Github Link: https://github.itu.dk/BDM-Autumn-2023/aoli_a2

MLproject Directory: aoli_a2/SVR

VM Public IP: 4.231.171.53

1. Introduction

In this project, I created a pipeline which is used for 1) fetching the wind forecasting and power data from the last 90 days, 2) preprocessing both datasets and combining them into one dataset, 3) training and evaluating different models on this dataset. When the data reach the model, there are two features, “speed” and “direction”, and one output, “total”.

2. Methods to join datasets

In aligning the data from two sources, I first left joined the wind forecasting dataset to the power dataset. Since the power dataset has much more time points, there are a lot of missing values in “speed” and “direction”. The missing values of “speed” was interpolated in linear since it is a numerical variable. The missing values of “direction” was filled forward by existing values. For example, if the direction is NNE at 15:00 and it is NW at 18:00, then the directions between 15 and 18 are filled with NNE. After doing this, the rows that are still with missing values were removed. However, some issues arise from this way of handling missing values. Since the power data is generated every minute but the wind forecasting is only generated every three hours, 179 out of 180 wind forecasting are simulated instead of being the real data, which may cause much bias in the analysis. To address this, I only kept the data points every 30 minutes, trying to reduce the bias created by pure simulated data but at the same time allowing more data for analysis. After joining the datasets, 4127 rows of data are ready for further analysis.

	time	Total	Direction	Speed
120	2023-08-21 12:00:00+00:00	20.540285	SSE	7.152640
150	2023-08-21 12:30:00+00:00	23.283391	SSE	7.301653
180	2023-08-21 13:00:00+00:00	21.050175	SSE	7.450667
210	2023-08-21 13:30:00+00:00	18.865174	SSE	7.599680
240	2023-08-21 14:00:00+00:00	20.834621	SSE	7.748693
...	...	...	...	...
123816	2023-11-19 07:30:00+00:00	2.980016	SSE	3.576320
123846	2023-11-19 08:00:00+00:00	3.167016	SSE	3.427307
123876	2023-11-19 08:30:00+00:00	3.014016	SSE	3.278293
123906	2023-11-19 09:00:00+00:00	3.006016	SE	3.129280
123936	2023-11-19 09:30:00+00:00	2.922016	SE	3.129280

4127 rows × 4 columns

Figure 1 Overview of joined dataset

3. Choices of models and evaluation metrics

To decide which models can be used for analysis, I first performed the exploratory data analysis to see the correlations between two features and the output (figure 2,3,4). From figure 2, we can assume that the relationship between wind speed and power generation is probably not linear. From figure 3, we can see different wind direction also affects the power generation. However, comparing the x labels of figure 3 and figure 4, which are arranged by the averages of y labels, we can identify that the orders of x labels are similar. This implies that the variations of power generations caused by wind direction may be partly explained by that wind speeds are different in different directions.

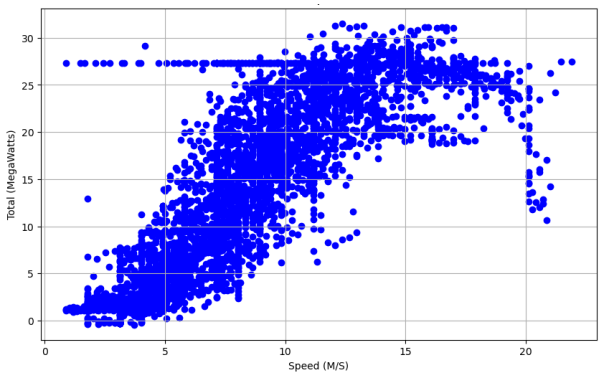

Figure 2 Relationship between wind speed and power generation

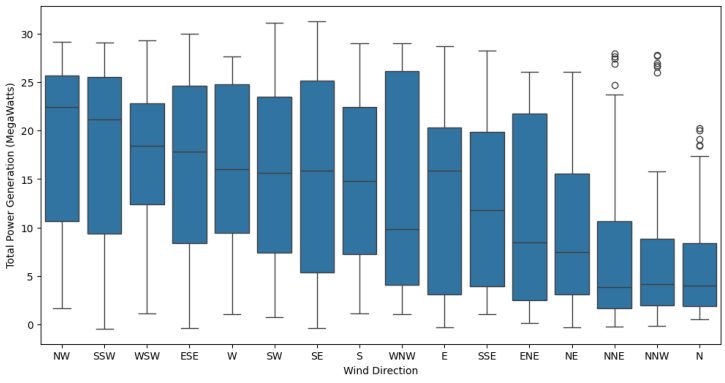

Figure 3 Distribution of power generation by wind direction

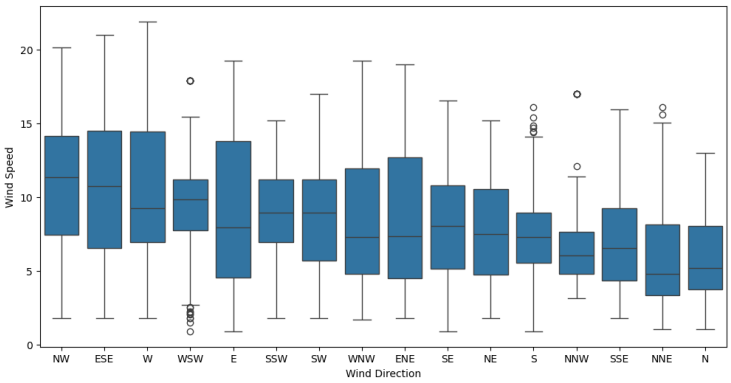

Figure 4 Distribution of wind speed by wind direction

Taking into consideration of variable types and relationships of different variables, I chose four kinds of models generally suitable for supervised regression tasks, and trying to cover models with different statistical foundations. Therefore, the chosen models are polynomial regression, support vector regression (SVR), random forest regression, and sequential neural networks. For different models, I tested 1-2 hyperparameters that are usually perceived important for each model. Specifically, I tested different degrees for polynomial regression, different C values for SVR, different numbers of estimators and maximum features for random forest regression, and different optimizers for sequential neural networks.

To compare the performances of different models, I chose two metrics, mean squared error (MSE) and explained variance (EV). MSE is chosen because many regression models are optimized based on minimizing the MSE loss function, making it a direct measure of model performance. However, it is not normalized and lack of interpretability in some cases. Therefore, EV was chosen to complement MSE, since it is normalized and provides an intuitive sense of how much of the output variation the model can explain.

4. More questions regarding the design of the system
    a. How to decide if the wind direction is a useful feature for the model

There are several ways to decide if wind direction is a useful feature. First, we can see this from the exploratory data analysis above, or calculating the correlations between wind direction and power generation, and decide if wind directions explain some variations of power generations. A more quantitative approach can be evaluating the feature importance after training some models such as random forest. We can also train two models with different feature spaces, one with both features of speed and direction, and another only with speed, then comparing the EV of the two models.

    b. Intervals for fetching data

To evaluate of 90 days is a good interval for analysis, we can evaluate the model performances with different intervals. If expanding or narrowing the interval improves performance metrics significantly, it may suggest a better fitting interval. Besides, domain knowledge is crucial in this case to decide the length of the historical timeframe that are relevant to future predictions.

Although a larger interval may improve the performance considering the wind forecasting is only generated every 3 hours, there are some trade-offs to consider. First, more data require

more computational resources and storage space. Besides, data from a long time ago can be outdated and leading to training data with low qualities. Therefore, the accuracy may not necessarily increase by including more data especially when the data contain noise or irrelevant information. Besides, the wind forecasting can be quite seasonal and varies a lot between different timeframes. In this situation, a short timeframe with recent data can be more predictive.

c. Different parameters within the same model

I tried different parameters for different models as stated above. Sometimes the parameters change the performances significantly within the same model, for example, in the polynomial regression, the degree 2 performs much better than degree 3 and 4. However, in other models, changing parameters did not improve or decrease the performances significantly. Overall, the impact of changing parameters is less than changing models.

5. Improvements

However, the pipeline can still be improved in several ways. First, I created two “for loops” to streamline the process of testing different models and hyperparameters. The logic is that it first runs with the first model and tests all its hyperparameters, and then proceeds to the next model and its hyperparameters. However, in most cases, the model itself decides more if the model is good for the dataset, which means we do not need to test all hyperparameters for models that do not perform well for this task. It would be better to build a pipeline that tests all models first, and then only tests hyperparameters for the model that performs the best.

Second, the pipeline now does not fit every model well. For example, it is a good practice to include scaler of numerical variables in the Sklearn pipeline for polynomial regression and SVR, but it may not be good to include the scaler for random forest regression. Besides, neural networks have more parameters than other models in the fit/transform functions, such as learning rates and epochs, which are hard to be included if we also want this pipeline fitting other models.

Third, in the evaluation metrics, only the means of MSE and EV are calculated after cross-validation for now, but the standard deviations should also be taken into consideration to evaluate the stability of the performances across different datasets.

6. Advantages of packaging models in the MLFlow and comparison with other reproducibility options

MLflow provides a standard format for packaging ML models, which includes both the environment and the dependencies. This encapsulation ensures that the model can be reproduced and served in different environments. In addition, MLflow allows users to track and log different runs, including code versions, datasets, parameters, and metrics. This function, together with the UI provided by MLflow, allows user to compare and analyze different experiments and runs.

Comparing to other reproducibility options, such as the Docker containers, while Docker containers can also encapsulate environments, it does not track experiments and runs. Overall, Docker can be better applied in terms of environment compatibility with broader software deployment practices, while MLflow is more well-suited for ML related tasks.

7. Reasons to deploy the model

Deploying a model to make it available to others can have several benefits. First, deployment allows for easy access and utilization of the model's capabilities, which is particularly important in business where different teams can benefit from the deployment. Second, deployment helps the model to be tested in real-world practices, where we can get valuable feedback on its performances. In addition, deployment can also foster collaboration and improve operational efficiency.