HUB Trentino Innovazione

FINAL REPORT OVER THE INDUSTRIAL AI CHALLENGE EXPERIENCE:
Prevision of the water flow at the inlet of the Stenico purification plant

Participants:
Alessandro Assirelli
Luca Del Giudice
Andrea Di Luca
Mattia Sartori
Marcus Vukojevic

Mentors:
Luca Di Persio
Stefano Di Persio

Client company and representative:
NIRIS™, Andrea Turso

Academic Year 2021-2022

Summary

1 The Industrial AI Challenge

The following work has been realized in the context of the ”Industrial AI Challenge” organized by HUB Innovazione Trentino with the collaboration of the University of Trento. The objective of the challenge is to perform the optimization of an industrial process with machine learning - oriented methods, which would be applied starting from the analysis of an already existing database of the aforesaid industrial process. The present report will therefore discuss in detail the steps that have been implemented to reach the final result and what ideas have been taken into consideration to deliver the custumer a predictive model which would be efficient and reliable at the same time.

The specific challenge faced by the team, was provided by NIRIS™, a company specialized in monitoring systems which currently manages the safety and control systems of more than 1400 water treatment plants in the Garda area and in Trentino region. The aim of the proposed challenge is to detect correlations between historical data and the data of anthropic and meteorological impact on water treatment plants in order to implement electrical energy cost-saving procedures and predictive maintenance. The team could rely on the help of the mentors at HPA™(High Performance Analytics). The first step for the team, the mentors and the company, was to restrict the scenario to a single treatment plant in order to better understand the problems and define feasible objectives. Hence, the Stenico treating plant was chosen as case of study.

1.1 The Stenico water purification plant and the task of NIRIS™

In order to get an idea of how the industrial process worked, the team decided to pay the plant a visit. This was a crucial step since it helped the team understand the meaning behind the variables the database is made of. Being the Stenico one a very complex and articulated plant, a quick description of its main features will be given so to grasp the main functions of this industrial process.
The plant itself can be divided into two areas, outer and inner respectively. The outer part is dedicated to collecting in a single pipe the water conveyed by the 4 collectors that arrive to the plant. This area is used as a degraveler, a compartment that has the purpose of retaining heavy debris. On the other hand, the inner area lodges machinery, tanks and filters used for the actual purification. The wastewater undergoes two further physical pre-treatments, consisting of fine screening for the separation of suspended solids and de-oiling-desanding for the separation of fine sands, oils and greases. Subsequently to the de-oiling-sand removal sector, the wastewater is then divided into 3 different treatment lines (Figure 1) where the following purification phases take place:

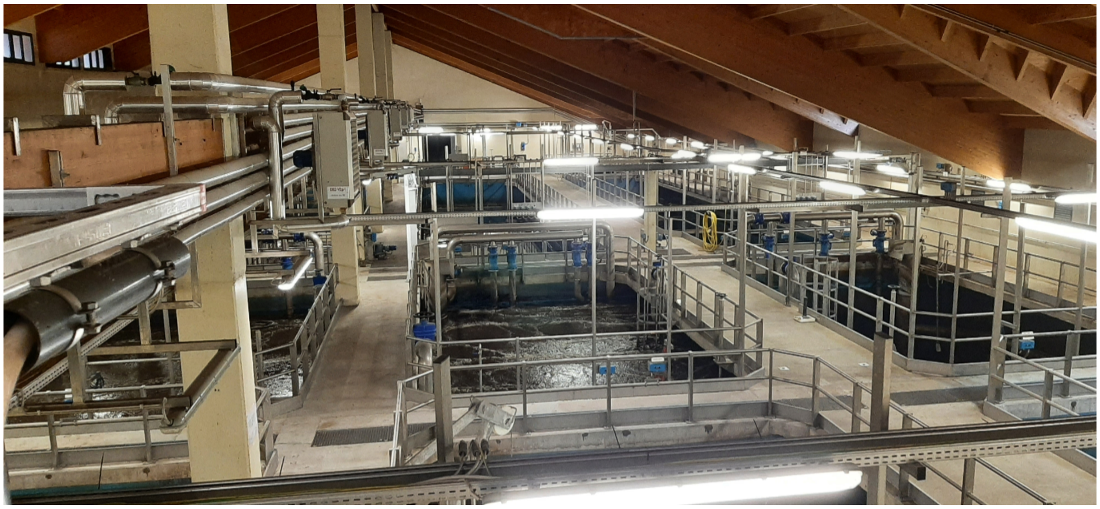


Figure 1: An overview of the 3 lines

- In a first tank denitrification takes place, which is a biological process in an anaerobic environment through which specific bacteria carry out the transformation of nitrates into gaseous nitrogen.
- In a second tank nitrification takes place, which is a biological process in an aerobic environment through which specific bacteria carry out the oxidation of ammonia nitrogen to nitrates, a

molecule that is harmless for aquatic environments. Compressors provide for the delivery of air blown into the bottom of the tanks with silicone membranes and distributed in micro-bubbles in the mixture.

- In a third tank, the sedimentation of the biological sludge takes place (the bacteria mentioned above) with consequent reintroduction into the first tank thus implementing a partial re-circulation.
- In a fourth tank, the ”surplus” mud thickens, i.e. the sludge to be removed from the system to maintain balanced the concentration in the denitrification and nitrification tanks, with respect to the biomass growth that occurs with biological processes.
- In a fifth tank the ”stabilization” takes place, a specific mineralization treatment for the ”excess” biological sludge residuals.
- Finally, in a last tank the ”final filtration” takes place for the recovery of suspended solid particles that the sedimentation phase has possibly failed to retain.

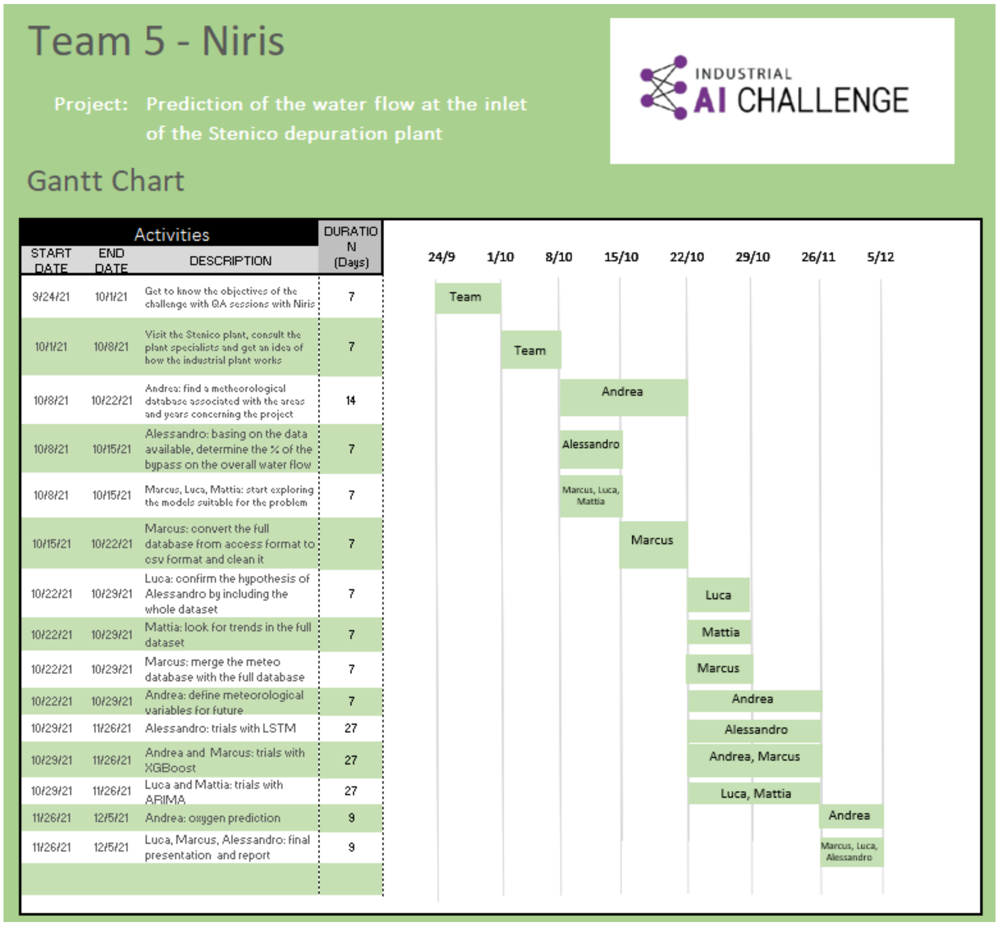
The deep interconnection between all the phases of this chemical process implies the existence of a very effective control strategy that would deal with malfunctions, critical meteorological events and maintenance of all the machines. It is within these regards that NIRIS™was tasked to improve the aforesaid control strategy by creating an additional variable which would predict the inlet flow at the entrance of the plant given real time meteorological information about the collectors that surround the purification plant. This variable is meant to detect future criticalities that would help the plant staff to plan maintenance and, ideally, to better manage the setting of the power of the compressors so to cut on expenditures.

1.2 Setting up the challenge

Given the complexity of the industrial process and of the objectives set, scheduling was believed of paramount importance: according to the different fields of expertise of the team members, tasks were assigned and a Gantt chart was compiled and weekly adjourned in order to keep track of all the efforts and activities yet to come (Figure 2).

Moreover, two meetings were planned to be held weekly: the former within the team members (in order to let all members keep up with all the updates going on and help each other), the latter also with the customer, Andrea Turso, and the mentor, Luca Di Persio, in order to decide the steps for the next week. In particular, Andrea focused on providing the team with all the needed data and contacts about the Stenico personnel, while Luca supported the team with bibliographical references and advice whenever some doubts occurred.

As far as the software part is concerned, the team agreed to work jointly with Python in order to facilitate import/export operations of the code pieces, which were uploaded on GitHub. A MS Azure virtual machine was used to run the algorithms and to store the database.



Team 5 - Niris

Project: Prediction of the water flow at the inlet of the Stenico depuration plant

INDUSTRIAL AI CHALLENGE

Gantt Chart

Activities			DURATION (Days)
START DATE	END DATE	DESCRIPTION	
9/24/21	10/1/21	Get to know the objectives of the challenge with QA sessions with Niris	7
10/1/21	10/8/21	Visit the Stenico plant, consult the plant specialists and get an idea of how the industrial plant works	7
10/8/21	10/22/21	Andrea: find a metheorological database associated with the areas and years concerning the project	14
10/8/21	10/15/21	Alessandro: basing on the data available, determine the % of the bypass on the overall water flow	7
10/8/21	10/15/21	Marcus, Luca, Mattia: start exploring the models suitable for the problem	7
10/15/21	10/22/21	Marcus: convert the full database from access format to csv format and clean it	7
10/22/21	10/29/21	Luca: confirm the hypothesis of Alessandro by including the whole dataset	7
10/22/21	10/29/21	Mattia: look for trends in the full dataset	7
10/22/21	10/29/21	Marcus: merge the meteo database with the full database	7
10/22/21	10/29/21	Andrea: define meteorological variables for future	7
10/29/21	11/26/21	Alessandro: trials with LSTM	27
10/29/21	11/26/21	Andrea and Marcus: trials with XGBoost	27
10/29/21	11/26/21	Luca and Mattia: trials with ARIMA	27
11/26/21	12/5/21	Andrea: oxygen prediction	9
11/26/21	12/5/21	Luca, Marcus, Alessandro: final presentation and report	9

Figure 2: Gantt chart of all the activities

2 A deeper focus on the structure of the database

2.1 The initial database

At first, the team was given an initial dataset made of measurements taken during the year 2020 of all the sensors contained in the Stenico water-treatment plant (this due to an impossibility, by the company representative, to gather all the data at once). All the sensors were taking measurements every two seconds and, every five minutes, the database was adjourned with the minimum, maximum and mean value concerning this period. Given that the data was coming from real world sensors and could be subject to errors, the database was first cleaned from outliers so that the best timeseries possible could be fed into the future models. To do so, the team searched values which were three standard deviations above the mean and negative values. Such values were corrected by replacing them with the mean of the previous five measurements to avoid random drops in the database's values.

2.2 Weather data

Extreme raining events close to the Stenico water-treatment plant can cause increases in sewage inlet flow rate. Indeed, rainwater can reach the Provincial sewage collector and dilute the organic material contained in the sewage. Looking at the topography map of the lands around the purifier, shown in Figure 3, 10 weather stations were selected (see Table 1). Each station collects rainfall measurements expressed in millimeters (mm) of water collected with a frequency of 5 minutes.

Measurements are provided with a data quality flag. A complete list of quality flags is reported in Table 2. Missing data (code 255) are replaced with a forward fill strategy which consist in propagating last valid observation forward. As a first approximation, uncertain (code 140) and not validated measurements (code 145) were also replaced with a forward fill approach.

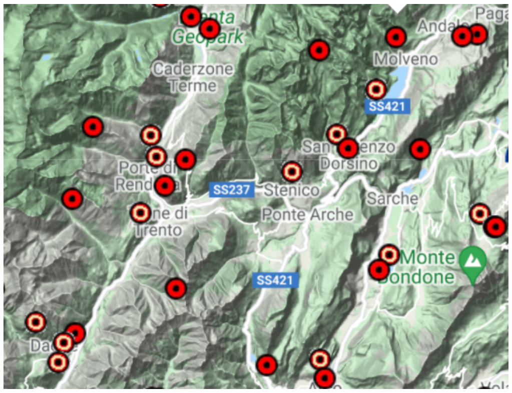


Figure 3: Map reporting weather stations around Stenico water treatment plant. Red circles represent active station. White circles represent non active stations (e.g. Stenico weather station was active until 31/12/2001)[1]

Name	Code	Active since
San Lorenzo in Banale	T0414	2007
Tione	T0179	1985
Pinzolo	T0175	1991
Val Di Breguzzo	T0177	1991
Villa Rendena	T0411	2011
Val Dambiez	T0413	2011
Zuclo	T0412	2011
Montagne	T0182	1990
San Lorenzo In Banale	T0414	2010
Giustino	T0426	2011

Table 1: Weather station downloaded from MeteoTrentino open-database.

Code	Meaning
1	Good
90	Snow (delayed)
140	Uncertain data [station malfunctioning]
145	Not validated
255	No data

Table 2: Rainfall measurement quality codes.

Assuming that rainwater has to travel before reaching the water-treatment plant, new variables were defined which would account for rainfall measurements integrated over different time windows. Table 3 shows a list of the 8 variables calculated.

Variable Code	Time Window
RM10T	10 min
RM30T	30 min
RM1H	1 hour
RM2H	2 hours
RM4H	4 hours
RM10H	10 hours
RM20H	20 hours
RM2D	2 days

Table 3: Rainfall measurement quality codes.

2.3 The full dataset and its conversion from .mdb to .csv format

To integrate the first part of the data we received a dump of a Microsoft Access database which weighted over 140 Gigabytes. The database contained the information and measurements regarding all the water treatment plants in the Trentino region. The period that is covered is from 2011 to 2019. It wasn't possible to directly use the information stored because of the .mdb file format. So, a bash script was built (file: importare_db.sh) to automatically extract and efficiently convert in .csv format all the files in the folder. Each .mdb file contained around 1500 .csv files. After that, the files which contained the measurements about the Stenico water treatment plant were selected. It turned out that Stenico started collecting data just from mid-2013 and thus more than two years of data is missing. All the useful information was merged in one .csv file and was applied all the data cleaning processes that were described in sections 2.1 and 2.2 in order to remove outliers and have the cleanest database possible to feed the future models. At the end of the process the data extracted from the files weighed around 400 Megabytes (file name Stenico_Integrale.csv).

2.4 The final dataset

At this point, the data about the flow rate and the weather data were needed to be merged. Among all the sensors contained in Stenico, the team could not dispose of a sensor that recorded the inlet flow. To tackle this problem, it was decided to take all the measures concerning sensor "CL-SQm-1", devoted to the measurement of the outlet flow, because it was the one which better approximates the flow values wanted (see section 3.1 for more details). By analyzing the weather measurements of all the stations, it was noticed that some of them were recording the same, but shifted, values (see Figure 4). This is probably due to the relative proximity of the stations. To maximize the data information gain and minimize useless features and additional weight, the stations of Tione, Montagne and San Lorenzo in Banale were selected. Moreover, flow integrated variables over different time windows were designed (as in section 2.2, Table 3) to provide additional features to the model. Before merging the two databases, variables were extracted from the date such as day, month, year, hour, day of the week and week of the year. This was to help the ML model learning seasonal trends in the flow rate, thus improving the overall accuracy of the algorithm. Finally, all the information was merged in one .csv file (file DB_con_meteo.csv) of a total weight of 744 Megabytes.

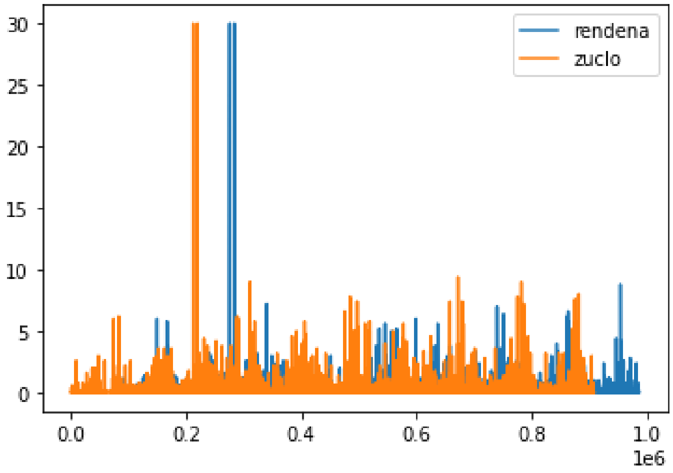


Figure 4: Similarity in weather measures

3 Forecast of inlet flow

3.1 An approximation of the inlet flow

Since, as mentioned, the plant did not have any sensor for the input flow, which was the prediction target, the team started looking at other sensors and data from other plants to retrieve comparatively useful information.

The first analysis (flow_rates_analysis.ipynb) exploited the data of the flow pumped by two wastewater lifting plants towards Stenico's treatment plant. In particular, the outflow time series of the plants of Comano and Cillà were available to be analyzed. Among the two, Comano plant was the most significant in this analysis since it is closer to Stenico plant and it collects the wastewater coming from the majority of the sewer pipes of the territory. Anyway, considered both the lifting plants were considered for completeness purposes. In Figure 5 a comparison of the mean outflows of the three mentioned plants is shown. It is worth noticing that the presented results are only referred to the year 2020, because the full dataset was provided only in a second time (around the 15th of October). For visualization purposes only, Cillà's data have been rescaled to allow a better lineup. In Figure 6 we can appreciate some details of the time series. From the plots, it is possible to see that the trend is almost the same between the different plants, especially the Comano and Stenico ones. To assess this similarity in a quantitative way the cross-correlation function (CCF) was computed between Stenico data and each of the two lifting plants outflow. In Figure 7 the results of such calculations can be appreciated. The plots show again that the Comano data are more meaningful. Considering

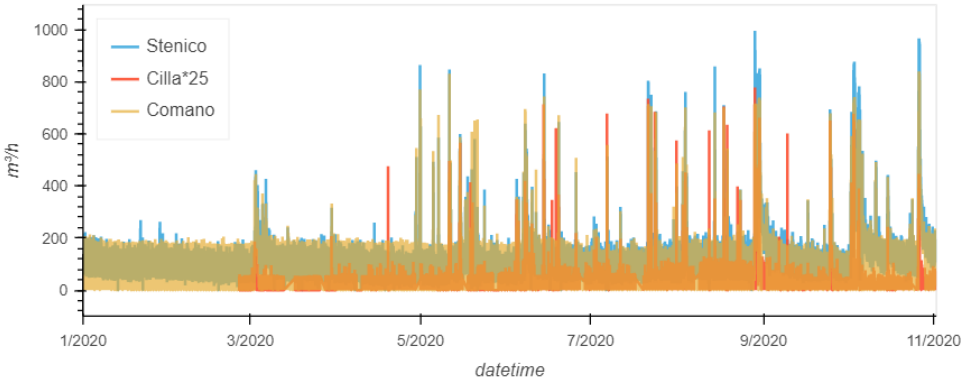
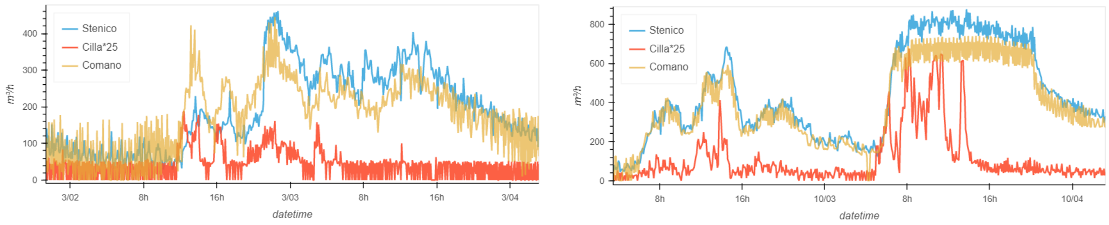


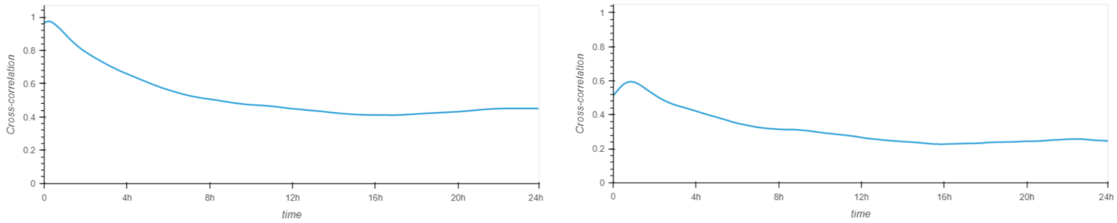
Figure 5: Mean outflow of treating and lifting plants, year 2020



(a) Detail of flow, 02-03 march 2020

(b) Detail of flow, 03-04 oct 2020

Figure 6: Details of mean outflow from Figure 5



(a) CCF Stenico-Comano

(b) CCF Stenico-Cillà

Figure 7: Cross-correlation between Stenico outflow and each lifting plant flow on a 24h period

only this plant, the peak of the CCF was extracted, equal to 0.975, which occurs at the average delay time between when the wastewater is pumped from the lifting plant to when it impacts on the treating plant's outflow. This average delay resulted to be approximately 15 minutes. Given the high correlation between Comano and Stenico outflows and given the fact that the mean delay by the two time series is relatively small, the team decided to approximate Stenico's inlet flow with the outflow.

Before confirming the above hypothesis it was important to consider the Stenico internal processes in order to understand how the wastewater coming in is purified and released out. The plant has sensors for the flow that comes out from the plant (OUT) and for the bypass flow (BYP) which does not go through all the processes and is not considered by the already mentioned "CL-SQm-1" value. Talking with plant experts the team decided to work with the assumption of a conservative flow: water that comes in, must also come out. By doing so we could approximate the overall fluid mass (TOT) as the sum of OUT and BYP.

$$TOT = OUT + BYP \tag{1}$$

Under this assumption, if BYP was negligible OUT could be used as our prediction target for our supervised learning algorithms. To ensure BYP played almost no role in the plant's dynamics, the percentage of the BYP on the overall mass flow was evaluated. The BYP for year 2020 is presented in Figure 8. Again, at the time of the analysis, only 2020 data were available. The results obtained considering the full dataset will be presented in the following paragraph.

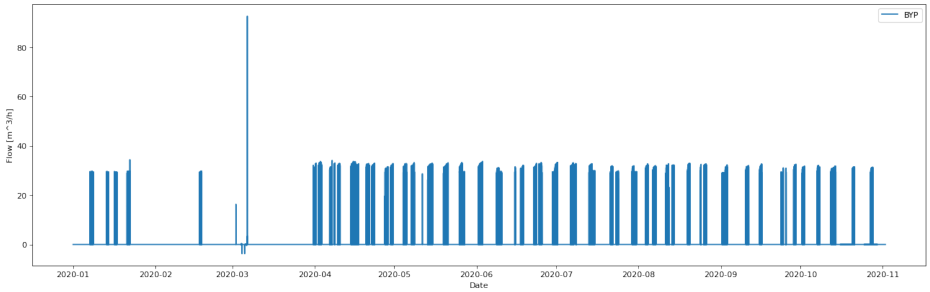


Figure 8: Bypass flow for year 2020

The analysis showed that for year 2020 the percentage of BYP on TOT is less than 1% and thus negligible.

Once the full dataset was given to the team, an additional analysis were performed regarding the bypass flow (PrimaAnalisiTimeSeries.py). The intent was to check whether any pattern occurred within the bypass dataset. By using the *pandas* library, the plot in Figure 9 was obtained, which shows a similar behaviour with the one previously found. The *rolling().mean()* methods were used in order to study the seasonality of the series according to a daily, weekly and monthly mean.

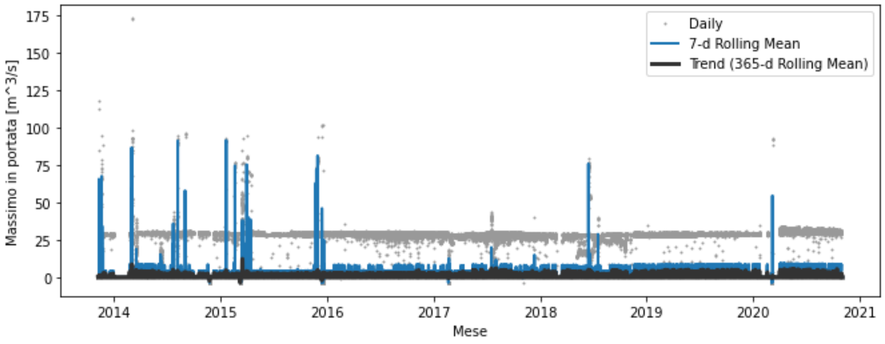


Figure 9: Bypass flow from 2013 to 2020

All the results combined led the team to finally neglect the bypass and to approximate the inlet water flow with the outlet water flow.

3.2 Looking for trends: a preliminary analysis before applying ML models

Before proceeding with the application of machine learning algorithms, several studies were conducted about the detection of trends that would prove useful for the design of time windows implied in the different algorithms. A first attempt to do so (SecondaAnalisiTimeSeries.py) was performed adopting methods and libraries similar to the ones used in the bypass analysis. Rolling means were again used and applied to the outlet flow: this way, the plot in Figure 10 was derived. The general trend which can be detected enlightens the most grievous situation in the autumn and winter months, which can be correlated to the increase in rainfall.

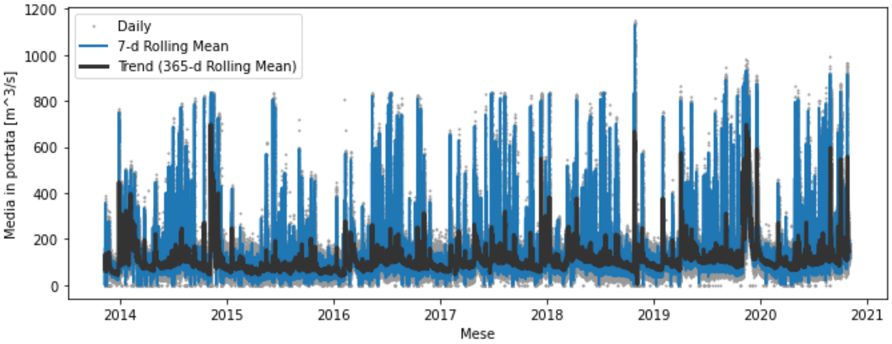


Figure 10: Outlet water flow behaviour from 2013 to 2020

One important thing that was noticed in our very first visualizations of the time series, was the presence of a daily pattern, strictly connected to the daily cycle of humans. The trend is mainly visible in periods with no extreme events in the flow, i.e. periods where the overall trend of successive days is nearly flat. The daily trend can be seen in Figure 11 by looking at the white dots that represent the mean outflow at every hour of the day. The plot also shows the distributions of the data samples for each hour of the day and we can notice that they have a gaussian shape. For what concerns the daily trend, it can be seen that lower values could be registered during the night and an increase in the flow in the morning. The peak is reached in the late morning and around lunch time. The flow decreases a bit in the afternoon, rises again to the second peak around dinner time and then decreases again. As mentioned above, this highlights a correlation with the daily cycle of humans.

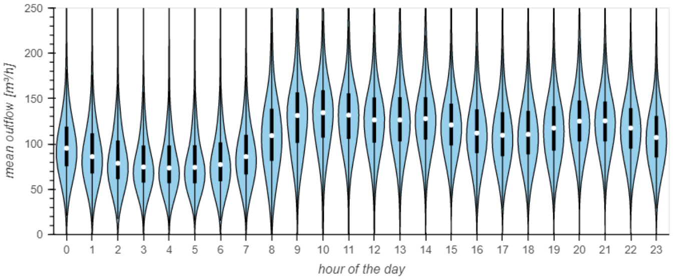


Figure 11: Daily trend of Stenico plant's mean outflow, violin plot

4 Exploring machine learning methods

In this section, the most relevant machine learning solutions analyzed are listed. Many models were skimmed, from more classic ones such as ARIMA [2] and LSTM [3], to the state-of-art such as XGBoost [4] and TCN [5]. Moreover, even if not listed in this document, also Prophet [6], MiniRocket [7], Transformers [8] and NeuralProphet [9] were tried and deserve a mention. It was decided not to include them since only just some trials were performed and close to the end of the Challenge, which prevented the team from getting relevant results. It can be worth considering them in the future work also given that the original papers of the last two models cited were submitted respectively on 24th

September 2021 and 29th November 2021, thus they could become widely popular in the near future and may lead to better results. Reference to the trials in the Miro board of the team (see section 7).

To compare the analysis and results, the persistence algorithm was used as baseline model. The persistence algorithm uses the value at the previous time step $(t-1)$ to predict the expected outcome at the next time step $(t+1)$. By comparing the developed models to the persistence algorithm forecasts, it can be easily measured how much the model is propagating the last measurement. Indeed, this property is not desirable when dealing with extreme event forecast and it should be controlled.

4.1 XGBoost

XGBoost [4] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. We tested this algorithm with different choices of input features. Here the most interesting attempts are described.

4.1.1 Training using only rainfall integrated measurements

To make the model less prone to propagating the last inlet flow measurement, a XGBoost regressor was trained using only integrated rainfall measurements, an integrated inlet flow measurement over the previous two days and two time variables for the hour and the day. Measurements were sampled every 30 minutes and 2000 estimators were used, while the learning rate was fixed to $l_r = 10^{-3}$. An early stopping procedure was implemented with a patience of 50 to avoid over-fitting . Mean Absolute Error (MAE) was used as a cost function. Since this was intended as a test, only the measurements coming from the weather stations of Tione, San Lorenzo and Montagne were used. Figure 12 shows the feature importance according the above described XGBoost model. The model was tested on the period after June 1st 2020. Figure 13 shows an example of model performance. In particular, it can be seen that the model is able to predict changes in the inlet flow using only rainfall measurements. Moreover, the model is observed to have low sensitivity to daily trends. Indeed, in absence of extreme events signature, the model tends to propagate the same value (see the blue line that is almost flat between August 27th 2020 up to August 28th 2020). This is easily explained by the fact that only an integrated variable over two days was given as input to the model.

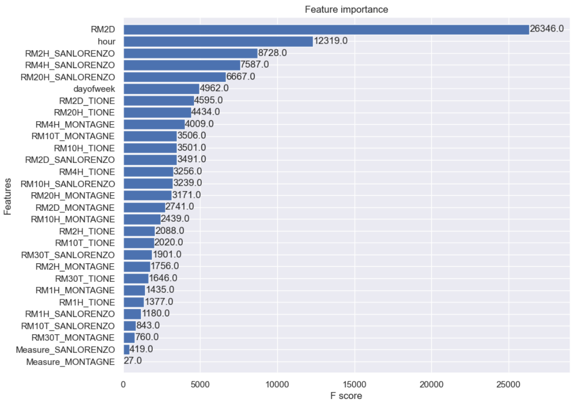


Figure 12: Feature ranking obtained using XGBoost model.

Then, the model performance was compared with the persistence model. The MAE value on the test set achieved by the persistence model is 28 while for the XGboost model is 38. The left side plot of Figure 14 shows the relative error of XGBoost and persistence model predictions, which is better for the second one.

To prove the forecasting ability of the model, the team restricted the test set to points where a rising trend in the inlet flux trend is expected. The right side plot of Figure 14 shows that the XGBoost model achieves lower relative errors. The MAE value on the test set restricted to rising edges achieved by the XGBoost model goes down to 28 which is the same scored by the persistence model.

To better understand the XGBoost performance, the distribution of the relative error as a function of the expected target was plotted, as shown in Figure 15. It is clear from the images that the main

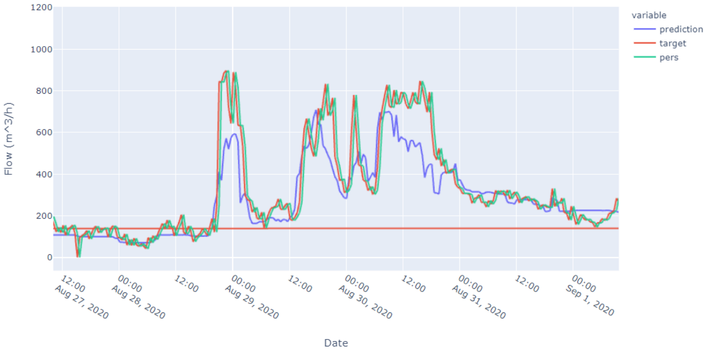


Figure 13: Comparison of XGBoost model prediction (blue line) to target (red line) and persistence model (green line). The time period for this visualizations starts on August 27th up to September 1st 2020.

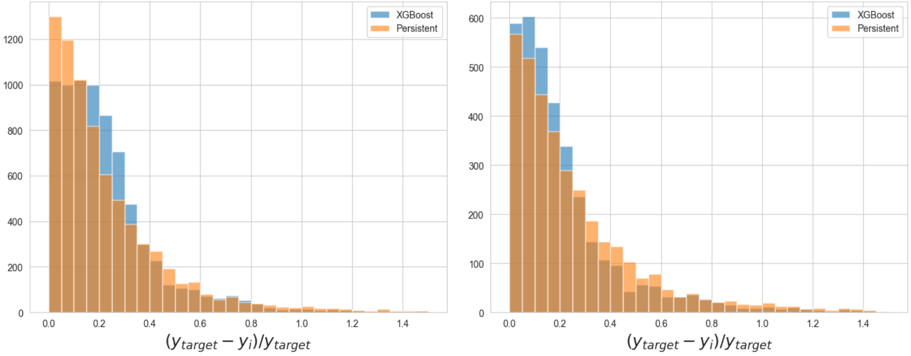


Figure 14: Relative error distribution of XGboost (blue) and persistence (orange) model predictions. (Left) Absolute distribution. (Right) Relative errors on rising edges.

source of error for the XGboost model comes from prediction during ordinary days. The bright spot centered at around zero for the persistence model appears to be centered around ~ 0.15. Also, the sparse region for the high values of the target highlight that the persistence model commits smaller errors respect to the XGboost model. Referring to Figure 13, it can be seen that the XGBoost model is capable to forecast correctly the change of trend but still the predictions are not very accurate. This can be explained since the only information on the inlet flow is averaged over two days and no information on the actual sewage presence in the tanks of the water treatment-plant is used (see subsection 3.1 for the adopted target definition of the inlet flow using the outlet flow).

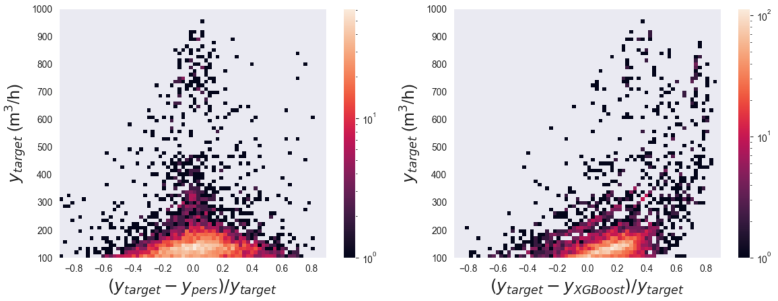


Figure 15: Relative error distribution of persistent (left) and XGboost (right) model predictions as a function of the expected target.

4.1.2 Walk-Forward validation

The Walk-Forward validation method aims to improve the accuracy of the predictions by retraining the model by including in the training data events that were previously used for test only. Figure 16 shows a rheumatic representation of this method. By doing this, the model is less sensitive to changes that can occur during time, e.g. electronic sensors degradation.

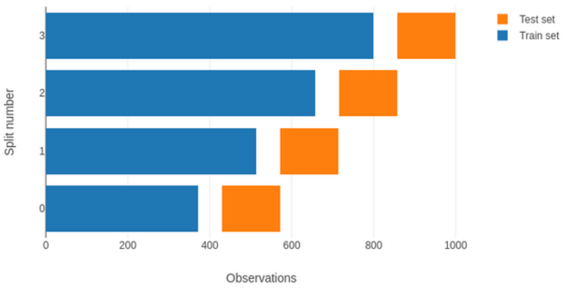


Figure 16: Schematic representation of walk-forward validation procedure.

This method was applied to XGBoost which was trained with one month of data averaged over an hour and re-trained by choosing different training frequency values. Two tests were made to compare the different performances: one with the training frequency of one over 6 hours (with 383 trainings), one with a training frequency given by every time a new value was predicted (with 767 trainings). Since this approach requires to train the model multiple times, the size of training data and the values of the model hyperparameters have to be carefully chosen in order to have the trained model ready to be used in time. Using the above cited training frequencies, an XGboost models was trained using walk-forward validation, with 500 estimators, a learning rate of 0.1 and Mean Absolute Error (MAE) as cost function. The features used are the same as in section 4.1.1 plus the information about the inlet flow (integrated over time and not). After the training, all results were compared also with the persistence model. As it can be seen from the two plots of Figure 17 the performances improved. In the first picture, the daily trend is reproduced and also the extreme events are predicted. But it is with a further zoom in that the effectiveness of the model can be appreciated. In the second picture, the zoom of the extreme event that occurred on October 26th 2020 is shown. Following the red line it can be seen that around 12:00 a.m., when the target is rising up in value, the model follows it almost perfectly compared to the other methods which are an hour late on the prediction. To verify that on that day there was actually an extreme event, the meteorological analysis of Tione's area in 2020, provided by MeteoTrentino[10] was checked. In this file, it is reported that October 26th 2020 was recorded as the highest rainfall day in the whole month. The MAE that is achieved with the persistence model is 19.05. Our model's MAE is 16.12 for the test with a training frequency of 6 hours, and 8.12 (almost half) for the one with training frequency of one hour. As shown in Figure 18, lower errors

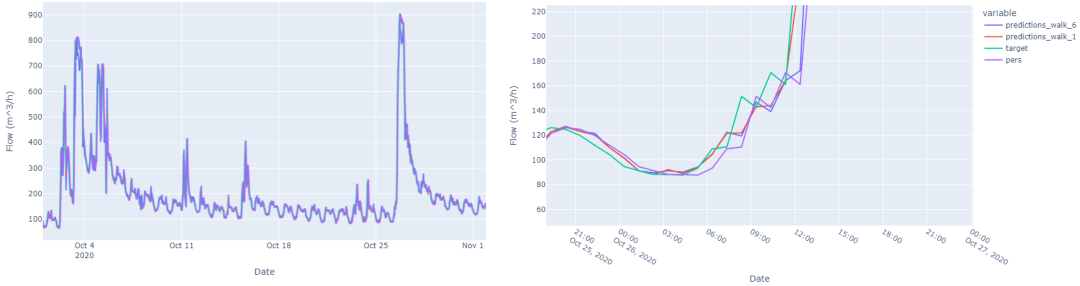


Figure 17: Performance over the test period and performance during an extreme rainfall event.

were made in comparison with the persistence model. In addition to this, Figure 19 shows that the error distribution is more centered around zero values in the case of 1 hour training frequency with respect to the 6 hour one appears to be more sparse.

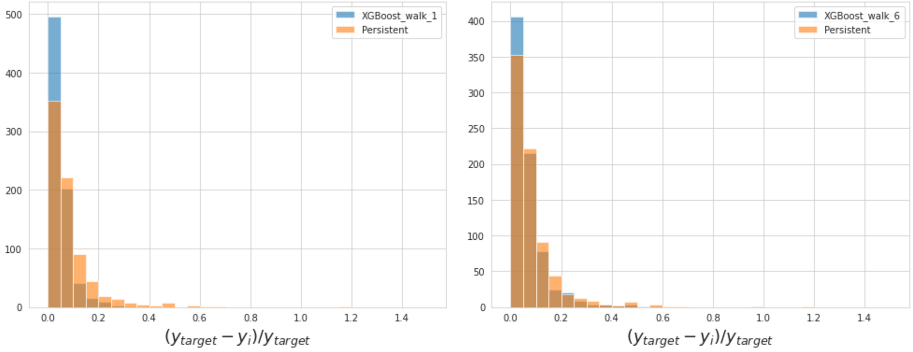


Figure 18: Relative error distribution of persistence and XGboost training frequency at 1 (left) Relative error distribution of persistence and XGboost training frequency at 6(right)

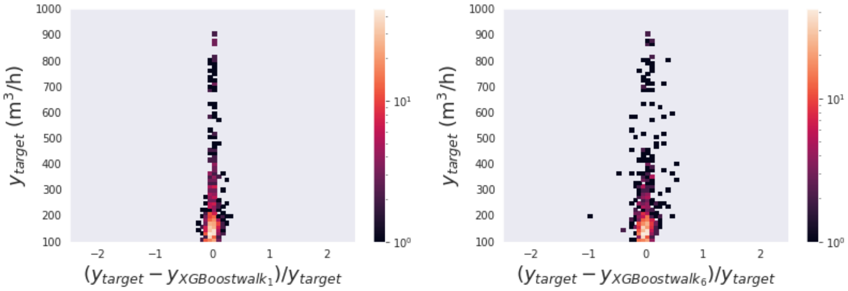


Figure 19: Relative error distribution of XGboost freq 1 (left) and XGboost freq 6 (right).

4.1.3 Classification

The control system that is currently integrated in the Stenico's treatment plant is based on a fuzzy logic system which combines different measurements coming from different subsystems. To easily integrate our model predictions in the control system, we investigate a different approach in which we cast the regression to a classification task. The flow rate was divided into four different classes (easily coded in bits): flow rate less than 150 m^3/h, flow rate between 150 and 400m^3/h, flow rate between 400 and 600 m^3/h and flow above 600 m^3/h. A feature was added into the dataset to describe this behaviour. This model was trained using XGBoost Classifier walk-forward validation with training frequency every hour. The hyper-parameters, length of the training and test sets are the same as section 4.1.2. Figure 20 shows an example of the classifier performance. One time-step delay can be observed for the 150 and 400 m^3/h, while extreme events forecast appear to be in phase with the target. The classifier accuracy on the test is equal to 0.98. From the confusion matrix, that is shown in the left side plot of Figure 21, it can be observed that most of errors are done by misclassifying the 150-400 m^3/h class with the < 150 m^3/h class.

The same type of prediction can be obtained by looking at the above described XGBoost regressor output and defining the same inlet flow bins. By doing this, we were able to compare the performance of the classifier with that of the regressor evaluated on the same test set. As for the regressor, an accuracy of 0.93 was obtained. The right side plot in Figure 21 shows the confusion matrix over the predictions for the regressor. It is worth noticing that the model is making some extra errors compared to the previous plot in left side plot of Figure 21 in classifying the 150-400 m^3/h class with the < 150 m^3/h class. Those errors are mainly done on events in the daily trend and they do not affect the accuracy on high inlet flow forecast classification.

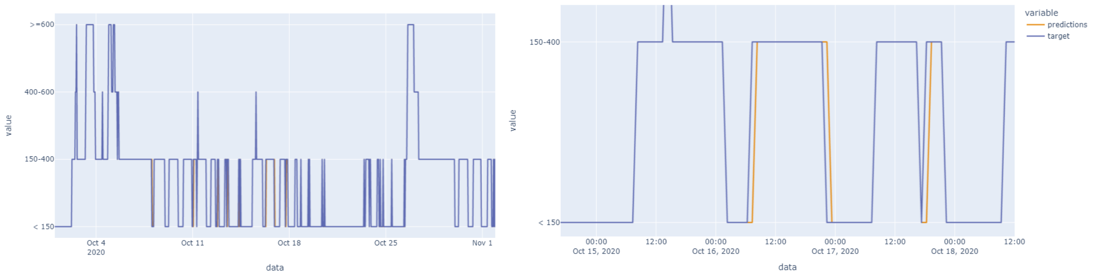


Figure 20: Comparison of XGBoost model class prediction (orange line) to target (blue line). Bottom plot shows a zoom on October 15th-18th 2020.

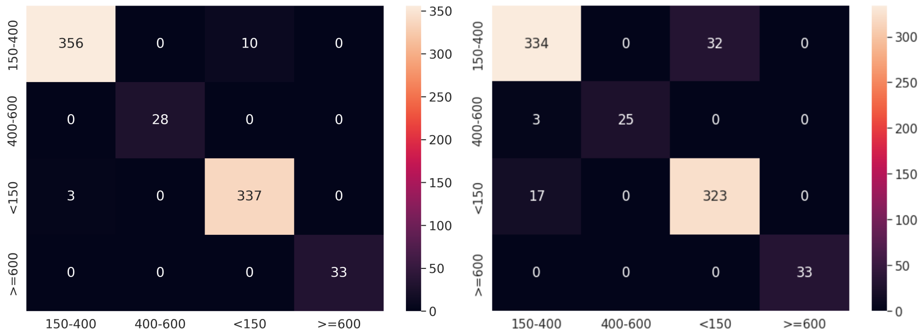


Figure 21: Confusion matrix on the classification problem. On x-axis the predicted class is shown while y-axis shows the correct target. (Left) Predictions evaluated using the XGBoost classifier. (Right) Predictions obtained using the regressor output with the same class bins of the classifier.

4.2 Long short-term memory

As timeseries prediction is a special kind of regression task, which involves knowledge of the past history of the signal, some special architectures have been developed to tackle this problem. Long short-term memory (LSTM) [3] is a kind of recurrent neural network whose neurons' connections are not only with their previous and following layers but also with themselves, thus keeping information about the past.

4.2.1 Sliding window

LSTM cells use sequence of data to make predictions and these sequences are often referred to as ”windows”. A window is a vector containing a number of timesteps equal to the length of the window. Each time-step also contains a certain amount of features. The dimension of the dataset is thus a 3D array with dimension: $[nsamples, ntimesteps, nfeatures]$. Figure 22 shows a schematic representation of such mechanism. By comparing different windows lengths, it was decided to set it at 6 timesteps. Since all the LSTM trials that were performed relied on a timeseries with hourly samples, this meant that each window contained 6 hours of past information.

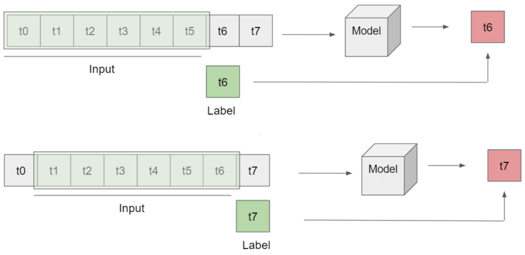


Figure 22: Sliding window data-loading mechanism.

4.2.2 Trial with LSTM

The data used to train this network are obtained by scaling the dataset by subtracting the mean and dividing by the standard deviation, the sampling was set to 1 hour. The dataset was split to get training data of the first 70% of training, the following 20% for validation and the final 10% for test. It was decided to implement an architecture with two distinct inputs, one for recurring variables ('min', 'max', 'mean') and one for all the other, including the integrated variables. The architecture is shown in figure 23. In this way, the LSTM stack is involved just in learning dependencies on the min, max, mean values of the timeseries, while the integrated variables are processed only in a second stage by the fully connected layers. The best hyperparameters were found by exploiting the hyperband algorithm [11]. An Adam optimizer was chosen and a 'MSE' (Mean Squared Error) validation loss objective. An early stopping scheduler with patience of 5 was used to prevent over fitting and to save time in the optimization procedure. The optimization variables are here described. Dropout1: dropout of first layer of lstm, Units 1: units for first LSTM layer, LSTM layers: numer of stacked LSTM layers, Units LSTM: units for each layer of the LSTM stack, Dropout2: dropout for each layer of the LSTM stack. Dense layers: number of stacked dense layer, Units dense: units for each layer of the dense stack. Firstly, an optimization with rough steps was run (Table 4) in order to get an understanding of which parameters are crucial for a good prediction.

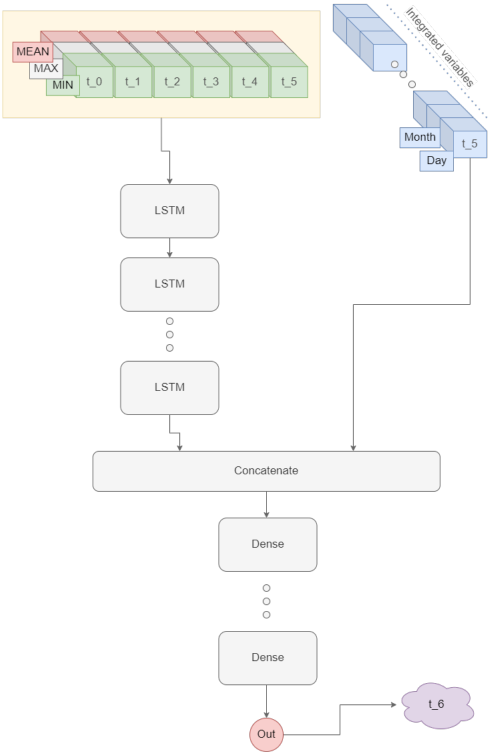


Figure 23: LSTM architecture

By looking at the relation between parameters and loss, the bounds for the next optimization step were chosen. A finer scale is used and the bounds are presented in Table 5. At this stage no better results could be obtained with this approach, and the optimization iterations were therefore stopped here. Finally a model with the best hyperparameters was trained. The training scheduling is presented in Table 6

The resulting MAE is 13 for LSTM and 16 for persistent model. A similar analysis to the one done for XGBoost is presented in Figure 24. In figure 25a the predictions over the whole test set are presented. The real and predicted flow are very close, meaning that the model is pretty good in guessing the right value of flow. The problem with this model is the delay, which is highlighted in figure 25b. A delay in prediction means that the model has not truly understood how the system behaves, it is just trying to guess the next step by shifting the actual value with a slight modification.

Parameter	Type	Min	Max	Step	Values
Dropout 1	Float	0.0	0.8	0.1	
Units 1	Int	32	512	32	
LSTM layers	Choice				0, 1,2,3,4
Units LSTM	Int	32	512	32	
Dropout 2	Float	0.0	0.8	0.1	
Dense layers	Choice				1,2,3,4,5
Units dense	Int	32	512	32	
Learning rate	Choice				1e-2, 1e-3, 1e-4

Table 4: First LSTM optimization step

Parameter	Type	Min	Max	Step	Values
Dropout 1	Float	0.0	0.8	0.1	
Units 1	Int	32	512	16	
LSTM layers	Choice				0,1,2,3
Units LSTM	Int	100	400	16	
Dropout 2	Float	0.0	0.8	0.1	
Dense layers	Choice				1,2,3,4,5
Units dense	Int	200	600	16	
Learning rate	Choice				1e-3, 1e-4

Table 5: Second LSTM optimization step

Parameter	Value
Dropout 1	0.4
units 1	480
LSTM layers	0
Units LSTM	-
Dropout 2	-
Dense layers	5
Activation	'relu'
Units dense	272
Learning rate	1e-3
Loss	'mse'
Epochs	100
Batch size	32

Table 6: LSTM final training parameters

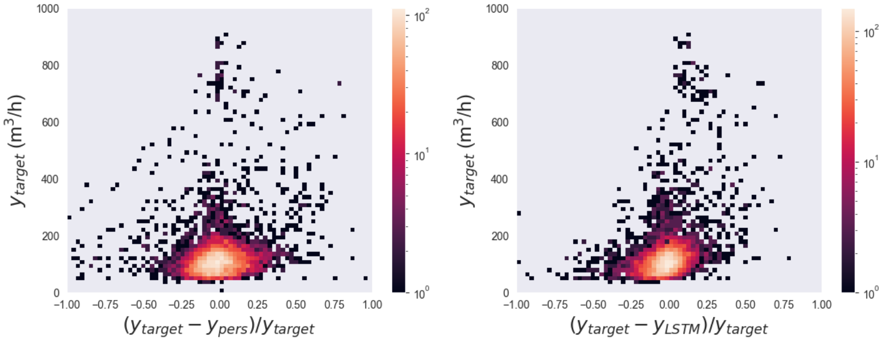
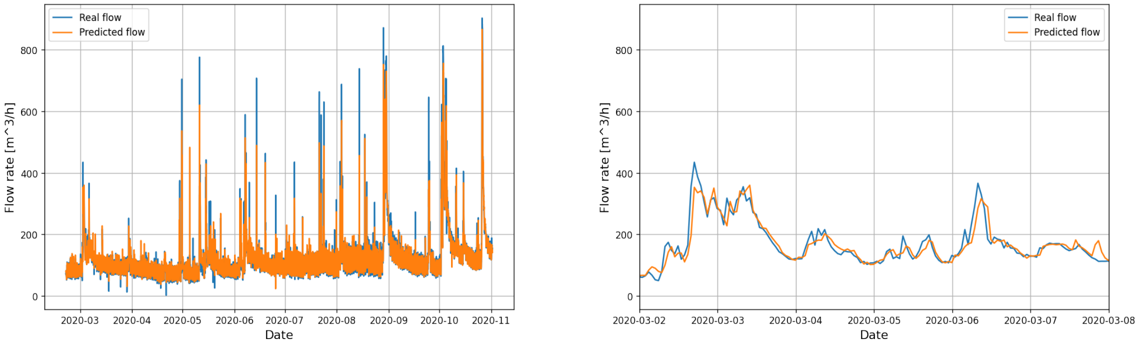


Figure 24: Relative error distribution of persistence (left) and LSTM (right) model predictions as a function of the expected target.



(a) LSTM prediction over the whole test set

(b) LSTM predictions show a delay of one time step

4.3 Temporal Convolutional Networks

Temporal Convolutional Networks (TCNs) [5] are deep learning models for time series forecasting. For each timestamp, the following variables were considered: hour and day of week variables, integrated inlet flux over 2 hours before, rainfall measurements integrated over two hours before for the Tione, Giustino, Val Di Breguzzo, Villa Rendena, Val Dambiez, Montagne, San Lorenzo In Banale. Measurements were sampled every hour. The target, in this case, is fixed to be the average inlet flux expected in the next hour and data were normalized to improve the model training. The normalization parameters are evaluated on the training dataset and only used to transform the test dataset.

The data loading step is similar to the LSTM approach described above. In this case, the look-back window is fixed to 12 timestamps, corresponding to 12 hours. The model architecture consists of one TCN layer with kernel size 3 and batch normalization [12]. Then, the TCN output are flattened to an array that is given in input to a Fully Connected Neural Network with two hidden layers of 64 nodes with SELU activation function [13] and one linear output unit. The model is developed using Keras [14]. The ADAM optimizer [15] with learning rate $l_r = 10^{-4}$ and MAE loss function was chosen. A learning rate scheduler was used to reduce the learning rate value when the loss function on the validation dataset is not improving after 5 epochs. The reduction factor is fixed to 0.2. An Early stopping procedure was implemented by monitoring the loss evaluated on the validation dataset with a patience of 10 epochs.

Figure 26 shows an example of model performance. In particular, the model is able to predict changes in the inlet flow and it can also be seen that the model is sensitive to daily trends (see the blue line that follows the daily trend between August 27th 2020 up to August 28th 2020).

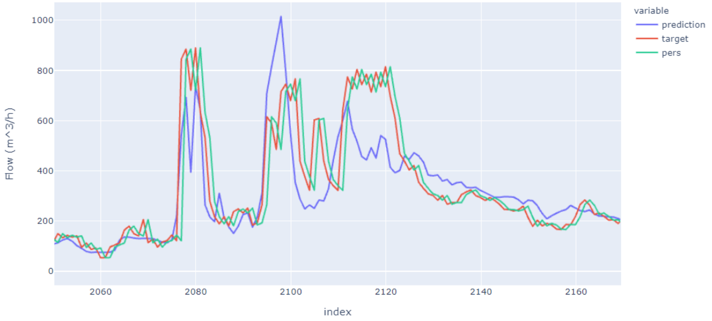


Figure 26: Comparison of TCN model prediction (blue line) to target (red line) and persistence model (green line). The time period for this visualizations starts on August 27th up to September 1st 2020.

Similar studies to the ones performed for the XGBoost model are performed to assess the performance of the TCN model with respect to the baseline persistence model. The MAE metric for the persistence model evaluated on the test set is 20 while for the TCN model is 17. Figure 27 shows the relative error for predictions obtained using the persistence and TCN models. The TCN model is obtaining better relative errors respect the persistent model on both the complete test set and on rising edges contained in the test set. Figure 28 shows how the TCN model can predict with a relative

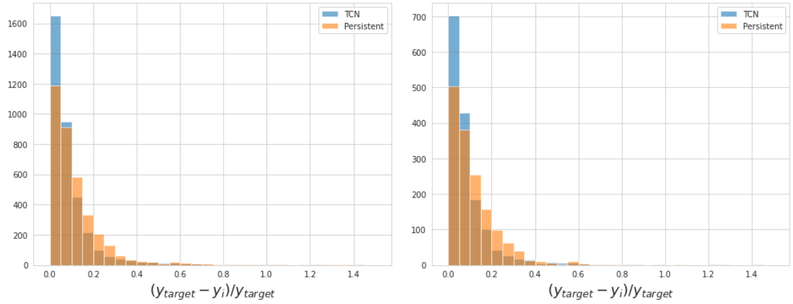


Figure 27: Relative error distribution of TCN (blue) and persistence (orange) model predictions. (Left) Absolute distribution. (Right) Relative errors on rising edges.

error centered to zero values also the daily trend of the inlet flux. This can be motivated by the fact that 12 measurements are given as input to the model. Moreover, the inlet flux variable used while developing this model is integrated over 2 hours, not 2 days as for XGBoost.

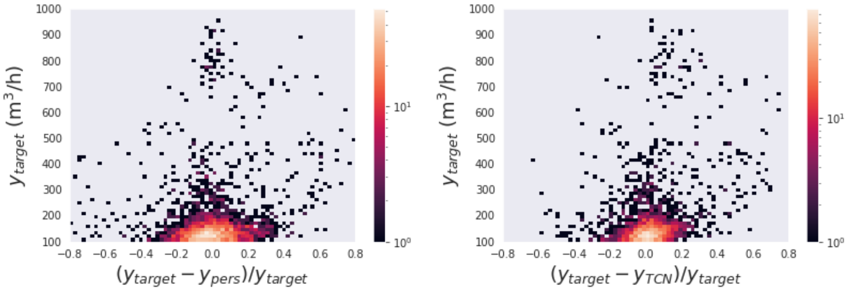


Figure 28: Relative error distribution of persistence (left) and TCN (right) model predictions as a function of the expected target.

4.4 ARIMA

A first trial with ARIMA[2] (Arima_FirstTrial.py) was carried out by using the library *pmdarima*, importing the method *auto_arima()*. Such a method would estimate, via an optimization algorithm, the parameters p,q,d which would best suit the problem. At the end of the process, the following numerical values were found: 4 for p, 0 for d and 1 for q, although the associated AIC (Akaiake Information Criterion) value was very high (of the order 10^5). These values were fed to the basic ARIMA model available in the library *statsmodels*. On the other hand, the database was divided into training (70%) and test (30%) sets. Once having set everything up, the algorithm was launched and residuals were calculated (Figure 29).

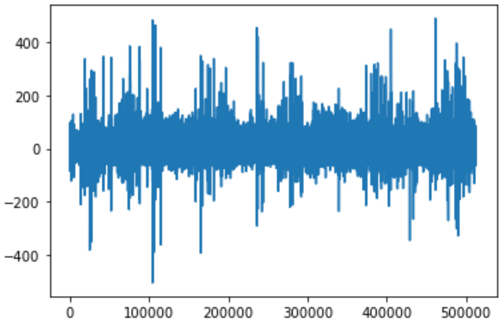


Figure 29: Residuals from the auto - ARIMA trial

As it can be seen, they proved to be really high and thus not very useful (but this could be forecasted from the AIC). So a different approach was tried out. In particular, we decided to delve deeper into the ARIMA model architecture to find out if it was a suitable algorithm for our problem.

Following this purpose, the second trial (ARIMA.ipynb) started with the statistical analysis of time series for the Stenico's hourly mean outflow. Since the complete time series proved out to be complex to model, it was decided to test the ARIMA architecture on a small period of time (20 days), shown in Figure 30, which does not show extreme events, but just "normal" daily activity.

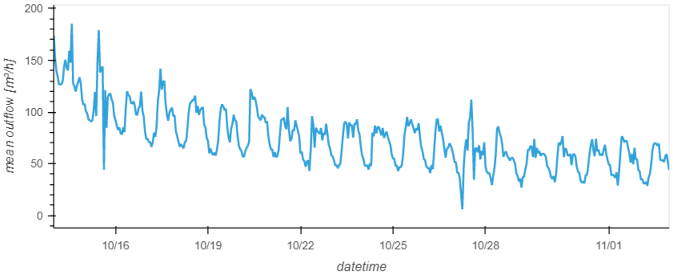
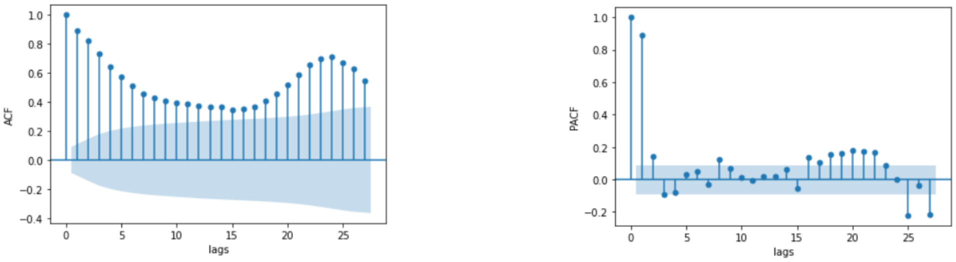


Figure 30: Analyzed period for ARIMA model training

As described below, this approach allowed the team to obtain results compatible with what can be found in literature and to try to fit an ARIMA model following some classical theoretical steps, namely the Box-Jenkins approach[16]. The first step of the analysis is the model identification and selection, i.e. finding the order of the ARIMA model. The team started by checking the stationarity of the series: this is because ARIMA models work with stationary inputs. The non-stationarity of the time series could be assessed both by qualitatively checking the plot and also with quantitative tests. These are the ADF (Augmented Dicker-Fuller) test and the KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test, which opposite null hypothesis are respectively the presence of unit roots and the stationarity. The non-stationarity can also be seen by plotting the autocorrelation (ACF) and the partial autocorrelation (PACF) of the series as in Figure 31.



(a) ACF

(b) PACF

Figure 31: ACF and PACF plots of the initial time series

From these graphs it can also be noticed the presence of a 24 hours seasonality. This matches the initial analysis of the daily trend. The seasonality has to be removed in order to better estimate the order of the non-seasonal components of the ARIMA model. In order to remove the seasonal effect, multiplicative seasonal adjustments were applied [17] which produced the time series in Figure 32.

To obtain a stationary time series, differentiation was performed, which consists in computing the step wise difference between the samples. The differentiated time series y_{diff} is hence computed as:

$$y_{diff}(t) = y(t) - y(t-1), \quad \forall t > 0 \tag{2}$$

The first differentiation was believed enough since the second one made the standard deviation increase leading to over-differentiation [2]. Repeating ADF and KPSS tests it was assessed that stationarity had been achieved. Analysing the ACF and PACF plots in Figure 33, the orders of the AR and MA components could be defined [18].

The plots show a mild over-differentiation which can be balanced by increasing the order of the MA term [2]. Considering this, an ARIMA(2,1,2) or an ARIMA(2,1,3) were opted for. To have better

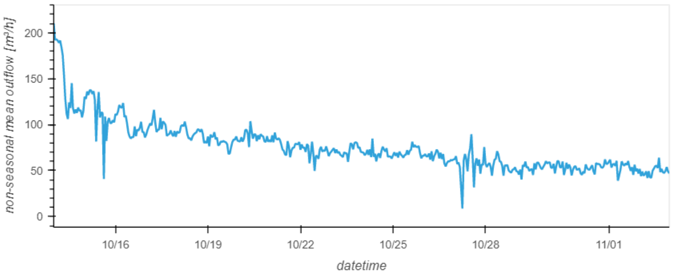
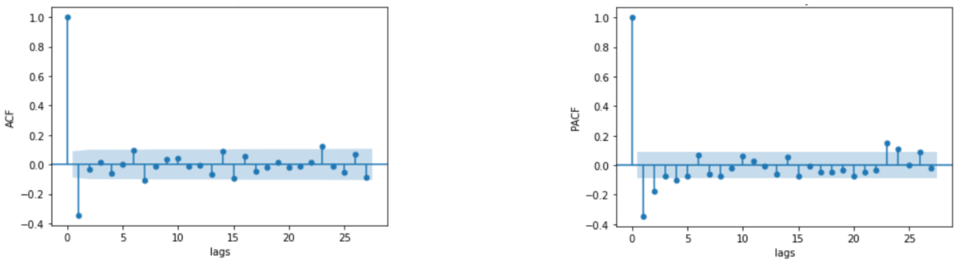


Figure 32: Seasonal adjusted time series



(a) ACF

(b) PACF

Figure 33: ACF and PACF plots of the seasonal adjusted and differentiated time series

feedback on the validity of our choice, a grid space of parameters was set over which to evaluate different ARIMA models, i.e. a model was trained for each combination of parameters and compared the results based on AIC criterion. As it can be seen in Table 7, the model with the lower AIC value, and therefore the most performing, turned out to be an ARIMA(2,1,3), confirming the previous calculations.

(p,d,q)	**AIC**
(2, 1, 3)	3442.50
(2, 1, 4)	3443.24
(3, 1, 3)	3444.10
(0, 1, 1)	3444.93
(1, 1, 1)	3445.07
(4, 1, 3)	3445.25

Table 7: Most performing models chosen via AIC

The second step of the Box-Jenkins method is the parameter estimation of the model with the identified order. This step is done automatically by *fit()* method of the class *statsmodels.tsa.arima.model.ARIMA* which uses maximum likelihood estimation.

Once obtained the model with the estimated parameters, the final step of the method is to perform a statistical model checking. This was done by analyzing the residuals as in the first trial of the application of ARIMA model. As can be seen in Figure 34 the residuals seem good and resemble white noise.

This is shown also by the distribution of the residuals in Figure 35.

The model was tested trying to make hourly forecasts for the two subsequent days with respect to the training set.

As shown in Figure 36, the predictions are delayed and resemble a persistence model. Since this is not the desired behavior for the application and the result was obtained on a period much simpler to model than the complete time series, we decided to discard the ARIMA option.

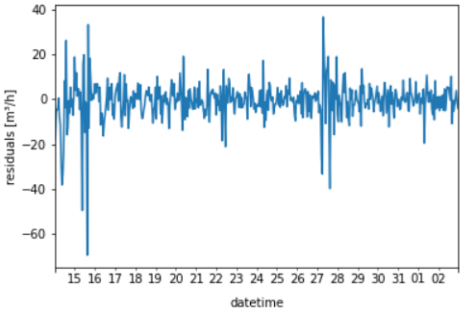


Figure 34: Residuals of the ARIMA(2,1,3) model

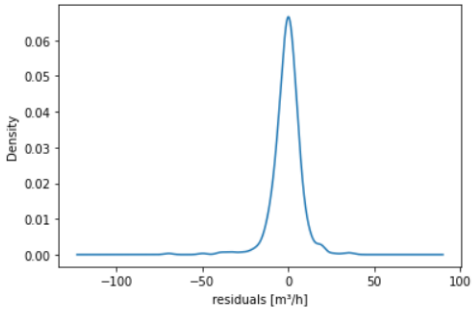


Figure 35: Distribution of the residuals of the ARIMA(2,1,3) model

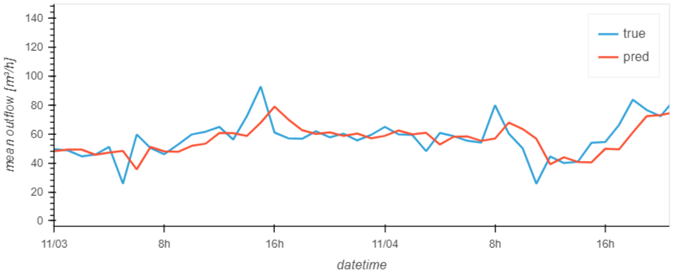


Figure 36: Forecasts of 1h with the ARIMA(2,1,3) model

5 Optimization of the regulation of compressors

The discussed obtained results on the inlet flow forecasting were shared with the experts of the Provincia Autonoma di Trento. During the discussion, the team proposed them a new project on electrical energy saving which could be achievable by optimizing the regulation of compressors that introduce oxygen in the water-treatment plant of Stenico. Such an optimization would not only allow saving electrical energy but also to reduce costs related to the maintenance of compressors.

5.1 Data description

The data used to accomplish this task were measurements of the volume of air introduced in the tanks by compressors, together with measurements of oxygen dispersed in the tank. The latter sensor is sensitive to the activity of bacteria during the aerobic biological process. From these measurements, one could produce integrated variables as it was done for rainfall variables. Additional information could be added using the inlet forecast that was produced with the models previously described. Indeed, during extreme rainfall events, the sewage that will arrive at the water treatment plant will be much more dilute. In these conditions, the aerobic environment for bacteria is much easier to maintain by introducing a lower amount of air.

5.2 XGboost model without inlet flow forecast

A XGboost model was built to predict within 30 minutes the expected change in the oxygen level in a tank. As input variable integrated measurements of air introduced in the tank and integrated measurement of oxygen level in the tank were used, with additional information about the time by introducing the hour and day variables. 2000 estimators were used and a learning rate of $l_r = 10^{-3}$ was fixed, and an early stopping procedure with a patience of 50 was also implemented to avoid overfitting . Mean Absolute Error (MAE) was used as cost function. Figure 37 shows the obtained feature ranking.

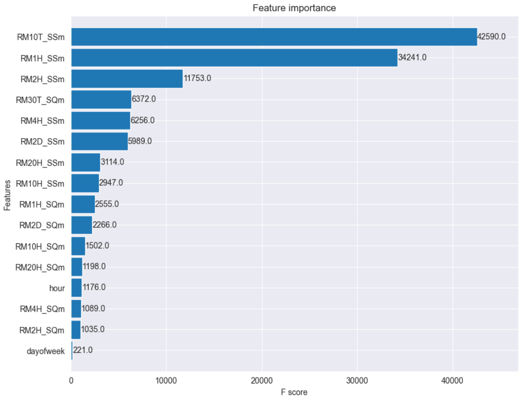


Figure 37: Feature ranking obtained using XGBoost model.

The model was tested on the period after June 1st 2020. Figure 38 shows an example of model performance. The developed model can predict the expected changes in oxygen level, while the persistence model is failing as expected with such an oscillating trend.

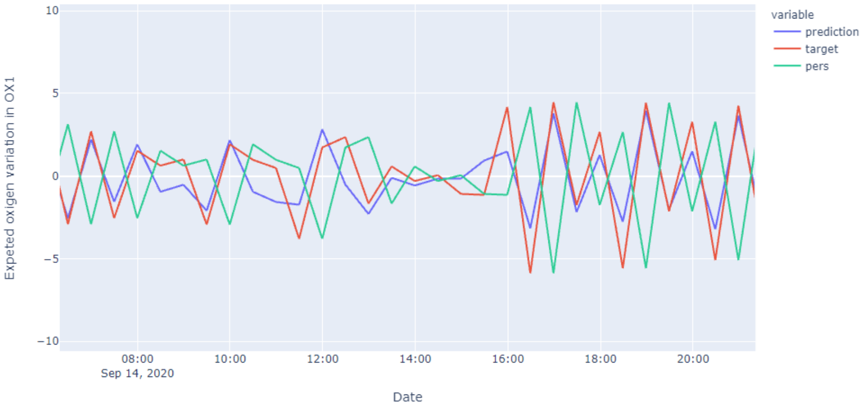


Figure 38: Comparison of XGBoost model prediction (blue line) to target (red line) and persistence model (green line).

Then, the team performed the usual studies to assess the performance of the XGBoost model respect to the baseline persistence model. The MAE metric for the persistence model evaluated on the test set is 4.2 while for the XGBoost model is 1.3.

Figure 39 shows how the XGBoost model can predict with a relative error centered to zero values while the forecast obtained using the persistent model are center around 2.

To get a fair comparison, a persistence model was also tested which is not propagating the last observation but the one occurred 2 time-step before. In this case the MAE score goes down to 2, which is still larger than the score obtained with the XGboost model. It is worth noticing that for this test no information of the expected inlet flux is used.

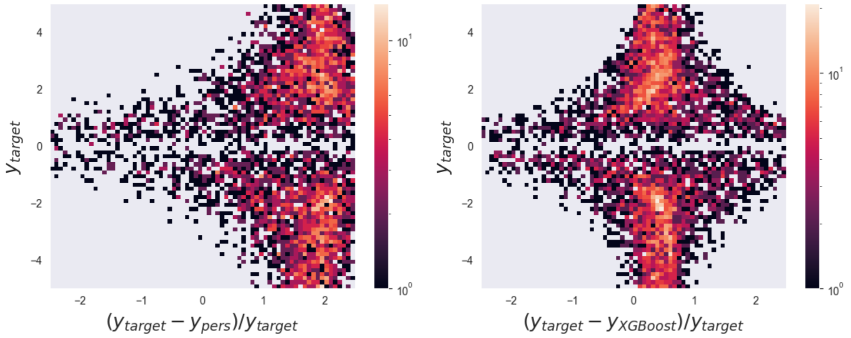


Figure 39: Relative error distribution of persistence (left) and XGBoost (right) model predictions as a function of the expected target.

5.3 XGboost model with inlet flow forecast

A XGboost model was developed to predict within 30 minutes the expected change in the oxygen level in a tank. The following input variables were given: the integrated measurements of air introduced in the tank, the integrated measurement of oxygen level in the tank, and the expected inlet flux in 30 minutes with additional information on the time by introducing the hour and day variables. 2000 estimators were used and the learning rate was fixed to $l_r = 10^{-3}$. An early stopping procedure with a patience of 50 was implemented to avoid over-fitting . Mean Absolute Error (MAE) was used as cost function. Figure 40 shows the obtained feature ranking. The expected inlet flux measurement ranked between the top 5 features according to the trained XGBoost model.

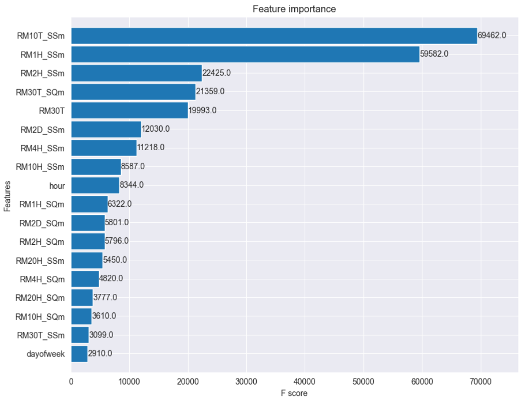


Figure 40: Feature ranking obtained using XGBoost model.

The model was tested on the period after June 1st 2020. The performance were really close to the previously described XGBoost model. Adding the 30 minutes forecast on the inlet flux allowed to reduce the MAE metric on the test set down to 1.2.

6 Conclusions and Further work

The present report describes the team activities during the ”Industrial AI Challenge” organized by HUB Innovazione Trentino with the collaboration of the University of Trento.

The aim of the proposed challenge was to detect correlations between historical and meteorological data and to forecast the impact on water treatment plants in order to implement electrical energy cost saving and predictive maintenance.

Different AI solutions were investigated to solve the specific task of predicting the inlet flux to the water-treatment plant of Stenico. The team was able to develop a XGBoost model with walk-forward validation which effectively predicts the inlet flux trend one step in advance. Moreover, for better integration with current methods, the model was trained to predict the range of the flow rate by solving a classification task.

After that, a second XGBoost model was developed in order to predict variations of the oxygen diluted in the tank with one step in advance. The model was able to predict correctly the trend, also by using the inlet flux prediction obtained using the above cited model.

The prediction of the inlet flow along with the forecast of the variations in the oxygen diluted in the tank are new variables that can be used in the algorithm which controls the regulation of compressors. These could allow for a more performing scheduling of the activation of the compressors in order to decrease the energy consumption and allow for reduction of the wearing of mechanical components of compressors.

The work presented can be easily implemented in current control systems and it is easily scalable to other water-treatment plants with a reasonable amount of collected data.

Acknowledgments

We would like to thank the owner of NIRIS™Andrea Turso for the great help and availability shown during the whole challenge. A great thank also to the mentors Luca Di Persio and Stefano di Persio from HPA™for the support during the Challenge.

The present work would not have been possible without the continuous exchange of information that took place with the technicians of the purification plant. In particular, we would like to thank very much Angelo Spadaro from Servizio Gestione degli Impianti of Provincia Autonoma di Trento who listened to us patiently and on many occasions expressed sincere and useful opinions about our work. We want also to thank Stefano Messana, who developed the present algorithm for controlling compressors, for useful discussions.

7 File repository

All the codes cited in this report can be found at: `https://github.com/MarcusVukojevic/Niris.git`
As for the Miro board, the link is the following: `https://miro.com/app/board/o9J_lwCnA10=/`

8 Contacts

Students
Alessandro Assirelli: alessandroassirelli98@gmail.com
Luca Del Giudice: luca.delgiudice@studenti.unitn.it
Andrea Di Luca: andrea.diluca@unitn.it
Mattia Sartori: mattia.sartori-2@studenti.unitn.it
Marcus Vukojevic: marcusvuk@gmail.com

Mentors
Luca Di Persio: luca.dipersio@hpa.ai
Stefano Di Persio: stefano.dipersio@hpa.ai

Company representative
Andrea Turso: andrea@niris.eu

References

[1] Meteo Trentino. *Dati storici e Mappa Stazioni.* URL: https://www.meteotrentino.it/#!/content?menuItemDesktop=141.

[2] Robert Nau (Duke University). *Lecture notes on ARIMA models.* URL: https://people.duke.edu/~rnau/411arim.htm.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.* 2018. arXiv: 1803.01271 [cs.LG].

[6] Sean J. Taylor and Benjamin Letham. *Forecasting at scale.* 2017. URL: https://facebook.github.io/prophet/.

[7] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. "MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery; Data Mining.* KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 248–257. ISBN: 9781450383325. DOI: 10.1145/3447548.3467231. URL: https://doi.org/10.1145/3447548.3467231.

[8] Jake Grigsby, Zhe Wang, and Yanjun Qi. *Long-Range Transformers for Dynamic Spatiotemporal Forecasting.* 2021. arXiv: 2109.12218 [cs.LG].

[9] Oskar Triebe, Hansika Hewamalage, and Nikolay Laptev. *NeuralProphet: a simple and interpretable forecasting library.* 2021. URL: https://github.com/ourownstory/neural_prophet.

[10] Meteo Trentino. *Analisi Meteorologica 2020.* 2020. URL: https://content.meteotrentino.it/analisiMM/Analisi_meteo_2020_10.pdf.

[11] Lisha Li et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[12] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: 1502.03167 [cs.LG].

[13] Günter Klambauer et al. *Self-Normalizing Neural Networks.* 2017. arXiv: 1706.02515 [cs.LG].

[14] François Chollet. *keras.* https://github.com/fchollet/keras. 2015.

[15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[16] Wikipedia. *Box-Jenkins Method.* URL: https://en.wikipedia.org/wiki/Box-Jenkins_method.

[17] Robert Nau (Duke University). *Lecture notes on statistical forecasting methods, Spreadsheet implementation of seasonal adjusting.* URL: https://people.duke.edu/~rnau/411outbd.htm.

[18] Rob J Hyndman and George Athanasopoulos. *Forecasting: Princples and Practice, 2nd edition.* OTexts, May 2018. Chap. 8.5 Non-seasonal ARIMA models. URL: https://otexts.com/fpp2/.