# Analysis of Courses of Study to Predict Consequences of Educational Choices Using Machine Learning

## Bachelor Thesis for the Bachelor of Science Cognitive Science at the Institue for Cognitive Science, Osnabrück University

This is the code base used to survey, analyze, visualize and process the NEPS dataset from the LIfBi.

### Structure

The main file is `core.py`. It runs the pipeline of loading the data -> filtering the data -> cleaning the data -> aggregating the data -> joining the data -> output a dataframe usable for machine learning tasks.

The pipeline code is found in the `thesis` package. If starting from scratch, the program expects the NEPS Stata `.dta` files in a folder in the working directory called `data` and inside a folder `raw` which holds all files. This path can be changed in the file itself. There is no command line available yet to configure the parameters from outside.

`thesis` consists of several subpackages, listed in the order the pipeline uses them:

- `preprocessing`: here the two modules `preprocess` and `aggregate` are found. They do the heavy lifting of reading in the data, filtering it, cleaning it, aggregating the several files and joining them then.
- `model`: contains modules for the ML algorithms used. `ffn`, `svm`, `decision_tree` and `random_forest` are wrapper for the scikit-learn implementations. They all host a class to describe the model, which derives from the base class `Model` defined in the module `models`. It offers capabilities for saving and loading, in extension to wrap the scikit-learn functions.
- `train`: hosts the grid parameter search and training of the models. It only contains the module `trainer`. The parameter grid is defined for each model independently, inside the python file.

Additionally the package `Prototyping` can be found here. It hosts prototyped code, that was tested and evaluated before being integrated into the pipeline or discarded. The `util` module in `thesis` contains functions for saving and loading Python structures to and from binary structures using `joblib`.

The `thesis` package does not yet contain functionality for evaluation and visualization of the results from training. These functions can be found in the jupyter notebook `Feedforwardnetwork, Plotting and Evaluation`, which - as the name implies - also contains code for the feed-forward network. Since the ffn never left the prototyping and testing stage, the TensorFlow code was not migrated into the automatic pipeline. All plots and analysis performed can be found in that notebook.

`proto.py` contains code from testing the feed-forward network implementation of scikit-learn as the two frameworks TensorFlow and scikit-learn were compared.

### Going through the code

When examining the code, I would suggest following the process of the pipeline. Starting at `core.py` and work through `preprocess.py`, which is probably by far the largest chunk, and then onwards to models and training. The notebook has the least documentation as it was mostly for ongoing work and plotting. The results from the notebook are discussed in the thesis itself.