

# Indoor Fire Forecasting with Machine Learning Techniques



*Pablo Descudet*

*In collaboration with Gabriela Melia and Ricardo Moreno,*

*Tutor: Celine Hudelot*

## **Index**

Introduction.....	3
Work Summary.....	4
• Theory.....	5
• Methodology.....	10
Encountered Difficulties.....	20
Conclusions and Prospects.....	21
Bibliography.....	22

## **Introduction**

This document serves as a description of the development of our project up to the JALON 3 submission date: 31st May 2018. A history of the project and the obtained results are presented.

The document consists of three main parts, excluding this introduction. In the first one, a description of the work process is done, including what each task consisted of and the main concepts learned. The next part is dedicated to the explanation of the difficulties that were faced during the project. Finally, conclusions and prospects for the future are presented in the last section of this document.

The final goal of this project was to develop a machine learning algorithm that could predict the behavior of a precise fire in an atrium. During the investigation of previous works of this field, it was found that the topic had not been explored much and the application of machine learning techniques had been barely considered. This, and the fact that machine learning has worked very well for cases similar to the prediction of indoor fires, made this project more interesting.

The project begun by the learning of machine learning techniques and the study of the state of the art, was followed by a long period of data analysis and ended with the design of the codes. During this process, there were times where the objectives were unclear and we failed to see the use of what we were doing, mainly during the data analysis, which was completely new to us. However, new solutions were coming up continuously, ways of improving the working methods and even achieving things which had been considered impossible at the start. In the end, the results are better than expected and most of the work needed to propose an algorithm which can predict every kind of fire in the atrium has been done.

## **Work Summary**

This section is going to be divided into two subsections. Whereas the practical part is the core of the project and what builds the solution, it would not have been possible to develop the algorithms without first understanding many concepts hidden behind. Indeed, on the one hand, much time was dedicated to the study of the statistical and machine learning methods and algorithms. On the other hand, there was their implementation, the practical work. Therefore, the first subsection contains the explanation of important concepts that have been discovered and the second subsection describes the methodology followed during the project.

## Theory

The very first part of the project, to be considered because it was essential and also because of the considerable amount of time it took, was the study of machine learning. In order to acquire a solid basis in the subject, the Stanford University's online course was done between the months of September and December of 2017. The course explained the meaning of machine learning, its applications, the difference between supervised and unsupervised learning and went through the most performant algorithms which exist nowadays. As the project begun in September, this course was thoroughly explained in the rapports of the previous semester, mentioned in the bibliography.

If this course was thought to be “the necessary knowledge” at the beginning, it was later proved to be necessary but not enough. New concepts, not just related to machine learning, came up continuously and needed to be well understood. They are presented in this part, in order of “arrival: Python array, dendrogram, curve's noise, curve smoothing, Keras model layer's type, activation function (sigmoid function), batch size and epochs and LSTM neural networks.

### Python array:

This concept was found essential during the project. As simple as it may seem, not distinguishing clearly the difference between an array and a list in Python could result in a waste of time when programming. Arrays and lists are both used to store data, but they do not serve exactly the same purposes. They both can be used to store any data type, and they both can be indexed and iterated through, but the similarities between the two do not go much further. Their main difference is the functions that can be performed to them. For example, an array can be divided by 3, and each number in the array will be divided by 3. However, a list cannot be divided by 3, an error would be thrown. Arrays need an extra step because they have to be declared while lists do not because they are part of Python's syntax. Nevertheless, to perform arithmetic functions arrays must be used. Additionally, arrays store data more compactly and efficiently, so they are more adapted when a large amount of data is treated.

### Dendrogram:

A tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering. As what respects to the project, given the simulation, a dendrogram was drawn to see the relationships between variables. This way, as the dendrogram takes into account the “distances” between variables, it was possible to spot which were similar enough to be able to eliminate the highest number of variables possible without losing information. The result of a dendrogram from a data file can differ depending on some parameters. These, which needed to be understood and defined, are the method and the metric. The method which was decided to use for these project's dendrograms was “centroid”, and the metric “Euclidean”. This assigned:

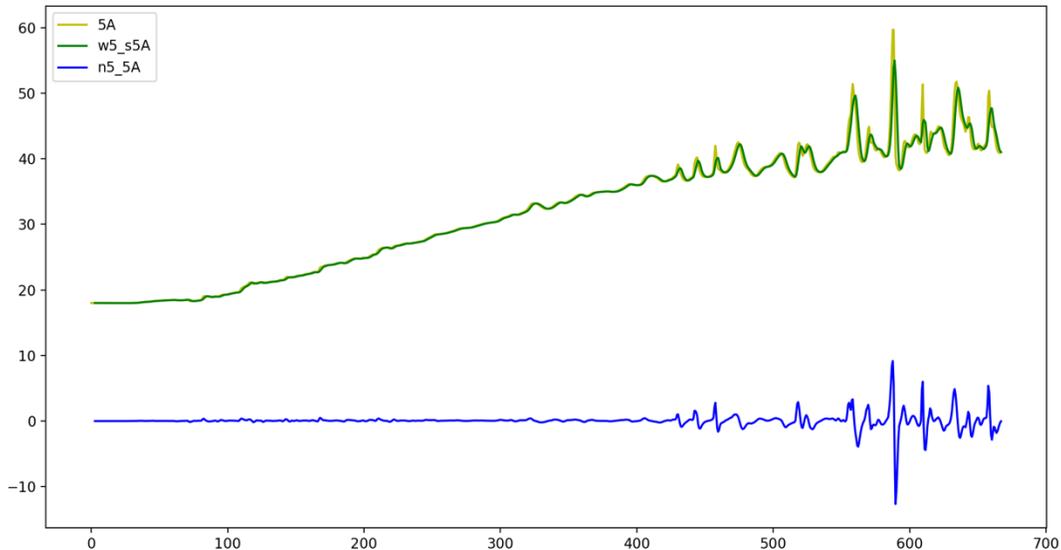
$$dist(s, t) = ||c_s - c_t||_2$$

Where  $c_s$  and  $c_t$  are the centroids of clusters  $s$  and  $t$ , respectively. When two clusters  $s$  and  $t$  are combined into a new cluster  $u$ , the new centroid is computed over all the original objects in clusters  $s$  and  $t$ . The distance then becomes the Euclidean distance (the line

segment connecting two points) between the centroid of  $u$  and the centroid of a remaining cluster  $v$  in the forest.

#### Curve's noise:

In curves where a tendency is recognized, the noise is the result of: deducting from the original function the one representing the spotted tendency. As it may remain unclear, here is a graphic demonstration of how it is done:



Three curves are appreciated in the image: the original curve (5A), the spotted tendency (w5\_s5A) and the noise (n5\_5A).

Moreover, to understand what the noise represented, some physics' knowledge was required. Seeing that, fire dynamics were considered. The conclusion of these studies was that there were two possible originators of the noise. Firstly, the presence of the smoke layer. This layer, formed by particles at a very high temperature would cause that when one of them arrived to the sensor's position its measurements underwent a large rise. Secondly, heat transfer theories demonstrate that the sudden temperature peaks could also be due to radiation.

#### Curve smoothing:

The action of smoothing implies reducing the noise. It was seen that many of the variables to deal with differed from each other due to their noise, their tendencies being very much alike, if not identical. To prove this, the smoothing of the curves needed to be done. It consisted of replacing each single measurement by an average of itself and a fixed number of previous ones.

#### Keras model layer's type (Dense):

The attribute "layer type" of Kera's library is related to how the neurons are connected to each other from one layer to another. In the models designed for this project, Dense layers

were used. This type of layers link each neuron to all the other neurons, they make as many links as possible. Further below it will be explained that other layer types (LSTM) have also been considered.

Activation function (Sigmoid function):

This concept was probably the hardest to understand. The neurons, which form the neural networks, work in the following way: they calculate a “weighted sum” of their inputs, they may or not add a bias and they then decide if their output is “fired” or not. The activation function is in charge of this last decision. Depending on the chosen function, the activation of the neuron means its output can be a 1 or a 0 (Step), its output can be [-inf, +inf] (Linear), its output can be [0,1] (Sigmoid), its output can be [-1, 1] (Tanh), etc. Each of the different functions has advantages and disadvantages which make them more adapted to some problems than others. Among all the function options, Relu and Sigmoid were the discussed ones for this project.

Relu function:

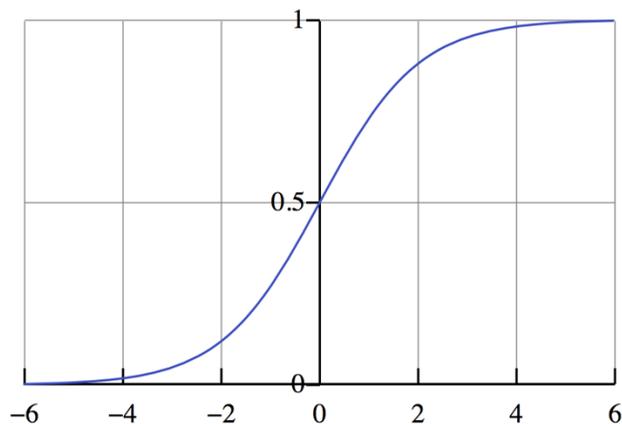
$$f(x) = x^+ = \max(0, x)$$

Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Both are nonlinear in nature as well as their combinations. This allows several neural network layers to be stacked (no matter how many layers there are, if combinations of functions were linear in nature, the final activation function of last layer would be nothing but just a linear function of the input of the first layer).

However, an important difference between them (specially in this case) is their domain. As it can be seen in the graphics below, whether the Relu’s output can be any positive value, the Sigmoid’s one is [0,1], which makes it not work for cases where other values are expected. Moreover, due to its form, in Sigmoid’s X values [-2, 2] the gradient is high and the range of values resulting from this zone is in consequence wide, but outside of it, all values tend to the same output.



Another important difference between them, is the “sparsity”. In a big neural network, the use of a Sigmoid would cause almost all neurons to fire in an analog way. That means almost all activations would be processed to describe the output of the network. In other words, the activation is dense, and this is costly. Relu allows some neurons in the network to not activate and thereby make the activations sparse and efficient.

Moreover, Relu is less computationally expensive than Sigmoid because it involves simpler mathematical operations.

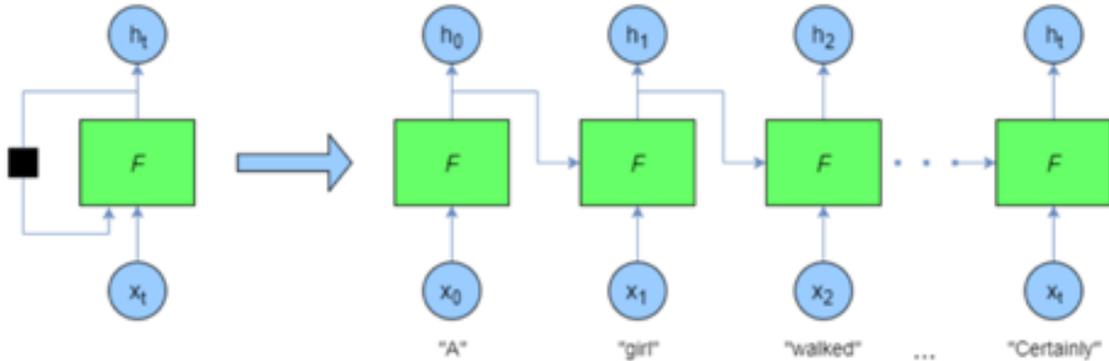
Nevertheless, not everything is perfect in Relu. Because of its horizontal line for negative values, the gradient can go towards 0. For activations in that region, gradient is 0 so the weights do not get adjusted during descent. That means, those neurons which go into that state will stop responding. This is called dying Relu problem. This problem can cause several neurons to die making a substantial part of the network passive. There are variations in Relu to mitigate this issue, and one of them would be the function used.

Batch size and epochs:

Another determinant aspect in the design of a neural network is the choice of batch size and epochs. This terminology appears when dealing with too much data, usually the case in machine learning problems. As it is not possible to pass it all at once to the computer, it is divided into smaller chunks, given one by one to the computer. Batch size is the total number of training examples present in a single batch (one of those smaller data chunks), and an epoch is when an entire dataset is passed once forward and backwards through the neural network. More than one epoch, many more than one in fact, is needed because the datasets are limited and an iterative algorithm (Gradient Descent) is used to optimize the learning process. Therefore, updating the weights with a single pass (epoch) is not enough.

Recurrent neural networks (LSTM):

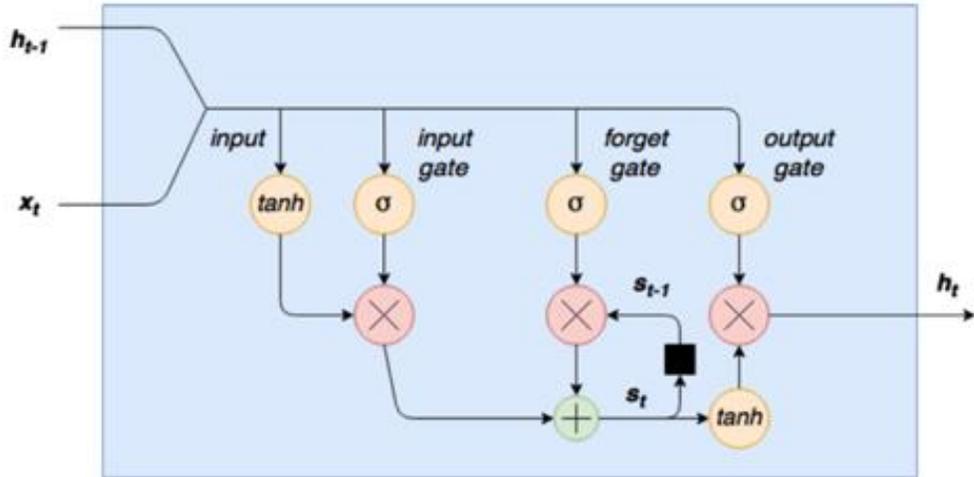
These kind of networks were not explained in the Stanford course. They may be of interest to the project because its complexity makes simple neural networks not good enough. The presence of a tendency in the variables indicates the recurrence may be adequate to achieve better learning. The recurrence consists on feeding back the output of a neural network layer at time (t) to the input of the same network layer at time (t+1).



**Unrolled recurrent neural network**

LSTMs (Long Short Term Memory networks) are a type of recurrent networks capable of learning long-term dependencies. They were explicitly designed to remember information for long periods of time, as there are many cases in which this is essential. In the example of a language model trying to predict the next word based on the previous ones, the antecedent may be a preposition with no information at all, and what is interesting is the verbs and nouns which came before.

They have become widely used because of this ability. To achieve this, they are formed by cell blocks in place of standard neural network layers. The key in these cells, is the top horizontal line: the cell state. It is kind of a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It is very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by the mentioned gates. Gates are a way to optionally let information through. They are composed of a Sigmoid neural network layer and a pointwise multiplication operation. The Sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through” An LSTM has three of these gates: the input gate, the forget gate and the output gate, named in accordance to their function. Here is a graphical representation of the LSTM cell:



LSTM cell diagram

## Methodology

Once the machine learning course completed, to know what techniques might be best suited for indoor fire forecasting, research on the state of the art of this field was required. During a couple of months, a thorough investigation on the modeling of fires was done. It was found that indoor fire modeling was a far less explored topic than outdoor fire modeling. Sadly, information on outdoor fires was not useful or applicable to our work. Please refer to the previous Jalon 3 if you are interested in reading the entire State of the Art.

When the techniques were learned and the research was finished, data analysis begun. This represents an important part of the project: the acquisition and preprocessing of data. As the project's goal is the forecasting of a fire in a large space, the data came from a series of simulations in an atrium. This project is realized with the support of a Spanish research company which is studying ways to optimize the prediction and forecast of fires. They are recently focused on finding innovative methods to apply and have decided to investigate in the field of machine learning. In consequence, much effort is being dedicated to the essential phase of collecting data. After carrying through several simulations, they created the data bases we received.

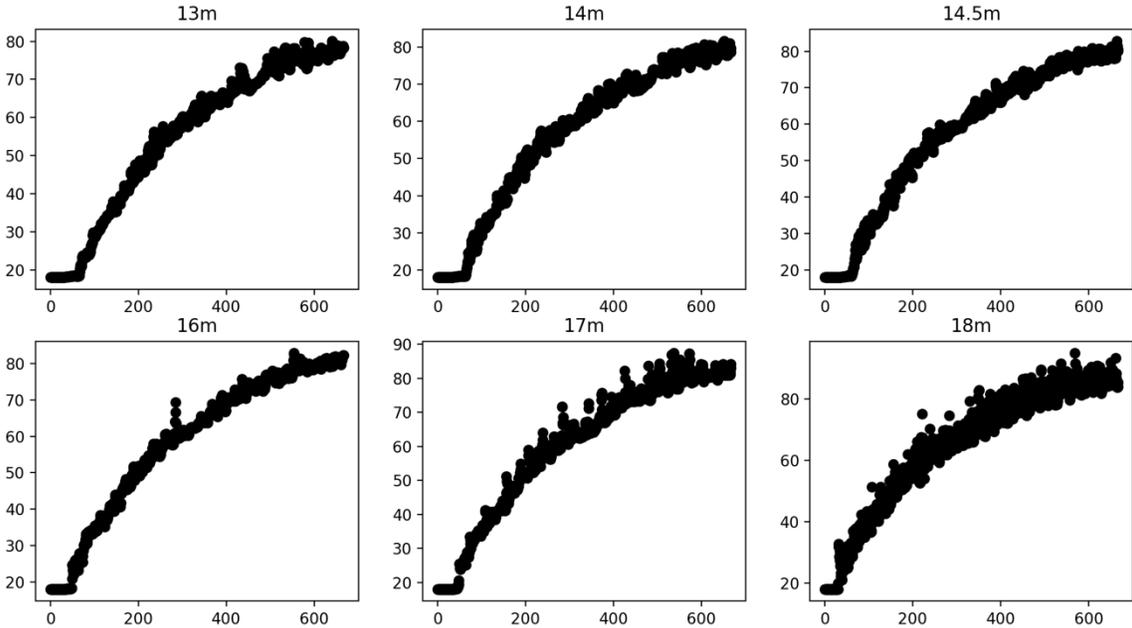
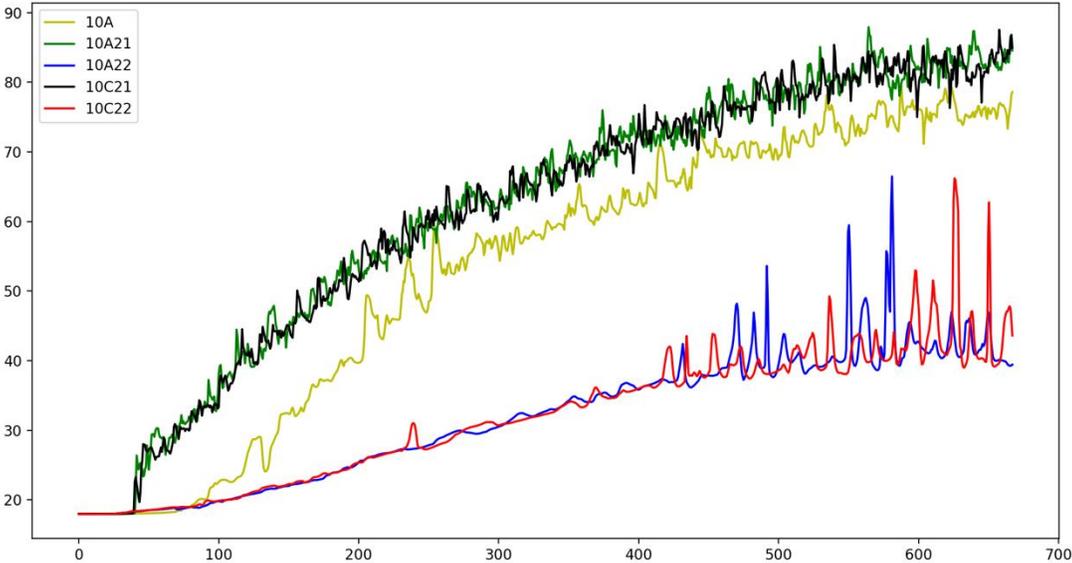
Along with the data base, a document explaining its contains was received. The rows corresponded to various instants in time and the columns to different type of data (temperatures, heights of the smoke layer, velocities...). Further in detail, taking temperature information as an example, its most relevant information was found. Columns 2-7 included temperatures given by thermocouples located in the center of the atrium at heights: 4.62, 6.62, 8.62, 10.62, 12.62 and 17.5 meters. Columns 8-13 included temperatures given by thermocouples situated in one of the atrium's walls, at heights: 5, 7.5, 10, 12.5, 15 and 17.5 meters. Finally, columns 54 and 55 were thermocouples situated above the fire source at heights 18.5 and 19.3 meters.

The data base received was a ".CSV" file containing many rows and columns and not yet organized with one piece of information in each cell. In consequence, the data was first arranged in Excel. Afterwards, with no previous experience in computing files, the solution was found in the Internet. "Numpy" library was the key and with it the "numpy.genfromtxt" function, which allowed obtaining all the data and transforming it into an array.

However, not all the data being numeric and not all the variables significant to the project, the following step was to select among the rows and columns those which were of interest. This facilitated the subsequent analysis of the data and a quick look of the values in the data base was able.

It was soon appreciated that many of the variables, representing temperature sensors in the atrium, had very similar values. This gave rise to the idea of clustering. The objective of obtaining several clusters with its respective representatives in order to reduce the number of variables. This was done with a tool widely used in data analysis: the dendrogram. Quality time was spent to make the decision of which method and metric to use.  $R^2$  (coefficient of determination, which relates two variables by its explained variability) or any similar statistics parameter could not be taken due to the variables' non-linearity.

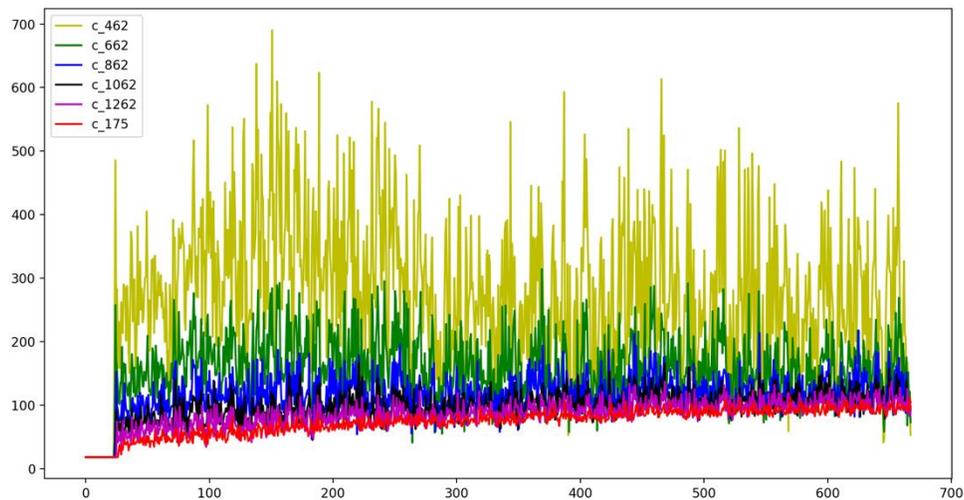
After a long period of reflection, it was decided that the understanding of the fire dynamics would be very helpful. To begin with it, all the variables were printed to see how the temperatures developed in the different points of the atrium.

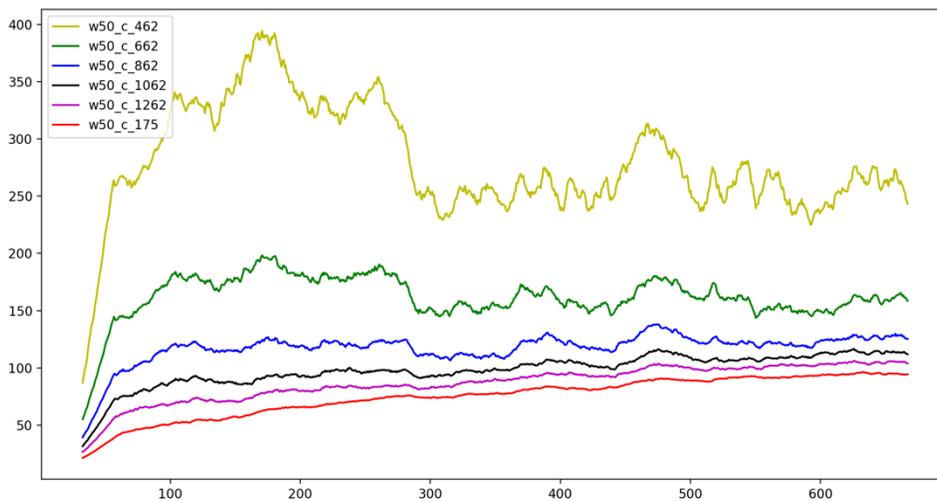
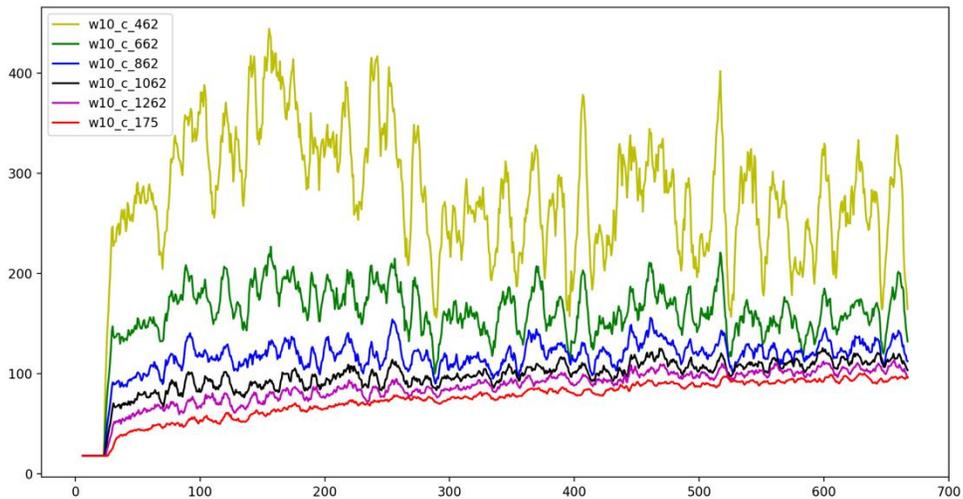


The first graphic shows the evolution of the temperatures measured by wall sensors situated at two different walls (A and C) and at a height of 10m. The second one shows the measurements at the middle point of a diagonal, at the different heights specified above.

Two conclusions were drawn from the analysis of all the plots. The first one, and very relevant because it proved clustering to be a good idea, was that many variables were nearly identical, if the noise was eliminated. The other one was that it could be easily seen at which time the smoke layer arrived at each point. This was clarified by the tutors' justification of how the smoke layer evolves. Their explanation was that this layer forms at the ceiling and starts descending until it reaches its final point. When the smoke reaches a thermocouple in the atrium, this sensor's measurements get very noisy. This is due to the difference between the high temperature of the smoke particles and the temperature there used to be at that point before the smoke's arrival.

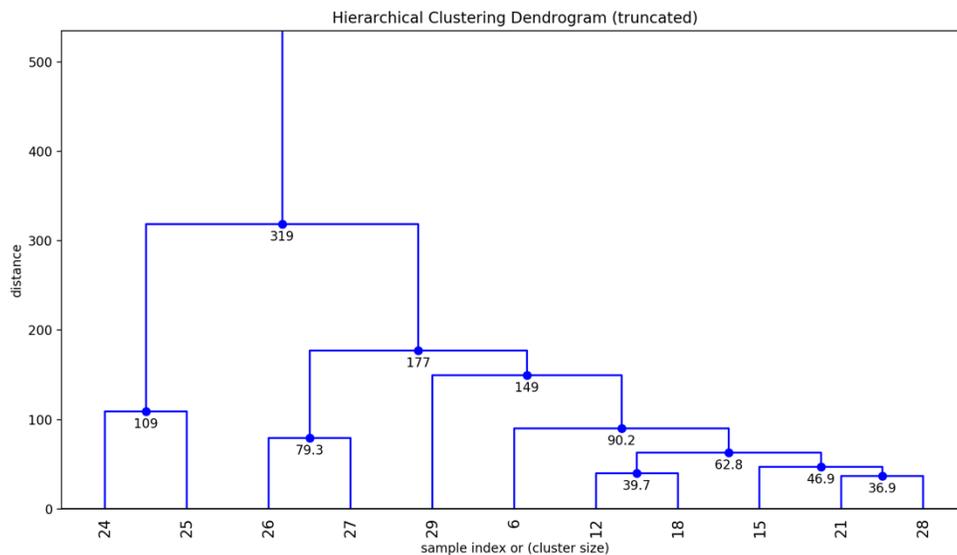
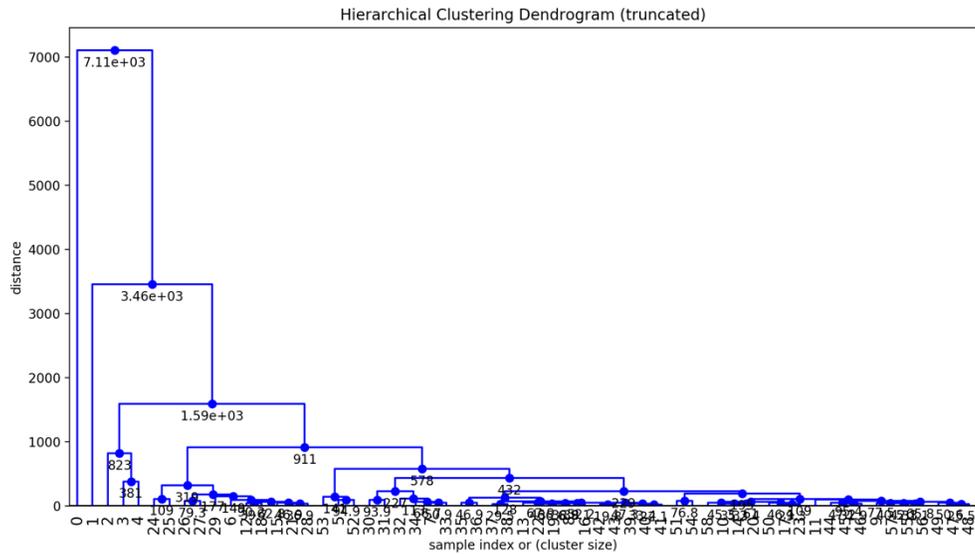
In consequence, by eliminating the noise from the curves, a clearer dendrogram, and easier to work with, could be reached. It was then decided to perform a smoothing of the curves previous to the drawing of the dendrogram. However, there were some points to clarify. Would the same smoothing be applied to every curve? Would the curve be separated into several pieces and the smoothing applied only to the noisiest? At the end it was decided to do personalized smoothing to the curves with the objective of assuring a maximum noise percentage for all of them, and to apply the smoothing to the whole curve. The function which was developed to do the smoothing separated the curve into same length pieces, searched for the one which had the highest variance and applied the smoothing technique in order to reduce its variance by 5%. This process was not simple because of the presence of values of the type NaN which needed to be transformed. They were substituted by the average of all their previous values, for instance, if it was the ( $t=0$ ) value, then it would remain itself.





The above graphics have been selected to show the smoothing process. They represent the evolution of the temperature measurements in the center of the atrium. These curves, being the closest to the fire, were also the noisiest ones.

Once the smoothing finished, the dendrogram was drawn and the representatives selected. There was a total of 5 clusters, and a few more “representatives”, as the maximum distance accepted to constitute a cluster was 400. These representatives, being the sufficient variables to avoid the loss of information, would be the inputs for the neural networks which constituted the next task.

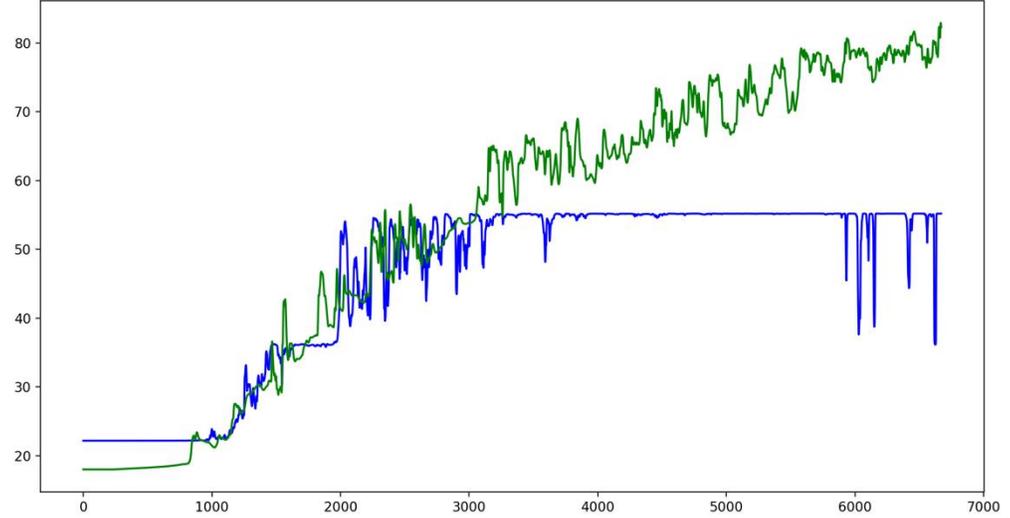
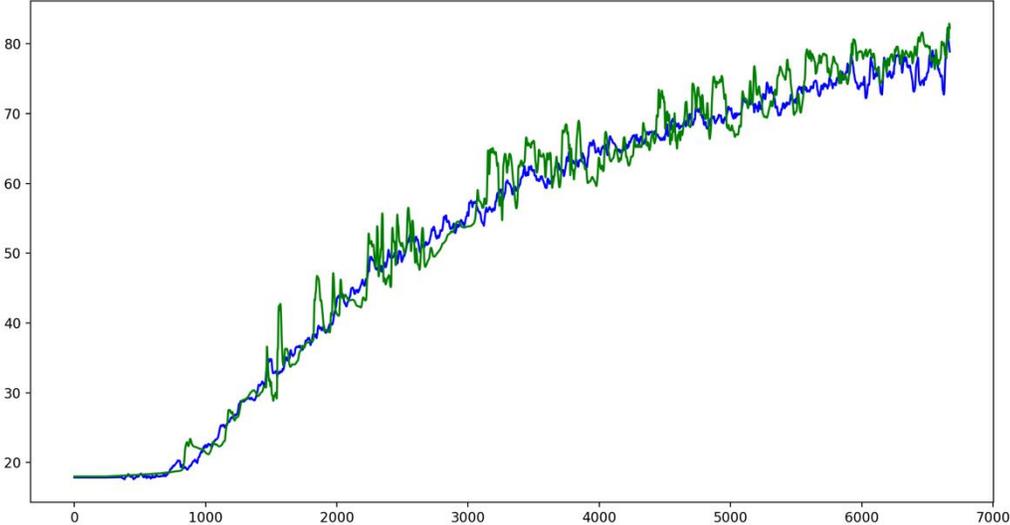


The two previous graphics are: the full dendrogram and a zoom of it. The zoomed one shows the components of the “most distant” cluster, the one with the maximum distance accepted. Even though, a distance of 319 is not a very high value, risking the loss of information was really not considered in the project. Accuracy has been aimed all along it.

Later on, the data crunch period finished, neural network encoding was faced. The first objective was to create neural networks which would be fed only with the representative of each cluster but would be able to predict every other value in the cluster. To better explain this, it meant having as many different models as variables had been eliminated. Each of the

eliminated variables would have its own model which would be built using a single input: the representative of the clusters it belonged to. This was achieved thanks to Tensorflow and Keras. The models developed consisted only of one hidden layer, used Dense layers and did not work with Sigmoid activation functions, not even in the hidden layer.

Here are the outputs of some of the models:



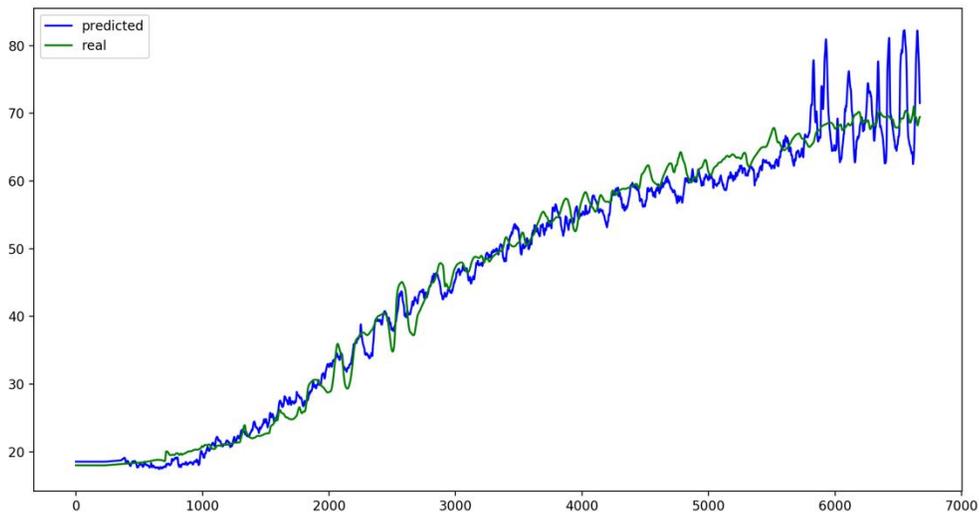
The first image shows a prediction without the use of Sigmoid (blue curve) and the second ones shows what happened when Sigmoid was applied to the hidden layer. All the other parameters (number of epochs, number of hidden layers and units, layer types...) remained unchanged.

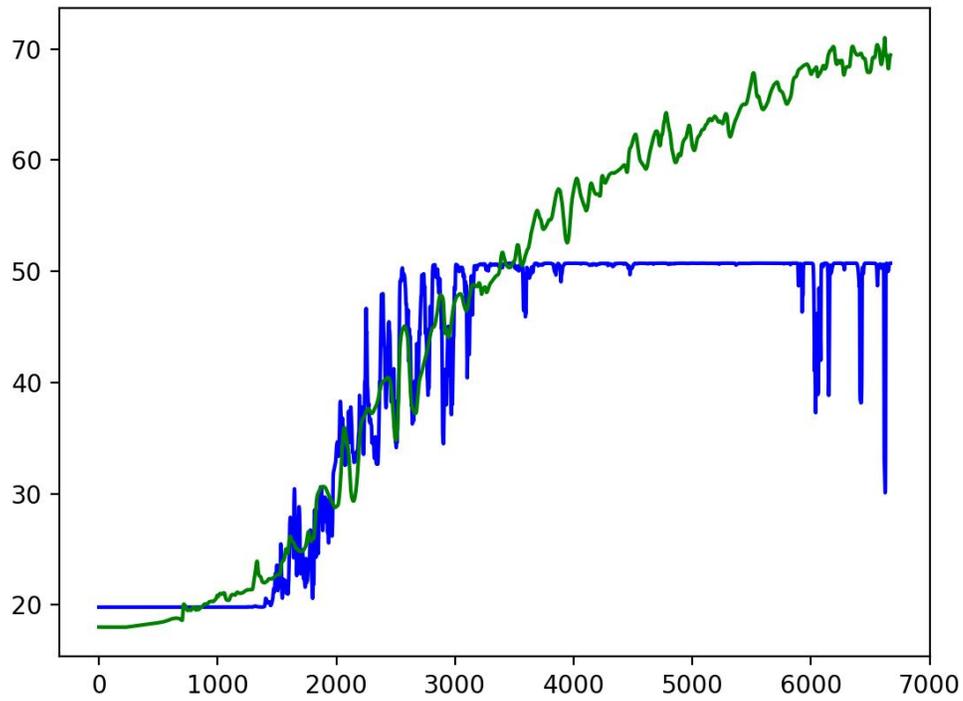
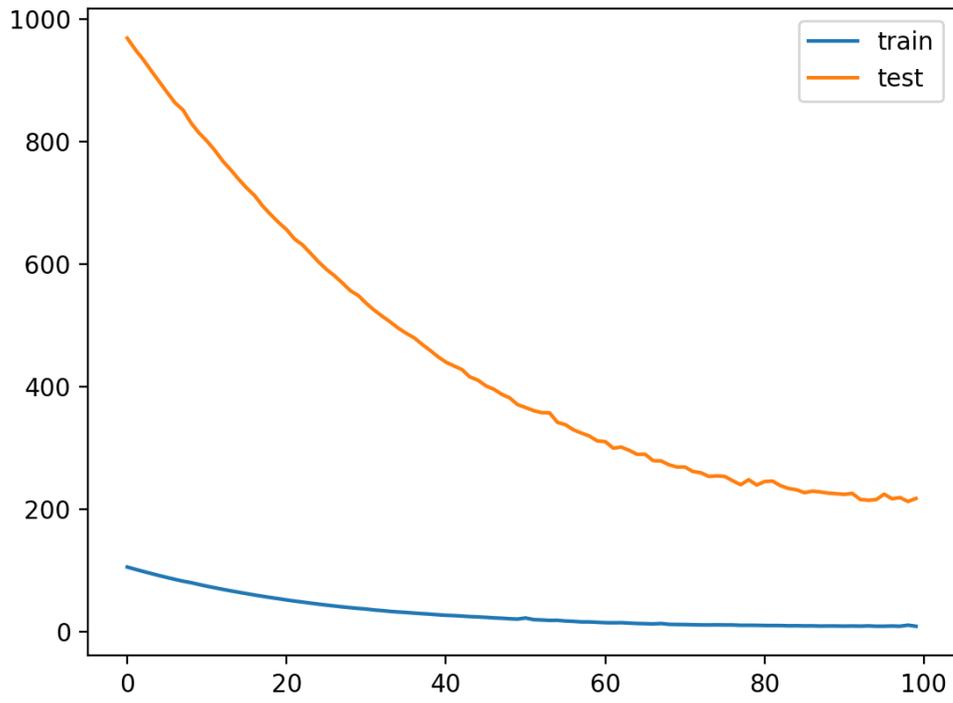
Consequently, what had been accomplished at that point was an algorithm which, given the characteristics of a point of the atrium where there used to be a thermocouple, looked for its particular model as well as for the cluster it belonged to (to know what had to be the input) and predicted what would have been the temperature measurements at that point.

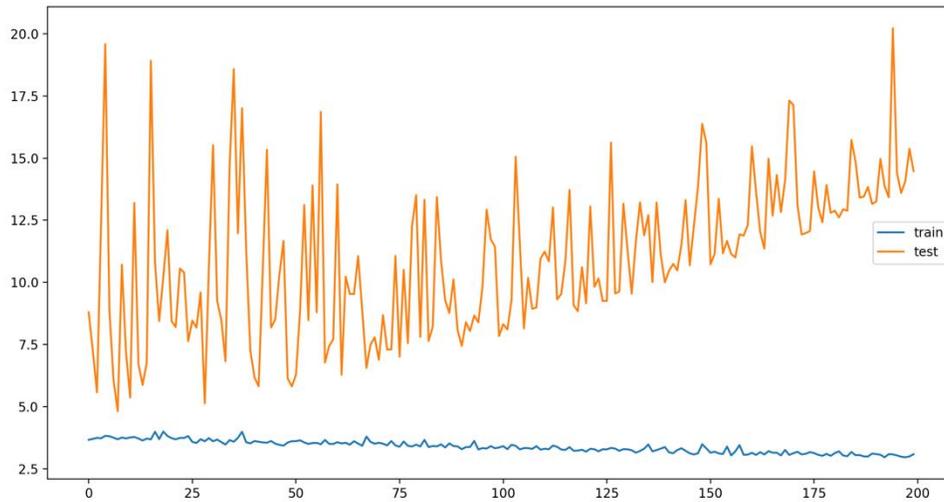
Even if the results were good and the predictions very accurate, this project had to main issues. On the one hand, there was the fact that some of the chosen representatives were located at impossible points of the atrium, floating, (whereas in the simulation they could be situated in the middle of nowhere, in the real atrium there would be no way to attach them to the air). On the other hand, these were great results for a very specific fire, but if its characteristics changed, the model would not be valid.

To make the problem more realistic, first it needed to be considered that the only points where there could be thermocouples were the walls. With this idea, the objective of the project varied. From then on, instead of having a single input for each model there would be as many as there were sensors at the walls. In consequence, the goal turned into having as many models as variables eliminated (all except the wall sensors) but they would all have the same inputs and they would all be trained with the data from several simulations.

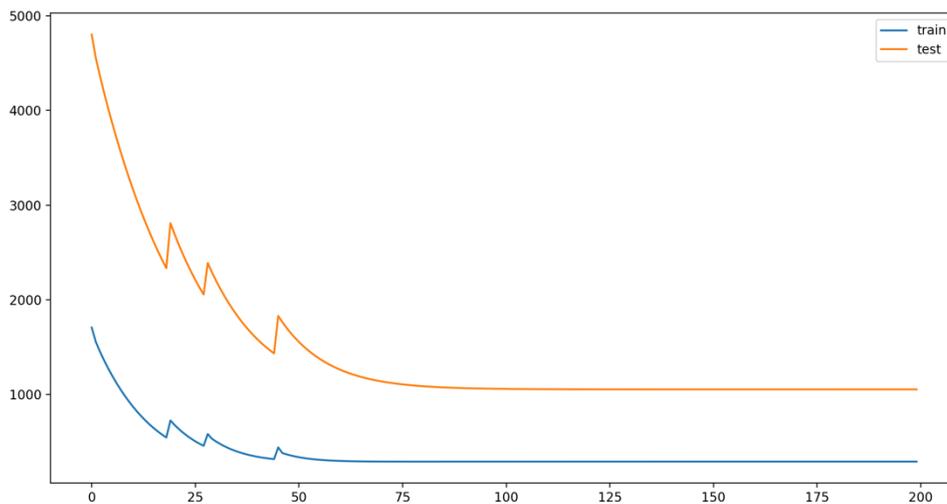
This was first tried with models with the previous characteristics: Dense layers, Sigmoid and not Sigmoid, just one hidden layer, etc. The results obtained are shown below. The first two plots are the outputs of models where Sigmoid was not used whereas for the last two it was applied to the hidden layer. They represent the contrast between the prediction error on the trained data and the error on the test data and then, as usual, the contrast between reality and prediction.

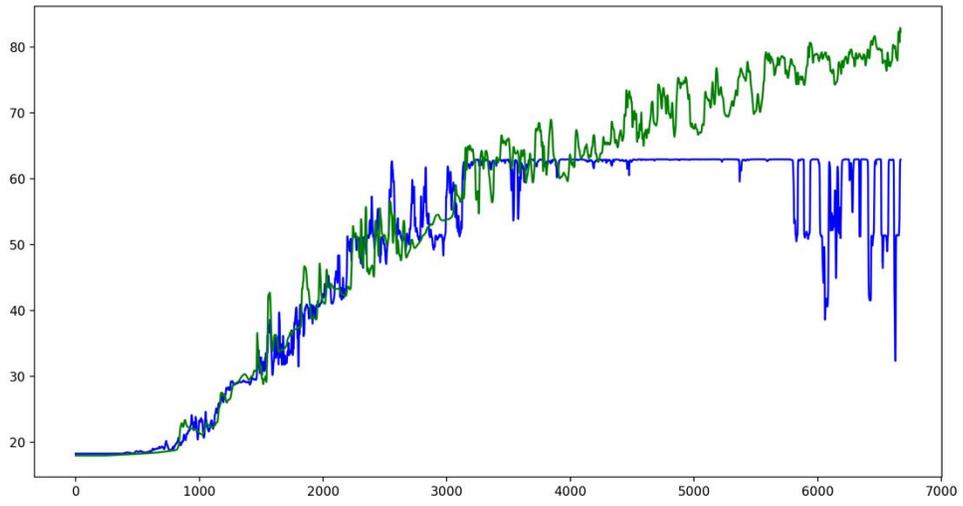






This new goal was more complex than the previous one, reason why it was harder to achieve the same accuracy. This motivated the study of more powerful networks. Based on the fact that the temperatures evolved following a pattern in time, recurrence was considered to be the solution to the problem. And more precisely LSTMs, which allowed to remember past facts in order to learn tendencies. Nevertheless, that has still not worked. At the moment, the predictions are still saturated.





## Encountered Difficulties

This section is added in order to explain the problems which came up during the development of the project.

Firstly, there was the issue of performing data analysis. For this task, as it has been the preferred method during the project, the solutions were searched either asking the tutors or looking on the Internet. It was them who brought up the idea of using dendrograms. Afterwards, the Internet was the source of information, it was there the algorithms were found as well as the explanations of how they worked and which values should be given to the parameters. It must be said, that quality time has been spent reading articles which were found in the Internet and specially trying to understand other people's codes.

Secondly, there were all the difficulties machine learning algorithms involve. In spite of having done the Stanford's course, there were always new concepts coming up. The machine learning field is a very complex one, and even if there has been a lot of progress from the beginning, it is still not at all mastered. We have realized that it is not the kind of domain where having a general idea of it is enough. Machine learning requires well understanding the concepts and how things work. Our two main obstacles have been the activation function and understanding how LSTM layers really work.

Both these problems have not yet been resolved. On the one hand, we still fail to understand the behavior of the curves when using the Sigmoid function in the hidden layer. We think it may be a problem of overfitting but we have not verified it. On the other hand, after a thorough study of the documentation on LSTMs, we still believe they are suitable for our problem but, searching for examples, we do not find any in which the inputs are what we would use. As a matter of fact, all of them train the neural networks with input  $X(t) = Y(t+1)$ . And this  $X(t)$  is not the predicted  $Y(t)$ , as we would have liked, as we thought the LSTMs did, but the real value of the variable trying to be predicted. This makes no sense for us, as either: their aim is to predict just a single value, or they use as input what they want to have as result (which would clearly be unavailable in our case).

## Conclusions and Prospects

The overall of this project is very positive. Not only we have advanced in a domain completely new to us, but we have succeeded in our results.

The reason we chose to get involved in a project like this, was because, as engineers, we wanted to be up to date in the domain which is causing a revolution in today's world: machine learning. We have done the course, and more important, we have tried the algorithms for ourselves and have faced difficulties from which we have learned a lot. We can now say, that we are able to understand how complex machine learning problems are solved, and that if it were to us to solve them we would manage.

Finally, with respect to the results, we have achieved the goals established at the beginning and we have even gone further. It was first demanded from us to make predictions for a specific fire in a specific room. We have accomplished this with the system of simple neural networks explained in the Work Summary. In addition, we have searched for a solution that could really be implemented. A solution where "floating" thermocouples would not be necessary. We have reached way better results than expected. What is more important, we have proved that with more time and data we would be able to predict any kind of fire's behavior in the atrium.

Even though this project is presented here as completed, for us there is still work left. Not happy with the conclusion that the system could be improved, we have decided to demonstrate it, continue until we reach the end. Therefore, we have contacted our tutors, and they have accepted to provide us with the necessary data for us to generalize the model. This way, we estimate that, with an additional month's work from the time we receive the new simulations, we will have a model able to predict any kind of fire's behavior in the atrium.

In conclusion, having the dimensions of an atrium and several simulations of what would happen in it with fires of different characteristics, our model is able to accurately predict the development of every possible future fire.

## Bibliography

- [1] «Probabilistic Graphical Models», MIT Press. [Online]. Disponible en: <https://mitpress.mit.edu/books/probabilistic-graphical-models>. [Accedido: 08-oct- 2017].
- [2] S. B. T. Rob J Hyndman, “Neural network models”, *Texts*. [Online]. Available in: <https://www.otexts.org/fpp/9/3>.
- [3] C. M. Bishop, “Pattern recognition and machine learning”, 2006.
- [4] S. Gollapudi, “Practical machine learning: tackle the real-world complexities of modern machine learning with innovative and cutting-edge techniques”, 2016.
- [5] D. J. Matich, “Redes Neuronales: Conceptos básicos y aplicaciones”, 2001.
- [6] Jalon 3, S7
- [7] Keras Documentation [Online] <https://keras.io>
- [8] Numpy and Scipy Documentation [Online] <https://docs.scipy.org/doc/>