

# Holiday Exercise for the lecture Big Data Analytics

WS 2025/2026

The top submissions for this exercise sheet will be presented in calendar week 3.  
Please make sure to implement the `get_group_number()` and `get_group_names()` methods  
in your submitted files. The deadline is January 8th.

The two PyTorch Crash Course sessions prepared you for this holiday task. You may reference the notebooks from these sessions, but make sure to use the code files provided in the `.zip` file for this task.

## 1 24-hour Day-Ahead Prediction

Unpack the `.zip` file and create the `mamba` environment using `mamba env create -f environment.yml` on the command line. Move into the `src` folder and run the script using `python train.py`. This will train the `SimpleEnergyNet` defined in `src/submission.py` using the hyperparameter values returned by the function `get_train_config()`.

Your task is to implement all functions matching their description in `src/submission.py` without changing the signatures or result types. Your goal is to train the best regression model predicting the `Spot` feature for the next 24 hours following the `forecast_history` context window. The best model is determined by the validation loss and qualitative assessments of the predictions (see below).

- You are free to design whatever network you like (CNN, LSTM, Transformer, ...)
- You are free to modify any hyperparameters returned by `get_train_config()`
  - You may not use target variables other than `[Spot, wind_mean, solar_mean]`<sup>1</sup>
  - You may only use  $1 \leq \text{stride} \leq 24$  (hours)
  - You may only use  $192 \leq \text{forecast_history} \leq \text{forecast_horizon}$
  - Does a different activation function improve the performance?
- Identify the best subset of features to use for your model. They must include the weather features `[wind_mean, solar_mean]` and at least one time feature (see `get_time_features()`).
- You are free to modify the training loop, but note that the evaluation script will not have any of your customizations<sup>2</sup>.
  - One such modification we discussed in the lectures would be learning rate scheduling (applied to the optimizer): You can modify or remove it from your version of `train.py`.
  - Data augmentation would be another, but can be implemented by returning an appropriate function in `get_my_batch_tsfm()`.

After you have trained your model, load the best weights and plot some predictions for various inputs. Does your model recognize the difference between a weekday and a weekend (or a public holiday)? Does it accurately predict absolute negative prices?

*Hint: You can learn how to plot your model's predictions from the `holiday_valid_spot.ipynb` notebook. You may have to add code to unnormalize the predictions to assess if the model predicts negative prices and to get the true dates on the x-axis.*

<sup>1</sup>That does not limit your use of time features - you can add them for the historic time steps, for future ones, or both.

<sup>2</sup>Which should not be a problem, since we are only evaluating your model, not training it. You can implement any kind of pre-training if it helps your final model.

## 2 48-hour Day-Ahead Prediction

Once you have familiarized yourself with the problem, train a neural network to predict the `Spot` feature for the next `forecast_horizon=48` hours following the `forecast_history` context window.

**This will be the only version you hand in.**

After you have trained your model, load the best weights and plot some predictions for various inputs. Does your model recognize the difference between a weekday and a weekend (or a public holiday)? Does it accurately predict absolute negative prices?

Hand in your fully implemented `submission.py` file and the `.pth` file with your best 48-hour model's weights.

## 3 Validate Your Implementation Using Our Notebook

All submissions will be evaluated by an automated script. This script uses the baseline implementation and the `mamba` environment we provided. For each group, the script will load the functions implemented in the `submission.py` file, assuming they have the described functionality, signature and return type declared in the version we provided. The evaluation script will not be able to use any additional dependencies. It will then attempt to load the model weights stored in the separate `.pth` file. It will assume the name is consistent with the output of `get_my_model_name()`.

If the evaluation script fails using your `submission.py` and `get_my_model_name().pth` files on the secret test set, we will not be able to consider your submission. You can validate your implementation with the `holiday_valid_spot.ipynb` notebook. It should run as is and not require any changes from your side.

All successfully evaluated models that exceed a realistic performance threshold for the 48-hour prediction on the test set will pass this exercise. The holiday exercise can make up for one of the three normal exercise blocks. The best-performing groups will present their model in the session in calendar week 3.