[PRD] Prediction

Note: this report summarizes the output that can be found on the Jupyter Notebook (`Phase2/src/PRD_Prediction.ipynb`) which also includes the source code in-line (html can be consulted). This report has been created with the sole purpose of meeting the requirement of 5 pages for the visual report.

Goal

The goal is to predict the variable 'WTURPower' from each asset based on the housekeeping telemetry reported by each asset on the previous day (sampled every 10 minutes) and the weather forecast (wind speed and temperature, sampled every 60 minutes for one week). The output should be sampled at 10 minutes.

1) Understanding of the input data

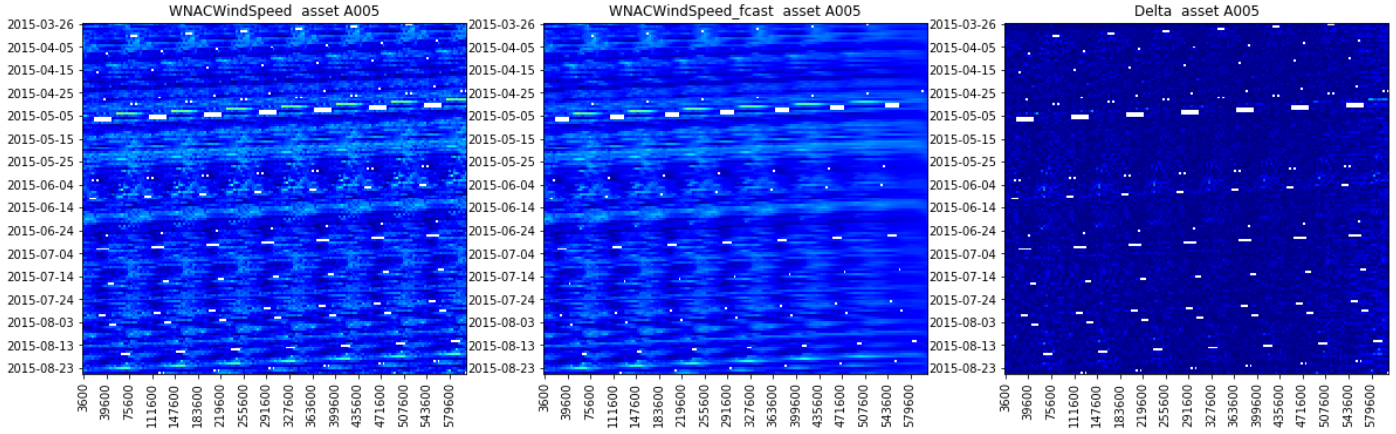
The HK (housekeeping, raw sensor) data allows us to understand the status of a given asset. If for example, a turbine is faulty, it would be reported in the HK data and the prediction should consider this.
The weather data is highly correlated to the output power and its information contributes highly to the prediction of the power. Note that this data is reported every hour, while the prediction needs to be done for every 10 minutes.

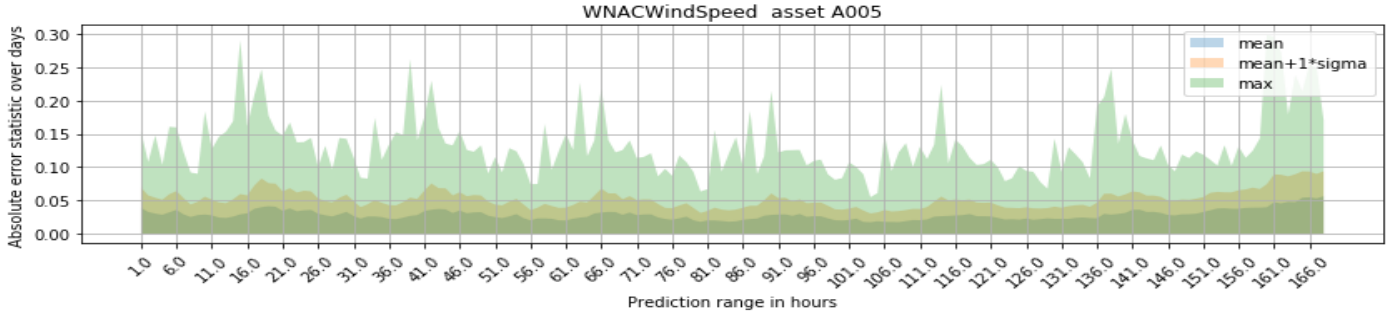
1.1) Analysis of forecasted variables (wind, temperature)

In order to understand the goodness of the prediction it is important to understand the accuracy of the input data. A prediction algorithm will not outperform the quality of the input data. This analysis shows how well the prediction variables and compare to their real values from the historical dataset.

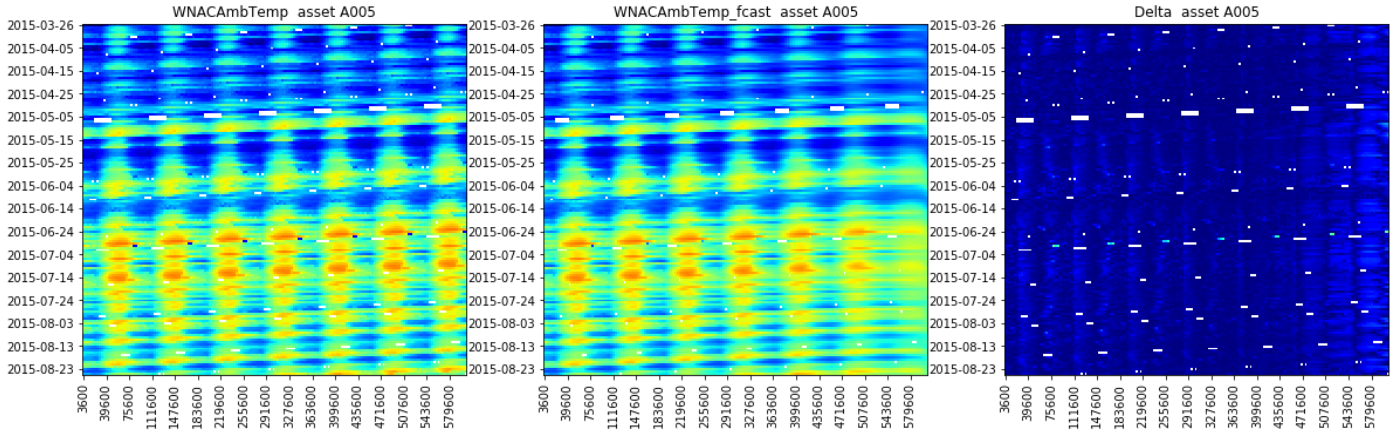
$$\Delta x_{wind}(\tau, d) = |\hat{x_{wind}}(\tau, d) - x_{wind}(d + \tau)|$$
$$\Delta x_{temp}(\tau, d) = |\hat{x_{temp}}(\tau, d) - x_{temp}(d + \tau)|$$

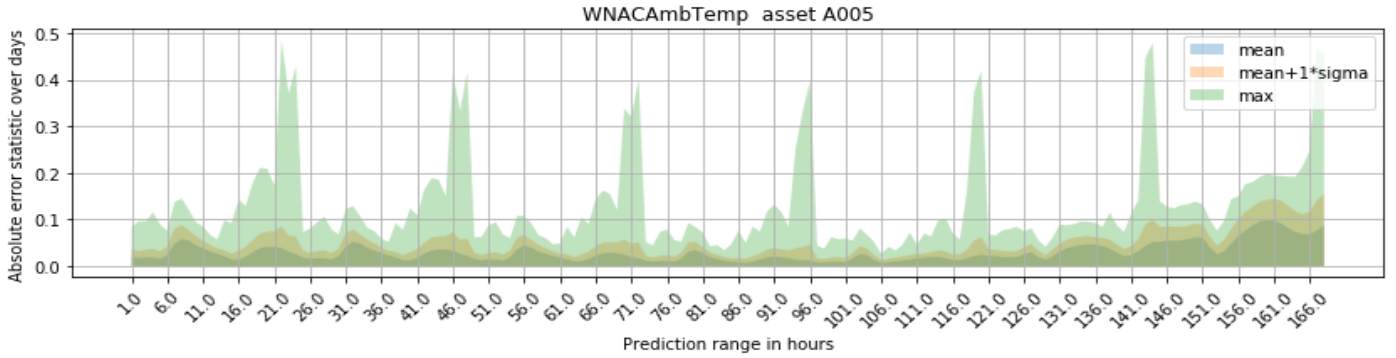
Being tau the prediction range and d the day of the prediction.
The correlation analysis on Phase 1 showed that the variable to predict (WTURPower) is highly correlated and dependable on the wind speed and temperature. The following plots show for asset A005, for every day (y-axis) and range (x-axis) the value of the variable and the difference (input is normalized). The timeline is, for each range (x-axis) the mean, mean+1sigma, and max of the estimation error over the days. MSE and R2 scores are also provided.





`MSE error = 0.0014627676256664844, R2 score = 0.6875463423249475`





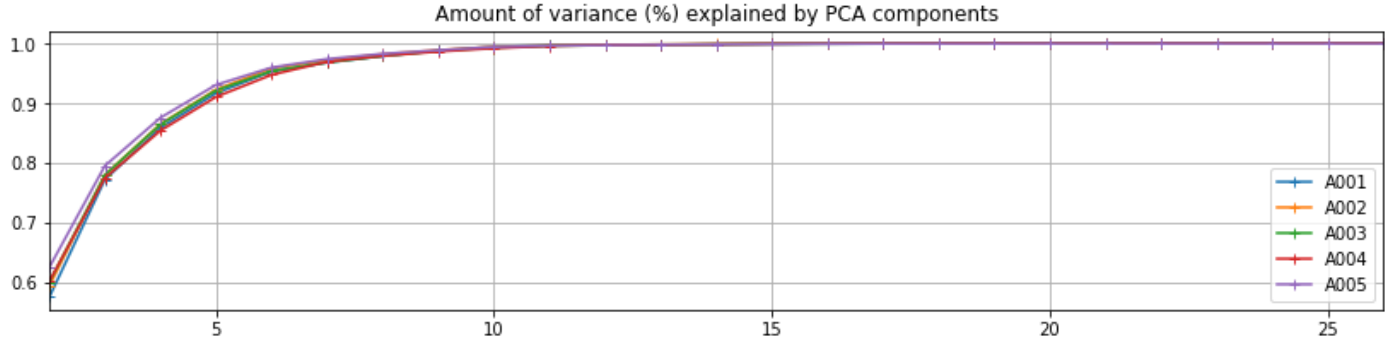
`MSE error = 0.0018523776922195995, R2 score = 0.9344818841087101`

Note the cyclical (daily) effects, which is more visible on the maximum error of the temperature (i.e. prediction gets worse for end of the day).

2) Dimensionality reduction

As shown in *Phase1*, the input data is highly correlated. In order to pick which variables to keep and which to let go, it is better to use an algorithm to do this for us, that would retain the most information. PCA is used. The goal is:

- To ensure that only the required (maximum) information (entropy) is flown into the predictor.
- To ease the computing load.



Based on this input and analysis, keeping **12 components allows us to retain 99.7% (3-sigma) of the variance.**

3) Learning

3.1) Approach

A two step approach is proposed:

- To predict the output power every hour for the next seven days.
- To interpolate the predicted points (@60min) to derive a higher frequency (@10min).

Note that a good algorithm design would also parametrize the sampling frequencies in a way that either the input weather forecast data is interpolated or the output is interpolated. For memory management reasons, and because in any case we are incurring into self-interpolation, the latter is chosen.
The idea is to fit a function (e.g. linear coefficients, SVR, MLP weights...) that from the HK data of the previous day and the weather forecasts from today to the next seven days, can compute the output power at 1h sampling, applying an interpolation afterwards.

$$\hat{x_{pow}}(d,\tau) = h(X_{raw}(d-1), \hat{x_{wind}}(d,\tau), \hat{x_{temp}}(d,\tau))$$

3.2) Optimization and prediction

The approach is based on `GridSearchCV` component from `scikit-learn` that allow to split the input dataset into a train and cross-validation dataset and optimize the output performances: given an input pipeline the object would determine which combination of the input parameter space provides the optimum results. The pipeline performs the following steps:

- Reduce the dimensions to 12 (raw) +2 (weather) components.
- Introduce polynomial features (degree 2).
- Scale the input features, to bring all of them to the (-1, 1) range.
- Perform a prediction based on one of the input regressors and configuration parameters.

3.3) Cross-validation and testing

The learned coefficients need to be cross-validated and tested. The cross-validation (i.e. ensuring that the configuration parameters of the regression algorithm are the optimum) is taken care of by the `GridSearchCV` function. It will test all the possible permutations of the configuration parameters (given by the dictionary `params`).
The learned parameters are then tested against a true referenced (not used in the training set). To have a clear snapshot of the requested prediction it was chosen not to isolate random records of the input feature matrix, but to isolate full days.
This way the current algorithm first splits the dataset into a train set and a test set. **The test set contains 5 random days**, which are not used for the learning exercise, but only to test the performance.

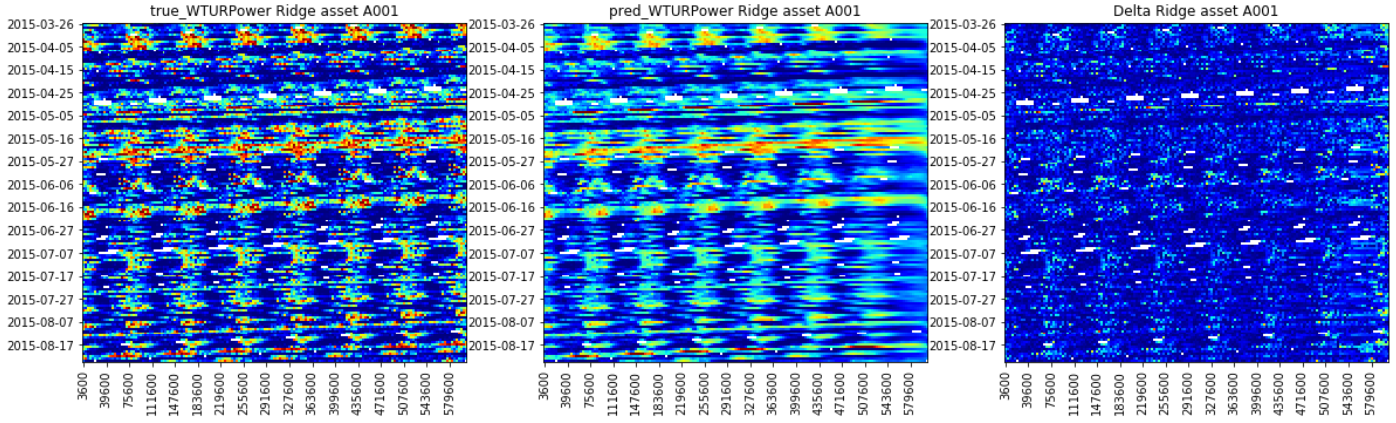
4) Results with Linear Regression

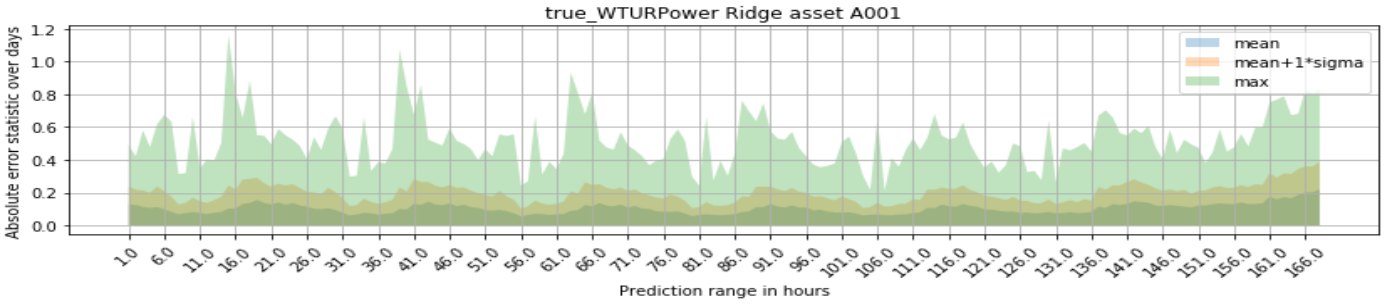
A linear regression with regularization is proposed as a first prediction algorithm. The regularization parameter would ensure a best fit based on bias-variance.

4.1) Training error

The algorithm actually chooses heavy regularization since the amount of input data is high and it would incur in over-fitting.

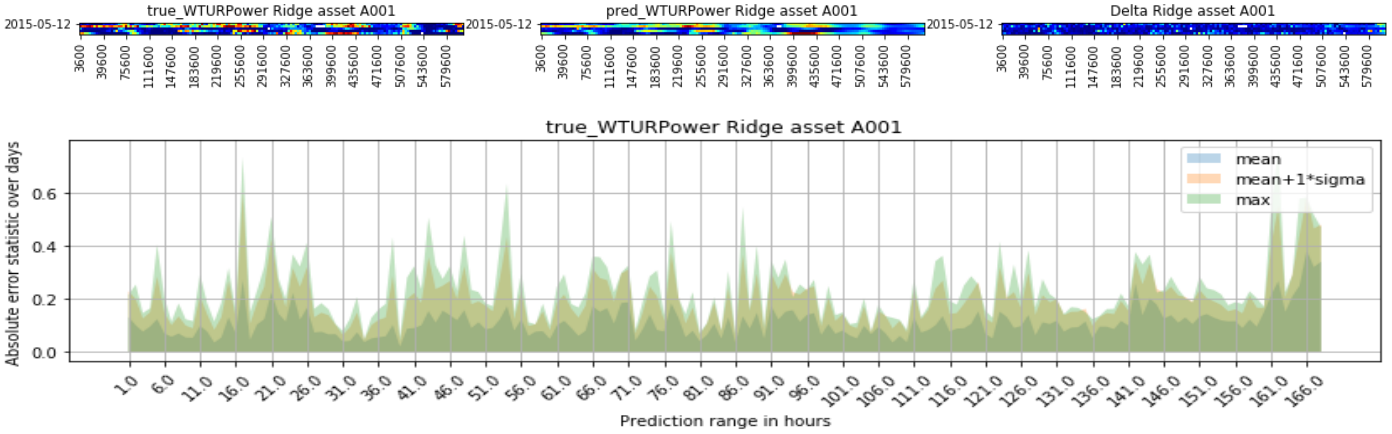
```
Best parameters: {'prediction__alpha': 100.0}
```






```
MSE error = 0.021408067339734l5; R2 score = 0.6684386203074613
```

4.2) Test error (5 random days)



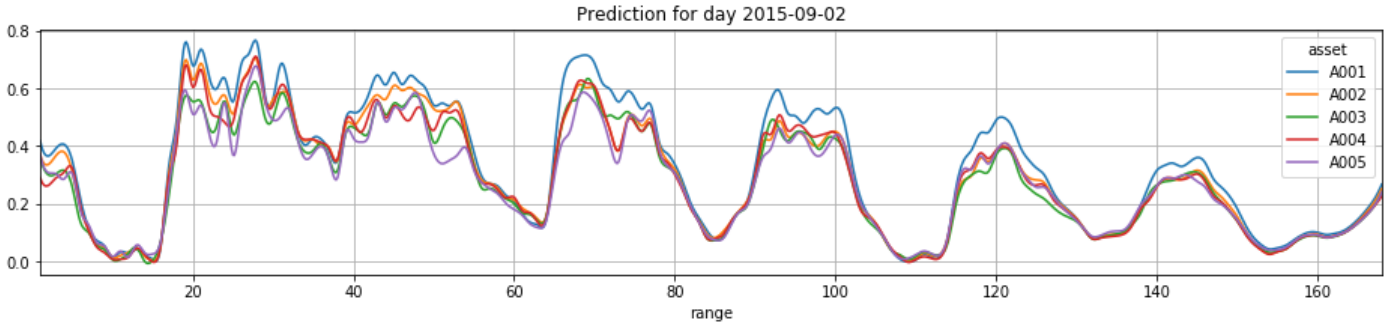

```
MSE error = 0.023966958723029 82; R2 score = 0.7244822513277698
```

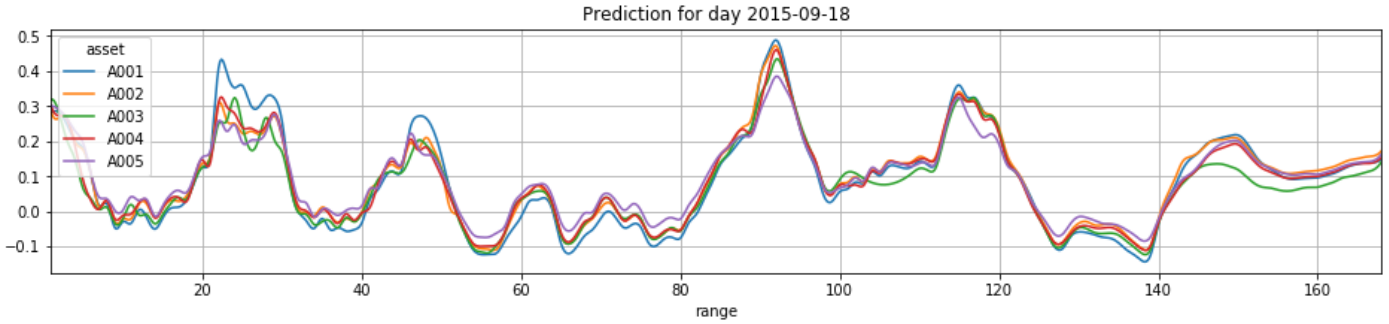
One can see that the training error and test error are similar, with an R2 score of 0.7. The MSE (mean square error) is considerably bigger than the input variables, but not the R2 score (wind has also 0.7).
The training error has a cyclical daily component, coming from the input weather data.

5) Prediction and file creation

Based on the best parameters obtained by the cross-validation exercise, every asset is trained with a predictor of that nature and the evaluation set is fed in order to retrieve the output file. It is important to notice that for the prediction of the evaluation set, all the historic data is used for training (no test days are left out).

- Cross-validation and learning will tell us which prediction algorithm configuration delivers the best results.
- To do the prediction, each asset will be trained separately with that configuration; and from this, a prediction at 1h sampling will be computed.
- An interpolation of the output values will provide an output at 10min.





6) Conclusions

- An approach to predict wind turbine output power using sensor HK data from wind-turbines and weather forecasts has been designed and proven.
- The approach is flexible: it can use any regression algorithm from scikit-learn and will optimize the performance using GridSearchCV.
- Several preprocessing steps are piped including: polynomial features, PCA and scaling of the input features.
- Some days for test (not used for training or cross-val) has been set aside to test the goodness of the prediction, providing similar results to the training set error.
- From the tested algorithms, the algorithm with better performance is a Neural Network with two hidden layers and regularization. It does not improve dramatically, however, the performance of a standard linear regression with regularization.

Limitations and future work

The lack of available personal time to work on this challenge penalized the final score achieved by the prediction. The following are points that, with more time would be investigated and researched.

- The current approach does not mix data from one asset to the other. I personally do not have insight on how much reliable is the entropy of one asset to the other, but the algorithm should allow to use the maximum volume of data. The current code allows to mix all the data from the assets (`train_test_split_day` parameter assets) but it results in a memory overflow due to permutations, with my current hardware setup (laptop).
- Given the pre-processing step of moving average (10 min) and the point above about the number of assets, one could think of using other technologies such as **Apache Spark MLib** to the machine learning.
- `scikit-learn neural-networks` aren't optimized. On the other hand, **TensorFlow** uses graphs to compute symbolic gradients instead of numerical gradients. This results in a better and faster estimation. Personal work done with TF can be found on https://github.com/drublackberry/prove-it/blob/master/TensorFlow/TensorFlowAdvanced.ipynb.
- More time would have resulted in a better research on which algorithms or parameters provide the best estimation.
- The interpolation to retrieve data at 10 minutes sampling is done a posteriori - over the estimated output power. It should be analysed whether it is better to interpolate the input data (weather forecasts) and learn over the interpolated parameters. The penalty for this technique is an increase in data, and hence, the learning process is more computing hungry.
- The interpolation used is a spline. Given the amount of points >> 3, experience shows that it provides better results than linear or other interpolation techniques. However, it should be validated and evaluated if another interpolation technique improves the accuracy of the prediction.