Apprentissage Data Challenge
Population forecast over SCNF train stations in IDF.

Mathis Embit & Gurvan Richardeau

March 2024

Abstract

SNCF-Transilien runs 6,200 daily trains, serving 3.2 million passengers in Île-de-France, with 2.3 million smart card validations per day, growing by about 6% annually from 2015 to 2019, emphasizing the need to prepare for increased demand to improve service and operations. The challenge consists in predicting the daily validations per station in the medium to long term, aiming to anticipate future passenger volumes. Anticipating and understanding the growth in validation volumes at its stations is crucial for SNCF-Transilien. To do so we proceeded in three main steps, a data exploration, some feature engineering and finally the construction of machine learning models. Concerning the models we started from the most basic to more advanced techniques. We ultimately reach a MAPE score of **99.45** on the public test set.

You can check our implementation details on https://github.com/mathisemb/challenge_sncf_2024.

Contents

0 Data exploration

You can see some data samples of the training set in appendix A.

At first looks, data seem globally balanced, nevertheless, after testing our prediction algorithms, we managed to spot some stations that have missing data (see appendix B).

We can see covid, strikes (as the big one of 2019 December), and probably renovation periods.

This data challenge presents one particular difficulty: the test set (i.e. the first six months of 2023) is probably quite different from the train set (2015-2022). We think this for two reasons.

First, by looking at tendencies over years on many different stations, it seems that people were not using transports the same way that before covid. Our main guess is that it is due to remote working.

Secondly, the MAPE (Mean absolute percentage error) we had on our own test sets (i.e. test set coming from a splitting of the train set) were very different from what we had when we made submissions. We had scores between 1 and 10 whereas our submission were around 200 for instance. There is one subtlety though, the MAPE presents an issue for zero values and one needs to make a choice on the behaviour to have in this case. In our model, we have chosen a threshold below which the value is set at this threshold. If the choice of the challenge provider is not the same, this could partly explain the observed differences.

This difference between the training set and the test set means that splitting our training set to have our own test set and making measurements on it with different models is not really a good way of assessing the quality of our models. So, in a way, we were able to test our model only once a day (challenge submission rules), drastically limiting the hyperparameters research etc.

As said above, temporal dynamics did not appear so clearly to us, Covid-19 seems to have broken a lot of dynamics. So we chose to tackle the challenge without any time series approach at first.

1 Feature engineering

1.1 Day typing

So our main approach was to claim that the number of visit in a station is not really related to how it has been visited the previous days but much more by the type of day it is. To this end we set our day type classification as follows:

- 0. 'job' : means none of the other day types,
- 1: 'mid_holy' : holydays but neither first nor last day,
- 2: 'start_holy': first day of holydays period,
- 3: 'end_holy': last day of holydays period,
- 4: 'Noel_eve',
- 5: 'Noel',
- 6: 'New_year_eve',
- 7: 'New_year',
- 8: 'Labour Day',
- 9: 'May 8 1945',
- 10: 'July 14',
- 11: 'Assomption',

- 12: 'Toussaint',
- 13: 'November 11',
- 14: 'Easter',
- 15: 'Ascension',
- 16: 'Pentecost'.

As the vacation and holyday labels provided by the data challenge were not completely accurate, we recovered it from web and injected it in our train data.

1.2 Covid and strikes managing

We chose to remove the entire covid period plus the strike of December 2019. For the rest of unusual events, we counted on our anomaly elimination algorithm.

1.3 Time period selection

As the last year (year 2022) was pretty much different than the period 2015-2021, we tried two things:

1. Only base our prediction on year 2022.
2. Replicate data from year 2022 to give twice importance to this year.

To have a better understanding of what we mean by replicating year 2022, see appendix C.

1.4 Anomaly elimination algorithm

As we can see in appendix B, some stations are challenging. We then had to implement an algorithm that removes anomalies of the dataset. The algorithm is available in 'DataPreprocessingTools.py' as the function 'data_anomaly_elimination'.

We used a confidence interval based method, applied for each day of the week and then for each month. More precisely, the algorithm looks one station by one, computes the average number of visit on Mondays (for one station only), the corresponding standard deviation and computes a confidence interval of level alpha *as if the number of visit was following a normal law* and removes from data the points that are outside this interval. Then it repeats for Tuesdays, ... until Sundays, and then it does the same for each of the twelve months of the year. And finally repeats for the next stations.

We can see the results in appendix B.

1.5 Features choice

We tested many different feature combinations among:

- 'day_numeric' : day number of week (0 to 6)
- 'days' : day number of the month (1 to 31)
- 'week' : week number of the year (1 to 52)
- 'months' : month number of the year (1 to 12)
- 'day_type' : See 1.1

Finally, here are the features we have kept:

- 'day_numeric'
- 'week'
- 'day_type'

With hesitation between week and month as we thought that maybe it encoded too much information. Indeed with 'day_numeric' and 'week', a model can theoretically retrieve a specific day of the year.

2 Models

We don't see why the fact that one station has so many travellers would influence the number of travellers in another station. Also, since our approaches are 'non-longitudinal' (i.e. don't really take into account the sequential aspect of time, and are simply making categories (features)), even if the number of travellers in one station would be dependent of the number in other stations, our models wouldn't be able to leverage this.

We will call the overall model a station-wise model, in the sense that there is one single model for each station.

As we said we tried several strategies to predict the number of validation given a station. We first decided to try basic machine learning algorithms. Also given our data exploration we did not immediately feel the need to use time series.

2.1 Station-wise average

We differentiated several types of day and averaged the number of validations in the given station for each of these day types.

Let s^* be a given station and d^* a given data. Let job(d) be a boolean being true if $d \in$ {Lundi, Mardi, Mercredi, Jeudi, Vendredi} and false if $d \in$ {Samedi, Dimanche}. We estimate the number of validation v_{s^*,d^*} by $\hat{v}_{s^*,d^*} = \frac{1}{\sum_{d \in D} \mathbf{1}_{\text{job}(d)=\text{job}(d^*)}} \sum_{d \in D} \mathbf{1}_{\text{job}(d)=\text{job}(d^*)} v_{s,d}$, where D is the set of available dates for the station s^*.

We then tried different regressions. For some time the best score has been 170 by a Random Forest. The leading a score finally became 99.45 achieved by an MLP.

2.2 Station-wise Linear Regression

We tried to capture some fashion in the number of validation with a linear regression, even polynomial, but it did not predict better than the average.

2.3 Station-wise decision tree

The first idea we had with the day type averaging is in fact a decision rule. Following the inputs job, ferie, vacances we decide which lines of the csv we want to use. Hence using a quiet deep decision tree we get better score than the day type average.

2.4 Station-wise random forest

We can improve decision tree by bootstraping and aggregating (baging), namely random forest. We improved the score with this model.

Once we tried a few algorithms we need to choose one. We thought of 3 possibilities when making a prediction:

- choosing best score model prediciton
- averaging the predictions of several model
- training a new regressor taking models predictions as input

As random forest clearly gave us best score than other models we decided not to try averaging. But we decided to try the 3rd option which consist in a linear regression, decision tree and MLP stacked model.

2.5 Station-wise LR+DT+MLP stacked model

Surprisingly this model performs less well than random forest. But we also tried the MLP alone, which gave us good results.

2.6 Station-wise MLP

We got our best result (99.45 on the public test set) using an MLP with those parameters:

```
input_dim = 3
n_layers = 2
hidden_dim = 512
nb_epochs=50
learning_rate=0.001
```

The inputs of the MLP are

```
week
day_numeric
day_type
ferie
vacances
```

Again, it does not take the previous day number of validations as an input. There is no consideration about the time. We can see that even if the inputs are based on some decisions, our MLP performs a lot better than decision tree. This can be explained by the fact that the hidden layers has 512 neurons which is big compared to the depth of the decision tree. Hence the MLP can combine features to create meaningful new features and use them to predict.

2.7 Autoregressive Models

We finally tried a Time Series based approach, ARIMA and SARIMAX.

- ARIMA: Autoregressive integrated moving average, that takes essentially two elements: dates and one value column (here the number of validation per day).
- SARIMAX: Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors, that can take exogenous features, here we tried with the day type feature for instance.

As we were running out of time and as those methods were not very familiar to us, we haven't really delved into them. We only computed the projection on three stations and it looked really bad and quiet cost expensive in term of computation (+ 1min per station). As we can see on the figure, the prediction begins at the end of 2022 and is clearly distinguishable.

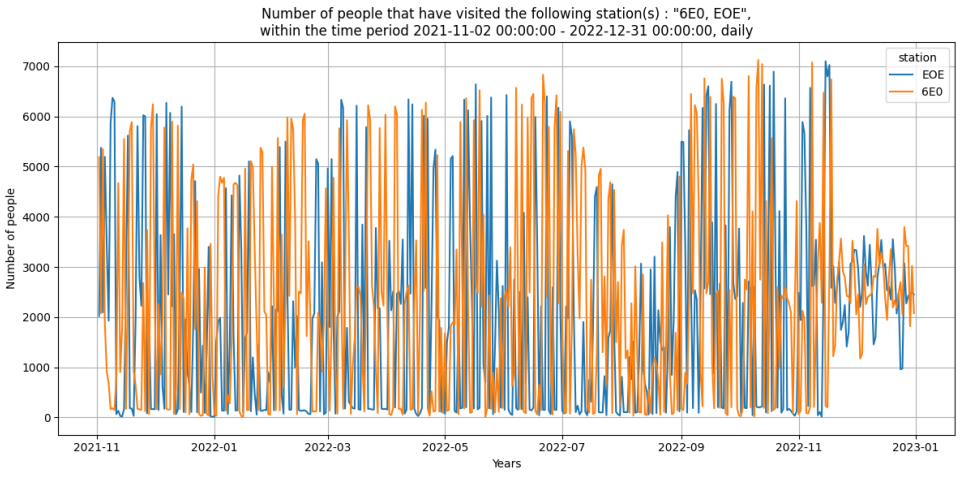


Figure 1: Example of SARIMAX prediction attempt on two stations. The standard deviations of the predictions become drastically lower.

3 Results

The following scores are from the public test set.

Method 1:

- time period considered: 2015-2022
- Anomaly elimination: Yes, around 2% of data has been removed
- features used: week, day of week, day type (from section 1.1)

Method 2:

- time period considered: 2022
- Anomaly elimination: Yes, around 2% of data has been removed
- features used: week, day of week, day type (from section 1.1)

Method	Average Model	Linear Regression	Decision Tree	Random Forest	MLP
Method 1	289	None	169	184	**99**
Method 2	296	218	200	170	**107**

Table 1: Performance Results on the Public Test Set. To know further details on hyperparameters, see the code on https://github.com/mathisemb/challenge_sncf_2024.

4 Conclusion

We have shown that our hypothesis about the lack of relevance of the time series approach might not have been wrong, since the score we obtained was not very far from the best result obtained in the data challenge, or at least we have shown that it was possible to have good results without time series.
After many thoughts, we finally think that a time series approach could be powerful, but as said in 2.7, we haven't done a proper attempt. We would really like to hear from some other teams about their success in adopting this approach, if any.
The best results were obtained with Multi Layer Perceptron approach, which was not especially expected, as the problem did not seem to present any subtle features at first. Indeed, if the MLP works better than classic regressors methods, that means in a way that it has found new features, different from the ones we gave to other regressors.

5 Overall comments

During this data challenge, which was our first one, we gained a lot of insight on some basic machine learning algorithms that we have just learned theoretically. Indeed, we had many thoughts regarding whether an algorithm would perform better or not before testing them out. Then we had to understand why our performance predictions were incorrect if this was the case.

A Raw data samples

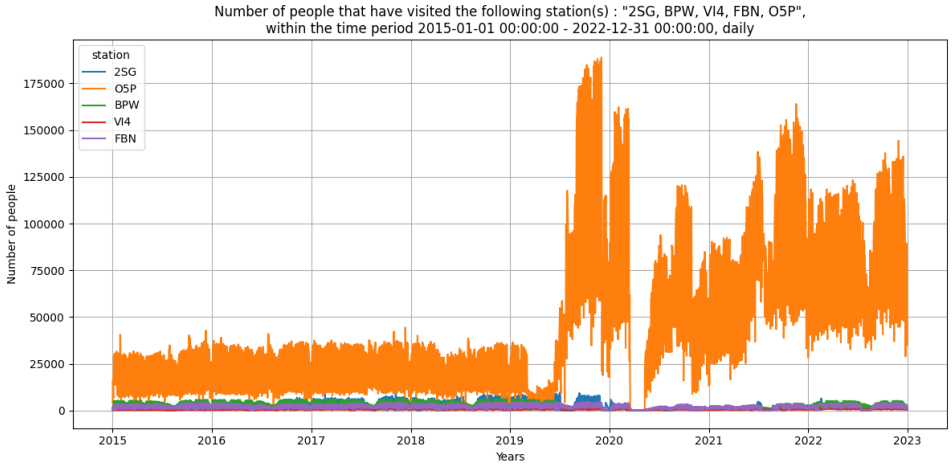
See Figs 2 & 3.

B Challenging Stations

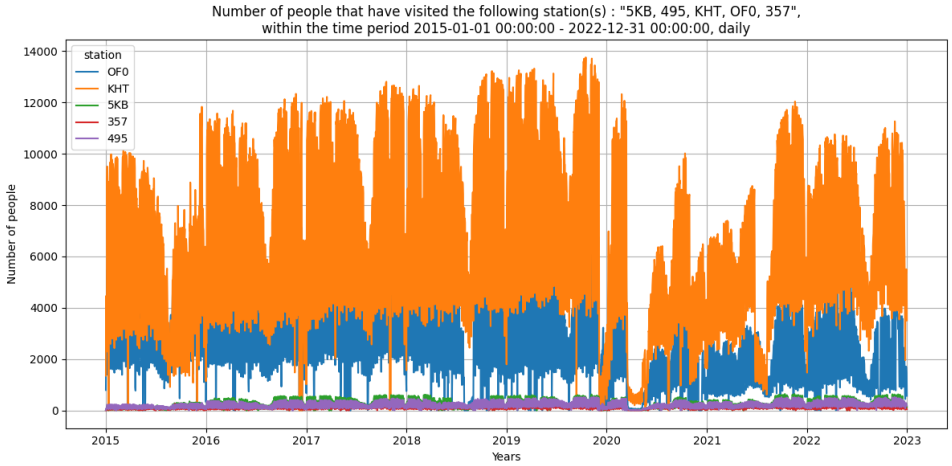
At first looks, data seem globally balanced, nevertheless, after testing our prediction algorithms, we managed to spot some station that missing data.

See Fig 4 to see how do they look like, and see Fig 5 to see how do they look like once the algorithm of data elimination has been run.

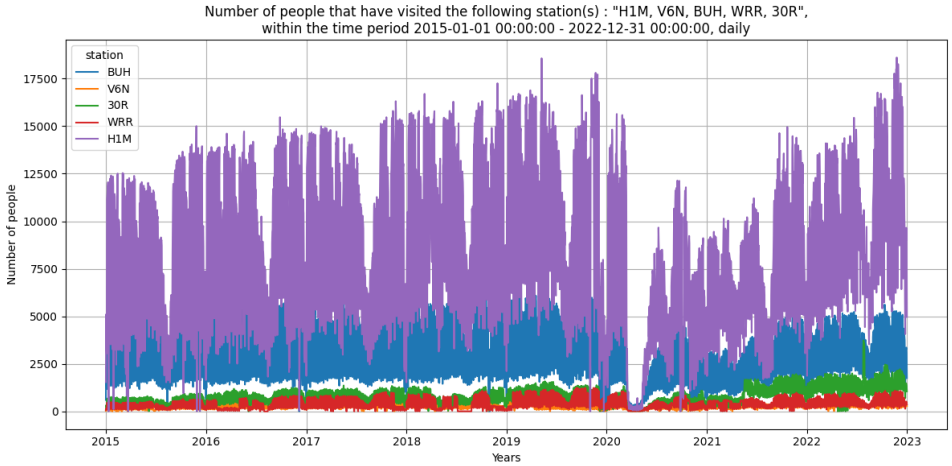
See Fig 6 to see how does the distribution of eliminated stations look like.



(a)

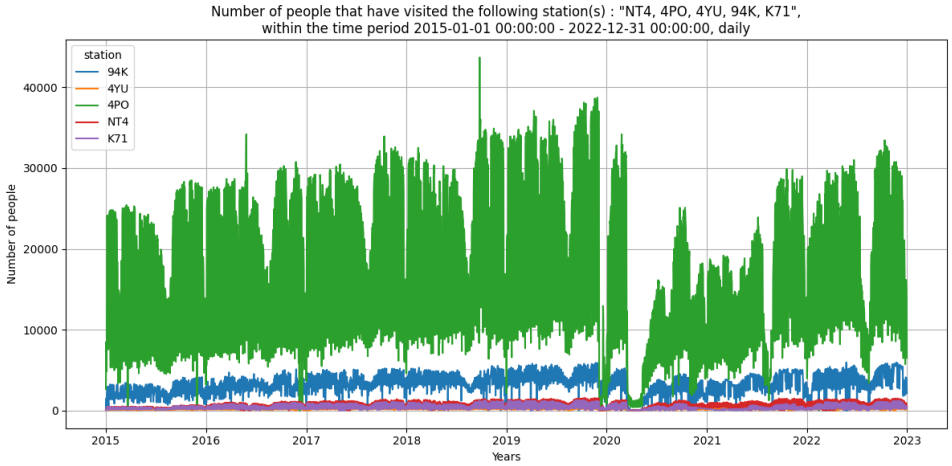


(b)

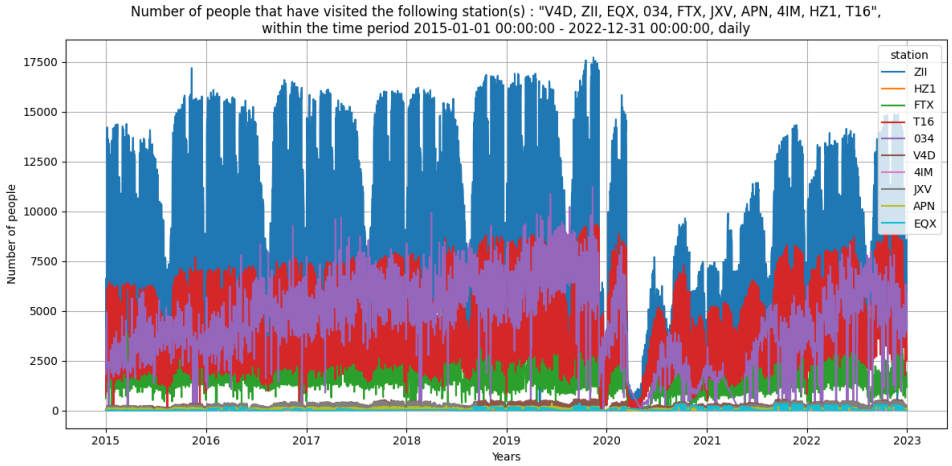


(c)

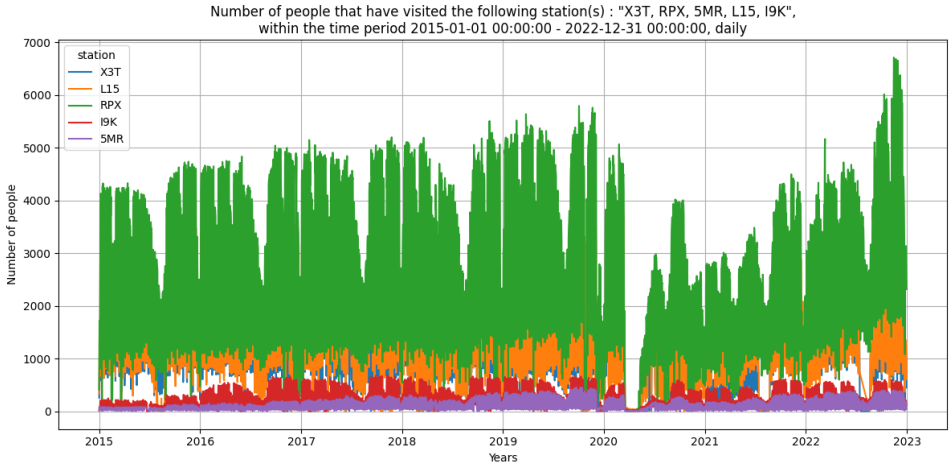
Figure 2: Data samples



(a)

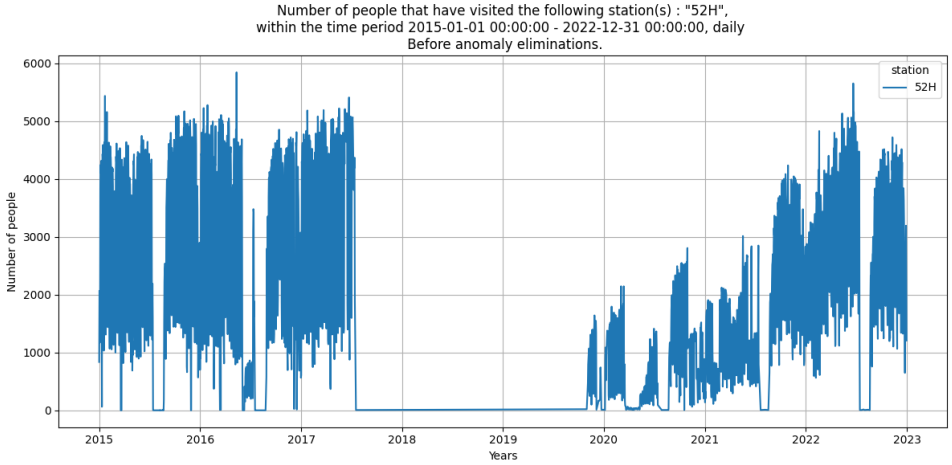


(b)

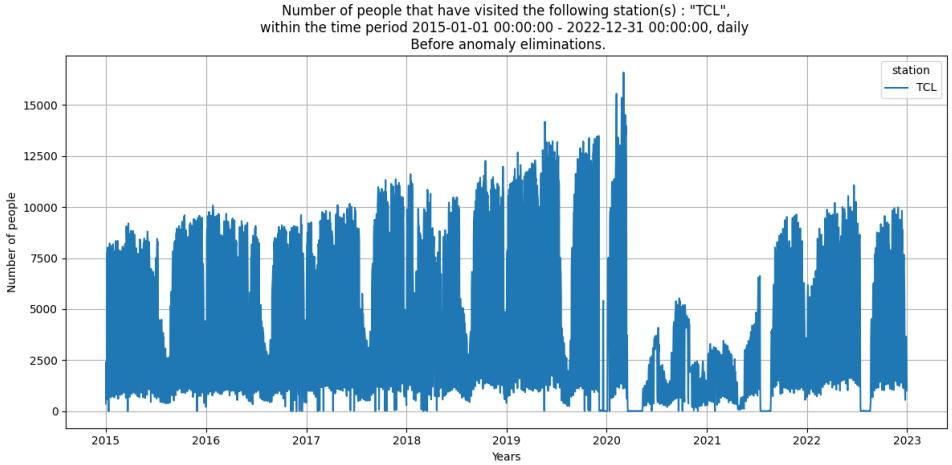


(c)

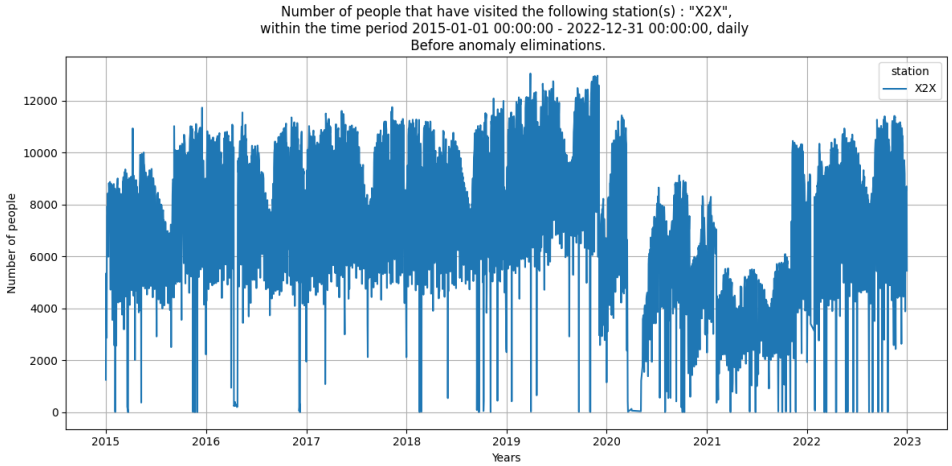
Figure 3: More Data samples



(a)

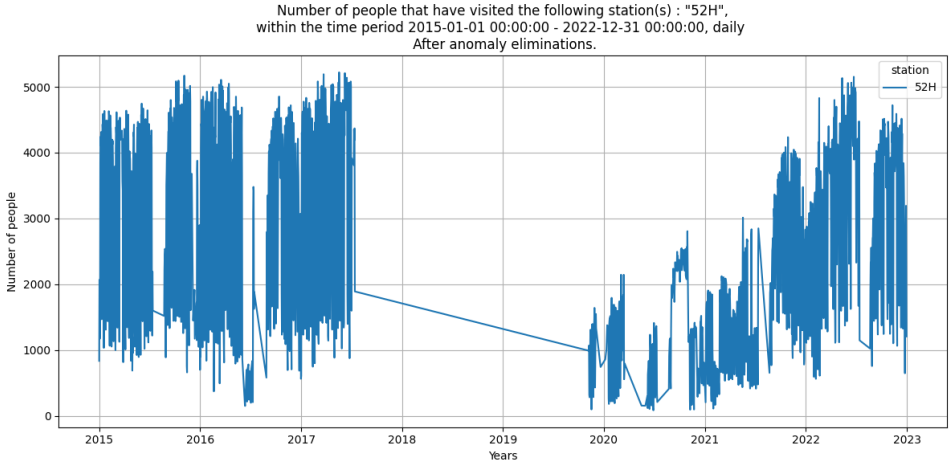


(b)

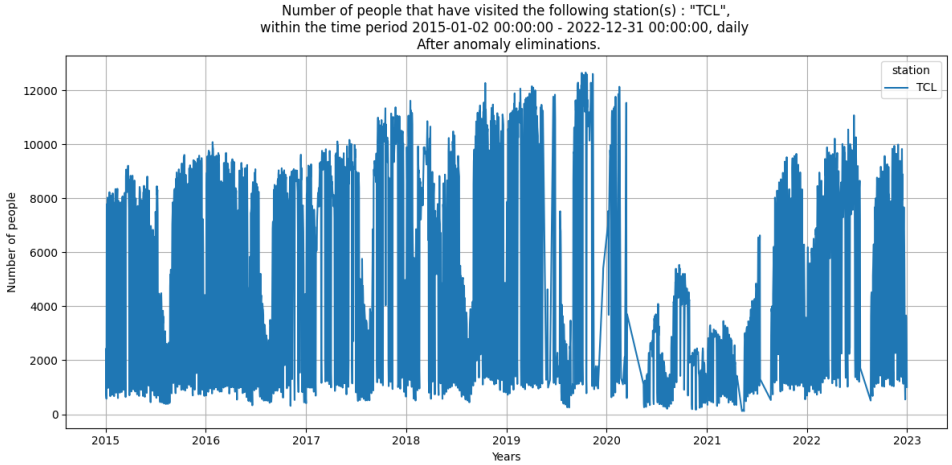


(c)

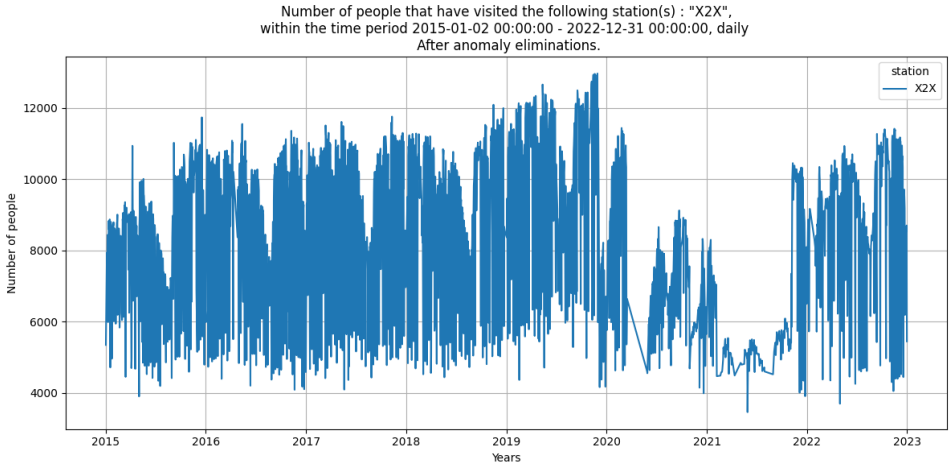
Figure 4: Station with anomalies detected. We can see two types of anomalies, some period records are completely made up of zero persons (a) and (b), some presents strange zero singular records (c) and (b)



(a)



(b)



(c)

Figure 5: Challenging stations once the algorithm of data elimination has been run. We can see that the two anomaly types have been successfully managed.

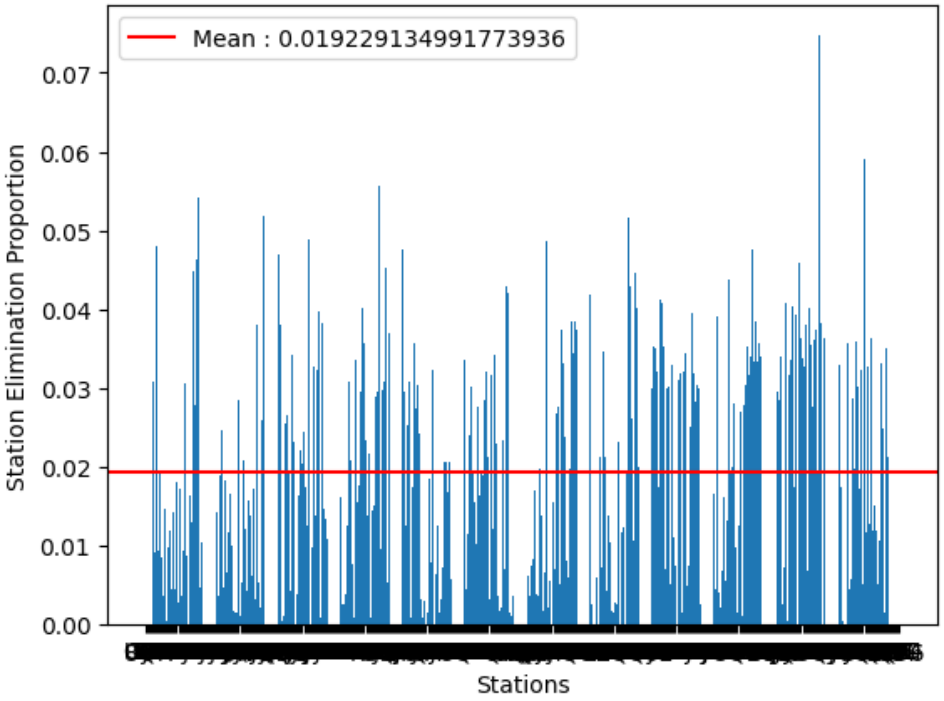
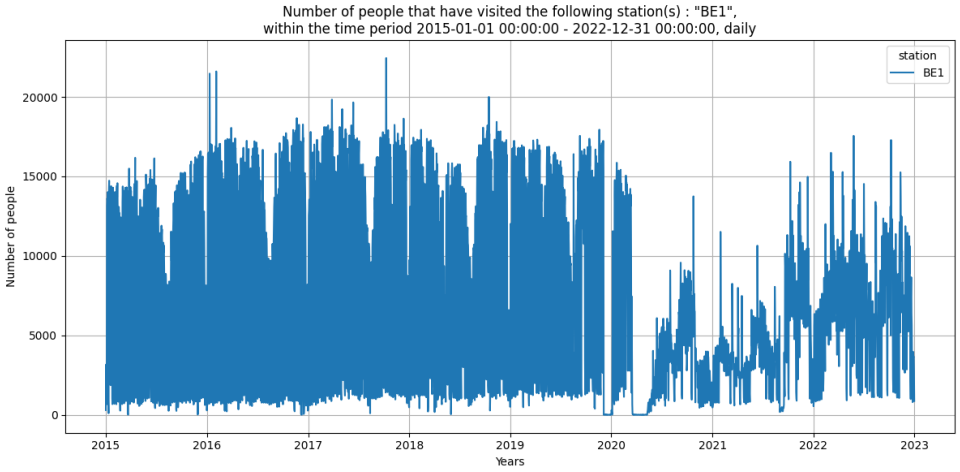


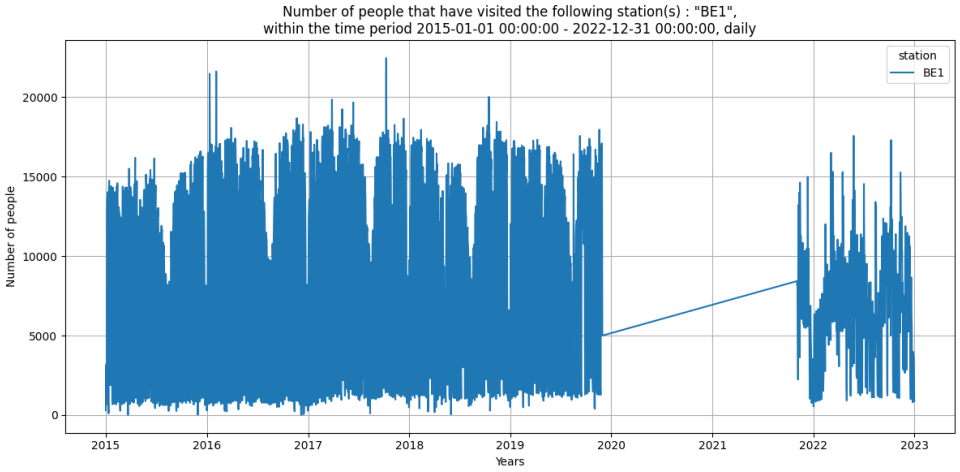
Figure 6: Each peak represents the proportion of data eliminated of one station. In average, almost 2% of data has been removed. But the standard deviation obviously matters a lot and is very high.

C Covid removing and replacing

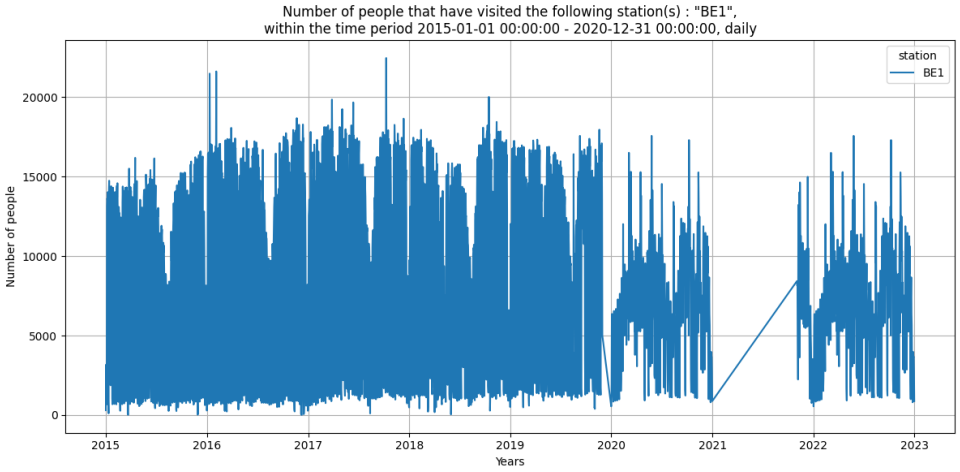
See Fig 7.



(a) Data with covid



(b) Data with covid removed



(c) Data with year 2022 replicated

Figure 7: Understanding our data engineering, with the example of one station. This approach only works if we have a non-longitudinal approach, as the date sequence is interrupted by empty data.