@author: Fred Coerver – 2022 https://github.com/FredC123/Fast-Anomaly-Detection

Description how this model works

Anomaly detection is a concept, which is discussed in many broad area's, disciplines and mathematical concepts. Various ways can be used to approach the data and find anomalies. In most cases, just like this method the dataset can be huge. This model however is used to find anomalies in a fast way and with a flexible sensitivity. This method is very successful in smaller datasets due to the fact that it is very fast and very precise. The input data is x and y and the model return the x values and identifies this value whether it is an anomaly or not, given the hyper parameters set by the user.

Thus the model is flexible, precise, can be used for a huge set of x,y values as well all shorter datasets and a dataset with x, y1, y2,….,yn can be processed in the loop of a python script very easy and remains still very fast.

The concept of this algorithm is not like machine learning like used in scikit.

Though the concept of testset and trainingset is applied. The testset is the evaluated value (x[j],y[j]) and the trainingset is (x[j-w] + x[j+w],y[j-w] + y[j+w]), where x and y are lists and w is the trainingset window parameter (integer w>=1).

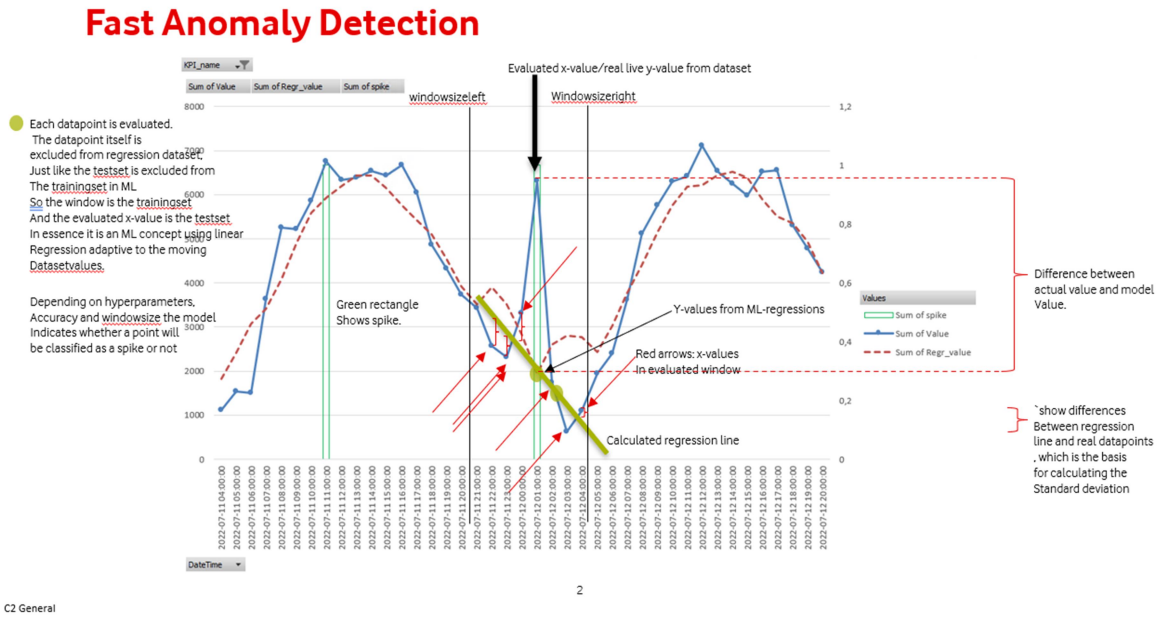
The trainingset is calculated via linear regression and the testset is calculated via slope/interception.

Each x[j] is calculated, resulting that the regressed polynomial is close to the original. See below graph.

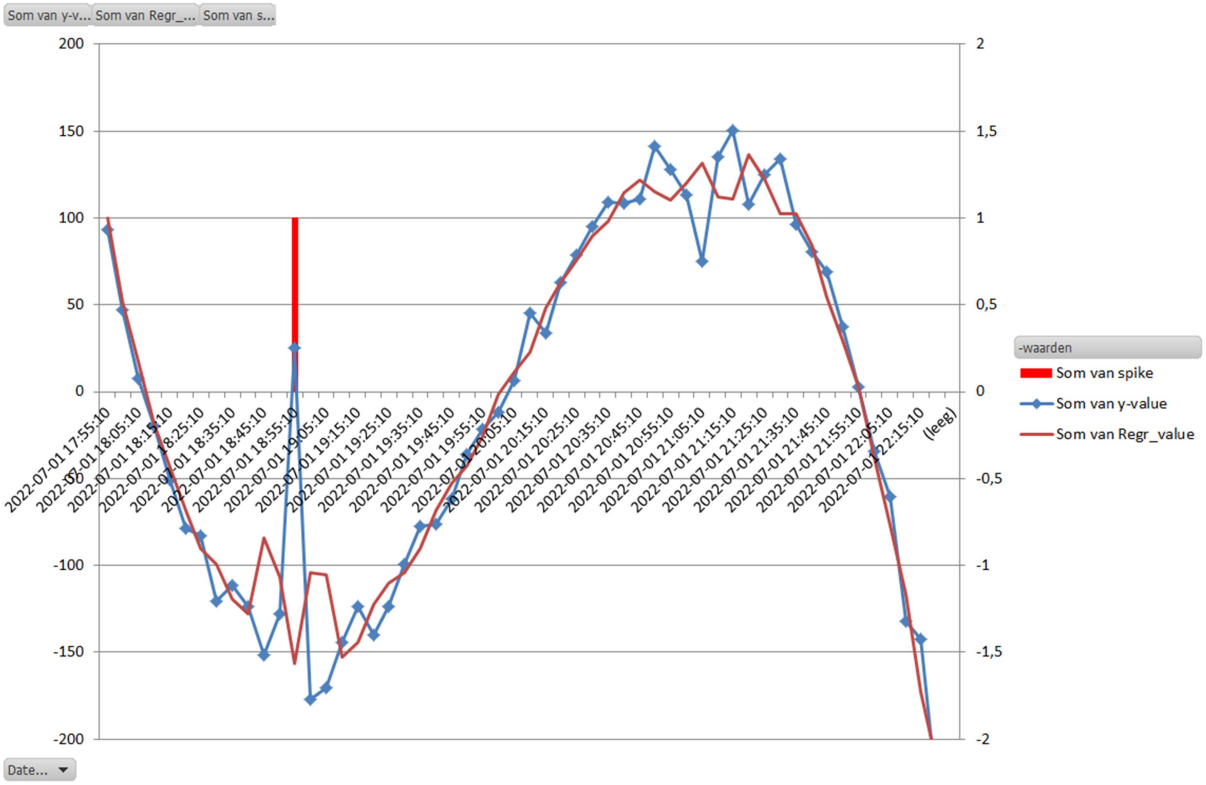
Blue are the original values, and the red dotted line is the model polynomial. From there the spikes are identified via the hyperparameters.

The overall similarity between ML-algorithms in Scikit and this model, is that the setting of the hyperparameters is extremely important and must be optimized for each x,y[j] combination.

A good example of applying this model is fast anomaly detection of recent data, from a live data-feed

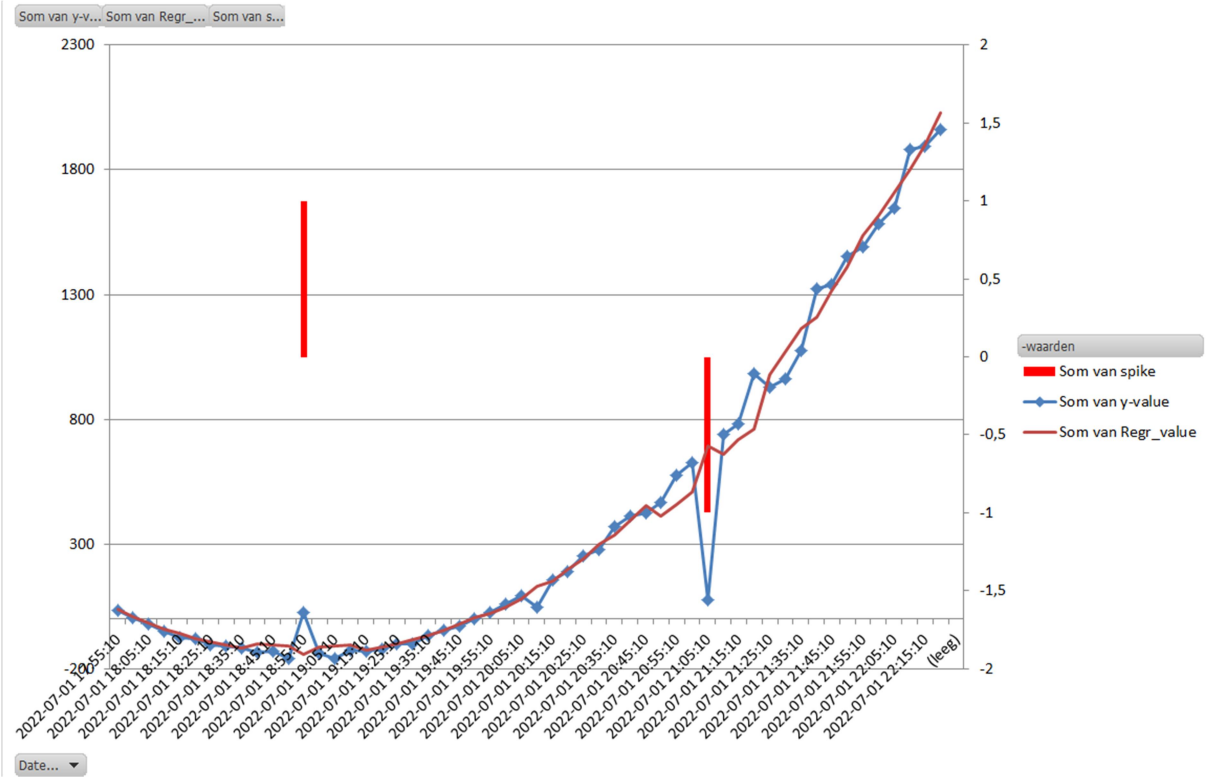


An indication of the sensitivity is shown below:



Below graph shows that the model is very good in following increasing trends, which is giving you a hard time with scikit models.

The downside of this model is prediction, which is really not possible if you understand the concept. Nevertheless for anomaly-detection this is not needed anyhow and prediction is not in scope of this model.



INPUT for the model

This module takes a pandas frame as input, together with several hyperparameters.

The pandas data frame must have a straight forward x and y column including an index. Not more Not less.

The output are two pandas dataframes. One table with statistical information of the input and whether a value x is identified as a spike or dip in the dataset, based on the input parameters. And the second is a list of spikes/dips. More details about the output below. The algorithm creates a window of datapoints from x - windowsizeleft to x + windowsizeright

Linear regression is performed on this window/x-range with the give y-values. The regression value : y = slope*x + intercept using the standard Python module Scipy.

This window is iterated over the pandas frame and the regression values are put into the output table, along with the other statistical values.

Input dataframe structure :

	DateTime	y-value
3	2022-07-01 17:50:10	131,82
4	2022-07-01 17:55:10	87,87
5	2022-07-01 18:00:10	46,05
6	2022-07-01 18:05:10	8,75
7	2022-07-01 18:10:10	-23,48
8	2022-07-01 18:15:10	-51,48
9	2022-07-01 18:20:10	-76,28

1) pandas index as int
2) x or DateTime value format %Y-%m-%d %H:%M:%S {working on an update that also accepts floats as x}. Column name must be "DateTime" if Timescale is true and must be next/right to the index
3) y {float or int}. Columnn name is free.

e.g. Regression is internally in the model executed on :

x=[1657540800, 1657544400, 1657548000, 1657555200, 1657558800, 1657562400]

y=[115730244, 117300778, 116863076, 108493789, 97012607, 95430706]

The data frame can have more columns, but the script only analyse 1 columns at a time.

So in case you want to analyse more columns, you need to call the script for each column and slicing the data frame in a way that the input frame complies to 1), 2) and 3)

Hyperparameters

ignore_startsamples = 5 (in case you want to omit starting rows from your calculation of the dataset. This parameter does not delete the rows! = default is 5. In case you increase the windowsizeleft, you might need to increase this value as well)

ignore_endsamples = 3 (in case you want to omit starting rows from your calculation of the dataset. This parameter does not delete the rows! In case you increase the windowsizeright, you might need to increase this value as well)

P1inc = 10 ---> In case the standard deviation of a regression window is P1inc-times higher as the previous std, then this x value is marked as a spike

accuracy = 4 ---> If a y-value exceeds the regressed value +- accuracy * std then this x-value is marked as spike

window_sizeleft = 3 ---> the regresssionwindow is #windowsizeleft x-values from the analysed x-point and #windowsizeright x-values from the analyzed x-point. So suppose the algorythm calculates whether x[j] is a spike, then the window x[j-windowsizeleft] until x[j+windowsizeright] is considered as the window of x-points. From these x- an y-point the regression is calculated, and the regression value is compared with the actuel y-value (Note : x[j] itself is not part of the evaluated data for the slope and intercept)

window_sizeright = 3 ---> amount of datapoint right from the x-value, which is subject for analyzing whether it is a spike/dip or not

ignoreendsamples = 3 ---> the Endresult output dataframe is capped with [ignorestartsamples:-ignoreendsamples] before returning to main

Timescale ---> If x-values are different from time then any string is valid, in case the x-values need to be in timeformat then this parameter is True.

OUTPUT from the model:

1) Pandas dataframe with format columns:

'DateTime', 'y-value', 'KPI_name', 'Regr_value', 'Value', 'Unixtime',

'Std', 'spike', 'spikevalue', 'Slope', 'Intercept', 'Diff',

'Regr_value_plus_std', 'Regr_value_min_std'

Example of output data

0	DateTime	y-value	KPI_name	Regr_value	Value	Unixtime	Std	spike	spikevalue	Slope	Intercept	Diff
3	1-07-22 17:50	131,82	y-value	0	131,8246875	1656690610	1	0	131,8246875	1	1	1
4	1-07-22 17:55	87,87	y-value	105,3285938	87,87	1656690910	13,68991298	2	105,3285938	-0,147413611	244218894,9	17,45859376
5	1-07-22 18:00	46,05	y-value	63,27097657	46,0496875	1656691210	13,4872271	0	46,0496875	-0,132170486	218965745,8	17,22128907
6	1-07-22 18:05	8,75	y-value	25,35957032	8,75	1656691510	12,38891899	0	8,75	-0,117099392	193997594,5	16,60957032
7	1-07-22 18:10	-23,48	y-value	-8,072265625	-23,4815625	1656691810	11,53810345	0	-23,4815625	-0,102407118	169657025,7	15,40929688
8	1-07-22 18:15	-51,48	y-value	-36,92554688	-51,48	1656692110	11,58527116	0	-51,48	-0,089119167	147642983,3	14,55445312
9	1-07-22 18:20	-76,28	y-value	-61,78222656	-76,28125	1656692410	11,21849345	0	-76,28125	-0,075613854	125268836,5	14,49902344
10	1-07-22 18:25	-97,72	y-value	-82,89015625	-97,7175	1656692710	10,32442158	0	-97,7175	-0,063181215	104671775,9	14,82734375
11	1-07-22 18:30	-113,59	y-value	-100,7435547	-113,59375	1656693010	9,639655553	0	-113,59375	-0,051804826	85824593,02	12,85019532
12	1-07-22 18:35	-127,00	y-value	-115,0700781	-127	1656693310	8,938657961	0	-127	-0,040908472	67772677,18	11,92992188
13	1-07-22 18:40	-135,78	y-value	-131,2869922	-135,7846875	1656693610	8,249602908	0	-135,7846875	-0,039232882	64996733,53	4,497695315
14	1-07-22 18:45	-144,25	y-value	-138,7302344	-144,25	1656693910	14,09648906	0	-144,25	-0,02659375	44057564,94	5,519765623
15	1-07-22 18:50	-149,95	y-value	-143,3455859	-149,9534375	1656694210	16,60949336	0	-149,9534375	-0,015253229	25269793,1	6,607851561
16	1-07-22 18:55	-151,50	y-value	-145,4117969	-151,5	1656694510	17,92330031	0	-151,5	-0,004027153	6671616,486	6,088203125

2) Pandas dataframe as 1) but then sorted and 'spike' >=1 or <=-1

Same structure as frame in 1) but applied filter >=1 or <=1

"""

Attached script is included in the repository as an example.

```
7
8  import pandas as pd
9  import numpy as np
10 from scipy import stats
11 import sys
12 import datetime
13 import time
14 import matplotlib as plt
15 if not sys.warnoptions:
16     import warnings
17     warnings.simplefilter("ignore")
18
19 pd.options.display.float_format = '{:,.6f}'.format
20
21 #Userdefined module
22 from fad_v02 import Get_SandD
23
24 ######################### READ DATA ################################################
25 td = pd.read_excel("./Testdata_inout2.xlsx",sheet_name="inputdata",usecols="B:C")
26 writer = pd.ExcelWriter("./Testdata.xlsx")
27
28
29 ######################### INITIALIZE DATA   ########################################
30 kpi_input = td.columns.to_list()
31 t=1
32 P1inc=20
33 accuracy = 10
34 window_sizeleft = 2
35 window_sizeright= 2
36 Time_scale=True
37 ignore_startsamples = 4
38 ignore_endsamples =4
39 td_in = td[[kpi_input[0],kpi_input[t]]]
40 td_in = td
41
42 ########################  CALL MODULE   ###########################################
43 dfSin_gl, spikelist_gl = Get_SandD(td_in,
44                                    acc=accuracy,
45                                    windowsizeleft=window_sizeleft,
46                                    windowsizeright=window_sizeright,
47                                    sp_name=kpi_input[t],
48                                    P1=P1inc,
49                                    ignorestartsamples = ignore_startsamples,
50                                    ignoreendsamples = ignore_endsamples,
51                                    Timescale=Time_scale
52                                    )
53
54 ########################    write data to excel   ##################################
55 dfSin_gl.to_excel(writer,sheet_name="Sheet4")
56 spikelist_gl.to_excel(writer,sheet_name="Sheet5")
57 writer.save()
```

The definition GetSandD() is included in the repository