Université de Strasbourg

cemosis
Centre de modélisation et de simulation de Strasbourg

SYNAPSE CONCEPT
INGENIERIE & ETUDES TECHNIQUES

Project reporting
1^{st} year master CSMI

Building Data Processing

Ouachour Hanane Madani Sarra Amel

Under the leadership of:
Stoll Yannick , Djatouti Zohra
and
Prud'homme Christophe

May 2020

Acknowledgements

At the end of this work, we would like to express our deep gratitude to our supervisor **Mr. STOLL Yannick** for his follow up and for his support, which he did not stop giving us throughout the project. We learned a lot from him and appreciated the commitment he brought to this work.

We would also like to thank the people in charge of the project at CEMOSIS, **Mr. PRUD'HOMM Christophe**, the head of the CSMI specialty, and **DJATOUTI Zohra**, for the advice they gave us, which enabled us to overcome the organisational difficulties linked to the project.

We would also like to thank all those who have contributed in one way or another to this project. The work we have accomplished in this area is the expression of our deepest thanks.

Contents

List of Figures

1 Introduction

1.1 Context

In a recent report by the United Nations Environment Programme (UNEP) entitled "Buildings and Climate Change: State of Play, Challenges, and Opportunities", the international organization confirms that considerable gains can be made in the fight against global warming and that an appropriate mix of government regulations combined with greater use of energy-saving technologies and behavioural changes can significantly reduce carbon dioxide (CO2) emissions from the building sector.

Indeed, as the main priority of the French policy is to reduce greenhouse gas emissions, energy efficiency is now a new imperative for public companies. Achieving the objectives set by the Grenelle Environment Round Table by 2020 is based on regulatory and incentive systems that govern the construction, operation and renovation of buildings.

The application of BIM to most aspects of building design and operation has been studied in depth since it emerged as a generic term for the processing of data describing a building. Particularly in the design, simulation, and optimization of building performance, where plublications trends show an increased interest on BIM methodology these last years.

1.2 Ibat Project

iBat (Intelligent Building) is a deployment of connected objects in the context of commercially available intelligent buildings. It is composed of 200 connected objects spread over 10 levels of the ICube laboratory covering an area of more than 10,000 square meters. Each object is represented by a node made up of sensors which record the relevant physical fields (temperature, humidity, noise, light, motion detector) to study the energy performance of the building.[1]

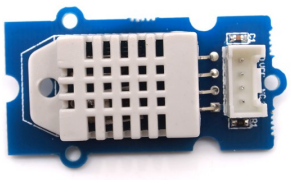

Figure 1: Temperature and humidity sensor model: Grove [2]

1.2.1 Presentation of the actors of the project

- **Cemosis**

  The Strasbourg Centre for Modelling and Simulation was created in 2013 by Christophe Prud'homme and supported by the IRMA Laboratory, the CNRS, IRMIA and AMIES Laboratories of Excellence. Cemosis offers expertise in the following fields: Modeling, Simulation and Optimization (MSO), High-Performance Computing (HPC), Signal and Image Processing (SIP), and Data Processing and Mining (DPM). Its objective is to be a driving force and an interface between mathematical research and academic research with companies, as well as to develop strategic partnerships (regional, national and international) with laboratories and companies on flagship projects that cover industrial fields such as automotive, energy, aeronautics, but also health fields.[3]

- **Synapse Concept**

  It is an engineering and technical studies company in the field of fluids, heating, air conditioner, ventilation, sanitary, and sanitation. Synapse Concept assists the client in the different phases of the project, it begins with the definition of the need for the project and calculates the dimensions to assess the impact of the project on the systems in place as accurately as possible, based on analysis of strengths and weaknesses of the facilities. After validation of the analysis phase begins the design stage by writing consultation plans and creation of detailed diagrams to have a good understanding of the project. [4]

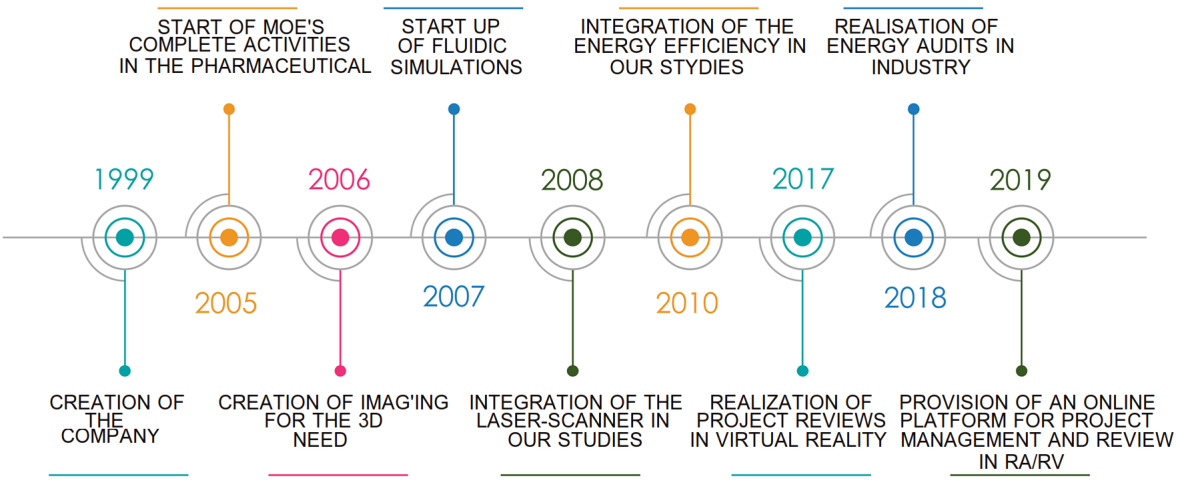

Figure 2: History of synapse concept [4]

1.2.2 Presentation of the project

The comfort level in the human body is an index that is always difficult to evaluate in a general and objective manner. Therefore, building owners and managers have been known to adjust environmental physical parameters such as temperature, humidity, and air quality based on people's subjective sensations to yield satisfactory feelings of comfort. Furthermore, electricity consumption could be reduced by minimizing the unnecessary use of heating and cooling equipment based on precise knowledge of comfort levels in interior spaces. To achieve the aforementioned objectives, this study undertook the following four tasks:

1. Providing visualization and smart suggestion functions to assist building managers and users in analyzing and developing plans based on the demands of space usage and electrical equipment.

2. Using Internet of Things technology to minimize the difference between real situations and those simulated in building information modeling (BIM).

3. Accurately evaluating interior environment comfort levels and improving equipment operating efficiency based on quantized comfort levels.

4. Establishing a persuasive workflow for building energy-saving systems.

The building studied in this project is one of the buildings of the API Cluster located in the Innovation Park of Illkirch-Graffenstaden. It houses several structures of the University of Strasbourg (Unistra) in the field of science and technology. It covers $45,000m^2$ on 6.2 hectares and has about 70 sensors that collect temperature, humidity, presence, light for 2 years inside and for a few months outside.

Figure 3: API building in Illkirch

1.2.3 Project's objective

The objective of this project is to provide numerical tools to process a 400 GB database in order to build virtual models of the behaviour of a building, by developing algorithms that will allow us to find missing data and thus be able to predict and simulate the results.

In order to best achieve our goals, we organize this under different tasks in the form of a roadmap :

1. Scaling up passage on the building :

   1.1. Benchmarking of DB queries
   Study of the database response: plot of the query execution time as a function of the number of nodes called for different typical period values.

   1.2. Implementation of data query by zones.

   1.3. Data aggregation by zones
   Implementation a data aggregation algorithm for a given field one serie per field and per zone. For example, one could take a volume-weighted average of the different rooms within a given zone.

   1.4. Testing the Random Forest (RF) algorithm for completion of missing data by zone

   - Survey of data zone by zone with study of the "out of bag" score (oob).
   - Study of the accuracy of the prediction algorithm over a few series by separating the data into training and test sets.

   1.5. Test of the data smoothing algorithm by field

   - Study of the root mean square error between raw and smoothed data.
   - Make the link with prediction algorithms (acuracy, seasonality etc...) i.e. to what extent does data smoothing improve the metrics of the prediction algorithms?

2. Descriptive statistics:

   2.1. Studies of the different quantities of interest per zone, i.e. average, median etc... for the different fields. We will also visualize and discuss the relevant histograms and the boxplots for the fields of interest (temperature, humidity etc...) for each zone.

   2.2. Studies of correlations between fields within the same area: for a given area, which fields are correlated with each others?

2.3. Study of correlations between zones: repeat the previous work for inter-zone correlations. For example, how well does the temperature in one zone correlate with that of an adjacent zone?

3. Missing data (extension to the case of LSTM neural networks):

   3.1. Implementation of a LSTM neural network for time series prediction.

   3.2. Comparison of LSTMs with random forest algorithms. For example, which algorithm has the best accuracy when predicting missing data?

4. Storage of statistical models and automation(optional):

   4.1. Implement the backup of the learned LSTM models, for later reuse.

   4.2. Implement a script which automates the querying of the database at regular intervals, and which updates the predictions of interest(temperature, humidity for each zone, associated statistics etc. ...).

5. Define measurement zones in accordance with the multi-zone model:

   5.1. Define the areas.

   5.2. Adaptation to the multi-zone model.

In order to have a visibility on the progress of the project, we have set deadlines for each of our objectives :

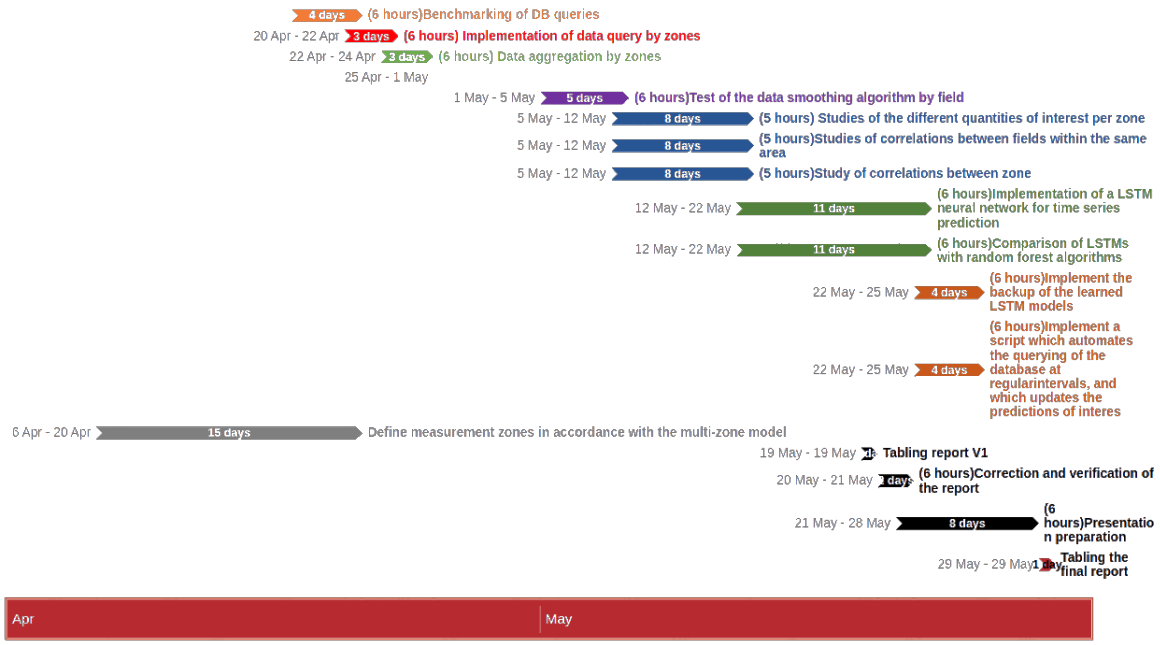

Figure 4: Gantt diagram of our project

To execute the tasks previously mentioned, we will use several methods, including Machine Learning techniques such as Random Forest, as well as different tools such as Github, Virtual Studio Code, Jupyter, and ElasticSearch.

2 Working environement

2.1 Github

Teamwork requires organization, for this we used Github. It facilitates collaboration using Git which is a distributed version control system that design to manage every type of project. This software allows us to publish new code modifications and to share functional versions of the project.

The source code of our project is hosted on Github, which allowed us to work in security while staying synchronised .

2.2 Virtual Studio Code

Visual Studio Code is an open-source code editor developed by Microsoft supporting a large number of languages thanks to extensions (python, c, Html,..). It supports syntax highlighting, debugging, and git commands. It is available for Windows, Linux, and macOS [5].

In our project, we use the python language which has high-level data structures and allows a simple but effective approach to object-oriented programming.

2.3 Docker

Launched in 2013 by Solomon Hykes, Docker is an open-source project allowing us to launch applications in software containers. The whole application is then inserted into a very light virtual container and run on a Linux server. It ensures a faster deployment of the application and all the management of the structure [6].

For our project, we used a Dockerfile script that contains all the commands and arguments that could be called on the command line to assemble an image and the various packages necessary for the realization of the project (elasticsearch, numpy, pandas, matplotlib,...)

The figure 5 below shows the use of the docker environment of our project under vscode :

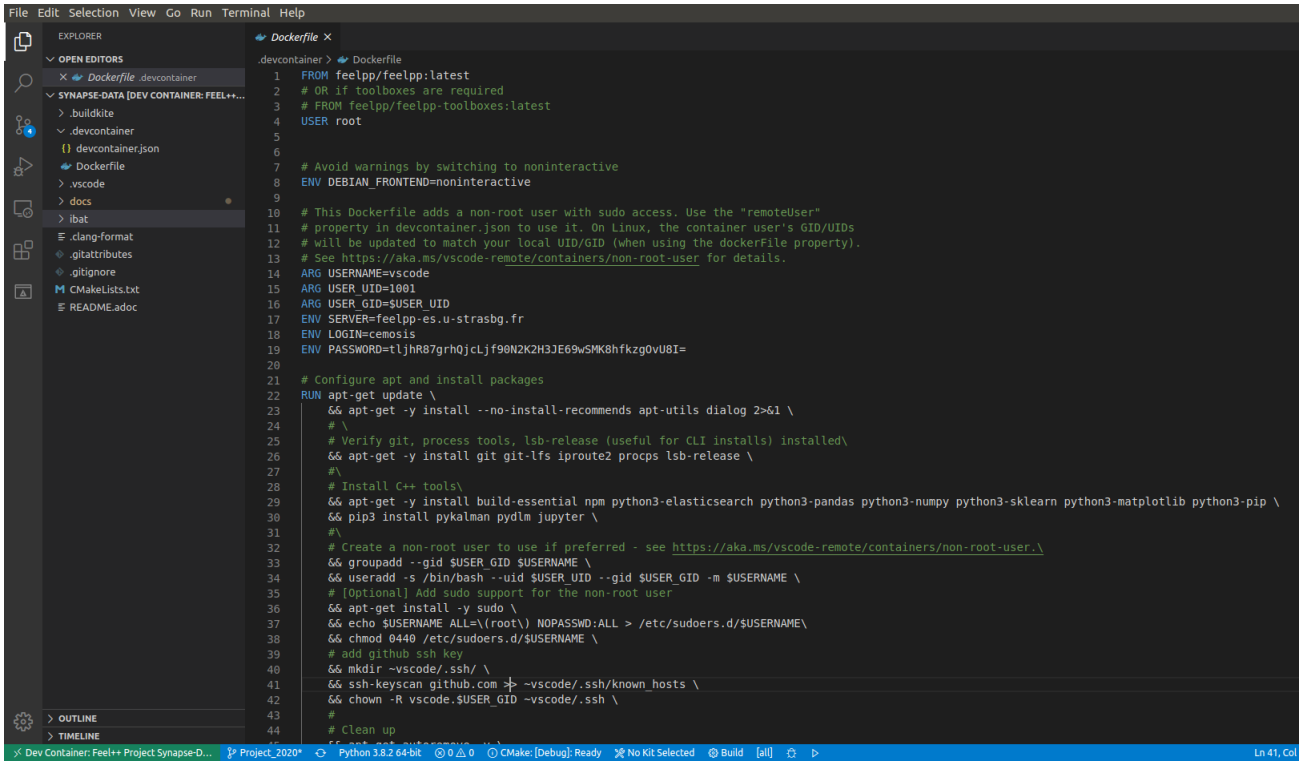

Figure 5: Dockerfile overview on vscode

3 Scalling up passage on the building

This part constitutes an important phase in the realization of our project. It consists of benchmarking different queries in order to elaborate an efficient study on the database. For this, we will need different tools that will allow us to perform this task.

3.1 Elasticsearch

The measurements collected by the sensors in our project are uploaded via the network to the Elasticsearch server which will allow us to extract data from the ibat database contained in the server.

Definition : Elasticsearch is a search engine distributed in real time and a very powerful analytical tool used to do full text and structured research. It was created by Shay Banon the founder of Compass Project in 2004. It is a software written in Java and published in open source under Apache license that is based on the Apache Lucene library .

It allows to carry out and combine various searches on structured, unstructured, geolocation or indicator data and display the search results in JSON format [7].

Configuration : To access the database, the environment variables SERVER, LOGIN and PASSWORD must be defined.

3.2 Database query benchmarking

3.2.1 Database query by nodes

We have adapted the existing script that extracted data from the database and exported it to a csv file so as to leave the choice to the user to select the set of nodes to be studied in order to calculate the execution time of the query according to these nodes. To achieve this, we can execute the following command :

```
python3 ibat_query.py --field "temperature" "humidity" "light"
"sound" "pir" -sd 2019-01-01T00:00:00Z -ed 2019-01-31T23:59:59Z
--in 60 -c 1,4,7-10,81
```

This will request the temperature, humidity, light, sound and pir fields between 2019-01-01 and 2019-01-31, every 60 minutes, on nodes 1,4,7,8,9,10 and 81. The results will be exported in a csv file, called export.csv, by displaying the values of the different fields for the different nodes.

Then we have created a script benchmarking that performs several tasks, among them plotting the execution time of the query according to the number of nodes called for different typical period values, in order to study the response of the database.

Here is an example of a command that, when executed, exports the results in a csv file named time-elm-freq.csv and outputs the figure 6 which represents the execution time of the query in function of the different period values 30, 60, and 120 minutes for the 30 first nodes.

```
python3 benchmarking.py -o node -v -in 30,60,120 -n 30
```

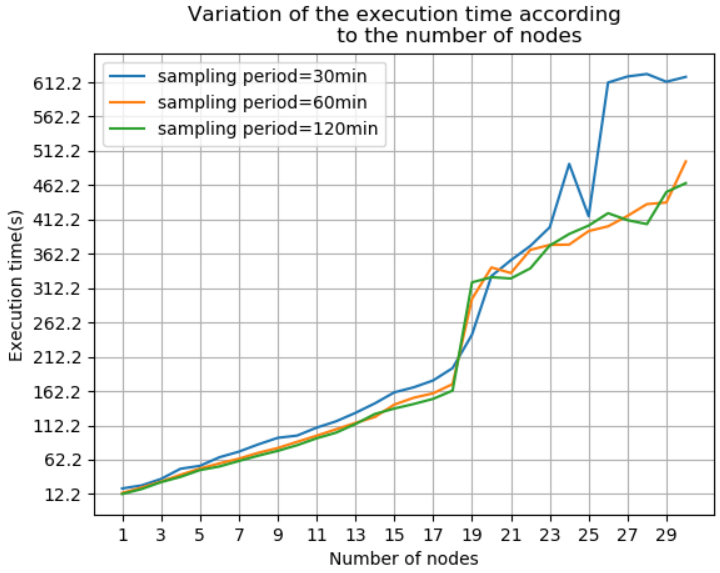

Figure 6: Representation of the execution time for 30 nodes for each period

By examining this graph, we can see:

- a positive correlation between the number of query nodes and the query execution time. Indeed, the study of the database response takes more time when more nodes are queried.
- a strong increase of the execution time when more than 18 nodes are queried. This can be due to the database management.
- a negative correlation between the query execution time and the sampling period. As the latter decreases, the execution time increases.

3.2.2 Database query by zones

In this part, we will implement in the ibat query script the query of data by zones.

For this, we used the multi-zone model created by *Djatouti Zohra* which allowed us to visualize the distribution of the different zones of the studied building:

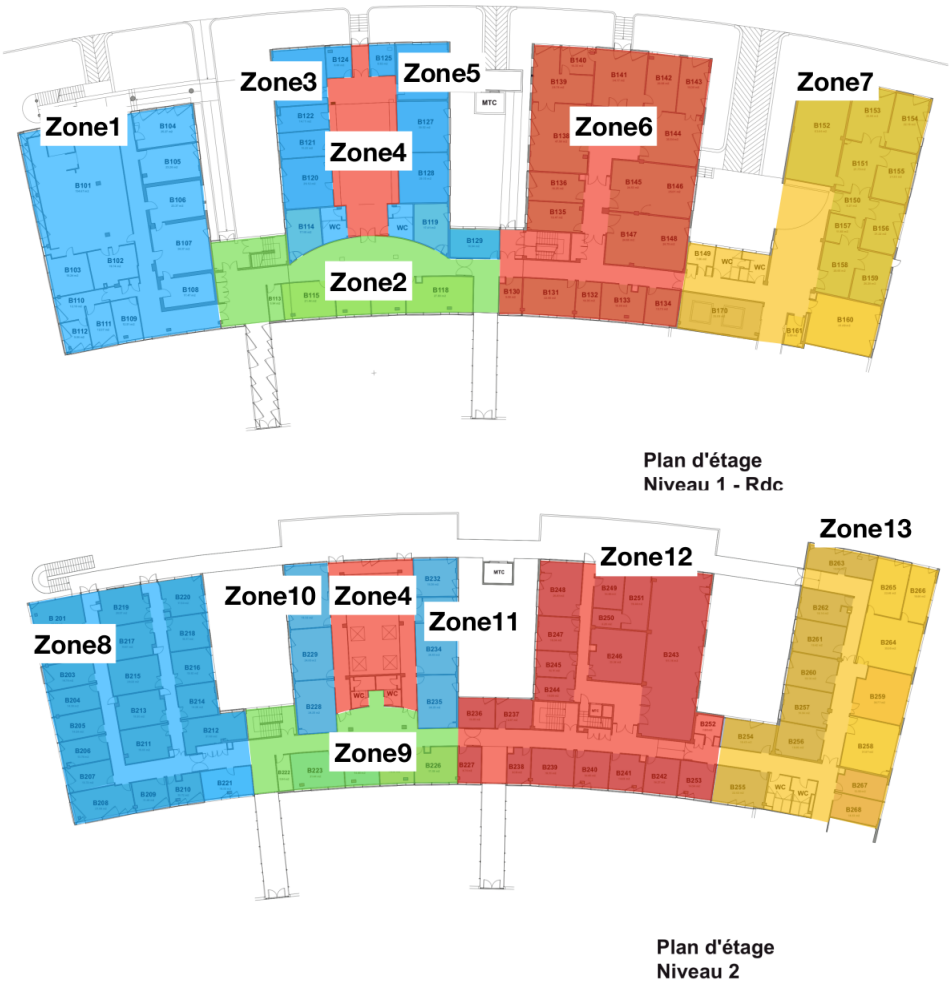

Figure 7: Preview of the multi-zones model courtesy of Z. Djatouti

The building is oriented in such a way that the zones 7 and 13 are on the north and zones 1 and 8 are on the south. On the other hand, the east is on the side of zones 2 and 9, and the west is on the other side.

The script will then have to read a json file that defines the nodes associated to each zone and then propose to the user an option that allows to select zones to be requested.

The different functions that allow to do this are implemented in script tools.

For exemple, if we want to request the temperature, light, and sound fields between 2019-03-16 and 2019-06-20, every 60 minutes, on zones 1,4,9,10,11,12 and 13 and output the results in the export.csv file , the following command must be executed:

```
python3 ibat_query.py --field "temperature" "light"  "sound"
-sd 2019-03-16T00:00:00Z -ed 2019-06-20T23:59:59Z --in 60 -z 1,4,
9,10-13
```

3.3 Imputation and data smoothing

3.3.1 Missing data

In a database, data may be missing : not all individuals are captured, which can introduce significant bias, make data processing and analysis more laborious and reduce the efficiency of statistical methods. What makes things a bit more complex is that there are several ways to examine and process these missing data depending on the case (this is called imputation). In our algorithm, we will replace the missing data with random forest regression values. Once the data are imputed, we apply the smoothing algorithm that uses the kalman filter.

3.3.2 Random Forests

Random forests are methods for obtaining predictive models for classification and regression.They implement binary decision trees, including CART trees proposed by Breiman et al. (1984).
The general idea behind the method is that instead of looking for the most important functionality when splitting a node, it looks for the best functionality among a random subset of functionalities. This results in a large diversity that usually leads to a better model.[8]

- **Maximum Tree Construction** : A decision tree is a scheme representing a prediction model, it is composed of 3 elements:
  - Node : each node of the tree tests a condition on a variable, this node is a separator variable by which the data will be partitioned.

- Leaf : correspond to a label. To predict it, we cross the tree of the root by following the answers to the test until we get to that label.
- Branch : represents a test result.

Here is an example that illustrates the representation of a decision tree :

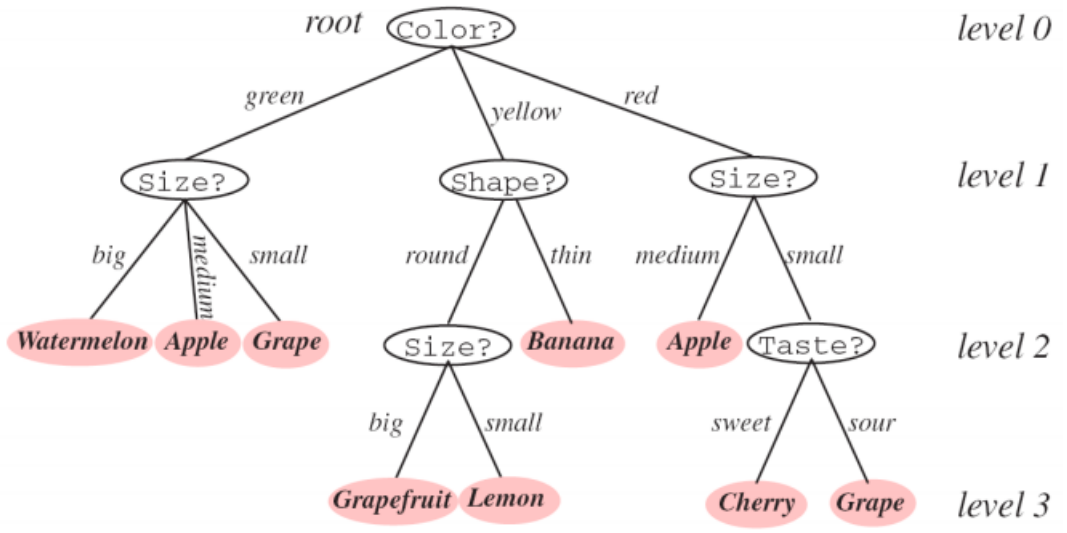

Figure 8: Exemple of a decision tree [9]

The principle of this method is [10] :

1. Taking a study space R^p where p is the number of variables.
2. Start from the root of the tree, which contains all the observations of the learning sample.
3. Cutting at best this root into two child nodes, so we follow an iterative process which consists in devising each time the elements of the nodes which checks the given criteria in two subsets.
4. Repeat the step 3 until the maximum tree-top depth is reached or the predictions are not improved.

The cutting criterion consists in choosing at each iteration the cutting point (j,s) which minimizes a certain cost function with $j \in \{1, ..., p\}$.
Assuming n observations $\overrightarrow{x}^1, ..., \overrightarrow{x}^n$ of space X labeled by $y^1, ..., y^n$, and r regions $R_1, ..., R_r$:

- **In regression:** We try to minimize the quadratic error.

$$\underset{(j,s)}{\operatorname{argmin}} \left(\sum_{i:\overrightarrow{x}^i \in R_l(j,s)} (y^i - y_l(j,s))^2 - \sum_{i:\overrightarrow{x}^i \in R_r(j,s)} (y^i - y_r(j,s))^2 \right)$$

Where $y_l(j,s))$ (resp.$y_r(j,s)$) is the label associated with the region$R_l(j,s)$ (resp $R_r(j,s)$)).

- **In classification:** we try to minimize the impurity of a class.

$$\underset{(j,s)}{\operatorname{argmin}} \left(\frac{|R_l(j,s)|}{n} \operatorname{Imp}(R_l(j,s)) - \frac{|R_l(j,s)|}{n} \operatorname{Imp}\left(R_l(j,s)\right) \right)$$

Such as :
$\operatorname{Imp}(R) = \sum_{c=1}^{C} p_c(R)(1 - p_c(R))$ represents Gini's impurity ($p_c(R)$ indicates the proportion of R region training examples that belong to class c).

- **Bagging** One of the problems with the construction method described above is that the maximum tree has a very high variance and low bias.
To solve this problem, Leo Breiman in 1996 proposed the Bagging (bootstrap aggregating) whose steps are presented as follows[11]:
  - Generating from the set of random subsets formation with discount.
  - Building a decision tree for each subset.
  - Combining the predictions of the models obtained by averaging (for a regression) or voting (for a classification).

3.3.3 Kalman filter

The Kalman filter, named after its Hungarian inventor Rudolf Kalman, and also known as linear quadratic estimation, is a recursive estimator that uses an observed time series. It estimates the current state, using estimates of the previous state and current measurements.
The state of the filter is represented by 2 variables :

- $\hat{x}_{k|k}$: the estimated state at time k.
- $P_{k|k}$: the error covariance matrix.

The Kalman filter has two distinct phases :

- **Prediction** : uses the estimated state of the previous instant to produce an estimate of the current state.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \text{ (predicted state).}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \text{ (predicted estimate of covariance).}$$

such as :

- F_k : matrix that links the previous state k-1 to the current state k.
- u_k : command input.
- B_k : matrix which connects the control input u to the state x.
- $P_{k|k-1}$: a priori error covariance estimation matrix.
- Q_k : process noise covariance matrix .

- **Update** : current time observations are used to correct the predicted state to obtain a more accurate estimate.

$$\tilde{y}_k = z_k - H_k\hat{x}_{k|k-1} \text{ (innovation)}$$
$$S_k = H_k P_{k|k-1} H_k^T + R_k \text{ (innovation covariance)}$$
$$K_k = P_{k|k-1} H_k^T S_k^{-1} \text{ (Kalman optimal gain)}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k \text{ (updated status)}$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \text{ (updated covariance)}$$

with :

- z_k : observation or measurement of the process at time k.
- H_k : matrix that links state x_k to the measurement z_k.
- $P_{k|k}$: a posteriori error covariance estimation matrix.
- R_k : measurement noise covariance matrix.
- I : appropriately sized identity matrix.

The formula for the covariance update is valid only for an optimal Kalman gain. The use of other gain values requires more complex formulas.[12]

3.3.4 Application

As part of our project, we have performed some tests on the efficiency of the imputation and smoothing algorithms that allow us to fill in missing data at the database level.

- **Imputation Algorithm** : The class which implements the RF algorithm used for data completion is avavaible in Impute. This has allowed us to impute the missing data from the export.csv file.
  The following command allows us to impute the missing data associated with zones 1,2,7,8,9,10,11,12, and 13, for the light and sound fields every 30 minutes and export the results in the export-test.csv file :

```
python3 ibat_query.py --field "light" "sound" -in 30
-sd 2019-02-01T00:00:00Z -ed 2019-08-28T23:59:59Z
-z 1,2,7-13 --impute
```

Once the results are obtained, we can choose between plotting the total execution time (query and application of the imputation algorithm) on the data against :

1. The number of zones for each period.

   By executing the following command:

```
python3 benchmarking.py -in 30,60,120 -v -o zone -n 13
--data_algo impute
```

   The results will be stored in the time-elm-freq.csv file and ploted as follow :

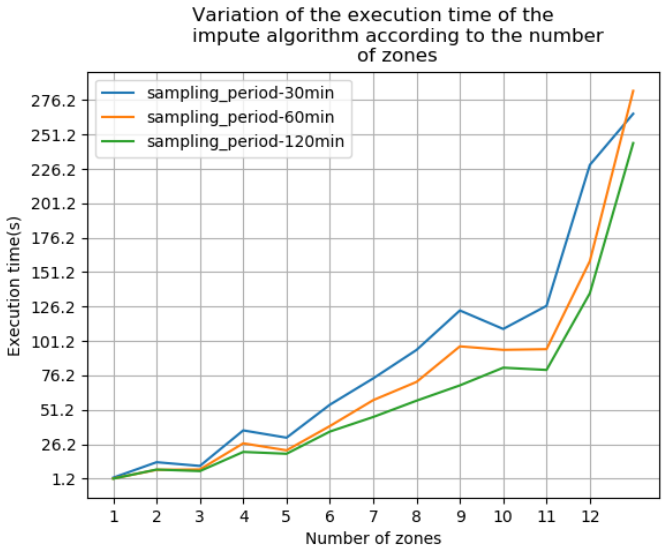

Figure 9: Representation of the execution time for 13 imputed zones for each period

   Such that the total execution time will be based on the number of zones to be requested (up to 13 zones) for the 30 min, 60 min and 120 min sampling periods.

   We can observe on the graph 9 the same remarks as those concerning the benchmarking on the nodes, except that here we have zones, and each zone contains several nodes.

2. The number of moths for each zone.

   By executing the following command:

```
python3 benchmarking.py -in 1440 -v -o zone -sd 2019-01-01
T00:00:00Z -ed 2019-12-31T23:59:59Z -z 4,6,8,9,13
--data_algo impute
```

We will obtain results stored in the time-date-freq.csv file and then will get the plot which is shown in the figure 10 below:

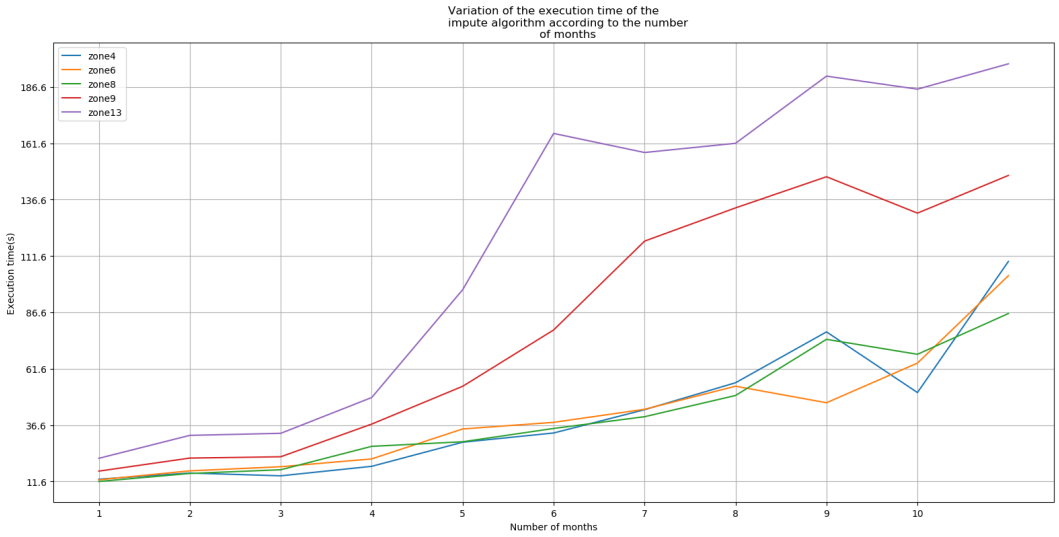

Figure 10: Representation of the execution time for 1 year for each imputed zone

This figure represents the total execution time (query + application of the imputation algorithm) as a function of the number of months spent during a year (from 01/01/2019 to 31/12/2019) for a sampling period of one day, and this for the different zones selected (4, 6, 8, 9, and 13).

We can see that :

- as the number of months increases in the query, the total execution time also increases because it means studying more data.
- the total execution time taken for zone 13 is more important compared to the other selected zones because it includes more nodes compared to the other zones (4 nodes). Then comes the zone 9 which includes 3 nodes.
- the rest of the zones (4, 6, 8 and 9) to be queried and to which the imputation algorithm has been applied take almost the same execution time and are faster. This comes down to the fact that they comprise only 2 nodes.

- **Smoothing Algorithm** : For this algortihm, the script associated is represented in the KalmanSmoother class. Once the data is completed, this algorithm will allow us to "smooth" the data, i.e. to unruffle data that are basically completed.

  The following command allows us to smooth the data associated with zones 2, 6, 9, 10, 11, and 13, for the light and sound fields every 30 minutes and export the results in the export-test.csv file.

```
python3 ibat_query.py --field "pir" "humidity"  -in 30
-sd 2019-01-16T00:00:00Z -ed 2019-10-16T23:59:59Z  -z 2,6,9-11
,13 --impute --smooth
```

On the other hand, by executing the following command, we will calculate the total execution time (query and the application of the imputing and smoothing algorithms) on the data against :

1. The number of zones for each period :

```
python3 benchmarking.py -in 30,60,120 -v -o zone -n 13
--data_algo smooth
```

The results will be stored in the time-elm-freq.csv file and then ploted like the figure below:

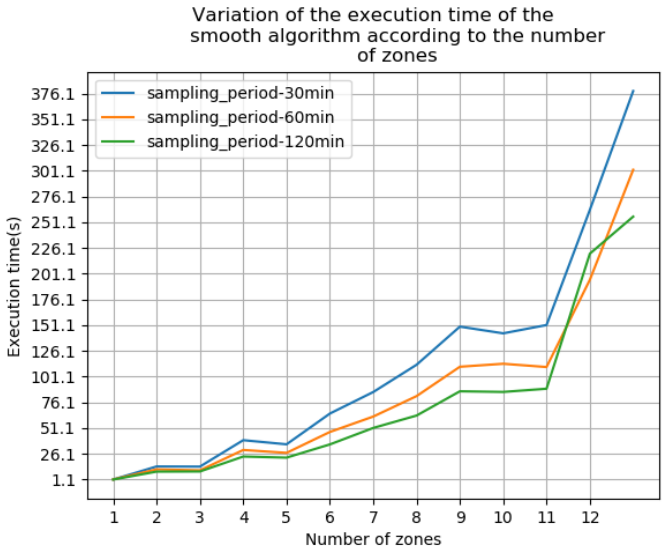

Figure 11: Representation of the execution time for 13 smoothed zones for each period

This figure represents the total execution time (query + algortihm imputation + smoothing algortihm) as a function of the number of zones to be queried (up to 13 zones) for the different sampling period values (30 min, 60 min and 120 min).

We observe the same variability and the same remarks as in the case of the imputation algorithm, except that it takes a bit longer because in this case, we first have to fill the missing data using the imputation algorithm and then smooth the data by applying the smoothing algorithm.

On the other hand, we can see that the application of the smoothing algorithm takes less time than the application of the imputation algorithm.

2. The number of months for each zone :

```
python3 benchmarking.py -in 1440 -v -o zone -sd 2019-01-01
T00:00:00Z -ed 2019-12-31T23:59:59Z -z 4,6,8,9,13
--data_algo smooth
```

The results will also be stored in the time-date-freq.csv file and ploted as shown in the figure below that the total execution time (query + application of the allocation algorithm and the smoothing algorithm) as a function of the number of months in a year for a period of one day, for each selected zone :

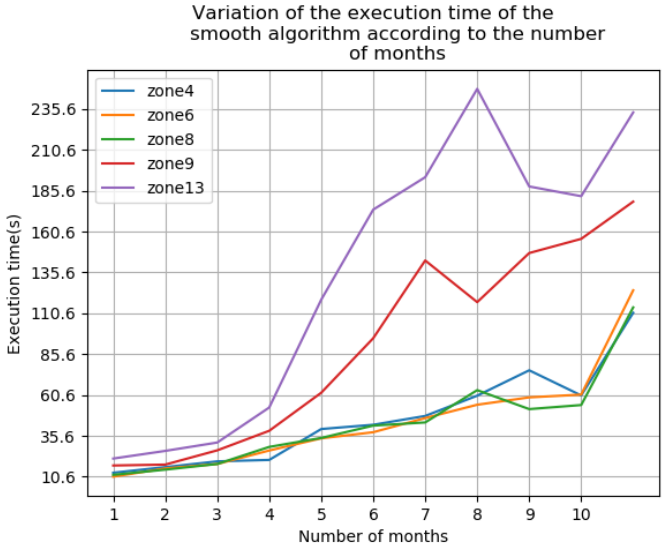

Figure 12: Representation of the execution time for 1 year for each smoothed zone

We notice that the total execution time here has the same behaviour as in the case of the imputation algorithm, except that it takes more time due to the application of the smoothing algorithm in addition to the imputation algorithm.
But, we also notice that the application of the smoothing algorithm only takes less time than the imputation algorithm.

3.4 Data aggregation

In this part, we have implemented the aggregation of data by fields. We used an area-weighted average : for a given zone, and for a given field, an average is computed, weighted by the rooms' areas associated to each node in the zone of interest.

To obtain these results, the following command can be executed:

```
python3 ibat_query.py --field "temperature" "humidity" "light"
 "sound" "pir" -sd 2019-04-01T00:00:00Z -ed 2020-03-31T23:59:59Z
 --in 1440 -z 4,6,8,9,13 --impute --smooth -a
```

The option "-a" will aggregate the temperature, humidity, light, sound and pir fields between 2019-04-01 and 2020-04-30, for a sampling period of one day, on zones 1,4,9,10,11,12 and 13 from the file export-test.csv which results from the application of the imputing and the smoothing algorithms and output the results in the aggregated-export.csv file.

3.5 Conclusion

In this part, we carried out a study on the behaviour of the ELASTICSEARCH database by performing different benchamrking. This allowed us to see that the more the number of nodes to request increases, the more the execution time of the request increases. This is also the case when performing a grouping of several nodes per zone.

We also noticed that the sampling period played an important role on the time to query. Indeed, a period value that is too small in relation to the study duration will take more time to query.

Subsequently, we applied the imputation and smoothing algorithms in order to complete the missing data at the database level. Benchmarking against study duration was applied. The execution time was found to be positively correlated with the number of queried months.

We will use the aggregated-export.csv file obtained to perform a statistical analysis on the different results obtained for the selected zones (4, 6, 8, 9 and 13) that we will study in the following section.

4 Descriptive statistics

In our project, we have used descriptive statistics to provide a statistical summary of the results of the aggregation of zones. This has allowed us to rigorously study time series of interest by observing values at regular time intervals.
To achieve this, we used the Jupyter notebook

4.1 Jupyter notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents containing Markdown formatted text or executable computer code. It was developed for the programming languages Julia, Python and R, now supporting nearly 40 different languages [13].
Uses of jupyter nootebook include: data cleansing and transformation, numerical simulation, statistical modeling, and data visualization.

4.2 Data treatment

In this section, we will execute different functions that are implemented in the jupyter notebook analyses-data, and that will allow us to make an adequate statistical analysis on the selected zones (4, 6, 8, 9 and 13) during one year for a sampling period of one day. We can found it in the jupyter notebook results-analyses-data.

In order to exploit the data present in the aggregated-export.csv file, we retrieved these data in the form of DataFrame using the **panda** library.

Our goal is to study the different correlations between :

1. Temperature and humidity fields for each zone.
2. The differents selected zones for earch field.

In order to better compare the different distributions, the correlation matrix between the variables is trained :

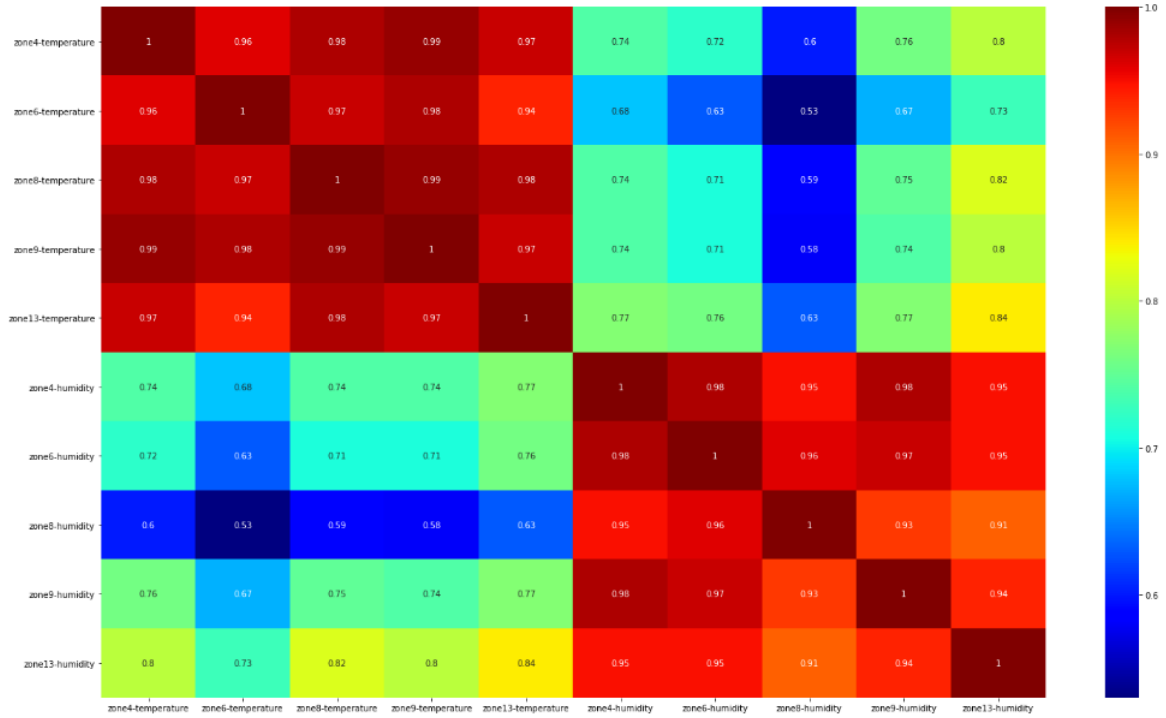

Figure 13: Correlation matrix

For the purpose of highlighting the correlation between temperature and humidity in the same zone, we performed some scatter plots :

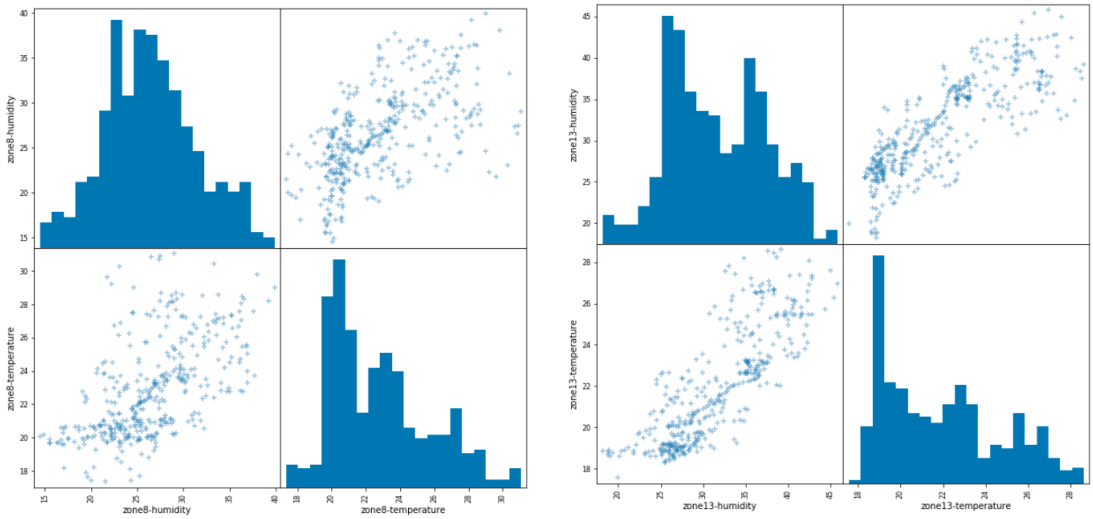

Figure 14: Scatter plots

We notice that there is a strong correlation between humidity and temperature variables, both between zones and within zone.

The histograms allow a better visualization of the relationship between the "temperature" and "humidity" fields and the different zones selected:

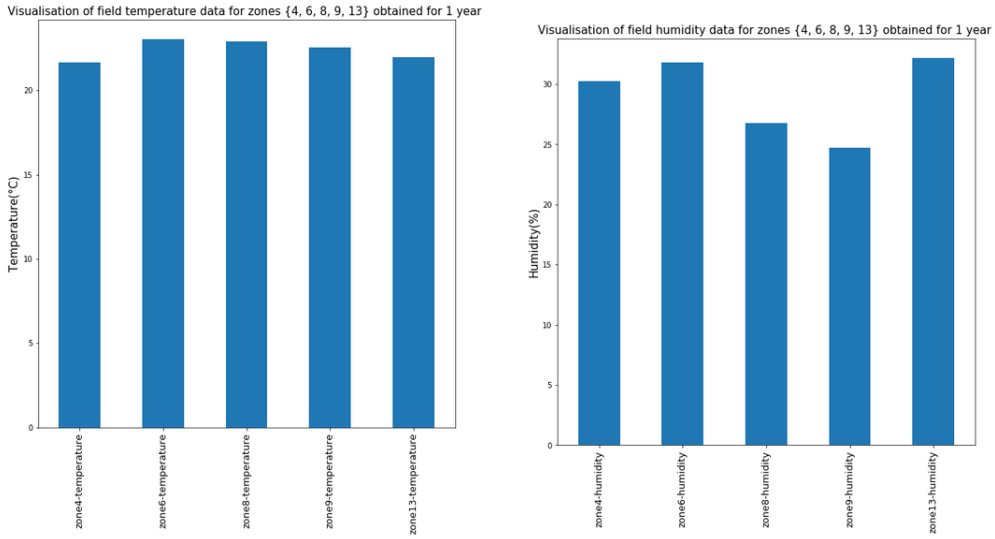

Figure 15: Histogram plots

By analyzing these histograms, we can see that :

- The temperature varies between 20°C and 25°C for all zones but zones 6 and 8 are slightly warmer because zone 6 benefits from the sun on the east side during the morning and on the west side during the afternoon. Zone 8 is located to the south of the building and is therefore exposed to the sun on the east and west sides and therefore to heat throughout the day. On the other hand, zone 4 is the least hot because it is a hall.

- Zone 13 is the wettest because it is located to the north of the building and is therefore generally not illuminated by the sun. On the other hand, zone 9 is the least humid because it is located to the east of the building and doesn't have a lot of office space. The entropic effect due to the presence of people is then very low, resulting in a low humidity level.

The distribution of the different values of the "humidity" and "temperature" fields for zone 9 is shown in the following histograms:

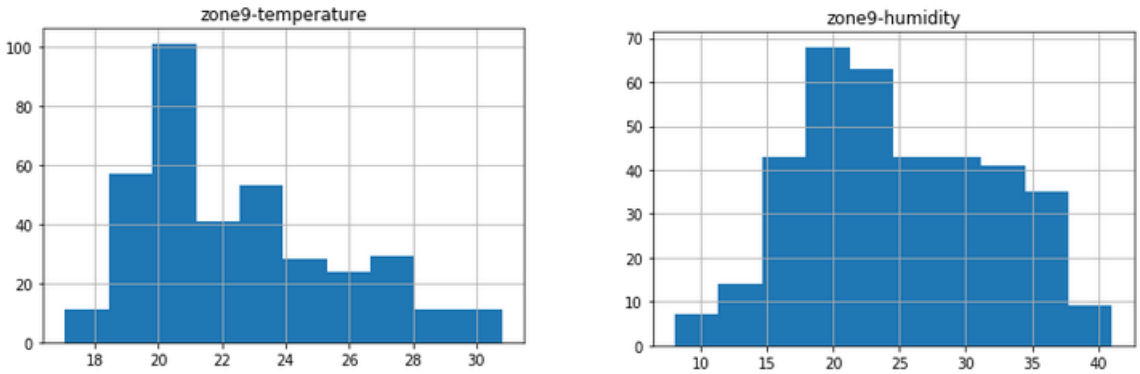

Figure 16: Distribution histogram plots

We can see that the dominant temperature during the year in this zone is between $20°C$ and $22°C$, while the dominant temperature is $18°C$ or between $28°C$ and $30°C$. With regard to humidity, it can be seen that values between 19 % and 25 % dominate the other values throughout the year, while values between 8 % and 12 % are dominated.

The following visualization gives a clearer idea of the evolution of the different physical fields over time :

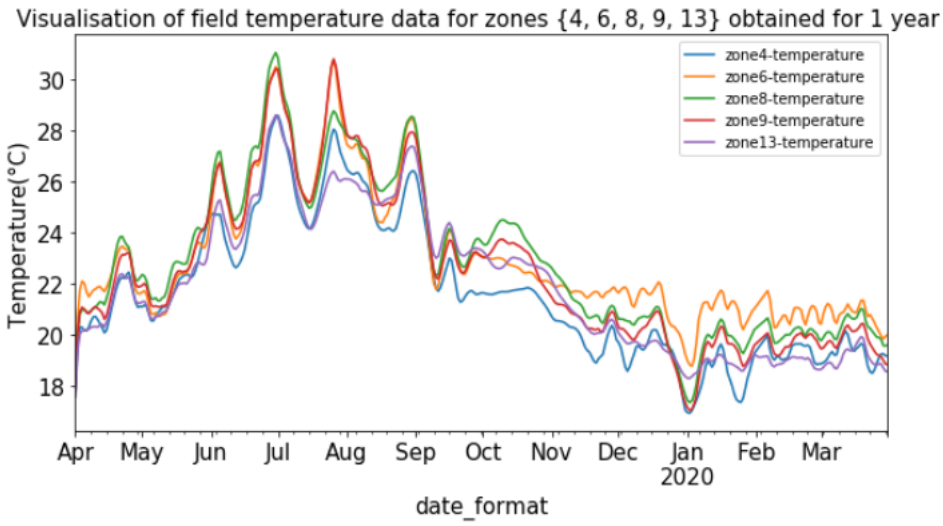

Figure 17: Temperature time series

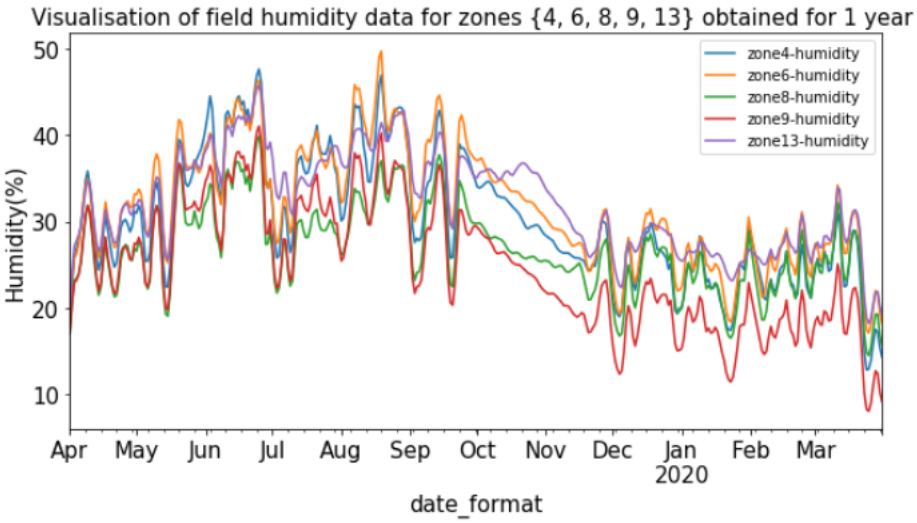

Figure 18: Humidity time series

We can see on these time series that the two fields "temperature" and "humidity" have the same behavior for all the zones; i.e., the values of these last ones increase in summer period and decrease in winter period.

And in order to have all the necessary information to carry out our study on the temperature and humidity fields, it is necessary to carry out a descriptive statistic which is represented in the table below:

	zone4-temperature	zone4-humidity	zone6-temperature	zone6-humidity	zone8-temperature	zone8-humidity	zone9-temperature	zone9-humidity	zone13-temperature	zone13-humidity
count	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000
mean	21.560652	29.496914	22.977991	31.323366	22.810699	26.461992	22.493388	24.129754	21.946304	31.628850
std	2.651421	7.282262	2.574530	6.741111	2.893586	5.008877	3.032705	7.264667	2.731899	5.769146
min	16.943562	12.838175	18.352839	17.109582	17.360579	14.496885	17.055071	8.049605	17.581695	18.208480
25%	19.533778	24.519830	21.177225	26.350327	20.538768	23.048838	20.083176	18.450160	19.409913	26.921648
50%	21.056972	28.272204	22.022879	30.358834	22.301358	26.128651	21.904181	22.960654	21.481375	30.992802
75%	23.020415	34.568078	23.898520	36.497836	24.510079	29.666520	24.159537	29.785736	23.774466	36.159629
max	28.596384	47.638227	30.768663	49.582633	31.052653	39.900402	30.804491	41.049224	28.612000	45.763940

Figure 19: Descriptive statistics

The boxplots are then called upon to represent this :

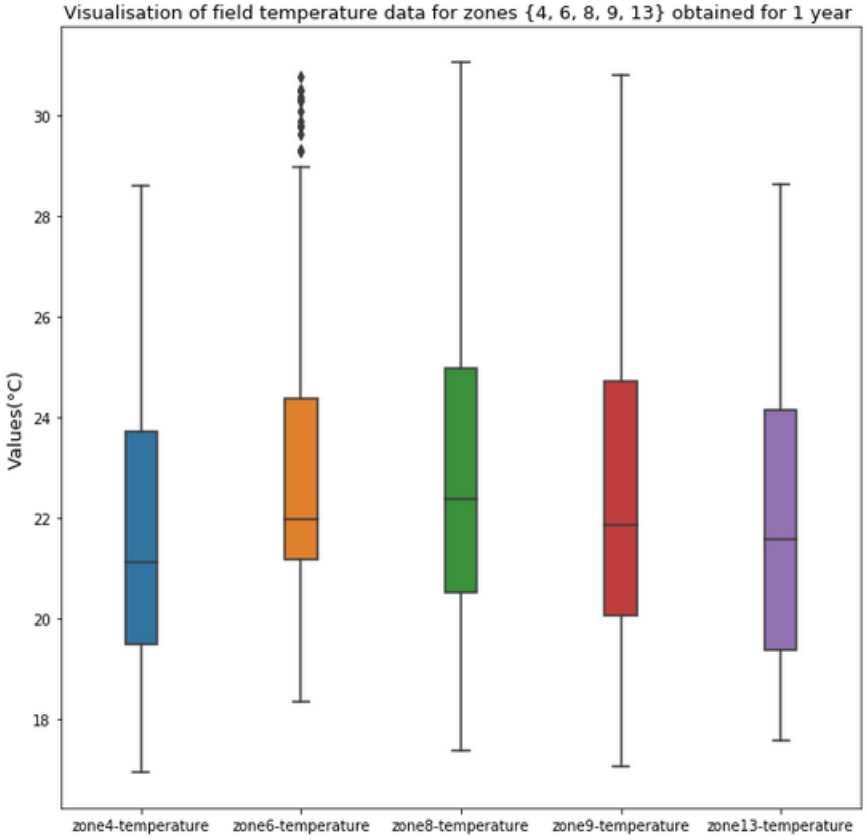

Figure 20: Temperature zones boxplots

Concerning the temperature, we notice that :

- the zone 13 has greater variability in temperature values than the other zones and the distribution of its values is symmetrical.
- the location of the box which represent the zone 8 (resp. 4) indicates that this zone contain the maximum (resp. minimum) temperature value.
- the box which represent the zone 6 contains outliers (extreme maximum). This means that during the study period, unusual high temperatures were recorded.
- the median values are different for all zones, and the moustaches of their boxes are asymmetrical.

For the humidity field, we can see in the boxplot below that :

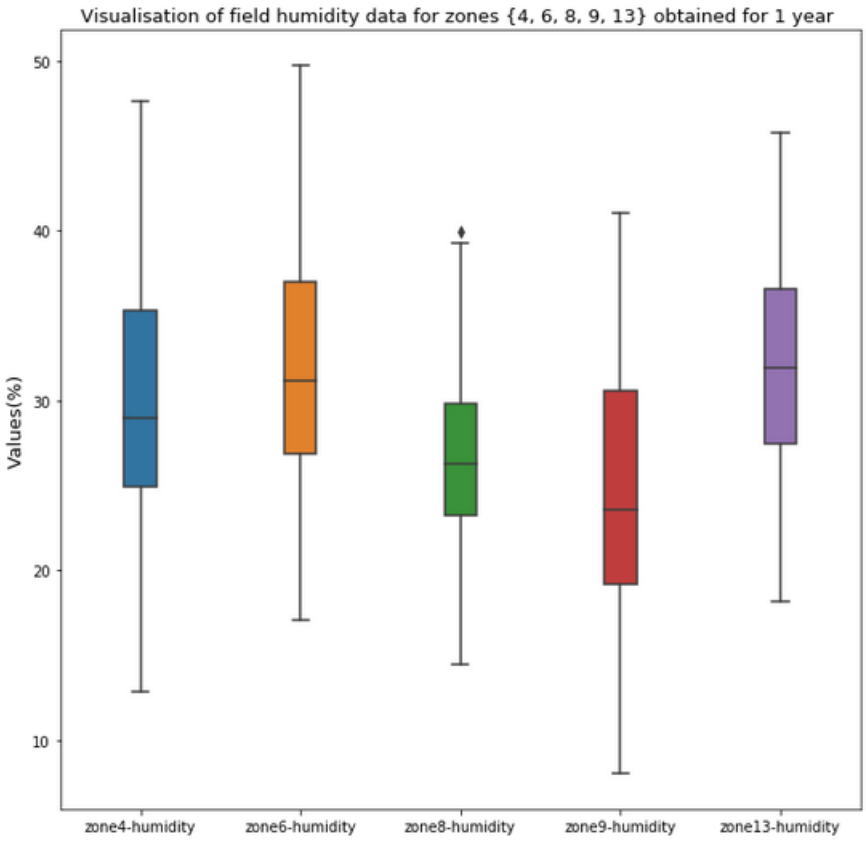

Figure 21: Humidity zones boxplots

- zone 9 has more humidity variability than the other zones.
- the location of the box which represent the zone 6 (resp 9) indicates that this zone contains the maximum value of humidity (resp minimum).
- the box which represent the zone 8 contains outliers (extreme maximum) which means that during our study, there are days when there are more people in the offices, so there may be an entropic effect due to the presence of people or climate change.
- the median values are different for all zones and the moustaches of the boxes that represent the zones 4 and 13 are symmetrical.

The following boxplots represent the variation of the values of the temperature and humidity fields for zone 4 for each month of the year:

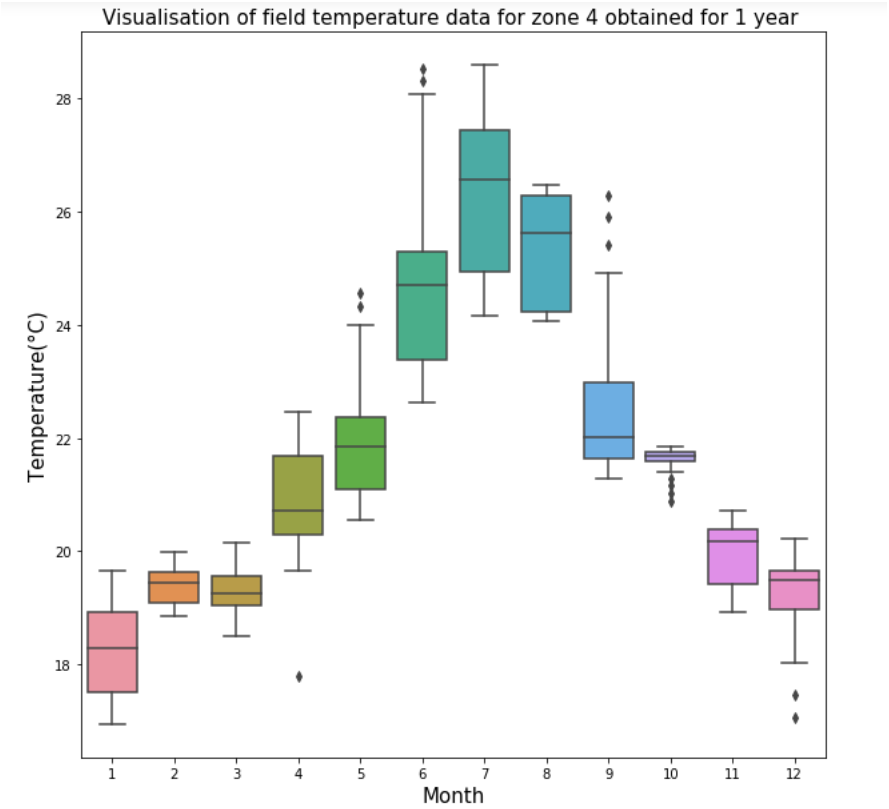

Figure 22: Zone 4 temperature boxplot

From the analysis of the boxplot of the zone 4 temperature, we have come to the following conclusions :

- the zone 4 has more temperature variability in summer.
- the location of the boxes for the month of July (resp January) indicates that this period contains the maximum temperature value (resp. minimum).
- the boxes corresponding to the months of May,June and September contain outliers (maximum extreme). This means that during these months, there are days when the temperature is higher than usual.
- The boxes corresponding to the months April, October and December contain outliers (minimum extreme). This means that during these months, there are colder days than usual.
- the median values are different for each month and all the moustaches of the boxes are asymmetrical.

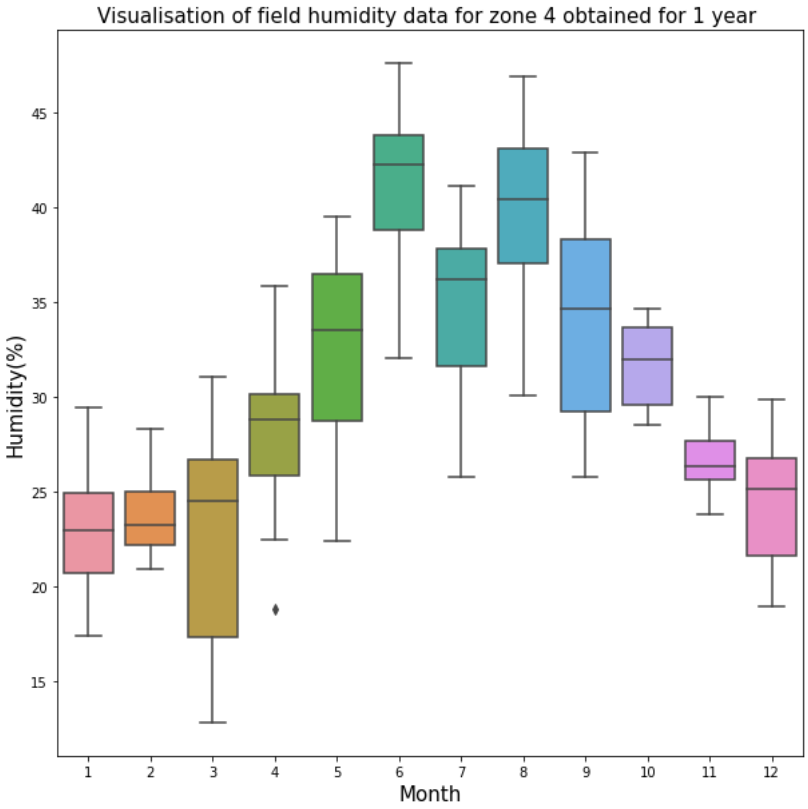

Figure 23: Zone 4 humidity boxplot

After studying the boxplots of the humidity field in zone 4, we found that :

- the zone 4 has more humidity variability in March and September .
- the location of the boxes corresponding to the month of June indicates that this box contains the maximum temperature value. Because in summer, the air is hot and humid and the halls are less ventilated, resulting in high humidity.
- the boxes corresponding to the month of march indicate that this box contains the minimum humidity value.
- all the boxes do not contain extreme neither minimum nor maximum, except the box which represent the month of April.
- the median values are different for each month and the moustaches of all the boxes are asymmetrical.

5 Conclusion

Thus we conclude this master project **"building data processing"** which was presented in this report and where a set of tasks was carried out.
The benchmarking of the requests on the database allowed us to study the behavior and performance of this database. We also exploited the multi-zone model where missing data were imputed and then smoothed using the Random Forest algorithm and the Kalman filter. This allowed us to better represent the results for a thorough statistical analysis.

During this project, we had the opportunity to put into practice the courses assimilated during our master CSMI, as well as the different tools and methods that we now master. It has also greatly contributed to the expansion of our skills and the acquisition of new technologies.
It introduced us to project management skills: we learned to manage pressure and workload, deal with various unforeseen events and problems, demonstrate team spirit and good organization with team members.

We are interested in several other perspectives that will allow us to complete this project during our internship, including :

- the exploitation of an important number of data. This is due to the fact that the database has a slightly erratic behavior at high load.
- the randomization of the nodes because this would have generated several modifications in the code.
- completion of the time series using LSTM neural networks (Long Short-Term Memory).

References

[1] Idrissa Niakh, Réduction d'ordre et assimilation de données, application à l'aérothermie, master 2 CSMI, university of strasbourg, March 2020.

[2] Génération ROBOTS, https://www.generationrobots.com/en/401879-grove-temperature-and-humidity-sensor-pro.html, consulted on march 2020.

[3] Christophe Prud'homme, Cemosis, Cemosis, le centre de modélisation et de simulation de Strasbourg, une agence math-entreprise en Alsace, http://smai.emath.fr/canum2016/resumesPDF/prudhomm/Abstract.pdf, consulted on 7/04/2020.

[4] Synapse concept, https://www.synapse-concept.com/presentation, consulted on 10/04/2020.

[5] Framalibre, https://framalibre.org/content/visual-studio-code, consulted on 16/05/2020.

[6] Docker, https://www.logiciel-libre.org/s/docker, consulted on 16/05/2020.

[7] Elasticsearch-Indexation et recherches simple, https://stph.scenari-community.org, consulted on 14/04/2020.

[8] Forêts aléatoires de classification et de régression,https://www.xlstat.com/, consulted on 14/04/2020.

[9] Arbres de décision et Forêts aléatoires, http://perso.mines-paristech.fr/fabien.moutarde/ES_MachineLearning/Slides/coursFM_AD-RF.pdf, consulted in may 2020.

[10] Chloé-Agathe Azencott, Introduction au Machine Learning, 179p.

[11] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems, Aurélien Géron,United States of America : Nicole Tache, 2017, 542p.

[12] Kalman filter, https://en.wikipedia.org/wiki/Kalman_filter,consulted on may 2020.

[13] Cours de Python, https://python.sdv.univ-paris-diderot.fr/18_jupyter, consulted on 16/04/2020.

[14] A Beginner's Guide to Neural Networks and Deep Learning, https://pathmind.com/wiki/neural-network, consulted on 14/04/2020.

[15] Du BIM à un modèle numérique du batiment : modélisation géométrique et mise en données physiques, university of strasbourg, August 2019.

[16] Zohra Djatouti. Amélioration de la prédiction de quantités d'intérêt par modélisation inverse : appli- cation à la thermique du bâtiment. Thermique [physics.class-ph]. Université Paris-Est, 2019. Français. NNT : 2019PESC2006.