Data Science Practicals: Final Assignment

Adil Ibraheem Koyava, Dieter Kalwa & Keerti Belmane

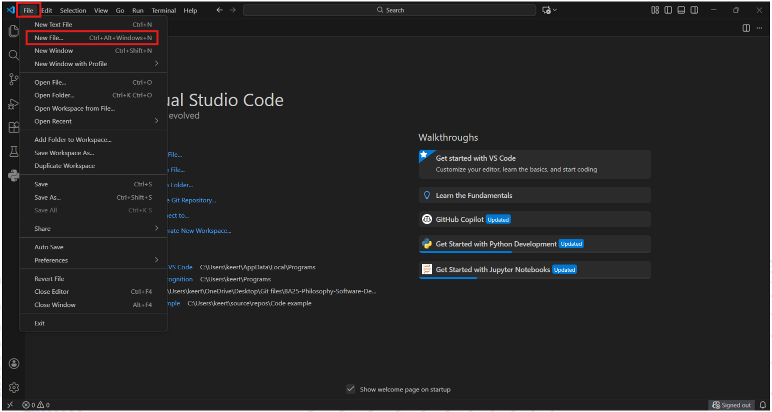
16. Dezember 2025

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Creation of .ipynb file

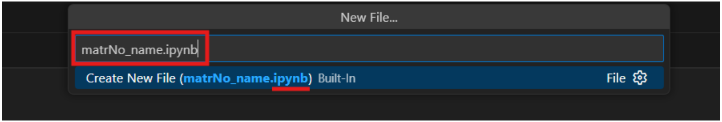
University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Creating the file



- Click on "File"
- Click on "New File..."

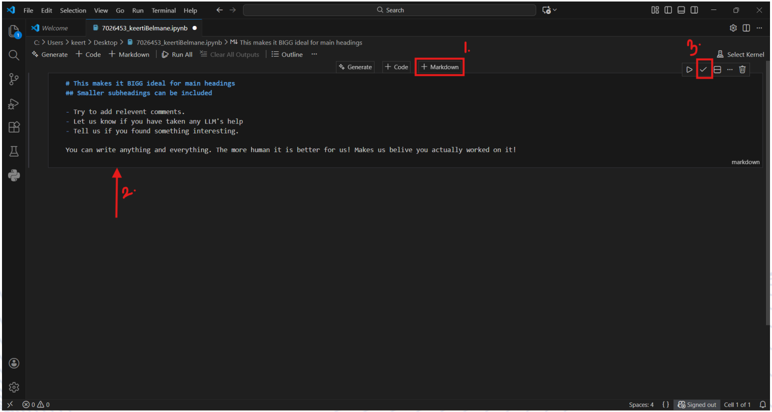
Naming the file



- Name the file with your **MatriculationNumber_Name.ipynb**.
- **Not .py**!! Save it as **.ipynb**, this makes it the interactive jupyter notebook that we are looking for.

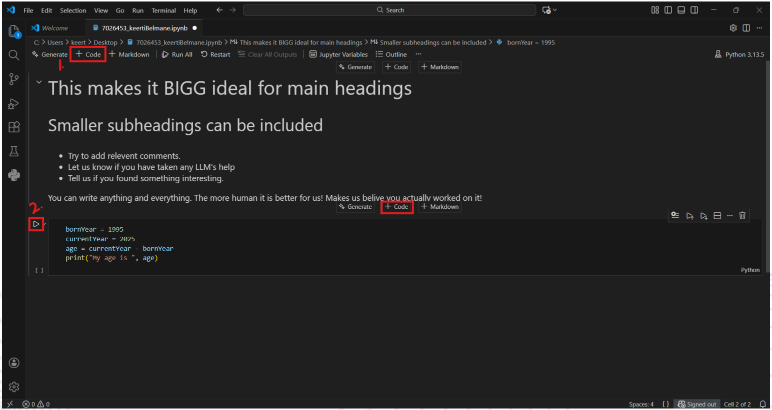
University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Adding Markdown Comments



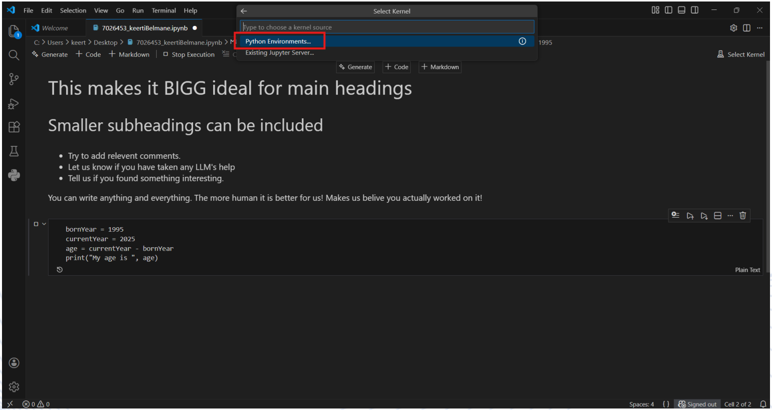
- Click on the Markdown, and add your comments.
- Click on the tick mark on the right up corner of the markdown cell. This will save it.

Adding Code cells



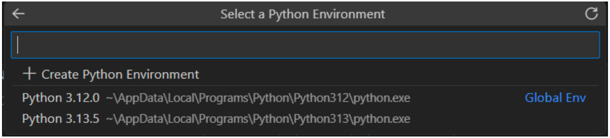
- Click on the code, to create the code cell. Here write all the code you want to implement.
- Click on the execute button, to execute the code.

Setting the python variable



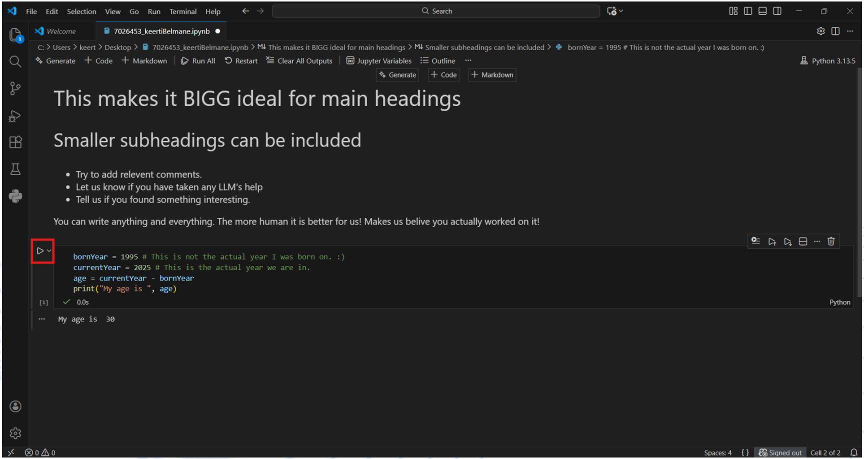
- 1st time when you try to execute, you will have to set your python environment.
- Click on Python Environment

Select the python version



- Select the python version you would like to use.

Happy Coding to you! :) All the best



Final Assignment: Overview

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Bike Sharing Demand Dataset I

The final project uses a real-world bike sharing demand dataset (CSV file) based on historical usage of a public bike rental system.

The dataset contains:

- A **target variable** `count`: number of rented bikes in a given time period.
- **Time information**: a `datetime` column (date and hour or date and day).
- **Context features**: weather, temperature, humidity, windspeed, season, holiday and working day indicators.

The overall goal is to build:

- A model that predicts bike demand from given conditions (“what-if” prediction).
- A time-series model that forecasts the next 30 days of total demand, following the pipeline from the lecture.

Dataset Variables I

Variable Name	Type	Description
`datetime`	Datetime	Date and time of the observation (hourly or daily).
`count`	Numeric	Total number of bikes rented (target variable).
`season`	Categorical	Season of the year (1=spring, 2=summer, 3=fall, 4=winter).
`month`	Numeric	Month of the year (1–12).
`hour`	Numeric	Hour of the day (0–23), if hourly data.
`weekday`	Categorical	Day of the week (0=Monday, ..., 6=Sunday).
`holiday`	Binary	Whether the day is a holiday (0=no, 1=yes).

Tabelle: Bike Sharing Dataset Variables (1/2)

Dataset Variables II

Variable Name	Type	Description
`workingday`	Binary	Whether the day is a working day (0=no, 1=yes).
`weather`	Categorical	Weather condition code (1=cle-ar, 2=mist, 3=light rain/snow, 4=heavy rain/snow).
`temp`	Numeric	Temperature in °C.
`atemp`	Numeric	"Feels like" temperature in °C.
`humidity`	Numeric	Relative humidity (%).
`windspeed`	Numeric	Wind speed (km/h or similar unit).
`casual`	Numeric	Number of casual users. *Do not use in Task 2 or 3.*
`registered`	Numeric	Number of registered users. *Do not use in Task 2 or 3.*

Tabelle: Bike Sharing Dataset Variables (2/2)

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Dataset Variables III

Note: The `casual` and `registered` columns are derived from `count` (they sum to `count`) and should **never** be used as input features to avoid data leakage.

Task 1: Describe the Dataset

Task 1: Data Understanding I

Goal: Understand the structure and basic properties of the dataset using Python (NumPy, pandas, Matplotlib/Seaborn).

What to do:

1. Read the CSV file with `pandas.read_csv` and parse the `datetime` column.
2. Report:
    - Number of rows and columns.
    - Time range covered by the data.
    - Target variable and list of feature variables (names and data types).
3. Create a **variable description table** (see above for reference).
4. Check for:
    - Missing values per column.
    - Duplicated rows (if any).

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 1: Descriptive Statistics and Visualisation I

Descriptive statistics:

- For numeric variables: calculate mean, standard deviation, minimum, maximum, and quartiles (e.g. with `df.describe()`).
- For categorical variables: show frequency tables or bar charts (e.g. distribution of seasons or weather types).
- Check for missing values in each column and comment on how you will handle them.

Visualisation:

- Plot the time series of total bike demand (`count`) over the full period.
- Plot distributions of key numeric variables (e.g. histograms of `temp`, `humidity`, `windspeed`).
- Plot aggregated demand by season, day of week or hour of day (e.g. bar charts or line plots).

Task 1: Descriptive Statistics and Visualisation II

Write a short text summary (3–5 sentences) describing main patterns you observe (seasonality, daily patterns, influence of weather, etc.).

Task 2: Predict Demand from Conditions

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 2: Supervised Regression Setup I I

Goal: Build a supervised regression model that predicts bike demand count from given conditions (features such as weather, temperature, time of day).

Input and target:

- Target variable: `count` (total number of bikes rented in that period).
- Example input features:
  - Calendar: season, month, day of week, hour of day, holiday, working day.
  - Weather: weather situation code, temperature, humidity, windspeed.
- Do **not** use columns that directly leak the target (e.g. `casual`, `registered` if present).

University of Applied Sciences
HOCHSCHULE
EMDEN·LEER

Task 2: Supervised Regression Setup II I

Preprocessing:

- Encode categorical variables (e.g. one-hot encoding for season, weather, weekday).
- Split the data into training, validation and test sets (for example 70 % / 10 % / 20 %) and scale numeric features using statistics from the training set only.

Task 2: Train Model and Implement User Prediction I

Model training:

- Choose at least one regression model using a machine learning library such as scikit-learn, TensorFlow/Keras or PyTorch.
- Examples: Linear Regression, Random Forest, Gradient Boosting, XGBoost, Multi-Layer Perceptron (dense neural network).
- Train the model on the training set and tune hyperparameters / monitor performance with the validation set.

Evaluation:

- Evaluate the final model on the test set using at least:
  - RMSE (Root Mean Squared Error).
  - MAE (Mean Absolute Error).
  - R^2 (coefficient of determination).
- Plot predicted vs actual demand on the test set (scatter plot and/or line plot).

Task 2: Train Model and Implement User Prediction II

User-input prediction function:

- Implement a Python function (or cell) that:
  - Accepts user inputs (e.g. season, weather, temp, humidity, windspeed, hour, weekday, is_holiday, is_workingday).
  - Applies the same preprocessing and scaling steps.
  - Uses the trained model to output a predicted `count` of bikes.
- Demonstrate several example predictions for different scenarios and briefly interpret the results.

Task 3: 30-Day Forecast

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 3: Time-Series Pipeline I

Goal: Build a time-series model that forecasts bike demand for the next 30 days (or 30 time steps), following the forecasting pipeline from the lecture.

Data preparation:

- Use the chronological order of the data to create a time-based split:
  - Training period (e.g. first 70 % of the timeline).
  - Validation period (e.g. next 15 %).
  - Test period (e.g. last 15 %).
- Engineer **time-series features**:
  - Lag features of `count` (e.g. 1, 7, 24, 168 steps back).
  - Rolling statistics (e.g. 7-step and 30-step rolling mean/standard deviation of `count`) using shifted values to avoid data leakage.
- Scale features using statistics from the training period only.

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 3: Forecasting Model and Metrics I I

Modeling options:

- Sequential models (e.g. LSTM, GRU, 1D CNN, sequence-to-one dense networks):
  - Create input windows of length 30: past 30 time steps as input, next step as target.
  - Input shape: $(\text{batch}, 30, \text{num\_features})$.
- Tabular models (e.g. Random Forest, Gradient Boosting, XGBoost, dense neural nets):
  - Use lag and rolling features directly as a 2D feature matrix.

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 3: Forecasting Model and Metrics II I

Training and evaluation:

- Train the model on the training period and monitor hyperparameters / early stopping with the validation period.
- Evaluate performance on the test period using at least:
  - RMSE (Root Mean Squared Error).
  - MAE (Mean Absolute Error).
  - MAPE (Mean Absolute Percentage Error).
- Plot true vs predicted demand on the test period as a time series and compare to a naive baseline (e.g. last observed value or moving average forecast).

Task 3: 30-Day Forecast and Interpretation I I

Multi-step forecast (30 days):

- Starting from the end of the known data, generate predictions step by step:
  - At each step, use the most recent history (and features) as model input.
  - Append the predicted demand to the series and recompute necessary lag/rolling features.
  - Repeat until a 30-day (or 30-step) forecast horizon is reached.
- Plot the full history of demand together with the 30-day forecast on a single chart.

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Task 3: 30-Day Forecast and Interpretation II I

Interpretation and comparison:

- Discuss whether the forecast captures observed patterns (e.g. seasonality, weekday/weekend effects, weather influence).
- Compare the forecasting model to:
  - The naive baseline.
  - The user-input regression model from Task 2 (when and why each is useful).
- Summarise the strengths and limitations of your approach in a short written conclusion.

Final Assignment: Evaluation

Declaration of Authorship I

I hereby declare that I, the undersigned, am the sole author of this submission. All sources consulted have been listed; all quotations and references have been properly cited. No version of this submission (in whole or in part) has been used previously for an academic degree or other examination.

I understand that false statements in this declaration may be punishable by law.

If ChatGPT or other LLM-based tools were used, clearly specify in the notebook:

- Which tool(s) were used (e.g. ChatGPT, Copilot)
- For what purpose (e.g. syntax help, explanation, debugging)
- Which sections or cells were influenced

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Evaluation Criteria — Mark Allocation (100 points) I

Criterion	Pts
Authorship & LLM disclosure	**4**
Reproducibility	**4**
Error handling	**7**
Feature engineering	**4**
Data splitting	**11**
Data leakage prevention	**7**
Model architecture	**7**
Hyperparameters	**11**
Model evaluation	**7**
Final prediction	**2**
Comments and explanations of code	**7**
Code quality	**7**
Environment specification	**7**
Submission format (`.ipynb`)	**7**
Delighter	**8**
Total	100

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Evaluation Criteria — Explanation (I) I

Criterion	Explanation
Authorship & LLM disclosure	A declaration of authorship is included. Any use of ChatGPT or similar tools is explicitly stated and described.
Reproducibility	The notebook executes top-to-bottom on a clean environment and produces consistent results. Random seeds are fixed where appropriate.
Error handling	Potential runtime errors are anticipated and handled (e.g. using `try/except`), with informative messages for the user.
Feature engineering	Meaningful features are added to the raw data (e.g. Fourier features for seasonality) and their purpose is explained.
Data splitting	Training, validation, and test sets are constructed correctly and suit the nature of the data (time-aware where applicable).
Data leakage prevention	Scaling, encoding, and feature transformations are fitted on training data only; validation and test data are never used improperly.

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Evaluation Criteria — Explanation (II) I

Criterion	Explanation
Model architecture	A suitable TensorFlow/Keras/Other model is built with appropriate layers, activations, and input dimensions.
Hyperparameters	Key hyperparameters (learning rate, epochs, etc.) are identified and briefly motivated.
Model evaluation	Appropriate accuracy metrics (e.g. MAE, RMSE) are computed correctly and interpreted in plain language.
Final prediction	The model produces a clear final prediction in the required format (e.g. future values).
Comments and explanations of code	Markdown cells explain the workflow clearly, and code contains concise, helpful inline comments.

Evaluation Criteria — Explanation (III) I

Criterion	Explanation
Code quality	Variables are clearly named, imports are minimal and relevant, and Python conventions are followed.
Environment specification	The Python version and key library versions (e.g. NumPy, TensorFlow) are explicitly stated.
Submission format	The final submission is a valid `.ipynb` file that opens and runs without errors.
Delighter	A thoughtful enhancement beyond the minimum requirements (e.g. baseline comparison, diagnostics, visualisations).

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Thank you

for your attention