Predict air quality at the street level
By Plume Labs

Thomas Opsomer
M2 MVA ENS Paris Saclay

Victor Pellegrain
M2 MVA ENS Paris Saclay

Abstract

During this project, we tried to predict three pollutant (NO2, PM2_5,PM10) concentrations in the air at different monitoring stations and at different times, based on previous temporal and static data, associated to different stations.

1 Introduction

This problem therefore consists in a regression task, where the algorithms are trained on data subdivided into static data (geographic status of a given buffer around the station), and meteorological temporal parameters, which behaves as time series. An important remark is that this is not a classic prediction task related to time series, in which the goal is to estimate the future observations. Indeed, we are here asked to predict pollution measures at the same time as our available observations.
We will first detail the way we explored and modeled features, then we will introduce the algorithms we used, and finally we will analyze the results obtained and interpret them.

2 Data Exploration

As mentioned before and in the subject, we first discriminated data into two subsets : temporal ones which keep evolve in the time, and static ones. We also made a basic correlation analysis between the three pollutants, which led to these results :

Table 1: Average on all stations of correlation between each pollutant

	NO2	PM10	PM2_5
NO2	1.0	0.44	0.36
PM10	0.44	1.0	0.85
PM2_5	0.36	0.85	1.0

Thus we concluded that it was necessary to realize a different model for each pollutant, or a model for NO2 and another one for PM2_5 and PM10, as the latter two pollutants are pretty correlated, but quite different from NO2.

We then analyzed the correlation of the features with the pollutant concentrations. Figure 1 shows correlation between raw temporal features as well as with the NO2 concentration. We observe no particular correlation between any features and the pollutant concentration.

Another interesting remark is that the static features do not take many different values. Hence, when applying different algorithms, we observed that it was difficult for them to learn on these features. Combining this with the time series model of the temporal features, we decided to exploit more these ones.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

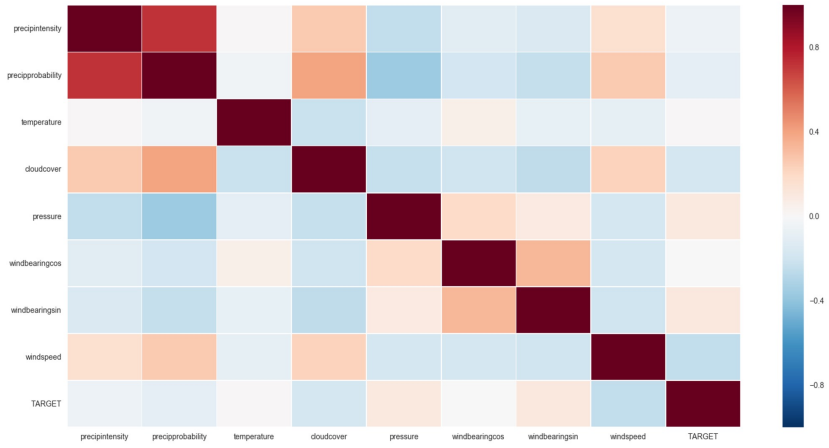


Figure 1: Average correlation for NO2 between temporal data and concentration (TARGET)

3 Adding new features and preprocessing

Given the nature of the variables provided in the dataset to predict the pollution concentration, we developed different kinds of features for each kind of data : temporal data and static data.

3.1 Generic Features

We used the `zone_id` field as an indicator for the zone of the station observation. From the `daytime` which represents time at every hour, we extracted two time features: a `hour_of_the_day` in range 0, 23 and a `day_of_the_week` in range 0, 6.

3.2 Temporal Features

Temporal data are defined by the fact that the value at time t is very correlated with the passed values. Temporal data also often show some seasonality at different frequencies. For instance temperature shows a strong seasonality every 24 hours. We constructed features that express those properties:

- **Delay**
  The first thing that comes in mind when trying to get information from the past within time series is to add previous values to the current observation. In order to guess what delays could be interesting and efficient we looked at the correlation between pollutant concentration and delay series of temporal data. For instance on figure 2 we observe that the concentration of $PM2\_5$ with delayed pressure is maximum at lag 12h and 24h.
- **Rolling mean and standard deviation**
  We used a range of rolling mean and rolling std for a range of window like [6h, 12h, 48h, 96h, 120h]
- **Trend, seasonality decomposition** In a deterministic way, time series can be seen as the sum of three components: a seasonal component, a trend component and a residual component (containing anything else in the time series).

  $$y_t = T_t + S_t + r_t$$

  This is the additive decomposition where T_t is the trend, S_t, the seasonality and r_t the residual. To estimate each of these components, we use a convolution filter to estimate the trend, then the average of this detrended series for each period computes the seasonal component of this period and the approximate residual is obtained by subtracting estimated trend and seasonality from the initial signal.
- **Continuous Wavelet Transform**
  Continuous wavelets transform defined in [2] as the convolution between a signal and a

dilated wavelet ψ:

$$Wf(u,s) = \int f(t)\frac{1}{\sqrt{s}}\psi(\frac{t-u}{s})dt$$

have been used successfully to predict pollutant concentration using temporal meteorologic data in [3]. The wavelet function is designed to achieve a balance between time domain and frequency domain. Very dilated wavelet (with large s) can detect short high frequency variation because of their fine localisation in time, and on the other hand for smaller scale their frequency resolution is much better.

We tried several mother wavelets and finally chose the $morlet$ wavelet as it gave us the richest spectrum as we can see on figure 3 for a sample of temperature and wind speed. We observe on this scalogram that the wavelet transform captures the low frequency components of the nearly sinusoidal part of the temperature series but also the high frequency variation in with larger scales.

Because wavelet transform coefficients of each scale are very correlated with each other, and that it creates many features if we want many scales, we reduce the dimension of each wavelet transform using PCA. We observe that for scales in $[1..144]$, 4 or 5 principal components are enough to represent more than 90% of the information.

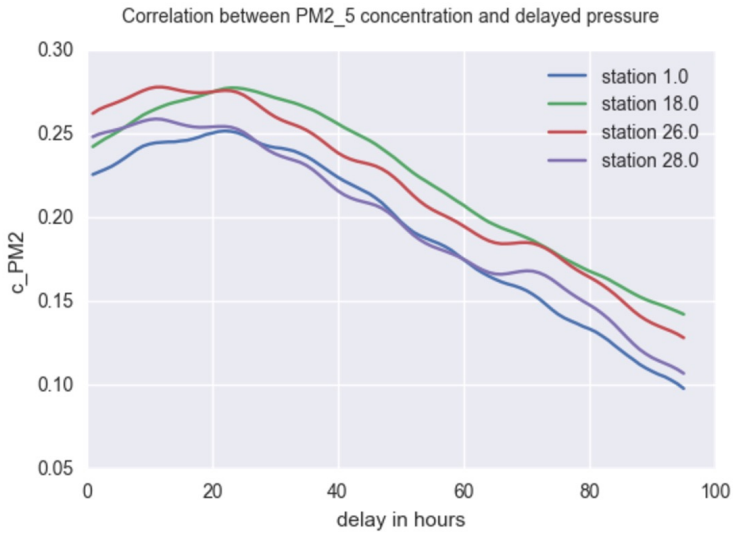


Figure 2: Correlation between PM25 and delayed pressure to estimated best delay to use in features

As we have to predict pollutant concentrations at the same time and in the same zone as some available train data, we finally decided to add the average of pollutant concentrations at the same time and in the same zone as a new feature. Unfortunately, it ended up in a large overfitting, so we decided to drop it.

3.3 Preprocessing

The first step of preprocessing was to ensure that there were not any missing or incoherent data. Thus, we replaced all "NaN" values in static data by 0, as it was mentioned in the problem statement that any missing value is due to the fact that no land use is encompassed within the buffer. We then rescaled the data between 0 and 1, but in different ways depending if the features are temporal or static. For static features, we just fixed the maximal absolute value to 1 and kept the same scaling coefficient for other values. For temporal features, we used a scaler which is more robust to outliers, as some features contain few outliers which seem not to be incoherent. In that way it enables to keep these outliers and to conserve the structure of other values, without compressing them.

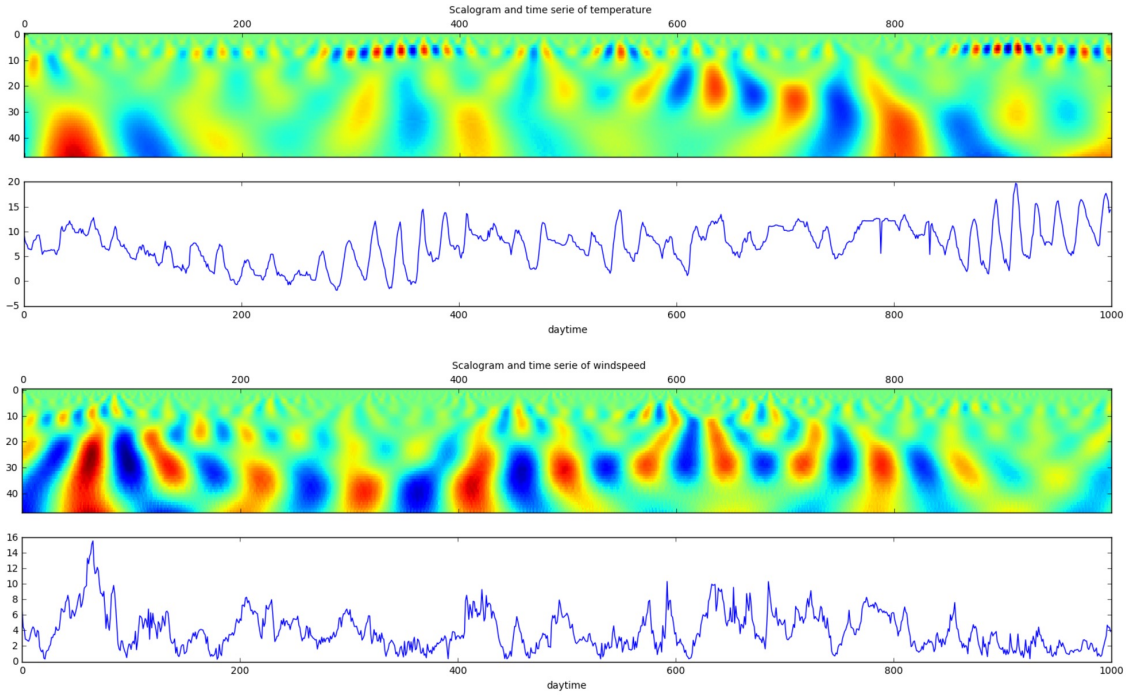


Figure 3: Scalogram for the wavelet transform with a morlet filter on temperature and wind speed

4 Experiment

4.1 Protocol

As said before, it isn't a forecasting task: we are not supposed to predict concentrations in the future, but instead, given past and present observations of the concentration in different areas we want to predict the concentration in a new area.

Consequently one had to split the dataset into train and dev sets according to this observation. To do so, we chose randomly for each zone one station to put in the dev set and keep others for the train set. This allows to build a dataset generator to make a kind of cross-validation. We also used a fixed dev set where we arbitrary chose the stations to put in the dev set, in order to have for each zone one station in the dev set but also to maximize the number of points in the train set.

As say previously, we train a single regressor for each pollutant. Performance is measured by mean square error.

4.2 Model selection

First we tried different models, two linear models and two tree-based models: Given the size of the dataset, support vector machine regressions were not easy to use, consequently we choose to work on two linear regression method:

- L2 penalized linear regression
- Stochastic Gradient Descend for regression with a huber loss (for robust regression)

We also investigate tree based model, because they are known to be good at handling heterogeneous data, which is the case in our task with temporal quasi continuous data and static very discrete data:

- Random Forest regression [1]
- Gradient Tree boosting regression via XGBoost [4]

Figure 2 shows performance of each of these model, for our different features. Scores can seem quite low, but these experiment were realized on ten different split for the train and development set, in order to assess the robustness of different regressor given different station for training and testing at each round. We observe that best results are obtained for the tree based model (Random Forest and XGboost). Moreover one can notice that the standard deviation of the MSE on the ten splits is quite high, and XGBoost is the algorithm that is the most robust with changes in zones and stations in training. That is why our effort will focus on optimizing performance with the xgboost model.

Table 2: Average MSE over 10 random split (train-dev) with "never seen" stations in the dev set at each round. We also report the variance (STD) of the MSE over the different split.

	model	MSE	STD
Default features	L2 Linear reg	355.16	86.9
	SGD huber reg	367.96	107.2
	Random Forest	**310.67**	91.2
	XGBoost	314.82	**83.7**
Default + decomposition	L2 Linear reg	355.42	87.05
	SGD huber reg	366.39	108.41
	Random Forest	317.65	90.57
	XGBoost	**311.59**	**82.97**
Default + Decomposition + Rolling means	L2 Linear reg	342.13	90.34
	SGD huber reg	4660.57	3290.0
	Random Forest	311.33	89.9
	XGBoost	**302.19**	**82.00**
Default + CWT	L2 Linear reg	355.34	87.0
	SGD huber reg	366.21	108.3
	Random Forest	319.87	85.4
	XGBoost	**313.47**	**84.0**

Finally we experiment with convolutional neural networks (called CNN-FC in our tables), see section 4.4.

4.3 Result on the challenge test set

Figure 3 gather our performance on the test set provided for the challenge. We observe that best scores are obtained using the XGBoost regressor. We managed to improve our performance by adding the additive decomposition and also a little bit by adding wavelets transforms of temporal data. Our best score of 207.43 gave us the 3rd place in the challenge.

We must say that there are high differences between our score on the fixed dev set we created and the test set, which was big issue in the challenge as it was hard to guess if our improvements on training / development data will lead to actual improvement on the test set. For instance we see that our model with a random forest perform much better than others on the dev set but was clearly not efficient on the test set.

Table 3: Results on test set (submitted), with regard to result on our fix dev set.

Features	**Model**	**MSE - test**	**MSE - dev**
Rolling mean & std + delay	XGBoost	213.7	16.5
Decompose + Rolling mean & std + delay +	XGBoost	207.47	98.1
Decompose + Rolling mean + delay + cwt	XGBoost	207.43	96.5
Decompose + Rolling mean + delay	Random Forest	237.3	82.1
Default features	CNN - FC	295.8	99.3

4.4 An attempt with Deep Neural Networks

Given the improvements and the good results we obtained by introducing rolling means and signal additive decomposition into the set of features, we thought of using models that learns that kind of transformation of the raw data. To do so, we built a model based on two neural networks, one for

the temporal features and one for the static features. We then merged both outputs and added a final fully connected layer.

In order to capture temporal patterns and invariance, we modeled the temporal part with a cascade of 1D convolutional network, with max-pooling or average-pooling layer in between and ReLU activation on each neuron. Rolling mean are basically just 1D convolution, with a filter constant filter, this idea of this architecture is to learn more expressive filter from the different temporal series. The static part was done using a more simple architecture that consists in two fully connected layers of sizes 20 and 10. The learning was performed using back-propagation with the Adam optimizer. The model was implemented using Keras.

After many experiments in order to find the right architecture in terms of number, and size of filter for the convolution layer, or in terms of different kinds of pooling layer, we finally found an architecture that makes the network learns quite fast and that does not over-fit, at least on our dev set.

- Temporal model
  - 1D - Convolution (48 filters, 3x3)
  - 1D - Max Pooling
  - 1D - Convolution (48 filters, 3x3)
  - 1D - Max Pooling
  - 1D - Convolution (48 filters, 3x3)
  - 1D - Average Pooling
  - Flatten
  - fully connected (144)
- Static model
  - fully connected (20)
  - fully connected (10)
- Merge both and add fully connected with one output.

However when applied on the test data, the network only achieve an MSE of 295.8, which was one of our worst score. The main explanation we can find is about how static features were handled by the network. Indeed, we can guess that the 1D convolution on temporal data created useful features because result were good on the train and dev set, however static features didn't bring enough information to perform efficiently on test set. The main issue with static features is that they are quasi discrete, and very sparse (many zeros), as a result the network isn't able to learn much from it, neural network are indeed more efficient with more continuous representation of data, and our attempt to embed static data with two fully connected layer didn't work well.

5 Observation and other ideas

Working with different kinds of data is hard

As we can see on the previous plots, our predictions guess the trend pretty well, but may have trouble in guessing sudden spikes.
One intuition that we got is that these spikes may from time to time be explained by the geography around the station (presence of roads, industries around), coupled with the appropriate wind speed and direction. Indeed, we mainly worked on temporal data which are continuous and widely exploitable, and put apart discrete static data. We first thought that we could add a latent feature which after training would have represented the position of roads/industries around the station ; but as our predictions are related to different stations, we wouldn't be able to access their relative latent variables.

Generalization over different area is hard

Figures 4, 6, 8 show prediction of the concentration versus the ground truth, on stations of the dev set, and figure 5, 7 show the distribution of error for each station and pollutant. We observe that we don't make the same kind of error on all stations. For instance for NO2, on station 16, (figure 4, 5) we miss many spikes. As a result, the error distribution is skewed toward positive values whereas

on station 23, still with NO2, our algorithm underestimate the concentration and errors distribution in skewed toward negative values. The same phenomenon happens for PM10 and PM25.

This observation shows that our algorithm doesn't generalise the same way to every station, which can mean two things: either static information isn't enough understood by the model, or there isn't enough information in the few static features we have.

Building two estimators: trend and variation

Looking at our prediction versus the real concentration signals, we observe that our main drawback is that our model fails at reproducing the spikes. An idea that we wanted to explore was to decompose the concentration of each pollutant into a trend and variable components using additive decomposition of wavelets and then train an estimator for each component. We observe that our model manage to follow quite well the trend and seasonality of the concentration signal, so an estimator working only on high variability may be more likely to reproduce the spikes. All component estimators would then be merged as an additive / multiplicative model.

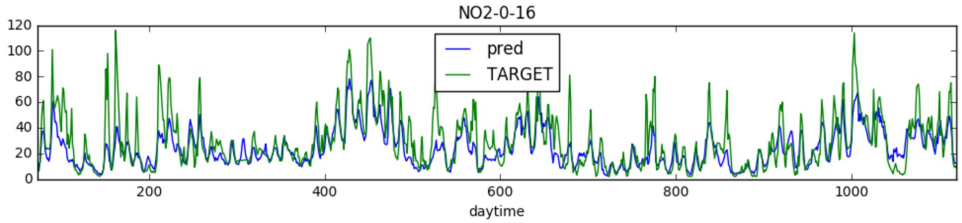
Reference

[1] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32.

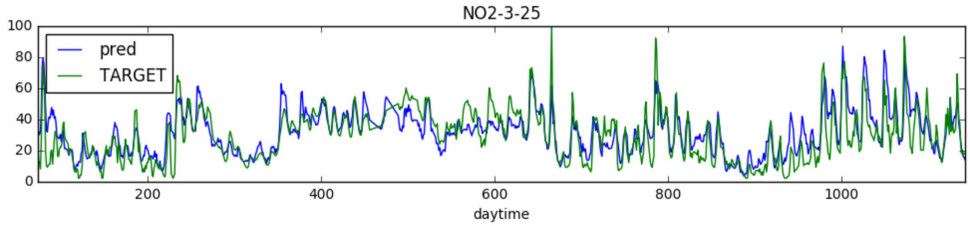
[2] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. 3rd. Academic Press, 2008. ISBN: 0123743702, 9780123743701.

[3] Krzysztof Siwek and Stanislaw Osowski. "Improving the accuracy of prediction of PM10 pollution by the wavelet transformation and an ensemble of neural predictors". In: *Eng. Appl. of AI* 25 (2012), pp. 1246–1258.

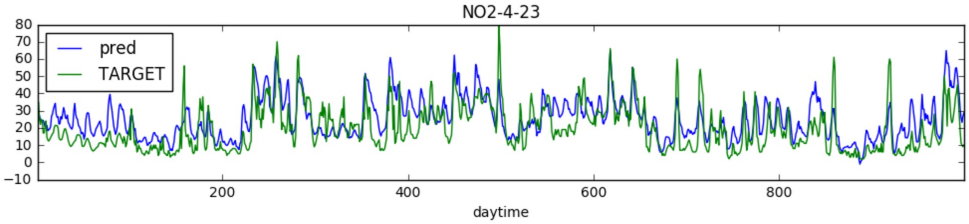
[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *KDD*. 2016.



(a) zone 0, station 16

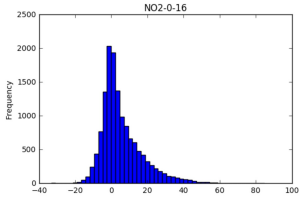


(b) zone 3, station 25

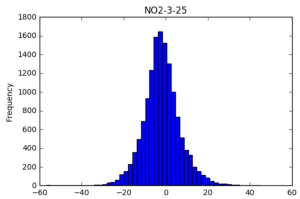


(c) zone 4, station 23

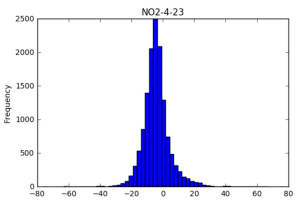
Figure 4: NO2 prediction vs ground truth on 3 stations of the dev set



(a) zone 0, station 16

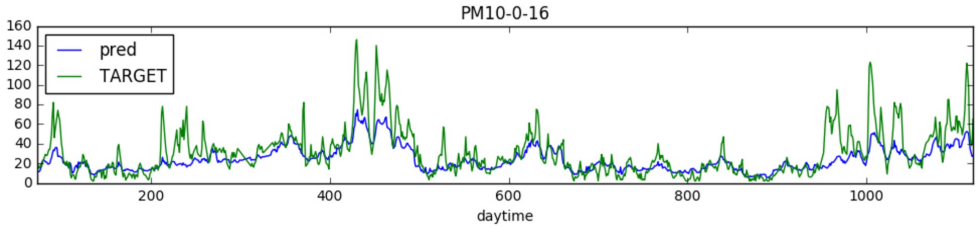


(b) zone 3, station 25

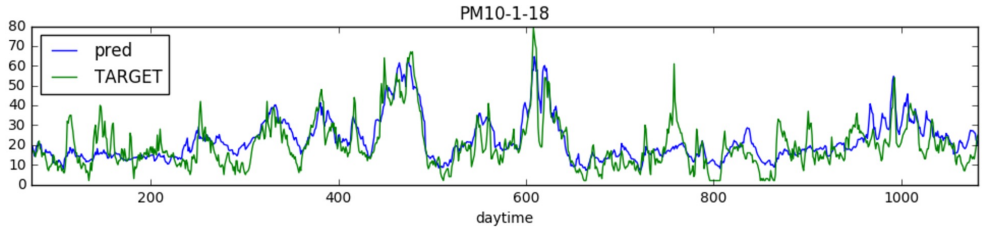


(c) zone 4, station 23

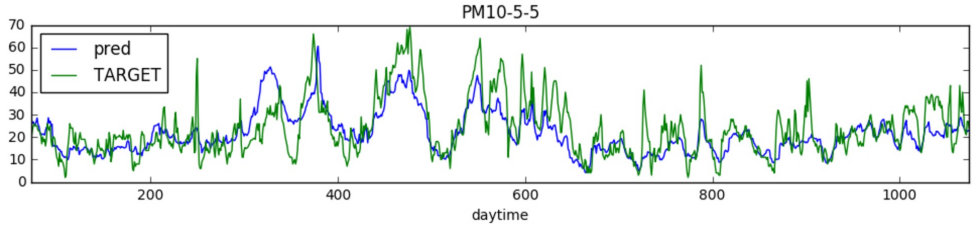
Figure 5: NO2 prediction error distribution per station in dev set



(a) zone 1, station 18

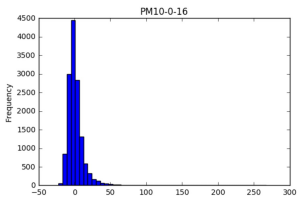


(b) zone 3, station 25

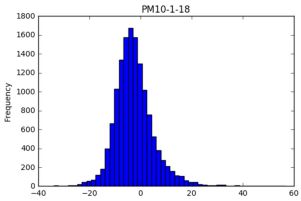


(c) zone 4, station 23

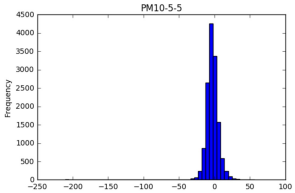
Figure 6: PM10 prediction vs ground truth on 3 station of the dev set



(a) zone 0, station 16

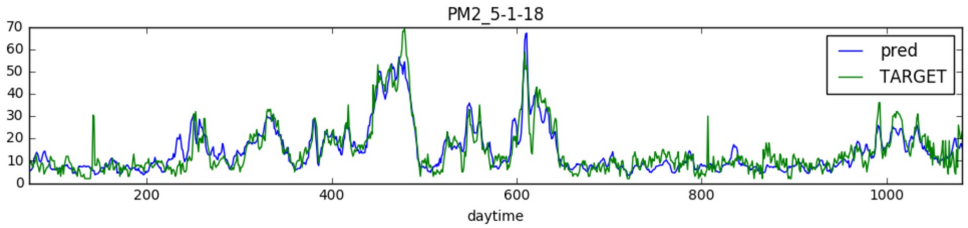


(b) zone 3, station 25



(c) zone 4, station 23

Figure 7: PM10 prediction error distribution per station in dev set



(a) zone 1, station 18

Figure 8: PM25 prediction vs ground truth on a station of the dev set