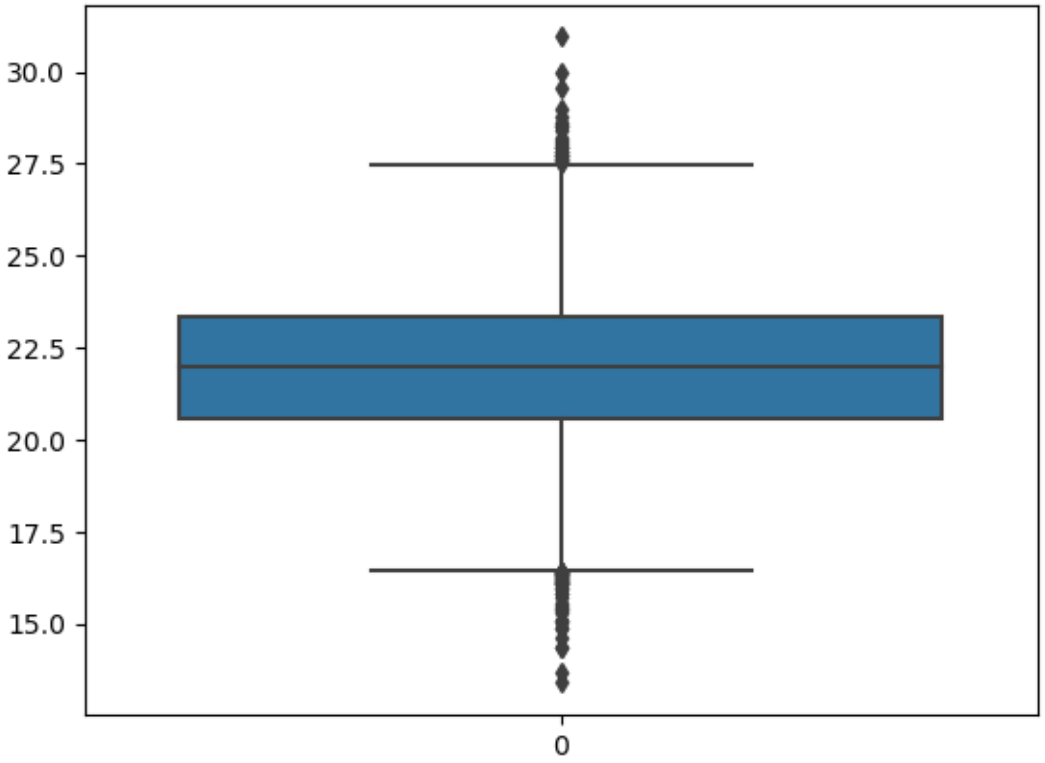
As can be seen 8 students are worse the then lower limit which makes them outliers. These are the same students as in exercise one plus 5 more.

[32]:
```
data = pd.read_csv("data/exercise_1/city_temperature.csv")
data.drop(data[data["AvgTemperature"] == -99].index, inplace=True)
data["AvgTemperature"] = (data["AvgTemperature"] - 32) * 5/9
data["date"] = pd.to_datetime(data[["Year", "Month", "Day"]])
data = data.set_index("date").groupby(["City", pd.
  ↪Grouper(freq="D")])["AvgTemperature"].mean()
data = data.reset_index()
```

```
/tmp/ipykernel_63548/3201813435.py:1: DtypeWarning: Columns (2) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("data/exercise_1/city_temperature.csv")
```

[41]:
```
sns.boxplot(data[data["City"] == "Brasilia"]["AvgTemperature"].to_numpy())
```

[41]:
```
<Axes: >
```



[48]:
```
ind = get_outliers(data[data["City"] == "Brasilia"]["AvgTemperature"]).index
data.loc[ind]
```

[48]:

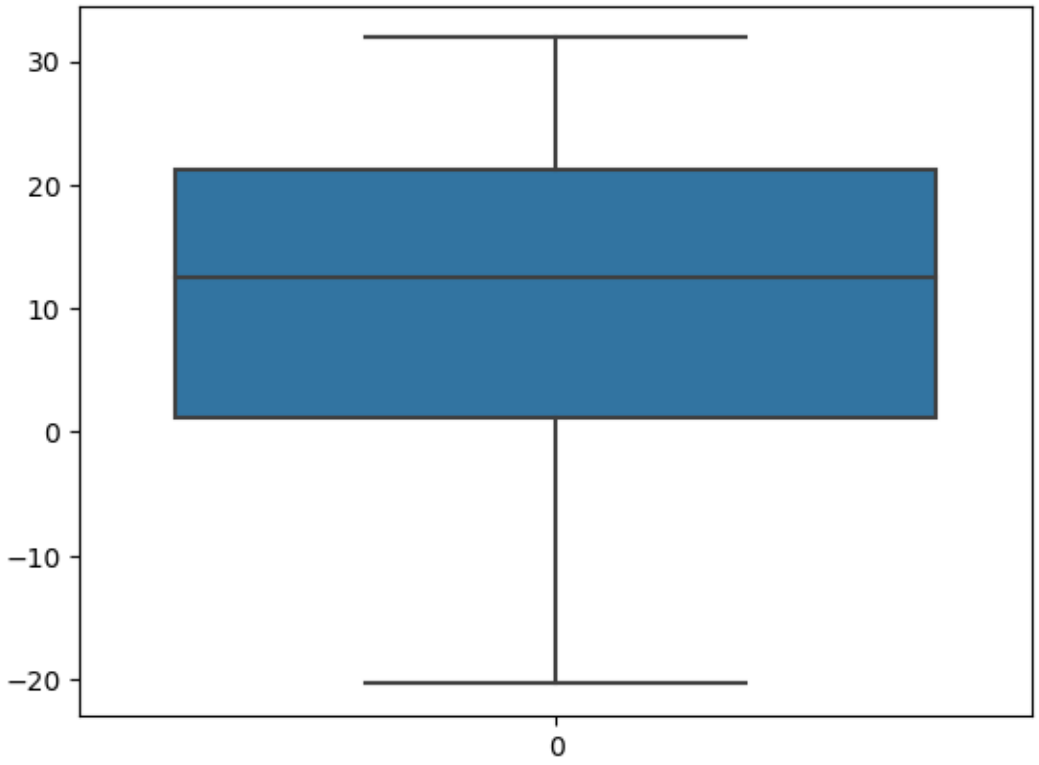
	City	date	AvgTemperature
405901	Brasilia	1995-06-08	15.722222
406288	Brasilia	1996-06-29	13.666667

```
406289  Brasilia 1996-06-30       14.333333
406608  Brasilia 1997-05-15       16.111111
406632  Brasilia 1997-06-08       15.944444
…          …        …              …
414742  Brasilia 2019-10-20       28.555556
414754  Brasilia 2019-11-01       27.944444
414764  Brasilia 2019-11-11       27.611111
414765  Brasilia 2019-11-12       28.500000
414766  Brasilia 2019-11-13       27.777778

[84 rows x 3 columns]
```

[49]:
```
sns.boxplot(data[data["City"] == "Pyongyang"]["AvgTemperature"].to_numpy())
```

[49]: <Axes: >



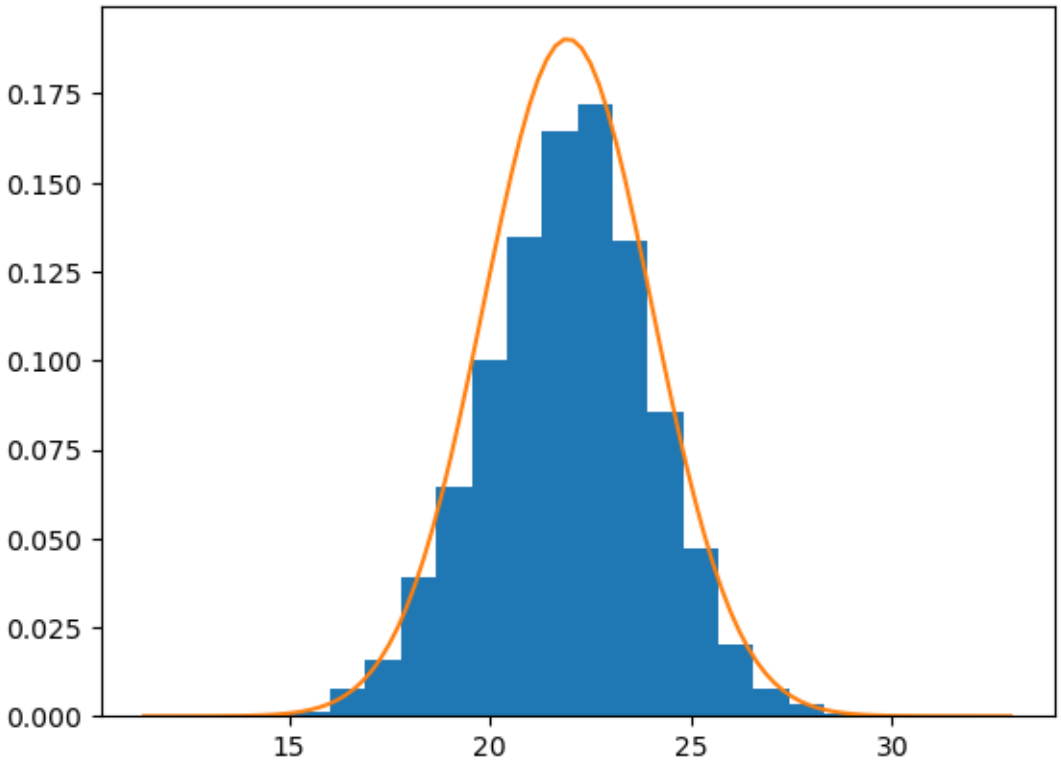
Again no outliers in Pyongyang because the data is not normally distributed.

1.1 Check for Normal Distribution: Draw Histogram and visually check if normal curve is close

Other statistical methods such as Kolmogorov-Smirnov, Shapiro-Wilk lead to hard facts

[71]:
```
from scipy.stats import norm

p_data = data[data["City"] == "Brasilia"]["AvgTemperature"]
xmin, xmax = p_data.min() - 2, p_data.max() + 2
mu, std = p_data.mean(), p_data.std()
weights = np.ones_like(p_data)/float(len(p_data))
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.hist(p_data, bins=20, weights=weights)
plt.plot(x, p)
plt.show()
```



[57]:

[ ]: