Identifying Outliers in a Data Series

Module 1 | Chapter 5 | Notebook 5

In this lesson, we'll shift the focus away from robust predictions when there are outliers in the data set. Instead, we'll now look at identifying outliers, building on the first lesson of this chapter where you already learned a lot about identifying outliers. You can use this knowledge to remove outliers, or alternatively, to take a closer look at the outliers. By the end of this lesson you will be able to:

- Identify improbable data points in a data series.
- Use robust measures for outlier detection.

Outliers as improbable data points

Scenario: You work for a company that manufactures hydraulic pumps. A large number of outliers were found in the data in a pilot project. As a result, management asks you to identify the times when the temperature measurements were unusual. Then the engineers will look into these afterwards.

We'll start by reading the temperature data from the database, see *Robust Feature Engineering*. The four DataFrames `df_temp1`, `df_temp2`, `df_temp3` and `df_temp4` each represent temperatures in degrees Celsius. Each row represents a machine cycle and every column represents a point in time during the cycle.

In [1]:
```
# module import
import sqlalchemy as sa
import pandas as pd

# connection to database
engine = sa.create_engine('sqlite:///hydro.db')
connection = engine.connect()

# read out temperature data of sensor
sql_query = '''
SELECT *
FROM temperature_sensor_{}
'''
df_temp1 = pd.read_sql(sql_query.format(1), con=connection)
df_temp2 = pd.read_sql(sql_query.format(2), con=connection)
df_temp3 = pd.read_sql(sql_query.format(3), con=connection)
df_temp4 = pd.read_sql(sql_query.format(4), con=connection)
df_temp1.head()
```

Out[1]:

	0	1	2	3	4	5	6	7	8	9	...	51	52
0	46.055	45.949	45.953	45.875	45.801	45.762	45.723	45.723	45.617	45.629	...	46.125	46.125
1	46.457	46.363	46.379	46.301	46.203	46.152	46.152	46.125	46.062	45.980	...	46.551	46.562
2	44.117	44.031	44.031	43.957	43.859	43.871	43.770	43.691	43.613	43.621	...	44.199	44.141
3	45.871	45.793	45.727	45.715	45.621	45.539	45.488	45.445	45.379	45.375	...	45.953	45.871
4	46.047	45.969	45.969	45.883	45.793	45.727	45.734	45.637	45.621	45.543	...	46.055	46.062

5 rows × 61 columns

How can you now find out which measured values are outliers?

How you identify an outlier depends entirely on what exactly you mean by an outlier. In general, an outlier is an extremely unusual data point. But how do you quantify "unusual"? How can the unusualness of a data point be summarized in a number?

Perhaps the most widely used method of identifying outliers is based on the assumption that outliers represent very **improbable data points**. For example, if you assume there is a normal distribution, then slightly less than 5% of the data are more than 2 standard deviations away from the mean, see the corresponding Wikipedia entry on this.

Let's look at the cycle `'C_456'` (row index 144) and temperature sensor 4 as an example:

In [2]:
```python
df_temp4.iloc[[144], :]
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	51	52
144	30.551	30.473	30.547	30.551	30.551	30.551	30.551	30.551	30.551	30.578	...	30.566	30.562

1 rows × 61 columns

The recorded temperature curve looks like this:

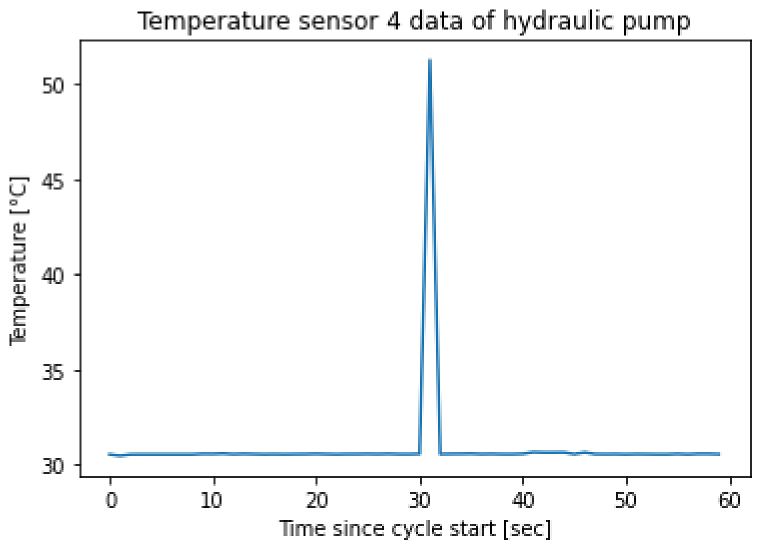
In [3]:
```python
# modul import
import matplotlib.pyplot as plt

# initialise figure and axes
fig, ax = plt.subplots()

# draw line plot
df_temp4.iloc[[144], :-1].T.plot(legend=False, ax=ax)

# optimize plot
ax.set(xlabel='Time since cycle start [sec]',
       ylabel='Temperature [°C]',
      title='Temperature sensor 4 data of hydraulic pump')
ax.set_xticklabels(range(-10, 71, 10))
```

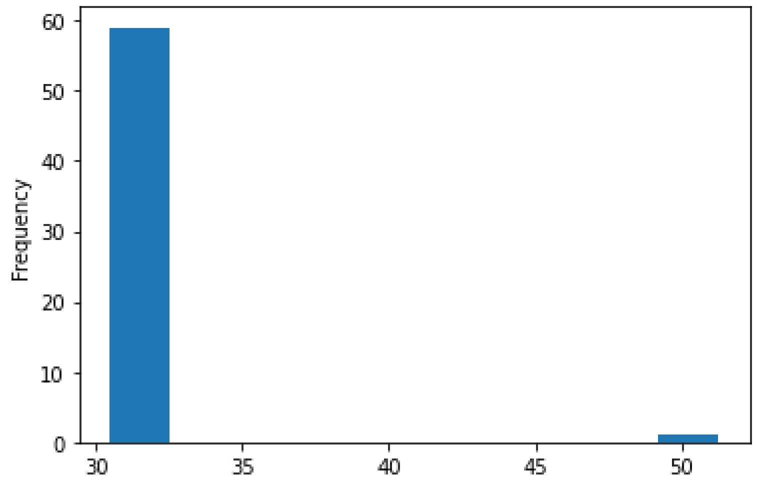
Out[3]:
```
[Text(-10.0, 0, '-10'),
 Text(0.0, 0, '0'),
 Text(10.0, 0, '10'),
 Text(20.0, 0, '20'),
 Text(30.0, 0, '30'),
 Text(40.0, 0, '40'),
 Text(50.0, 0, '50'),
 Text(60.0, 0, '60'),
 Text(70.0, 0, '70')]
```



You have learned previously that outliers can be detected very easily in histograms. Generate a histogram of this data (cycle `'C_456'` with row index 144) from the fourth thermometer.

In [12]:
```
fig, ax = plt.subplots()
df_temp4.iloc[144, :-1].plot(kind='hist', legend=False, ax=ax)
```

Out[12]:
```
<AxesSubplot:ylabel='Frequency'>
```



In both plots, it is clearly visible that actually almost all data is around 31°C. There is only one exception: an outlier at about 50°C. How many standard deviations ($standard\ deviation$) is this data point ($value$) away from the mean of the temperature curve ($mean$)? The following formula shows you how to calculate the distance in standard deviations:

\begin{equation*} distance_{SD} = \frac{|value - mean|}{standard \ deviation} \end{equation*}

Store the distance of the highest temperature value of `'C_456'` (line position number 144) from the mean temperature average of its cycle from thermometer 4 in the variable `distance_sd`.

In [22]:
```
distance_sd = abs(df_temp4.iloc[144, :-1].max() - df_temp4.iloc[144, :-1].mean()) / df
distance_sd
```

Out[22]:
```
7.616378058141554
```

The outlier is all of 7.6 standard deviations from the mean. Assuming a normal distribution of the data, this kind of data point is extremely improbable.

But how improbable must a data point be for it to be considered an outlier? There is no consensus among data scientists about this. The following probabilities are associated with the minimum distances specified, assuming a normal distribution of the data, see *Continuous Distributions: Measuring Values (Module 0, Chapter 3)*.

Distance to mean (standard deviations)	probability
1	31.7%
2	4.6%
3	0.3%

If you choose a lower value, you potentially get more outliers. Generally speaking, you make sure that only really very unusual data points are considered outliers. Therefore, three standard deviations from the mean is a good starting point for defining an improbable data point.

Based on this, how many measured values per cycle are regarded as outliers? The following code cell creates a `DataFrame' called` df_temp4_outliers_count `, which contains the answer to this in its` N_outliers` column. There are several ways to achieve this - in the following you'll see two possibilities that we have pre-programmed:

In [23]:
```
# create empty DataFrame
df_temp4_outliers_count = df_temp4.loc[:, ['cycle_id']]

################################################################################
# option without loop (faster)

# calculate distances to means of cycles
distance_to_mean = df_temp4.iloc[:, :-1].T - df_temp4.iloc[:, :-1].mean(axis=1)
absolute_distance_to_mean = distance_to_mean.abs()
std_distance_to_mean = absolute_distance_to_mean / df_temp4.iloc[:, :-1].std(axis=1)

# identify outliers
mask_outliers = std_distance_to_mean.T >= 3

# count outliers and add to DataFrame
df_temp4_outliers_count.loc[:, 'N_outliers'] = mask_outliers.sum(axis=1)

df_temp4_outliers_count.head()
```

Out[23]:

	cycle_id	N_outliers
0	C_312	0
1	C_313	0
2	C_314	0
3	C_315	1
4	C_316	1

Now we can identify how many measured values per cycle were identified as outliers. For example, no outliers were identified for the cycles `'C_312'`, `'C_313'` and `'C_314'`, whereas a single outlier was found in each of `'C_315'` and `'C_316'`.

Congratulations: Now you've almost completed the task. We can already count extremely improbable values for each cycle. However, management does not want to know **how many** values per cycle are considered outliers, but **which values** are considered outliers. We'll look at this next.

Identifying improbable values

Create a new `DataFrame` named `df_temp4_outliers_mask` with the same structure as `df_temp4`. The only difference is that there should now be a Boolean value in columns `0` to `59`: `True` if the value is extremely improbable (at least 3 standard deviations from the cycle mean) and `False` if it is not. Then print the first 5 rows of `df_temp4_outliers_mask`.

In [29]:
```
df_temp4_outliers_mask = df_temp4.copy()
df_temp4_outliers_mask.iloc[:, :-1] = mask_outliers
df_temp4_outliers_mask.head()
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	True	False	False	False	False	F

5 rows × 61 columns

The first 5 rows of `df_temp4_outliers_mask` should look something like this:

You could now hand this `DataFrame` over to the engineers in your company, who would then investigate what the machine was doing at those exact times. Also generate DataFrames indicating the times of extremely improbable readings for all temperature sensors. Store the corresponding DataFrames, with structures following that of `df_temp4_outliers_mask`, in a list called `list_df_temp`.

In [31]:
```
list_df_temp = []

for sensor_df in [df_temp1, df_temp2, df_temp3, df_temp4]:

    # copy DataFrame for structure
    df_tmp = sensor_df.copy()

    # calculate distances to means of cycles
    distance_to_mean = sensor_df.iloc[:, :-1].T - sensor_df.iloc[:, :-1].mean(axis=1)
    absolute_distance_to_mean = distance_to_mean.abs()
    std_distance_to_mean = absolute_distance_to_mean / sensor_df.iloc[:, :-1].std(axis

    # identify outliers
    mask_outliers = std_distance_to_mean.T >= 3
    df_tmp.iloc[:, :-1] = mask_outliers

    # add to list
    list_df_temp.append(df_tmp)

list_df_temp[1].head()
```

Out[31]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
2	True	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F

5 rows × 61 columns

The engineers could now use the stored `True` values to identify which values in which cycle for each temperature sensor are outliers.

Congratulations: You have identified values that are extremely improbable and marked them for each cycle and temperature sensor. Now we can investigate where these outliers come from. Although this approach to identifying outliers is very widespread, it is not without problems. We'll look at a better option next.

Robust Outlier identification

In the lesson (*Robust Feature Engineering*), you learned that the arithmetic mean and standard deviation are not robust against outliers. Now we have used these same values to identify outliers. This leads to the paradoxical situation that outliers strongly influence the calculation of whether they are outliers or not.

The following visualization illustrates the problem with example data. An extreme value (y-value of 17) appears like an outlier. But it also leads to a relatively large standard deviation of 4.9. As a result, this unusual point is only 2.8 standard deviations from the mean (3.5). If outliers are defined as being points further away from the mean than 3 standard deviations, this point is not recognized as an outlier, although it intuitively looks like an outlier.

Fortunately, however, in the lesson *Robust Feature Engineering* you have already seen robust alternatives to the mean and standard deviation: the median and the *median absolute deviation*. These values aren't as easily influenced.

If we redraw the visualization above with these dimensions, there are two noticeable changes. The median (3.0) is slightly lower than the mean (3.5) and the *median absolute deviation* (1.0) is much lower than the standard deviation (4.9). So the extreme value (y-value of 17) is now clearly identified as an outlier. At the same time, another, less conspicuous point (y-value of -2) is now also recognized as an outlier. By using robust measures, it's possible to identify outliers that you might not have noticed before.

Thanks to a normalization constant, you can interpret the *median absolute deviation* values output with the `mad()` function from `statsmodels.robust` in a similar way to standard deviations. You can find more on the documentation page. Assuming a normal distribution, the same probabilities apply:

Distance to median (*median absolute deviation*)	probability
1	31.7%
2	4.6%
3	0.3%

The following formula shows you how to calculate the distance in median absolute deviations: \begin{equation*} distance_{MAD} = \frac{|value - median|}{median \ absolute \ deviation} \end{equation*}

Now calculate the outliers which are at least 3 *median absolute deviations* away from the median for the fourth temperature sensor. Save them in the `DataFrame` `df_temp4_robust_outliers_mask`, which should have a similar structure to `df_temp4_outliers_mask`. Then print the first 5 rows of `df_temp4_robust_outliers_mask`.

In [38]:
```
from statsmodels.robust import mad
```

```python
# copy DataFrame for structure
df_temp4_robust_outliers_mask = df_temp4.copy()

# calculate distances to means of cycles
distance_to_median = df_temp4.iloc[:, :-1].T - df_temp4.iloc[:, :-1].median(axis=1)
absolute_distance_to_median = distance_to_median.abs()
mad_distance_to_median = absolute_distance_to_median / mad(df_temp4.iloc[:, :-1], axis

# identify outliers
mask_robust_outliers = mad_distance_to_median.T >= 3

df_temp4_robust_outliers_mask.iloc[:, :-1] = mask_robust_outliers

df_temp4_robust_outliers_mask.head(5)
```

Out[38]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	True	False	False	False	False	F

5 rows × 61 columns

If you look at the first 5 lines of `df_temp4_robust_outliers_mask` , you will notice that the robust method identifies several points in time as outliers, which were not marked as such in `df_temp4_outliers_mask` . The two tables only converge in their detection of outliers when the threshold value of the *median absolute deviation*, from which point a value is considered an outlier, is increased. You are welcome to try this in the previous code cell by changing the `mad_distance_to_median` threshold from 3 to 8.

Why does robust outlier detection identify so many more outliers? This is probably related to the fact that during a cycle, temperatures can also shift gradually. The approaches for identifying outliers that you have learned so far cannot cope with this. From the point of view of outlier detection using the *median absolute deviation* distance to the median, it looks as if as the temperature curve rises, early values are unusually far below the median and later values are unusually far above the median.

Identifying outliers based on the mean value

Both outlier detection using the standard deviation and using the *median absolute deviation* assume that the values of a data series **randomly** fluctuate around a central value (mean or median). If they don't, these outlier identification methods aren't so suitable. Then you might prefer regression-based methods to detect outliers

Regression-based outlier detection

Now you're done, close the connection to the database.

```
In [40]: connection.close()
```

Congratulations: You have learned how to detect outliers in the data by defining them as implausible or unusual values. If you follow this very common approach, you should use a robust method with *median absolute deviation* and the median instead of the more popular method with standard deviation and the mean value. However, either way, this approach cannot deal with the fact that values systematically shift within the data series. Next, we'll look at how to detect outliers in a robust way in a situation like that.

Remember:

- Outliers can be defined as improbable values
- If values fluctuate randomly, values greater than 3 standard deviations from the mean are only 0.3% likely.
- The *median absolute deviation* distance from the median is a robust estimate of the improbability of a measured value

---

Do you have any questions about this exercise? Look in the forum to see if they have already been discussed.

Found a mistake? Contact Support at support@stackfuel.com.

The data was published here first: Nikolai Helwig, Eliseo Pignanelli, Andreas Schütze, 'Condition Monitoring of a Complex Hydraulic System Using Multivariate Statistics', in Proc. I2MTC-2015 - 2015 IEEE International Instrumentation and Measurement Technology Conference, paper PPS1-39, Pisa, Italy, May 11-14, 2015, doi: 10.1109/I2MTC.2015.7151267.