ALEXANDRE Teddy
MARENGO Matteo
teddy.alexandre@ens-paris-saclay.fr
matteo.marengo@ens-paris-saclay.fr

SNCF Data Challenge
Apprentissage et génération par échantillonage aléatoire

18/03/24

Abstract

Time-Series forecasting, and especially train attendance prediction, is a challenging task that requires incorporating many data and information together to predict the future. A lot of methods already exist in Machine Learning. As a regression task the first approach was to use classical algorithms. Linear regression with regularization and Ensemble learning methods proved to be the best. When using deep learning methods, it surprisingly happens to have a better MAPE score even if the trend seems to be less good. It brings the question of if it is an accurate metric or not. Finally, more advanced methods have been or should be studied to even perform better on this task.

Keywords: Forecasting, Regression, Ensemble Learning, LSTMs, Feature extraction

1 Introduction

This report describes our work done for the challenge **"Anticipez l'affluence au sein des gares SNCF-Transilien !"** in the context of the course **"Apprentissage et génération par échantillonnage aléatoire"** given by Pr. Stéphane MALLAT, as well as our methodology and a critical point of view of the results obtained. All the code that has been used can be found on our **GitHub page**.

2 Presentation of the challenge

The challenge here is proposed by SNCF, the France's nationale state-owned railway company. In particular, they operate on most of the urban railway lines in the suburbs of Paris, allowing for about 3 millions of passengers to commute everyday with a park of 6200 trains. With a increased amount of daily commuters every year (about 6% between 2015 and 2019), the SNCF data scientists aim at anticipating this rise through the years.

In the context of the challenge, we have data that contains the number of daily validations per station between January 1st 2015 and December 31st 2022. The variable to forecast is the number of daily validations per station between January 1st 2023 and June 30th 2023. That is, we have a multivariate time series forecasting problem. Indeed, we have the two following variables :

- **date** : the date of the validations (format YYYY-MM-DD);
- **station** : the encoding of a station with 3 characters.

We are also provided with other feature variables :

- **job** : this variable describes whether the date is a business day or not (0 or 1);
- **ferie** : this variable describes whether the date corresponds to a public holiday or not (0 or 1);
- **vacances** : this variable describes whether the date lies into a holiday period (0 or 1).

The dataset is made of three CSV files :

- **X_train** : 1 237 971 entries corresponding to a combination date x station between January 1st 2015 and December 31st 2022 (2922 days).
- **Y_train** : same number of entries, but two columns : "index" and "y" which denotes the number of validations on a day for a certain station.
- **X_test** : 78 652 entries corresponding to the period between January 1st 2023 and June 30th 2023 (181 days).

For the challenge, the goal was to generate a CSV file **Y_test** with the expected number of validations per station on each day of our Machine Learning model in the first half of 2023. The metric of evaluation for the submitted CSV file is the **mean absolute percentage error (MAPE) [5]**, that is, given n is the number of couples date x station in our dataset :

$$MAPE(y, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \tilde{y}_i|}{y_i}$$

Where y are the ground truth values and $\tilde{y}$ are the predictions given by the model.

3 Related work

This challenge of time-series forecasting is not new and many techniques and reviews exist in the literature that will help us to design our models. Before starting the challenge, we made an extensive review of existing methods so we will be able to decide which one is the best.

3.1 Machine Learning models

First, some reviews present **machine learning** methods. At the beginning of our work, we looked for the most famous models for the specific task of predicting the number of validations in public transport. We punctually extended our research to different public transportation modalities (car, bus, subway...). Li et al. [4] based their approach on a **multivariate linear regression** model. Ensemble methods are also quite popular due to their scalability, as Ding et al. [6] who developed a model which uses **Gradient Boosting Decision Trees**, or Toqué et al. [14] who combined **Random Forests** with LSTMs (see below) for short and long-term passenger flow forecasting.

3.2 Statistical models

Some statistical models are employed in forecasting tasks, such as **ARIMA** and in particular **SARIMA** which takes into account the seasonal aspect, as mentionned in [12]. Such models are used a lot in the literature, especially since it generalizes the ARMA model to non-stationary time series. These models can be augmented by adding exogenous variables to measure their influence towards the target variable, with extends (S)ARIMA to (S)ARIMAX.

3.3 Deep Learning models

In the reviews and paper that we looked at [10], it is often the same architecture that is presented. The most important one seems to be **LSTMs (Long Short-Term Memory)** as they capture temporal dependence. This technique is therefore very interesting to study in the context of the data challenge. Wang et al. [16] combine this long short-term characteristic with **Convolutional Neural Networks (CNNs)** to further extract the characteristics of the sampled data, and improve the accuracy of the model.

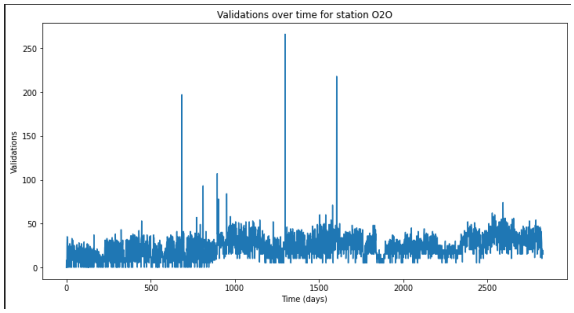
3.4 Other interesting frameworks

An interesting framework has been the **Prophet project** by facebook [13]. They claim to have developed a library that can forecast at scale time-series very easily. Finally, an other library has the same characteristics and it is **SkTime** [11].

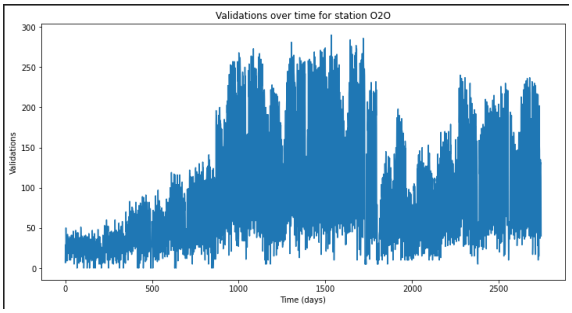
4 Data exploration

4.1 Attendance evolution through time

We remind that we have a **multivariate time series forecasting problem**. First, we make a visualization of some of our data by plotting train station attendance throughout time for two distinct stations in Fig 1a and in Fig 1b.
Some of these plots already evidence that some stations have more daily validations than others as shown in Fig 2. Some stations have the data only for one year (e.g 3) therefore we could guess that the forecast for these stations will not be as accurate as the other ones.



(a) Validations for the station O2O



(b) Validations for the station 1J7

Figure 1: Validations for two different stations

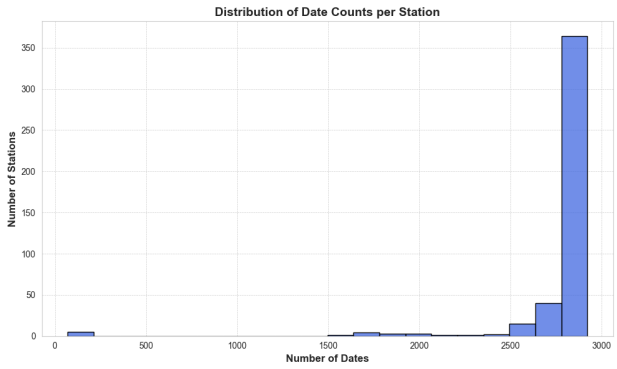


Figure 2: Distribution of date available per stations.

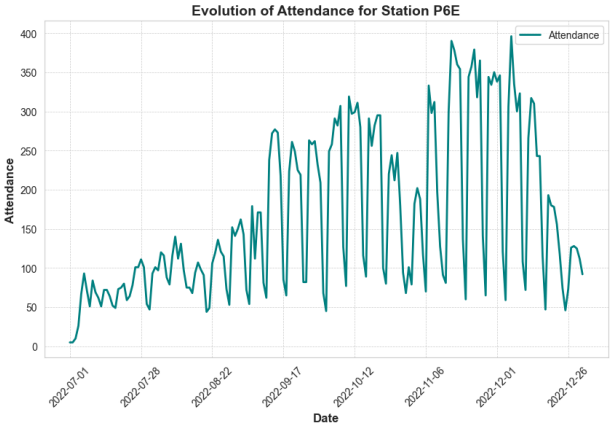


Figure 3: Attendance evolution for the station P6E

Therefore, the main hypothesis we made on the problem is the following: **each station is independent from the others**. That is, we run independent machine learning models for each station, each model trying to capture intrinsic characteristics for each time series given by a station. This hypothesis makes it somewhat easy to process each station, however, it is not quite realistic, since the amount of passengers at a certain station is going to impact the following ones.

4.2 Distribution of holidays, works & bank holidays

A major point to look at is the distribution of the holidays/work days throughout the year. Indeed, we can legitimately think that when it is during a holiday period the train station attendance will be lower than during working days. This trend is assessed in Fig 4 and in Fig 5. It can be further explored by zooming on a month where there is a real gap between week and week-end (as shown in Fig 6).

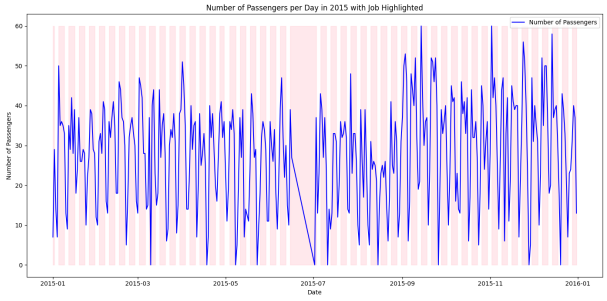


Figure 4: Evolution of the train attendance with the job highlighted

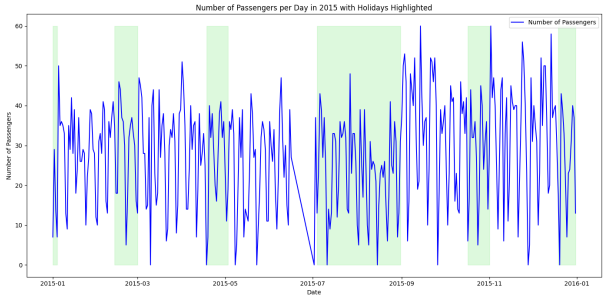


Figure 5: Evolution of the train attendance with the holidays highlighted

When plotting the distribution of each class for each day we visualize these two figures (Fig 7 & Fig 8). What we can conclude from it is that the working days are the majority but holidays for example represents nearly one third of the year meaning it is not an artefact but a component that will be taken into account in our models.

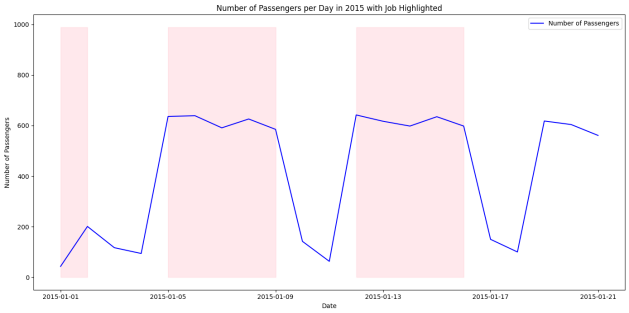


Figure 6: Evolution of the attendance during three weeks, working days are highlighted

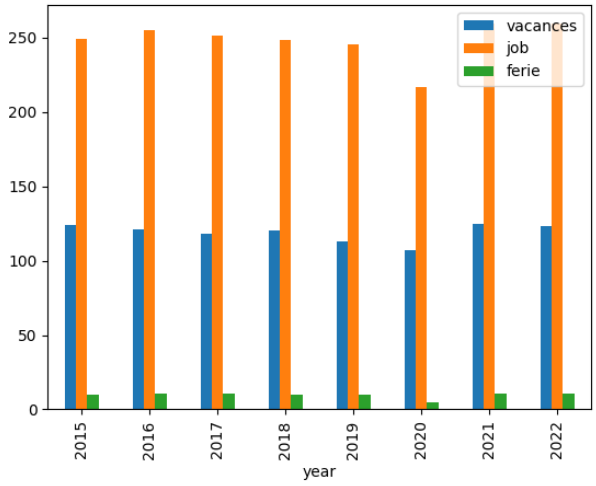


Figure 7: Distribution of the number of ferie / holidays / job

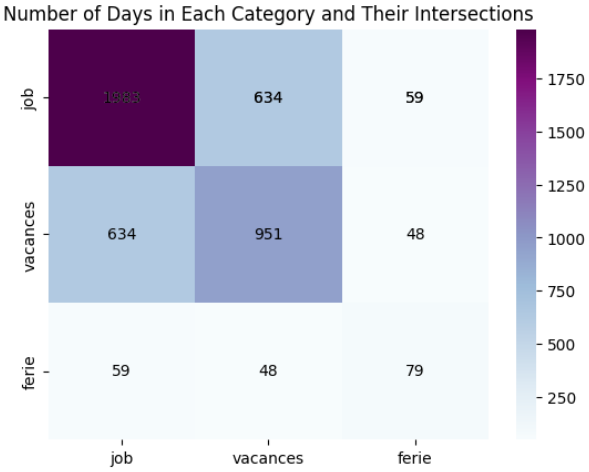


Figure 8: Overlap Matrix of the job / holidays / ferie

5 Feature selection

Starting from the original dataset, we are provided with three features : **job**, **ferie** and **vacances**. However, we estimated that these features were not sufficient for our model, and that some augmentation was required.

We know that we the data is provided with at a daily frequency. Our idea was to find a way to **encode the cyclical features** that could be present in our data (for example, a periodic pattern that could be learned by our models, over a week, a month...). This idea was inspired by the following blog [2]. For the challenge, we only considered the "week" level of seasonality. For this level, we proceeded with a two-dimensional transformation by computing a cosine and sine transformation of our data :

$$d_{cos} = cos\left(\frac{2\pi(d-1)}{7}\right)$$

$$d_{sin} = sin\left(\frac{2\pi(d-1)}{7}\right)$$

Where d is the number in $\{0, ..., 6\}$ which encodes the day of the week (e.g Monday is 0, Tuesday is 1...). These data were added to our dataset to capture recurring patterns in the time series and thus, potentially enhance the predictive power of our models.

6 Machine Learning methods

6.1 Linear Regression and Elastic-Net

The first fundamental model that emerged for our forecasting challenge is **linear regression**, a famous technique widely employed in predictive modeling tasks [4]. In multivariate linear regression, the goal is to establish a linear relationship between the input features and the target variable. **Elastic-Net**, an extension of

linear regression, combines the penalties of both **Lasso (L1)** and **Ridge (L2)** regularization methods, providing a balance between feature selection and coefficient shrinkage. This technique proves particularly advantageous when dealing with datasets featuring multicollinearity among predictors, as it helps mitigate overfitting while promoting model sparsity. Our inclination towards employing linear regression and Elastic-Net stemmed from their interpretability, ease of implementation, and effectiveness in capturing linear relationships within the data.

Let $X = (X_1, ..., X_n)$ denote the training features and $Y = (y_1, ..., y_n)$ represent the corresponding training targets (in our case the number of validations per station on each day). In multivariate linear regression, the algorithm aims to find the coefficients $\beta_0, \beta_1, ..., \beta_n$ that minimize the residual sum of squares (RSS) between the predicted and actual values. Mathematically, this can be represented as:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$$

Where $\hat{Y}$ is the predicted target variable. The model parameters are typically estimated using ordinary least squares (OLS) or gradient descent optimization techniques. Elastic-Net introduces an additional regularization term to the linear regression objective function, combining both L1 and L2 penalties:

$$\min_{\beta} \left(\text{RSS} + \lambda_1 \sum_{j=1}^{n} |\beta_j| + \lambda_2 \sum_{j=1}^{n} \beta_j^2 \right)$$

Here, λ_1 and λ_2 control the strength of the L1 and L2 penalties, respectively, offering a flexible approach to feature selection and model regularization. By tuning these hyperparameters appropriately, Elastic-Net facilitates robust and interpretable predictions, making it a valuable tool in our pursuit of accurate forecasting models.

6.2 Random Forest and Bagging

The second type of models that came up to our head when we began this challenge are the ones that fit into **ensemble learning**. In particular, **Random Forest** is an ensemble learning technique used in machine learning for classification and regression tasks. It is based on the **bagging** method, short for bootstrap aggregating, which involves training multiple models on different subsets of the training data and then combining their predictions to obtain a final output. Random Forest, a particular case of bagging, further decorrelates the models by introducing randomness in the feature selection process, and outputs the mean prediction of the individual trees (since we are in a regression task). It thereby enhances the robustness and accuracy of the predictions.

Our choice of using Random Forest in the first place was motivated by the literature (cf. [14]), where most papers mentioned it as a robust and scalable method in the domain of time series forecasting, and subsequently traffic flow prediction problems.

Let $X = (X_1, ..., X_n)$ be the training features and $Y = (y_1, ..., y_n)$ be the training target. A Random Forest regression algorithm is the following :

- Sample with replacement p elements from the training set (X, Y) (with $p < n$) : we call them X_p and Y_p.
- We train a decision tree ϕ_p on X_p and y_p and get a prediction $\hat{y}_p$.
- We repeat this process B times (B is a hyperparameter, the number of trees).
- After this, the random forest computes the predictions from the test set $X' = (x'_1, ..., x'_n)$ by computing their mean prediction :

$$\hat{y} = \frac{1}{B} \sum_{i=1}^{B} \phi_p(x'_i)$$

6.3 Gradient Boosting

Another significant technique widely used in machine learning for regression and specifically time series forecasting is **Gradient Boosting**. Compared to Random Forest which builds multiple decision trees independently and combines their outputs, Gradient Boosting builds **weak learners** (generally decision trees) sequentially, with each tree learning from the errors made by the previous one.

Specifically, Gradient Boosting optimizes a loss function by adding these trees to the ensemble iteratively. Each new tree is trained to predict the residual errors of the ensemble's current prediction, effectively reducing the

overall error. The final prediction is obtained by aggregating the predictions of all trees in the ensemble. This iterative process gradually improves the performance of gradient boosting, and makes more accurate predictions. Similar to Random Forest, Gradient Boosting has been successfully applied to time series forecasting tasks in the literature, demonstrating its effectiveness in capturing temporal dependencies and patterns in data (cf. [6]).

The training process of a Gradient Boosting regression algorithm can be outlined as follows:

- Initialize the model with a constant value, often the mean of the target variable:

$$\hat{f}_0(x) = \text{argmin}_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$

  where L is the chosen loss function, y_i are the true target values, and γ is the constant value.

- For each iteration $m = 1$ to M, where M is the total number of iterations:
  - Compute the pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, \hat{f}_{m-1}(x_i))}{\partial \hat{f}_{m-1}(x_i)}\right]$$

  - Fit a weak learner, such as a decision tree, to the pseudo-residuals:

$$\phi_m(x) = \text{argmin}_\phi \sum_{i=1}^{n} \Big(L(y_i, \hat{f}_{m-1}(x_i) + \phi(x_i))\Big)$$

  - Update the model by adding the contribution of the new weak learner with a learning rate η:

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \eta\phi_m(x)$$

- Compute the final prediction by summing the predictions of all weak learners:

$$\hat{Y} = \sum_{m=1}^{M} \eta\phi_m(x)$$

6.4 Recurrent Neural Networks and LSTMs

Recurrent Neural Networks (RNNs) and **Long Short-Term Memory networks (LSTMs)** are both powerful architectures designed for handling sequential data. RNNs maintain an internal state that allows them to process sequences of inputs, making them suitable for tasks such as time series prediction.

However, traditional RNNs suffer from the **vanishing gradient** problem, where gradients diminish exponentially over time during backpropagation, leading to difficulties in learning long-term dependencies. LSTMs were introduced to mitigate this issue by incorporating memory cells and gating mechanisms, enabling them to selectively retain or forget information over time. The LSTM cell consists of several gates, including an input gate, a forget gate, and an output gate, controlled by sigmoid activation functions, along with a memory cell updated by a tanh activation function. The equations governing the behavior of an LSTM cell are as follows, for some time t:

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}$$

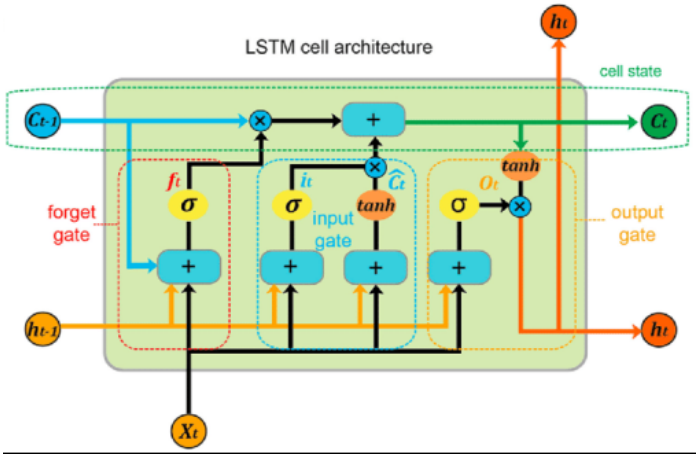


Figure 9: Cell of a LSTM (retrieved from [7])

Regarding the training phase of the LSTM, we beforehand wrote a function that creates sequences from a given dataset as input. Each sequence consists of a window of "past" data points (X), and the corresponding "future" data points (y). From these sequences, we can build our dataset, and pass batches of sequences to the LSTM so it can predict some values than can be compare to the "future" ones. We made use of Adam optimizer, with a learning rate of 0.001, and compared the outputs using the L^1 loss.

The LSTM networks suit well for our case of time series prediction, by their ability to capture and maintain long-term information as well as short-term ones. They are very often mentioned in the literature for the specific task of traffic flow prediction, and match with the sequential aspect of our data (cf. [16], [14]). Indeed, the number of validations per station on each day form multiple time series, which allowed us to train a LSTM for each station to capture the trend and the eventual seasonality on each station. This model was somewhat the core of our approach through the challenge.

6.5 Autoregressive Models: SARIMAX

Seasonal Autoregressive Integrated Moving Average with Exogenous Variables, commonly known as SARIMAX, is a widely used statistical model in time series analysis and forecasting. SARIMAX extends the traditional **ARIMA** model by **incorporating external predictors** or **exogenous variables**, as well as a **seasonal component**. It captures the temporal dependencies within the time series data while also considering the influence of external factors on the variable of interest. The SARIMAX model is defined by its parameters $(p, d, q) \times (P, D, Q)_s$, where p, d, and q represent the autoregressive, differencing, and moving average orders, respectively, and $(P, D, Q)_s$ denote the seasonal autoregressive, seasonal differencing, and seasonal moving average orders with period s. Additionally, SARIMAX includes exogenous variables denoted by X_t, which are integrated into the model to improve forecasting accuracy. The prediction equations for SARIMAX models are as follows:

$$
\begin{aligned}
&\text{Autoregressive (AR) component:} && \hat{Y}_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \\
&\text{Moving Average (MA) component:} && \quad + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \\
&\text{Seasonal Autoregressive (SAR) component:} && \quad + \Phi_1 Y_{t-s} + \Phi_2 Y_{t-2s} + \cdots + \Phi_P Y_{t-Ps} \\
&\text{Seasonal Moving Average (SMA) component:} && \quad + \Theta_1 \varepsilon_{t-s} + \Theta_2 \varepsilon_{t-2s} + \cdots + \Theta_Q \varepsilon_{t-Qs} \\
&\text{Exogenous (X) variables:} && \quad + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} \\
&\text{Error term:} && \quad + \varepsilon_t
\end{aligned}
$$

where Y_t is the observed value at time t, $\hat{Y}_t$ is the forecasted value, ε_t is the error term, and c represents a constant term. The coefficients ϕ_i, θ_i, Φ_i, Θ_i, and β_i are estimated from the data during model fitting.
SARIMAX models provide a flexible framework for time series forecasting, accommodating both the inherent dynamics of the series and the impact of external variables. In our case, the exogenous variables where **job**, **ferie** and **vacances**. However, despite trying different combinations of hyperparameters, it ended up not performing very well on the leaderboard, so we discarded it pretty quick.

7 Results

7.1 Random Forest Regressor

First, we looked at the performance of our models on validation set (validation split of 0.1 or 0.2). For example in Fig 10, Fig 11 and in Fig 12, we observe the regression results for the random forest regressor. MAPE score with this technique on the validation set 0.375 on the station ”1J7”. Furthermore, we observe that the trend seems to behave correctly even if the attendance is underestimated.

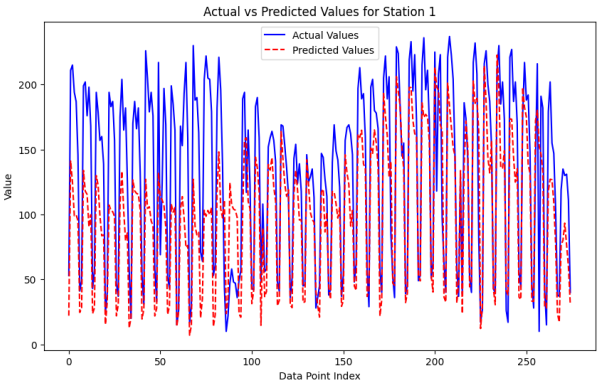


Figure 10: Actual vs Predicted values on the validation set for the RandomForestRegressor

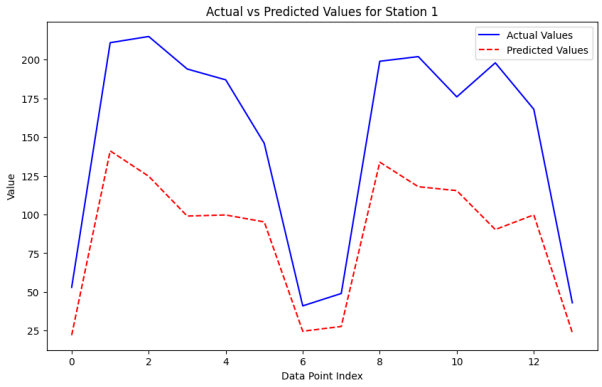


Figure 11: Actual vs Predicted values for two weeks on the validation set for the RandomForestRegressor

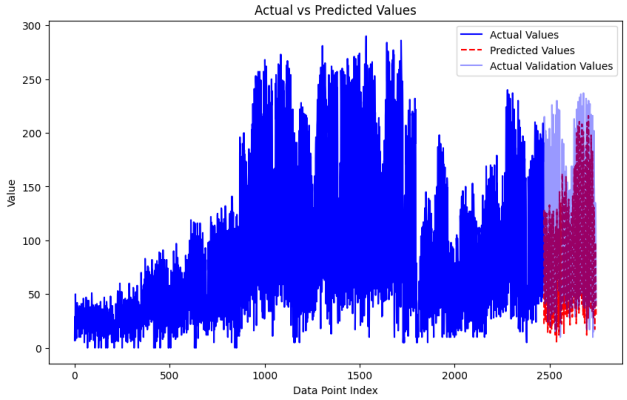


Figure 12: Training and Predicted values on both training and validation set

7.2 LSTM

Then results were looked with a LSTM network. Result on the validation set are shown in Fig 13, Fig 14 and in Fig 15. We observe that the trend seems to be well followed and that the magnitude is close to the reality. However, as soon as the actual values become high there is a high magnitude difference and it seems that it is more a mean than anything else. This trend is therefore confirmed as the MAPE score of the validation set is 0.435 on the station ”1J7”.

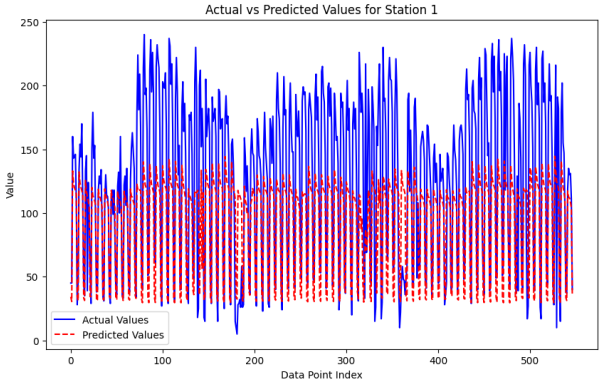


Figure 13: Actual vs Predicted values on the validation set for the RandomForestRegressor

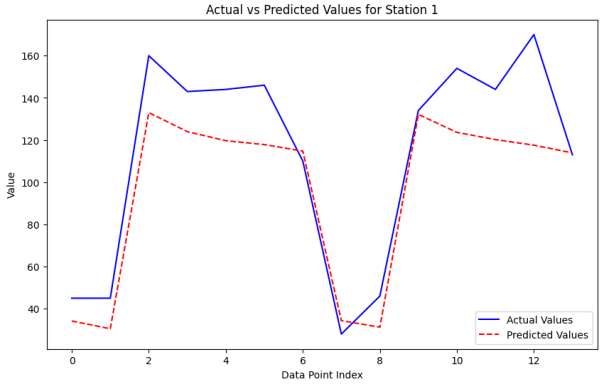


Figure 14: Actual vs Predicted values for two weeks on the validation set for the RandomForestRegressor

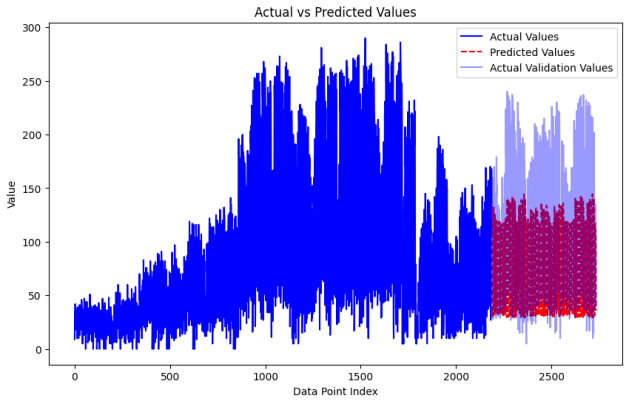


Figure 15: Training and Predicted values on both training and validation set

7.3 Prediction in 2023

In Fig 16 there is the predicted values for 2023 with XGBoost regressor and in Fig 17 is the predicted values for 2023 with a fully connected network (only dense layer). What we observe is quite strange. Indeed, the predicted values with XGBoost seems to follow the trend of the previous years whereas fully connected results only predict the same value around 100. But the most surprising being that when submitting fully connected layers results we get the best results on the public dataset (97) whereas with XGBOOST performs around 200. This unexpected result will be further discussed.

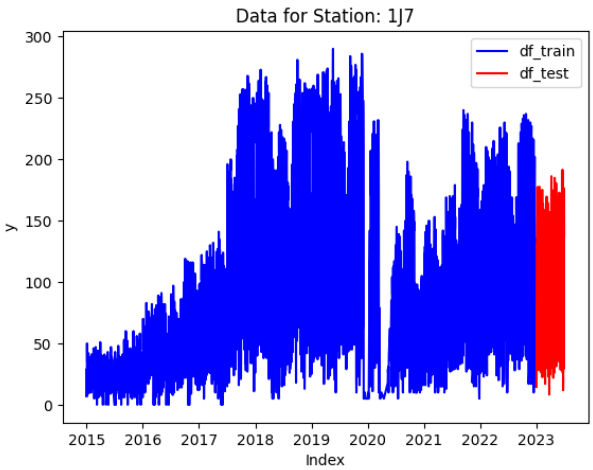


Figure 16: Predicted values for 2023 with XGBOOST

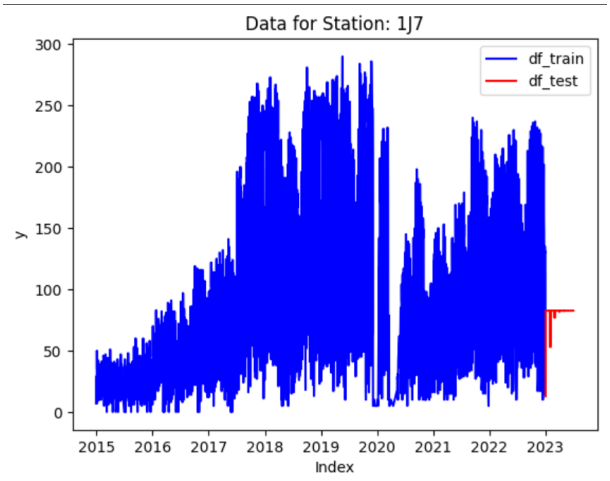


Figure 17: Predicted values for 2023 with Fully Connected network

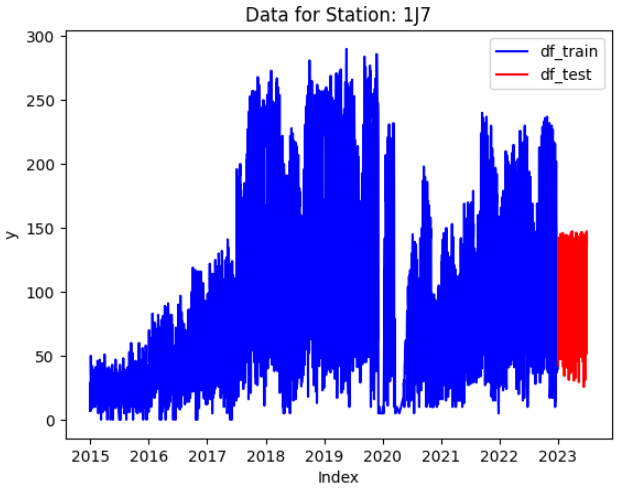


Figure 18: Predicted values for 2023 with SARIMAX

Method	Score
Fully Connected	97,09151732495503
LSTM	158,39529272034247
RF_XG_SVR	179,5819200648702
Elastic-Net	196,82225037110496
SARIMAX	242,9657721334913
SVR	285,7352720941828
Prophet	355,8057465828663

Table 1: Comparison of different methods and their scores.

8 Discussion

8.1 The MAPE metric

We had a reflexion about the relevance of the evaluation metric : the **mean absolute percentage error**. While it is a very simplistic measure of the error in percentage of our model, and it allows simple comparison between models at various scales, it has many limitations. The first one is that it is **very sensitive to the values that are close to zero**, or exactly equal to zero. However, in our data exploration phase, we found out that some ground-truth values are very close to zero on some days and some stations. That makes it difficult to evaluate whether our model performs well or not, when there is a risk of division by zero, leading to undefined quantities or artificially large MAPE values.

A second limitation of this metric is that it is **not symmetrical**. Indeed, the MAPE treats overestimations and underestimations differently, and favors models that underforecast than those who overforecast.

All in all, despite its easy-to-compute aspect and its interpretability, we think that it has many drawbacks that forces to consider alternative metrics, such as the **mean absolute error** or the **(root) mean squared error**. There is also the **symmetric mean absolute percentage error**, which corrects the lack of symmetry of the MAPE, or the **mean absolute scaled error**.

8.2 The ”Covid” hole

We have noticed on most of the stations that in 2020 for several months there was a drop in attendance because of the COVID pandemic (e.g in Fig 19).
This drop might explain why some predictions are not as good as others and that maybe if we try to predict behavior without these artifacts we might perform better. However, this trend reminds us that nothing can be predicted perfectly and that sometimes exogenous data might lead to the destruction of our algorithm. It is one caution with machine learning methods.

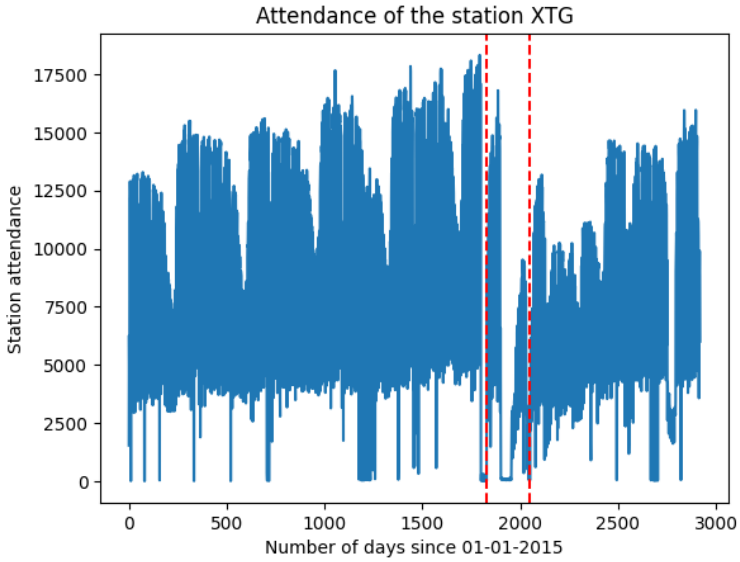


Figure 19: Attendance of the station XTG with the covid gap

9 Outlooks

In the furnished literature on time-series forecasting, we identified interesting methods that might be worth it to explore to increase our forecasting performance.

9.1 Time-Series Feature Extraction

During our work, we stumbled across some libraries that can extract additional features from the time-series such as **Time-Series Feature Extraction Library (TSFEL)** [1] that would allow us to extract additional features from our time-series so there is more data the model can rely on to do its prediction. Some features are for example; the mean value, the standard deviation, or the maximum frequency (there are more than 80 !). If it is a good idea it might also include noise in the model and then provide the wrong expected behavior. Other relevant libraries that do similar work are **Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests** [3], or the **Cesium Library**.

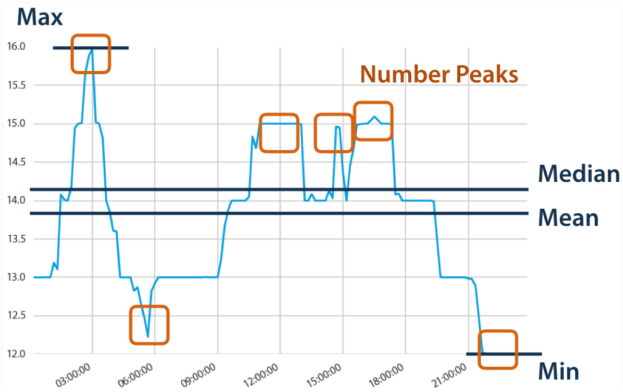


Figure 20: Examples of some features that can be extracted with the TSFRESH library. Retrieved from [3].

9.2 Transformer models

Recent techniques leverage from the attention model and the **Transformer** models [15]. Some of the promising models are Temporal Fusion Transformers (TFT) [9] or Informer [17]. The Informer forecasting architecture is shown in Fig 21.

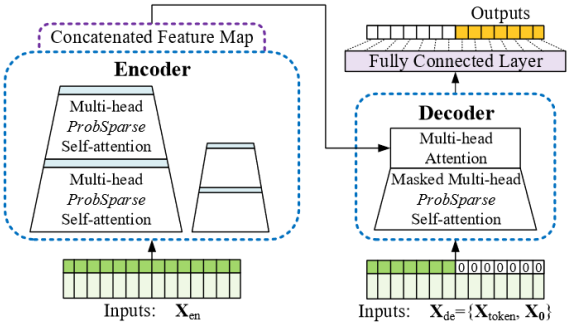


Figure 21: Network used by Informer based on self-attention mechanism. Retrieved from [17].

9.3 Lagged features

An other interesting track to further improve our results would be to use lagged features in the training. Meaning that you add in the training set information about future behavior (like the attendance in one week on the same day). It would certainly help to learn better trends [8].

10 Conclusion

All in all, what we conclude from this challenge is that forecasting time series is not an easy task. There are **many parameters** to take into account and huge efforts were made to incorporate exogenous information into our models. We appreciate it as we had a scientific method to start with basic models and finally finish with more complex ones. Our experiments have revealed good results as **we do not overfit** between the public and the private leaderboard, even improving on the private leaderboard (cf. Fig 23 compared to the public one, Fig 22).

6	24 février 2024 17:10	Talexandre150 & Matteo_Marengo	97,0915

Figure 22: Public academic final ranking on the data challenge

4	24 février 2024 17:10	Talexandre150 & Matteo_Marengo	97,3480

Figure 23: Private academic final ranking on the data challenge

References

[1] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[2] Pierre-Louis Bescond. Cyclical features encoding, it's about time! 2020.

[3] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[4] L.I. Dahui. Predicting short-term traffic flow in urban based on multivariate linear regression model. *Journal of Intelligent Fuzzy Systems*, 39:1–11, 06 2020.

[5] Arnaud de Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, June 2016.

[6] Chuan Ding, Donggen Wang, Xiaolei Ma, and Haiying Li. Predicting short-term subway ridership and prioritizing its influential factors using gradient boosting decision trees. *Sustainability*, 8:1100, 10 2016.

[7] Rui Kang, Bosoon Park, Qin Ouyang, and Ni Ren. Rapid identification of foodborne bacteria with hyperspectral microscopic imaging and artificial intelligent classification algorithms. *Food Control*, 130:108379, 06 2021.

[8] Scikit Learn. Lagged features for time series forecasting. 2024.

[9] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020.

[10] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, February 2021.

[11] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. sktime: A unified interface for machine learning with time series, 2019.

[12] Milos Milenkovic, Libor Svadlenka, Vlastimil Melichar, Nebojsa Bojovic, and Zoran Avramovic. Sarima modelling approach for railway passenger flow forecasting. *Transport*, 33:1–8, 10 2015.

[13] Letham B. Taylor SJ. Forecasting at scale. *PeerJ Preprints*, 2017.

[14] Florian Toqué, Mostepha Khouadjia, Etienne Come, Martin Trepanier, and Latifa Oukhellou. Short long term forecasting of multimodal transport passenger flows with machine learning methods. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 560–566, 2017.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[16] Yu Wang, Zhifei Wang, Hongye Wang, Junfeng Zhang, and Ruilong Feng. Prediction of passenger flow based on cnn-lstm hybrid model. *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, pages 132–135, 2019.

[17] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.