

Assignment 6: Predicting Air Quality with Machine Learning

Solve the following exercises and upload your solutions to Moodle by the due date.

Important Information!

Please try to *exactly match the outputs* provided in the examples.

Use the *exact filenames* specified for each exercise (the default suggestions from the heading). Your main code (example prints, etc.) must be guarded by `if __name__ == '__main__':`. Unless explicitly stated otherwise, you can assume correct user input and correct arguments.

You may use **only standard libraries**, plus modules covered in the lecture (Programming in Python 1+2).

In this assignment you should build a full, end-to-end machine learning pipeline consisting of the following steps:

1. Download and preprocess a real dataset.
2. Conduct exploratory data analysis, including visualisation.
3. Write auxiliary functions that compartmentalise training.
4. Train a model with PyTorch on the dataset.
5. Perform hyperparameter search for the task.
6. Present the results via an interactive Shiny app.

The exercises will lead you through these steps.

Exercise 1 – Submission: a6_ex1.py

30 Points

As the first step, we need to download and preprocess the dataset (<https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data>).

For this, write a function `preprocess_data(zip_path: str, station: str) -> pd.DataFrame` that does the following:

- It receives the path to the downloaded zip file and a desired station to model (e.g., Aotizhongxin).
- It handles missing values in an appropriate way and extracts useful features.
- It saves the cleaned dataset as `air_quality_cleaned.csv`.
- It returns the cleaned data as a DataFrame.

Exercise 2 – Submission: a6_ex2.py**30 Points**

As the next step, we want to plot the data to uncover interesting trends.

Write the following functions:

- `plot_pm25_trend(df: pd.DataFrame)` which plots the daily average PM_{2.5} and saves it as `eda_pm25_trend.pdf` (Figure 1).
- `plot_correlation(df: pd.DataFrame)` which plots a heatmap of the correlation between features and saves it as `eda_correlation_heatmap.pdf` (Figure 2).
- `plot_histogram_pm25(df: pd.DataFrame)` which plots the distribution of average PM_{2.5} per day and saves it as `eda_pm25_histogram.pdf` (Figure 3).

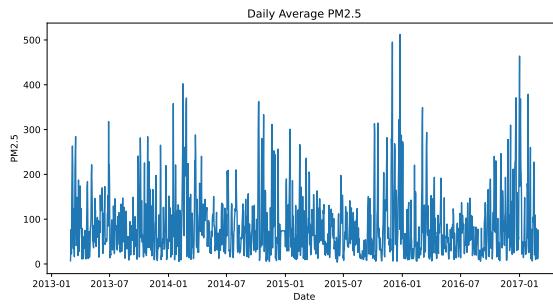


Figure 1: Daily average PM_{2.5} trend

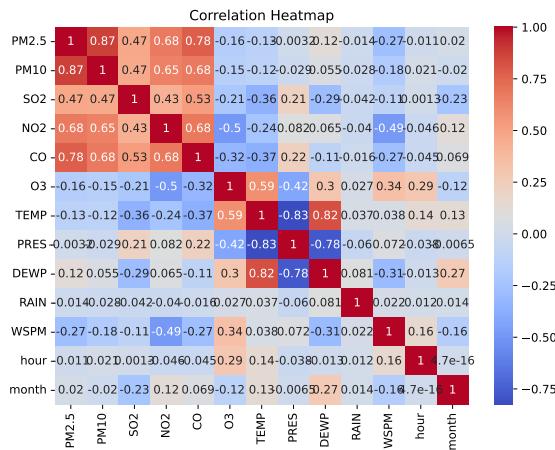
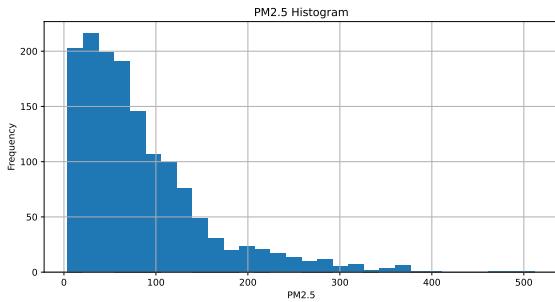


Figure 2: Correlation heatmap

Figure 3: Histogram of average PM_{2.5} per day**Exercise 3 – Submission:** a6_ex3.py**20 Points**

In this exercise you should prepare the data to be used for training your model. Write a function `get_data_loaders(df: pd.DataFrame, batch_size: int = 32) -> tuple[DataLoader, DataLoader, DataLoader]` that takes in the processed data from exercise 1 and returns DataLoaders of the split data.

- The data should be split 80/10/10 into training, validation and test sets.
- As usual, only training data should be shuffled.
- The data should be normalised, and the scaler should be saved for later use.

Exercise 4 – Submission: a6_ex4.py**20 Points**

In this exercise you should create a simple regression model for the given task. You have complete freedom to build a model class called `PM_Model` as you like.

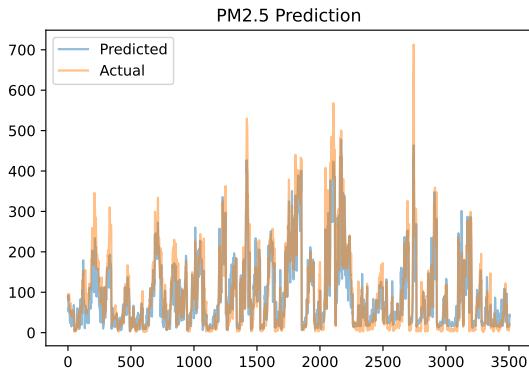
Exercise 5 – Submission: a6_ex5.py**30 Points**

In this exercise you should write a training loop for the model and the dataset. You should create a function `train_model(model: nn.Module, train_loader: DataLoader, val_loader: DataLoader, test_loader: DataLoader)` that trains the model on the dataset to predict PM_{2.5} from other features and reports performance throughout training:

- Every 50 epochs, print the current training and validation loss.
- You can freely choose training hyperparameters. Experiment with different options to optimise the loss on the validation set!
- After training, save the parameters of the model to `model.pt`.
- After training, plot the true and predicted PM_{2.5} on the test set and save the plot to `model_prediction.pdf`.

Example output (this completely depends on your model):

```
Epoch 0, Loss: 13766.7236, Validation Loss: 5325.7920
Epoch 50, Loss: 12944.0361, Validation Loss: 4987.3208
Epoch 100, Loss: 9163.0605, Validation Loss: 3811.4578
Epoch 150, Loss: 3197.1721, Validation Loss: 1841.0022
```

**Exercise 6 – Submission: a6_ex6.txt****30 Points**

In this exercise, you should perform a hyperparameter search for your model to find the best configuration for the given task.

- Identify reasonable parameter ranges and architectures.
- Train your model with at least 4 different parameter/architecture configurations.
- Report the training and evaluation performance for all 4 configurations in `a6_ex6.txt`.

Exercise 7 – Submission: app.py**40 Points**

In this exercise, you should build an interactive web app with Shiny. The web app should have the following functionality:

- Upload the cleaned CSV data (`air_quality_cleaned.csv`).
- After upload, a plot of the pollutants should appear with PM_{2.5} pre-selected.
- After upload, the sidebar should contain a dropdown to select pollutant(s) to display. A slider should adjust the rolling average window range in days.
- A checkmark should allow the user to also plot the PM_{2.5} prediction of a model. After clicking the checkmark, the user should be able to upload a scaler and the model parameters (you can import the model class in the code, you do not need to dynamically get it).

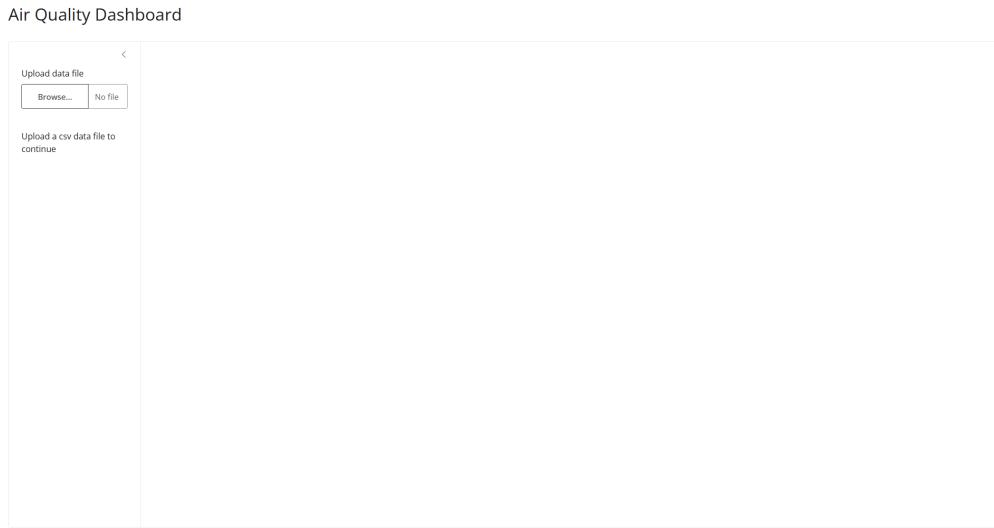


Figure 4: Initial state with no upload yet.

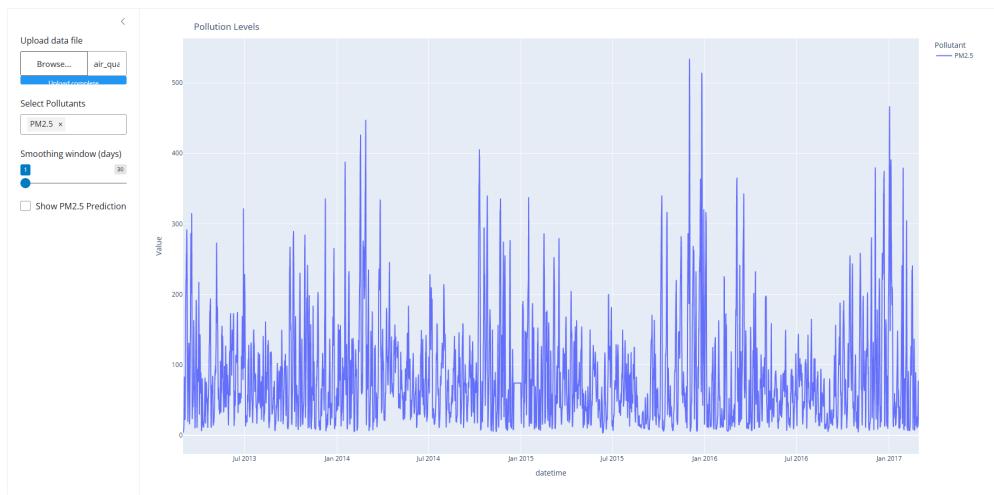


Figure 5: After uploading, the plotted pollutants, a smoothing slider, a selection menu of pollutants to plot, and a checkmark for model predictions should appear.

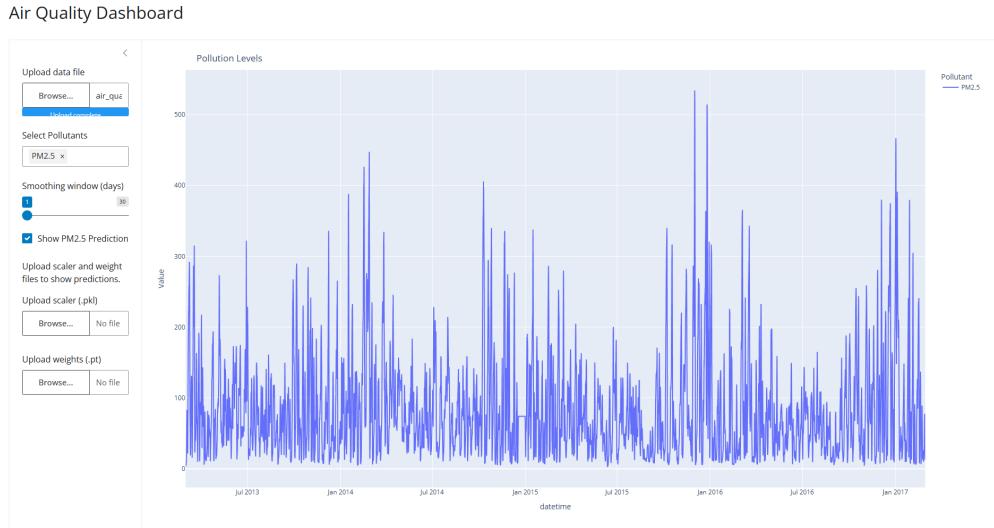


Figure 6: After clicking the prediction button, the user should be able to upload the scaler and model weights.

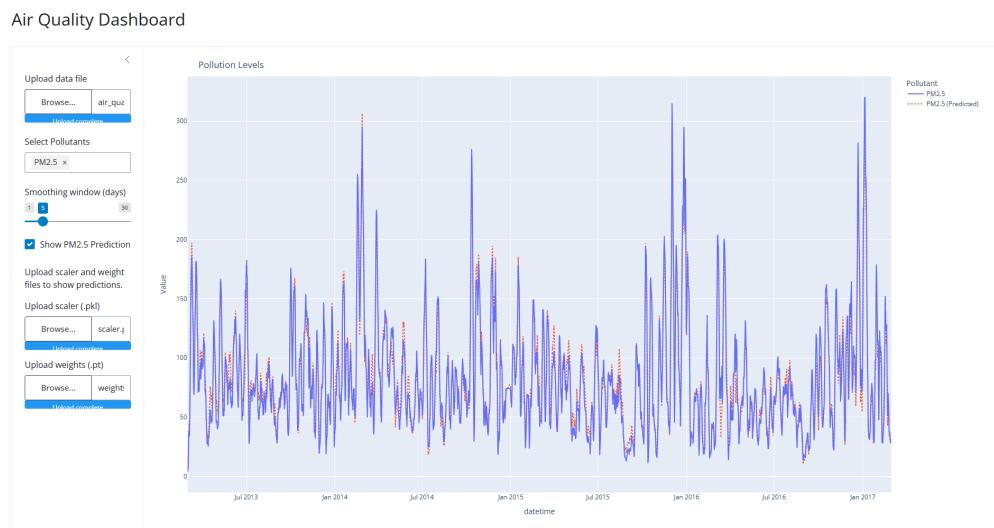


Figure 7: After uploading the scaler and weights, the predictions should appear in the main plot.

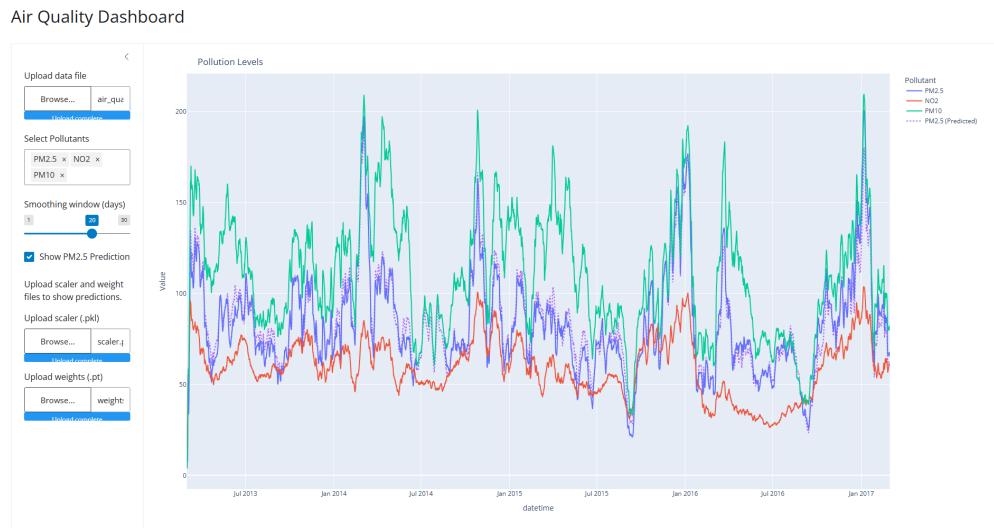


Figure 8: Plot after changing the smoothing window and adding more pollutants.