Assignment 2 | Big Data Management (Technical)

Link to Github repository: https://github.itu.dk/Big-Data-Management-2025/Assignment2tf.git

In the following report, I describe my design approaches and results. Thereby I used the 6 steps from the data analytics lifecycle:

1)Discovery and 2) Data Preparation

In the first part of the project, I explored and prepared the datasets that will later be used for model training. The three datasets provided are: historical power production (df_power), historical weather forecasts (df_weather), and future weather forecasts (df_future). The goal is to combine them into one clean and consistent dataset that contains both the weather features and the corresponding power output.

As a first step, I performed basic data quality checks. I inspected all columns for missing values and blank entries, but none were found in the raw datasets. After merging the datasets later, three missing values appeared in the Total column (power production). Since all of them occurred on the very first day and did not affect the continuity of the main data period, I removed these rows.A central challenge in this assignment is the mismatch in temporal resolution, which is also mentioned in the system requirements. The power data is recorded every minute, while the weather data is only available every three hours. To make both datasets compatible, I aggregated the power data to a 3-hour frequency using the mean. This approach preserves the information contained in the high-frequency measurements while aligning them with the weather timestamps. Afterwards, I merged the weather and aggregated power data using an inner join so that only matching timestamps remained.

In the next step, I prepared the features for modeling. The assignment mentions three possible methods for encoding wind direction:(1) One-Hot Encoding of the cardinal direction strings,

(2) Mapping each direction to an angle (degrees or radians), (3) Converting wind speed and direction into u/v vector components.

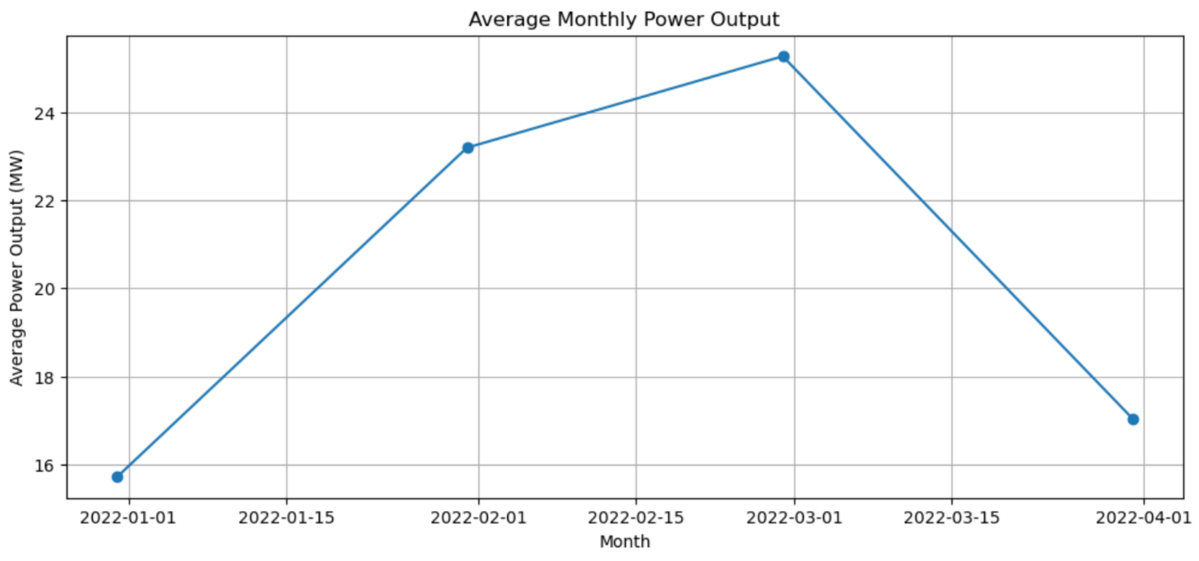
I chose to use the u/v vector representation.Although this approach introduces two additional numerical features and slightly increases model complexity,the main advantage is that it reflects the physical nature of wind as a 2D vector and avoids the artificial discontinuity at 0°/360°. Therefore, the benefits clearly outweigh the drawbacks for this forecasting task.

Before modeling, I created a chronologically ordered train/validation/test split,since shuffling would cause data leakage in time-dependent data. I used an 80/10/10 structure (implemented as 64% train, 16% validation, 20% test), which preserves the temporal order. Finally, I applied standard scaling to the numerical features as part of a scikit-learn pipeline, ensuring that scaling happens consistently across training, validation, and test data.The prepared dataset (df_merged) is then saved and used in the model-building phase.

3)Model Planning

In the model planning phase, I focused on identifying which features are relevant for the predictive task, selecting appropriate models to experiment with, defining a suitable hyperparameter search strategy, and determining how the models will later be evaluated. According to the assignment description, the key input variables are wind speed and wind direction, which I represent using the u/v wind vector components generated earlier.

To understand the relationship between the features and the target variable, I visualised wind speed, u, and v against power output. Wind speed shows a clear sigmoid-shaped relationship with power generation, confirming the non-linear behaviour described in the assignment. In contrast, the individual u and v components do not show strong relationships on their own, although the combined 2D scatter indicates that high wind power generally occurs when the overall wind vector magnitude is large. I also checked whether the temporal dimension contains useful predictive patterns by plotting the monthly average power output. No seasonal structure or recurrent temporal trend was visible, meaning that classical time-series models (e.g., ARIMA, RNNs) are not appropriate. Instead, the problem is treated as a supervised regression task.



→no correlation visible

Based on these findings, I decided to experiment with two tree-based ensemble models: Random Forest and Gradient Boosting. Linear models are not suitable because they cannot capture the non-linear power curve. Random Forest is recommended in the assignment because it naturally handles non-linearities and interactions. Gradient Boosting is included as a second model, as it builds trees sequentially and corrects previous errors, which often leads to higher accuracy in complex regression tasks. Although both models are tree-based ensembles, Random Forest trains trees in parallel, whereas Gradient Boosting trains them one after another.

I also defined a clear hyperparameter search strategy for both models. For the Random Forest, I vary n_estimators (100, 300, 500), max_depth (None, 10, 20), min_samples_leaf (1, 3, 5), and choose squared_error as the criterion, following the documentation. For Gradient Boosting, I use a moderate range for n_estimators (300, 500) to avoid overfitting, vary the learning rate (0.1 and 0.01), max_depth (3 and 4), and min_samples_split (2 and 5). Again, I use "squared_error" as the loss function, since the goal is to minimise MSE.

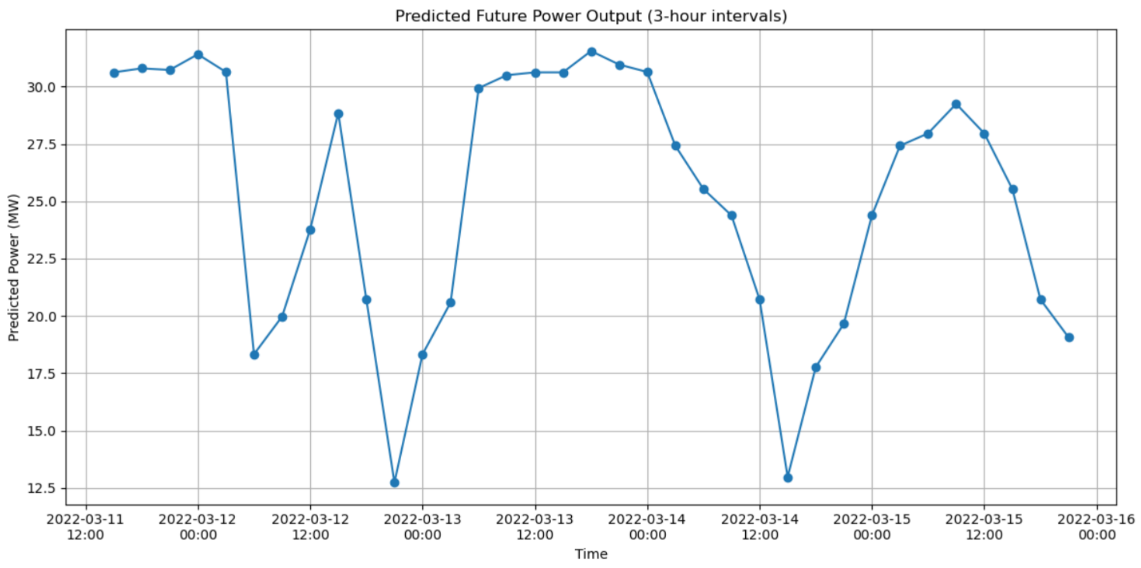
Finally, I decided that the models will be evaluated using standard regression metrics: MSE, RMSE and MAE. RMSE is especially interpretable because it is expressed in the same unit as the target variable (MW). The validation strategy respects the temporal structure of the data. The dataset is split sequentially into 80% temporary training data and 20% test data; the temporary training block is then split again into 80% training and 20% validation. This avoids data leakage and ensures that the model is always evaluated on unseen future data.

4) Model Building

In the model building phase, I implemented the decisions defined during model planning and trained the predictive models. The final train/validation/test split followed the sequential splitting strategy from the preparation phase: first, 80% of the dataset was used as a temporary training block, and the remaining 20% served as the test set. The temporary training block was then split again into 80% training and 20% validation, resulting in 64% training data, 16% validation data, and 20% test data. This approach preserves the chronological order and prevents data leakage.
The preprocessing and modeling steps were combined into a single scikit-learn Pipeline, consisting of a StandardScaler for the numerical features (Speed, u, v) and the chosen regression model. This ensures reproducible preprocessing and avoids applying transformations separately to each data split.
First, I trained a baseline Random Forest model and evaluated it using RMSE, MAE, and MSE. After establishing the baseline, I performed hyperparameter tuning with RandomizedSearchCV, which is suitable for small datasets. The search was tracked using MLflow, logging all parameters, metrics, and the final best model. The best configuration found included n_estimators = 200, max_depth = 10, and min_samples_leaf = 3. The tuned model showed a small but measurable improvement over the baseline (MAE ≈ 4.49, RMSE ≈ 6.20, MSE ≈ 38.05816877155546).
I also trained and tuned a Gradient Boosting Regressor for comparison. One time as a baseline model and one time with the same hyperparameter method as at the Random Forest model. Although Gradient Boosting often performs well on non-linear tasks, it achieved lower performance than the Random Forest in this case. Therefore, the best overall model was RandomForest_Tuned. I used this model to predict the power for the dates of the dataset future.csv and saved the results as a CSV file.

5) Communicating results

The power predictions generated by the model are provided as a CSV file named future_predictions.csv (uploaded under data in my github repository). In addition to the raw forecast data, a visualization of the predicted power output across the forecast horizon is included to support intuitive interpretation of the results.



To evaluate the forecast in context, the mean predicted 3-hour power value was compared with historical values from the dataset df_merged.csv:

- The mean predicted power over all 3-hour intervals is approximately 25.23 units.
- The mean actual power in the historical dataset is approximately 21.45 units.

This indicates that the forecasted power values are, on average, higher than the previously observed values.
Additionally, the total predicted power summed over all forecasted intervals amounts to 883.04 units, providing a measure of the cumulative expected output across the prediction period.