MACHINE LEARNING

SHEET 0

04.11.2021

10:15 A.M. UNTIL 11:45 A.M. VIA ZOOM

Join the meeting via this Zoom link:
https://uni-trier.zoom.us/j/82836620106?pwd=RUhrR1BxSjh4aDJhQVBHOFRrNlNtUT09

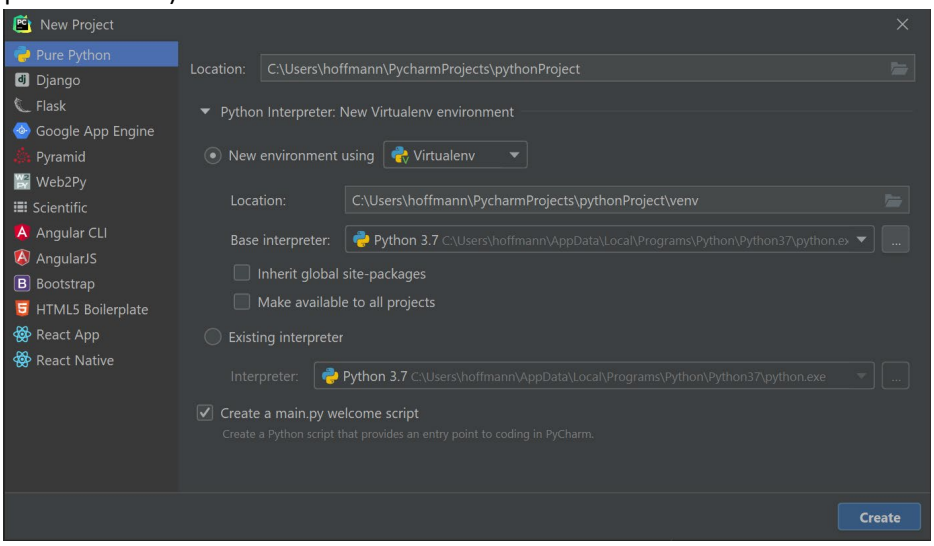
INTRODUCTION AND SETUP

The first step of setting up our programming environment is installing Python 3. For this course, we will use **Python 3.9**. You can find detailed instructions on downloading and installing Python at https://www.python.org/downloads/.

The IDE that we recommend is the latest version of **Pycharm** (https://www.jetbrains.com/de-de/pycharm/) since it allows convenient editing of code and provides further functionality, e.g., data visualization or remote access. Additionally, Pycharm's debugger is a very useful tool to inspect code, variables, and data collections which is essential for our purpose. Jetbrains offers a free license of their professional products (including Pycharm Professional) to students. We recommend to use this version since some functions that we use during the course might be not supported by the community version. Although we recommend Pycharm, this does not mean you have to use it for yourself. For instance, another popular way of creating Python code is with Jupyter Notebooks (https://jupyter.org/). Feel free to use the programming environment that you prefer. But please be aware that all solutions you hand in have to be in a given format (see section *Submissions*) and we can only give support for Pycharm.

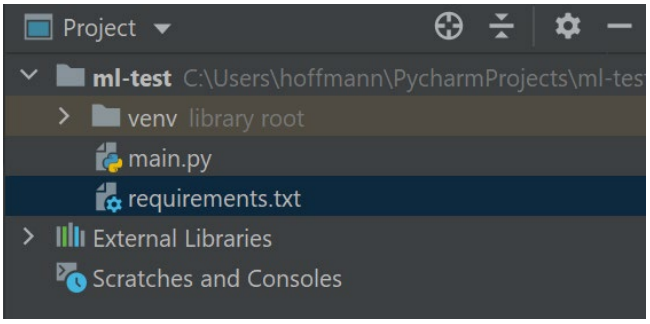
The following passage will guide you through the process of setting up Pycharm in the way we recommend:

1. Create a new local project. This can either be done in the start-up dialog or via *File* > *New project…* After that, a window will open that is used for setting some options (see picture below).

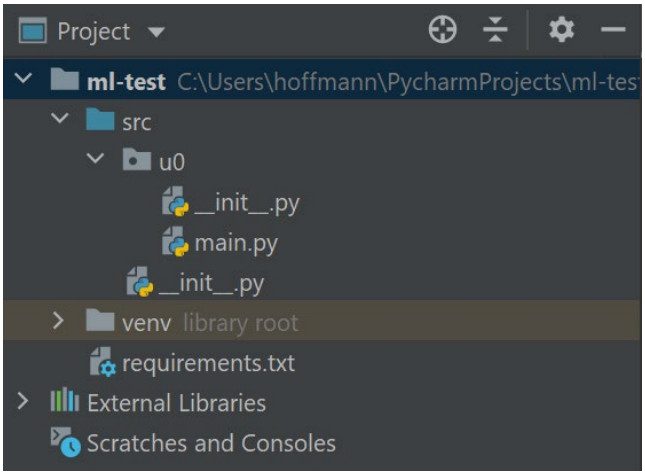


DEPARTMENT OF BUSINESS INFORMATION SYSTEMS II
PROF. DR. RALPH BERGMANN
UNIVERSITY OF TRIER

2. In this dialog, you should choose a pure python project and create a virtual environment that manages all of your project dependencies. We recommend using **Virtualenv** for this. The virtual environment should be used for executing all of your Python scripts, thus allowing you to separate the project dependencies from the dependencies on your host machine. You may have to select Python 3.9 as your base interpreter, i.e., the interpreter that your virtual environment uses under the hood.
3. After Pycharm initialized your project and created the virtual environment, it is time to setup the basic requirements used throughout the course. We use the dependency manager **pip** for this purpose. The dependencies are kept in a file called *requirements.txt* (available in StudIP). For the beginning, we depend on **Numpy** (https://numpy.org/), **scikit-learn** (https://scikit-learn.org/stable), **Tensorflow** (https://www.tensorflow.org/), **Pandas** (https://pandas.pydata.org/), and **matplotlib** (https://matplotlib.org/). You can read more about requirements in general at https://pip.pypa.io/en/stable/user_guide/#requirements-files. In order to install the requirements in the file into your virtual environment, right click inside of the editor view of the file and select the context option *Install All Packages*. Your repository should be displayed similar to the one shown below:



4. Next, we will create the recommended folder structure. All files should be located under the folder *src*. Right click on this folder in the Project panel and mark it as the sources root. Throughout the course, you should use subfolders of *src* for each exercise that you hand in. For instance, this exercise would be in the folder *u0* and the corresponding code in the file *main.py* within this subfolder (see picture below).



5. We fill this file *main.py* with some test code, that checks the correct installation of the dependency libraries. The file that is used for this will be uploaded to StudIP. Please

copy this file and see if everything works for you. Note that Tensorflow prints some debug messages in red although they do not specify errors.

EXEMPLARY TASK

The purpose of this task is to demonstrate how the real tasks on the coming sheets might look like. You do not have to solve it! It is inspired by some introductory examples of the used frameworks.

- Load the diabetes dataset that is provided by scikit-learn as a Pandas dataframe. Add a new column to this dataset that has the value age divided by bmi for every row.
- Train a linear regression model with scikit-learn and the created dataset. Use the trained model to predict the target values given the training data.
- Train a neural network model by using TensorFlow and the given dataset. The neural network should have three dense layers with 128 neurons and ReLU activation for the first layer, 64 neurons and ReLU activation for the second layer, and 1 neuron and no activation for the third layer. Train this model for 25 epochs with a batch size of 32, a mean squared error loss, and the Adam optimizer. Use the trained model to predict the target values given the training data.
- Compare both models in terms of their mean absolute error between the predicted target values and the true target values. Which model performs better? Are the results expressive if this would be a real-world scenario?

SUBMISSIONS

As already mentioned before, we expect your submissions in a certain format that allows us to review the submissions. So please stick to the following guideline:

- Submissions should be a single ZIP archive that is named in the following scheme: **[LastName]_[FirstName]_[StudentNumber]_e[ExcerciseNumber].zip**
- Within the ZIP archive, always submit your code in a single folder that is named in the format **e[ExerciseNumber]**. So, for example, the folder we created before has the naming scheme that is appropriate for exercise 0. The code for exercise 1 should be submitted in a folder named *e1*, the code for exercise 2 as *e2*, and so on. This folder is part of the ZIP archive, that could be named *Hoffmann_Maximilian_12345678_e0.zip*.
- Include all source code files and all additional files (e.g., datasets) in this folder to ensure the executability. Do not put other files, e.g., PDF files, that might have to be submitted into the folder of the source code. Those files should be put into the main ZIP file.
- The main Python file in the code folder is named *main.py.* It contains the main executable code.