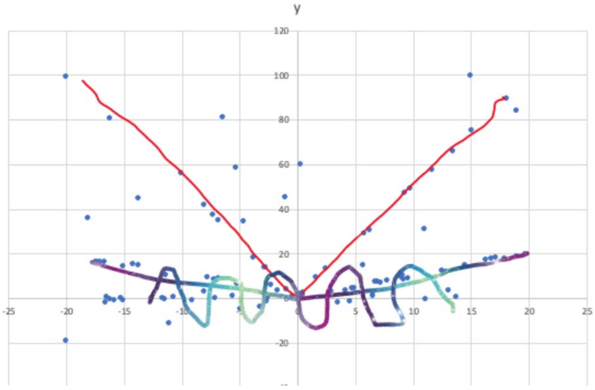
Helpful Tipps for doing the Python Programming Assignment!

Before we get into Python, I'll try to create a solution sketch using Excel. This is

sometimes, in reality, always useful, especially if the amount of data allows it.

Let's start at the target. We have a bunch of test data (test.csv) that represents some reality. If you look closely, you can do it by hand, as shown in the following figure,

i.e., different aspects of our reality are described by different functions.

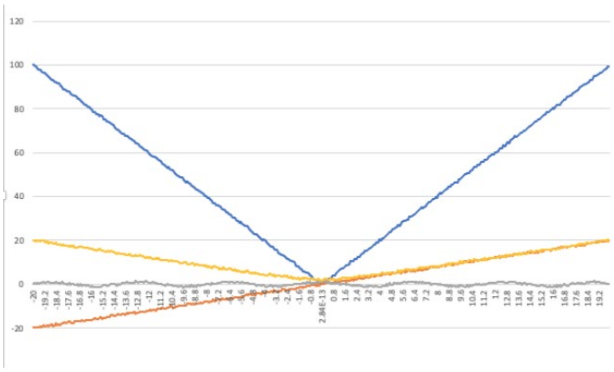


The Training Dataset contains these four different functions. Think of these functions as

Knowledge that represent the reality I have constructed. As you can see in the following figure

the four representations are somewhat noisy. Imagine, that these are individually measured

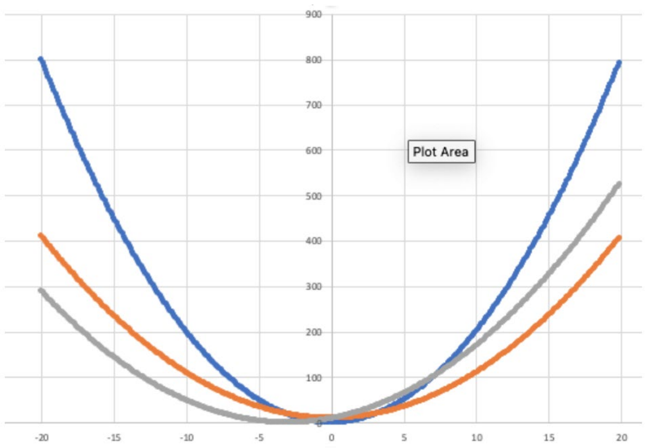
data or also generated data.



When approaching problems algorithmically, one would of course prefer to have a closed

function or at least a good mathematical function, which maps the noisy data relatively well.

In this task we provide 50 different versions of four functions that are a good approximation of our 4 functions that we think we can find in the test data set. We call these "ideal functions". Here are three examples from the "ideal functions" data set.



Now of course we don't want to paint it by hand and we don't want to do it with Excel, and so:

- * means: Find the 4 ideal functions that are closest to our model (the 4 noisy

training functions) come closest. Least Square and that gives us a noise-free

Data set for each of our 4 functions. Ie, instead of my noisy training functions,

I now use clean data from the ideal functions data set.

- ** means: Can I fit the 4 ideal functions to reality with a margin of error-fit? As we have already seen in my artwork above, in a few places this goes quite well. Others, like the sinusoidal curve, not so sure. How do we do this and what makes sense?

Some more tips for ** are:

- Go into each data point in the test/reality data set and find all points that are "close enough" (e.g. smaller than Sqrt(2)) to one of the 4 ideal functions, found in *. This means, I do this now four times, for each ideal function.
- Then you can say: I have n test data (where n<N with N=number of all test data), which can be fitted to one of my four ideal functions. Of course it can happen that a point is close to two functions, but that's ok. For each function we save the corresponding found n test data.

Source: Help for Python Hausarbeit IU (Deutsche Variante)