2. housing market, one variable
3. This is a report [211] in which I use ML for improving wall-functions for Large Eddy Simulations.

Next, look at the Python script *An example of Python code for Machine Learning for optimizing a k-omega turbulence model.* at the homepage. This script uses DNS data to optimize the coefficient C_μ in the $k-\omega$ model'where the turbulent viscosity is computed as $\nu_t = C_\mu k/\omega$, $C_\mu = 1$, see Eq. 16.11.

Assignment 1.9. Watch the recorded lecture on Appendix 1, Part II (you can download it from the course www page, Assignment 1). Try to understand the Python script. I use $\partial\bar{v}_1/\partial x_2$ as input (influence variable) and C_μ as output (target). When you execute the script, you find that you get different result (i.e. different RMS error) every time: why?

The RMS error is rather large (close to 20%). 80% of the DNS data (randomly chosen) are used for training. Maybe that's a bad idea; maybe it would make sense to only use data for which $|\overline{v_1'v_2'}|$ is large. Try that. What happens if you now change the C value in the `svr`-call? Does the ε value have any influence?

Assignment 1.10. Now you have developed a new ML $k-\omega$ model. Let's find out if it gives better results! I.e. implement the ML model in a Python CFD code. First you need to export your ML `svr` $k-\omega$ model by saving it to disk and then loading it in your Python CFD code. I show how to do that at pp. 13-14 in [211].
Now, which Python CFD code should you use? I give you three options.

1. In the CFD course in Study Period 2 you wrote a CFD code for fully-developed channel flow. Use that one.
2. You can use the Python code `rans-k-omega.py` which solves the fully-developed flow in a 1D channel (it solves $\bar{v}$, k and ω).
3. You can use my Python code **pyCALC-RANS** [212]. It can be downloaded here. I recommend that you compute fully-developed channel flow at $Re_\tau = 5\,200$, see Section 8 in [212]. The turbulent viscosity is computed in module `vist_kom`. At the end of this module, the module `modify_vis` is called which resides in file `modify_case.py`. Here you can re-compute ν_t using your ML $k-\omega$ model.

Assignment 1.11. Here you make a new ML model (and CFD predictions) as I describe in the recorded lecture. You will use $|\overline{v_1'v_2'}|$ as output and $\partial\bar{v}_1/\partial x_2$ and ν_t as input. Here we introduce C_μ in the Boussinesq assumption as $\overline{v_1'v_2'} = -C_\mu \nu_t \partial\bar{v}_1/\partial x_2$. The DNS data gives C_μ close to one everywhere.

Assignment 1.12. Finally you make a ML model for the mixing length model as I describe in the recorded lecture.