Report: Plume labs data challenge

Paul Dufosse, Matthieu Mazzolini, Konstantin Mishchenko

Abstract— **In this report we describe our solution to the Plume Labs data challenge, where we managed to get first place among 35 teams. We start with general description of the data, and then, in the second section, discuss the results obtained from linear models such as Ridge regression. In later sections we introduce our key idea – two-step model – and explain our other improvements. In the end of this report one can find technical details on cross-validation for this challenge, our proposal of additional enhancements, and, finally, our conclusion.**

I. Introduction

A. Description of the challenge

The aim of the Plume Labs challenge is to predict the level of air pollution in several French cities using similar data. For this purpose, the challenge relies on a Land Use Regression Model (LURM). It consists in using two types of variables. The land-use variable, which takes into account a given perimeter around a captor, and give an information about the area, such as the level of industry, nature, roads, etc. We refer to it as *location features* in our model.

Some of these features are provided for different area sizes. For example *green_500* describes the amount of green area in a radius of 500 meters around the station, and *roads_1000* counts the number of roads in a radius of 1000 meters.

We also have meteorological parameters which correspond to the monitoring of the stations, such as wind speed, rain intensity, temperature, pressure, etc. These features are time series and we refer to it as *weather features*. A similar work on this model was performed in the city of Gothenburg, Sweden [1] and in Montreal, Canada [2], as referred to in the literature.

B. Overview of the data

This is a regression supervised problem, and the score metric used is the classical mean squared error. The train data set is made of 448169 samples, and the test data set of 300891 samples. Besides the *weather* and *location* features explained above, we have to deal with 3 main categorical features:

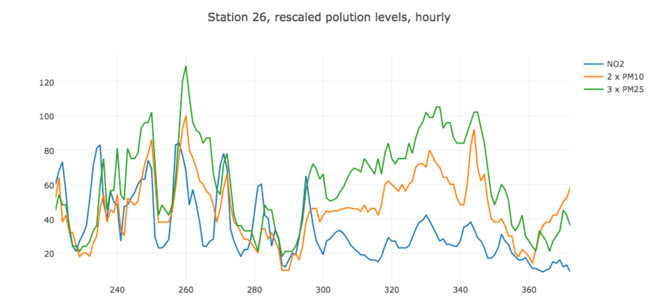
- *pollutant*: NO2, PM2_5, PM10, which matches with the 3 pollutants which levels are to be predicted.
- *zone_id*: ranges from 0 to 5. It corresponds to an anonymous city.
- *station_id*: integers, ID of the station. In each zone, or city, you can find from 1 to 3 different stations.

It is worth noting that the stations in the same city share the same *weather* features. In fact it is normal because the weather features are not measured precisely, and therefore the *windspeed* and the *temperature* features are the same over a given city at the same moment. However their *location* features are different. It was expected because the stations are spread in different part of the city, and have different surroundings.

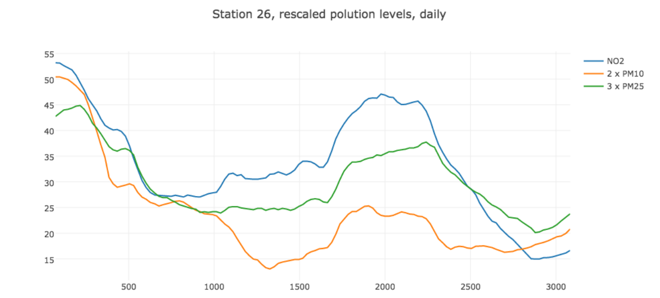
Another important remark is about the test set. It mainly contains measures from *stations_id* which do not appear in the train set. However these new stations appear to be in one of the 6 zones, i.e. cities, of the train set. This leads to very big similarity between observation from train and test set.

C. Pollutants

As we've mentioned there are 3 types of pollutants: NO2, PM2_5, PM10. We can take a closer look at different pollutants on same station, for example, station 26. If we rescale pollutants with coefficients, we can see that they are very similar at some moments and very different at others:

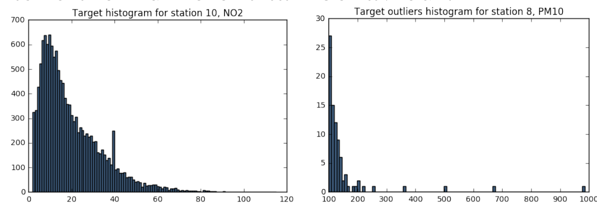


This difference can be seen with naked eye if we compare average pollution levels on daily basis rather than hourly:



D. Distributions

Another sign of big difference between pollutants is their different tail behaviour:



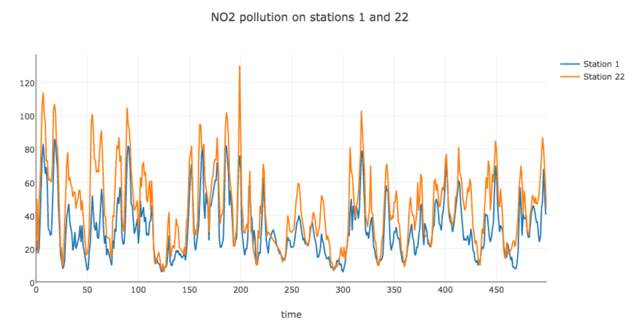
Despite PM10 pollution has long tails, its mean value is almost same for different stations. Meanwhile, one can see in the table below that the situation changes dramatically when we start looking at the NO2 pollutant:

Station id	1	4	6	8	9	26	28
Average NO2	36.11	18.55	47.9	14.85	20.56	23.44	38.35

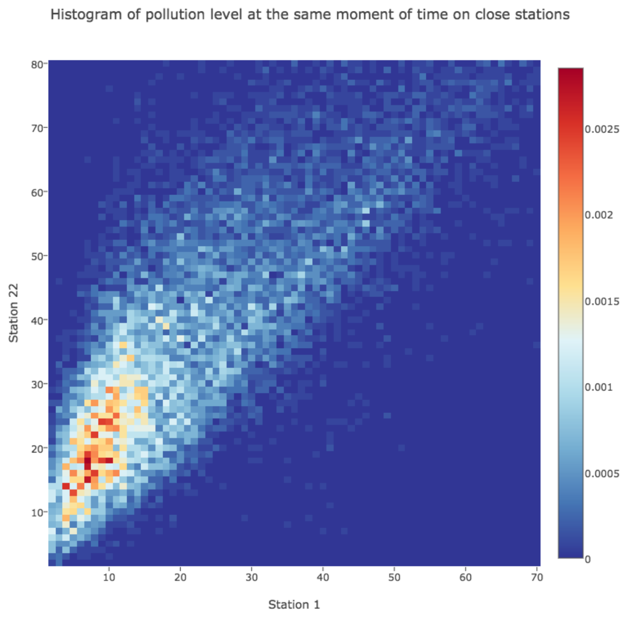
We can conclude by saying that the location features are especially important for predicting NO2 pollution.

E. Correlations and noise level

Let us now compare two stations located in the same city. It's clear from the graph below that their NO2 pollution levels are highly correlated:



At the same time one can mention that difference between them is often quite big reaching value of 40 at some moments. Taking into account that they are located very close to each other and have almost same weather at every moment, we can say that this difference allows us to estimate noise level. Let us plot joint histogram of their pollution levels:



While there is a clear correlation, inference is very complicated: if, for instance, the station 1 pollution level has value 20, then the station 22 pollution level can vary from 14 to 67.

II. Building a First Model

A. Dummy variables

As seen during the course, a first basic approach to deal with categorical variables is to build dummy variables. However it is not really relevant in our case for several reasons:

- *station_id* has different values in the train and test set, so it can't be used straightforward

- *pollutant* ranges and behaves differently regarding its value, as seen in the graphs and the distribution histogram before
- *zone_id* are common in the train and test data set. However the *location* features already aim to discriminate the prediction inside a given zone, and they are more accurate than just zone dummy variables.

As a conclusion, it is better to predict each type of pollutant separately. Therefore, we split our train and test sets in 3 different subsets for each pollutant. The question of training different models for each *zone* could also be tackled. We will keep it for the cross-validation, as explained in part V.

B. *Basic regression*

After splitting the train set in 3 different subsets for each pollutant, we trained a Lasso regression based on both *weather* and *location* features. However, the Lasso regression did not succeed in interpreting the *location* features efficiently. Looking at the coefficient of the Lasso regression, we noticed 2 main issues:

- The sign of the coefficient didn't match the common sense. We mean by this that some *roads* features had negative coefficients, whereas some *green* features had positive ones. This should be the opposite, once talking about pollution prediction.
- The range of the perimeter was also wrongly interpreted. A feature *road_100* could have a positive coefficient whereas *road_500* a negative one. In an ideal model we would instead expect both of them to be positive.

We also tried a Ridge regression on *weather* and *location* features, but it did not perform well, certainly for the same reason.

In fact, the *location* features are a bit misleading when dealt together with the *weather* features, which are time series. The test score is in fact better when we select **only** the time-series features, i.e. the *weather* features, and perform a basic linear regression on it. In the next section, we will explain how to take a better advantage of the *location* features, by building a two-step model.

C. *Random forests*

Once we start using only weather features, we can replace linear model with more complicated one. It happens that decision trees and random forests outperform them and can already give a reasonable prediction.

III. Two-step model

It's high time to take advantage of the location features. To this end, let's predict average pollution level on each station. We assume that the pollution level is a sum of constant location term and random weather term:

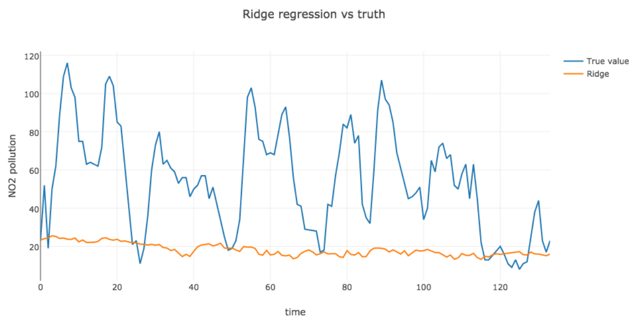
$$Y_{pol} = Y_{loc} + Y_{weather}.$$

One can note that each zone has special weather conditions, so our first approach was based on building a common weather model and subtracting predicted $\hat{Y}_{loc}$ to build then a model for prediction of

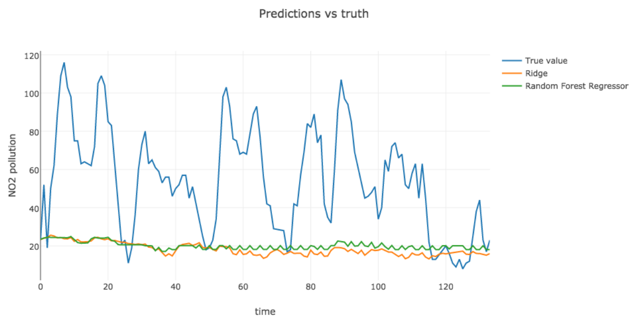
$$Y_{loc} \approx \hat{Y}_{pol} - Y_{weather}.$$

Then, our score sky-rocketed.

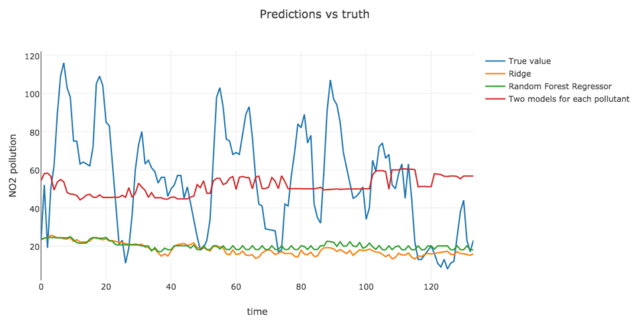
To sum up, let us illustrate the difference between several approaches discussed:



Ridge regression is too weak to predict big variance.



Random forests are a bit better and are less inclined to underestimate pollution level.



With two-step model we are able to predict average value and be more accurate with variance as well.

A. Important remark

From the very beginning of the challenge, we noticed that in the train set, each station in the same zone shared the same *weather* features, when compared at the same period. What we also noted, is that in the test set, the *station_id* features were different from the ones in the train set. Yet the 6 *zone_id* features are still the same.

What was not obvious, is that the period of time in the test set already appeared in the train set. Therefore the *weather* features of the test set were the same as the *weather* features of the train set, when compared in the same zone, at the same time. It negatively affected the quality of our models, because pollution levels on different stations can be significantly different even with the same weather.

After this notable observation, the basic idea was to compute the average of pollution between all the stations in the same city, and use it as a target variable for a new model.

IV. Final solution

A. Location before weather

One idea that can improve the last model is to reverse the order: first train location model and then use weather features. We can take into account different weather conditions of each zone by adding average weather characteristics such as average temperature. However, we later mentioned that feature selection suggests to ignore this characteristics.

This approach allows us to make a more accurate analysis of the location model, but we still have the same problem: we lack data. We have 15 observations for NO2 and 19 location features to make new predictions. Only linear models could be used in that situation, and even then we will have a really bad accuracy.

B. Elastic Net

Hopefully, we can try to use data description to understand what a linear model should look like. A good tool to perform constrained linear regressions is Elastic Net from *scikit-learn*, which has a flag called 'positive'. Assigning *True* to this flag forces all coefficients in linear regression to be non-negative. Clearly, only *green_5000* and *natural_5000* can decrease mean pollution level. Thus, we reverse the sign of these two columns and then we perform Elastic Net regression.

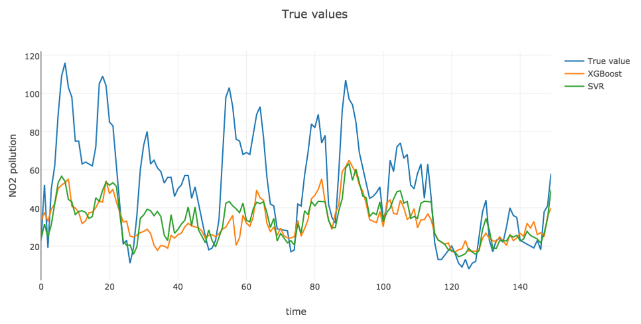
We can find out which location features are the most important by making Elastic Net more regularized. It turns out that number of roads and the inverse distance to the closest road play major role. It leads to an idea of generating new features based on everything that is related to roads. Thus, we add features of type $roads_{xxx} = road_inv_dist \cdot rout_{xxx}$, where xxx takes values in $\{300, 500, 1000\}$.

It turns out that solution of Elastic Net regression is sparse. The features that are selected by this algorithm are: $\{green_{5000}$, $route_{100}$, $hlres_{1000}$, $route_{1000}$, $natural_{5000}$, $route_{300}$, $hldres_{1000}$, $hldres_{500}$, $roads_{1000}\}$.

C. SVR and XGBoost

Our first approaches were based on as simple as possible models. And once we were finished with data understanding and feature selection, we started using tools that are most powerful. Trees based algorithms proved to be the best fit for the challenge data and XGBoost outperforms Random Forests compared with cross-validation.

SVR model, in turn, is weaker than Random Forests. Nevertheless, SVR tends to make overestimated prediction, when XGBoost makes underestimated one and vice versa, so they can help each other. We, thus, used a mixture with equal weights of this two models to make final submission. Below are the predictions for the station 28 and pollutant NO2 versus true value:



Clearly, we are still unable to predict variance, but now we are much closer than with original models.

A technical issue we faced is SVR's high memory and time consumption. Our dataset is too big to feed it to SVR, so we split the train set into 8 pieces based on the time feature. Each fold consists of observations within the same time interval and is used to train a model and then to make prediction on the corresponding part of test set. This allows to resolve the computational burden issue and does not seriously affect the cross-validation score.

V. Cross-validation

For every model described, we tuned the parameters by cross validation, using most of the time *GridSearchCV* method from the *scikit-learn* package. But the way of performing a cross validation is very tricky in our challenge, and is a matter of discussion. In the *scikit-learn* package you can specify different types for your cross-validation:

- **K-Fold** assumes that the samples are i.i.d., which is not the case because a part of the features are time series.
- **TimeSeriesSplit** is a special case of cross validation designed for time series. But we would catch too much information about the current station in the train set, which does not appear in the test set, and then the score would be too optimistic.
- **Leave One Group Out (LOGO)** works similarly as K-Fold, but the Folds are set beforehand into fixed groups, within the samples are assumed to be i.i.d.

In our case the LOGO set with the *zone_id* as group seems a good solution, because we can assume that pollution and weather measures are different in each zone. Therefore we tuned our parameters by letting one zone out for validation test and training on the 5 others zones. We could not use a pipeline and set the cross-validation to LOGO. It only works if you have a *transform()* method in your model while our solution consists in several steps. Therefore we had to implement it ourself.

VI. Possible improvements

A. Time series analysis

Despite pollution level is a time series, it's very hard to treat it in that way. The strongest tendency is low pollution level at night, but, unfortunately, pollution level is extremely random in the morning and in the evening. Another problem is that dependency between pollution at a moment and one moment ago is very weak, so adding features' values one moment ago does not improve prediction quality. This is a hard issue, but solving it should help a lot.

B. Feature engineering

Perhaps one way to improve the result would be to improve the LURM model by building new features. For example in our challenge *windbearing_cos* and *windbearing_sin* are respectively the cosine and the sine of the wind. We could imagine building new features in the same way, with the *pressure* or the *precipitation* feature. The problem is that we just lack expertise in this domain and would have to meet experts in meteorology to generate features that would make sense.

VII. Conclusion

In this report we presented our solution to the Plume labs challenge. It is the result of many trials and experiments, which have continuously evolved since the beginning of the challenge in November. It has no intention of being state-of-the-art and could surely be improved with more time. Eventually this challenge raised our interest, in a way that Plume labs is trying to use machine learning to warn people about the quality of the air they breath. By the mean of an app, they would be able to deliver with high precision the level of pollution in the surroundings of every app user. The geolocalisation would create customized *location* features for each user, and then deliver a customized prediction, regarding your position in the city. Such challenge is a great proof of an application of machine learning that helps people to live better on a daily basis.

References

[1] Land use Regression as Method to Model Air Pollution. Previous Results for Gothenburg/Sweden, Link

[2] A land use regression model for ambient ultrafine particles in Montreal, Canada: A comparison of linear regression and a machine learning approach, Link