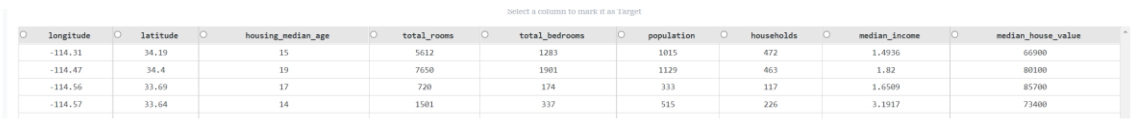
CASE STUDY: California Home Price Prediction.

In this case study we will use self-organizedmaps(SOM's)to train two SOM's,one with **rectangular** topology and the other with **hexagonal** topology. Both models will be compared and the relationships between **the**data **found** thanks to the nature of this type of neural networks will be analyzed.

1.Data used.

This study will use the" California Housing " dataset, which contains the average selling price of California homes in the year 1990 according to the census.

This dataset contains 17,000 rows with unique data and 9 features:



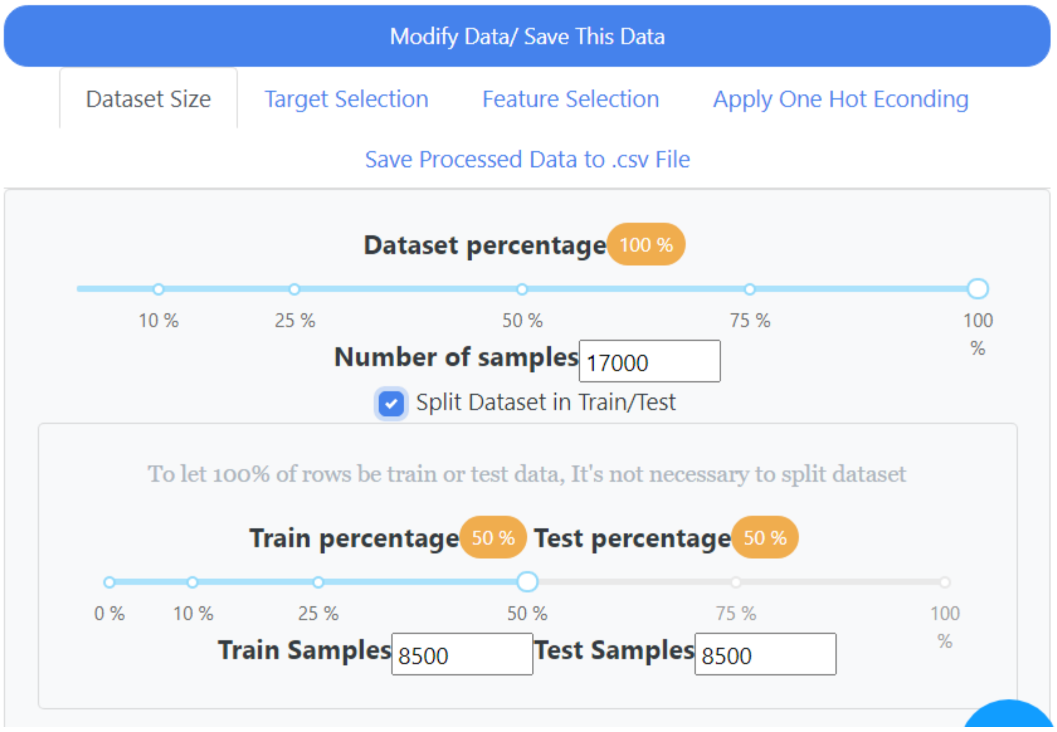
Select a column to mark it as Target

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
-114.31	34.19	15	5612	1283	1015	472	1.4936	66900
-114.47	34.4	19	7650	1901	1129	463	1.82	80100
-114.56	33.69	17	720	174	333	117	1.6509	85700
-114.57	33.64	14	1501	337	515	226	3.1917	73400

- **Longitude and Latitude:** Geographical coordinates of the house.
- **housing_median_age: Average** age of a house within a block
- **total_rooms :**Total number of rooms within a block of buildings.
- **total_bedrooms: Total** number of bedrooms within a block of buildings.
- **population:** Total number of people living in that block.
- **Households:** Total number of households in the block.
- **median_income: Average** income of people living in a household, for each block(in dollars)
- **median_house_value: Average** value of housing within a block(in dollars)

2. Data Processing.

- It will be selected as **a target** or characteristic to predict the **median_house_value.**
- The data(17,000 rows) **will be divided into** two sets, training and testing (train and test) **at 50%,** using the application.



Dataset percentage 100 %

Number of samples 17000

✓ Split Dataset in Train/Test

To let 100% of rows be train or test data, It's not necessary to split dataset

Train percentage 50 % Test percentage 50 %

Train Samples 8500 Test Samples 8500

3. Search for optimal hyper-parameters : Grid Search

In order to buy both rectangular and hexagonal topologies, on equal terms, both will be trained with the same hyperparameters.

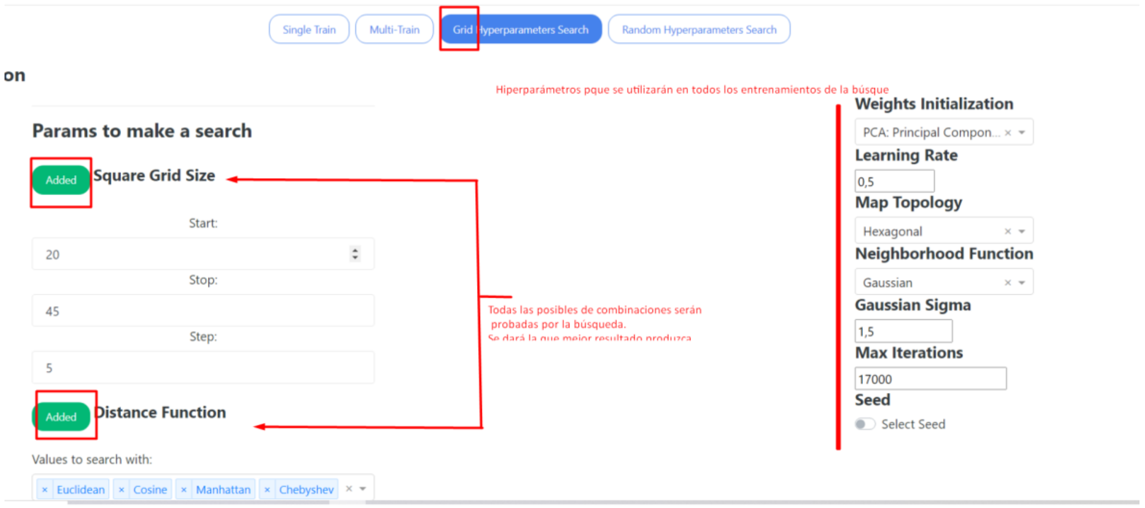
The application implements 2 hyperparametertuningalgorithms, to train them with values that produce optimal results, these are Grid Search and Random Search. Since the dimension of the data is small (17,000 data), we will use brute force search, that is, testing all combinations with **Grid Search.**

Since the initialization of the weights of the neurons of the model, empirically, have been shown to produce better results when using **PCA: Principal Component Analisys,** for this search we will not include this hyperparameter,which will always have the same value.

Of the 2 possible hyperparameters that we have left to do the search, we choose the following range:

- **Square Grid Size:** Size of the SOM to train(Number of NeuronsVverticales x Number of Horizontal Neurons), with a size of 20 (**20x 20 neurons)** to a final size to test of 45 (**45 x 45 neurons),** with jumps of **5** neurons. This leaves us with the following combinations of sizes: [20,25,30,35,40,45] to test.
- **Distance Function:** Activation function to choose the BMU in each round. We chose the **4** possible options for the search to test: **Euclidean distance, cosine, manhattan and chebysev.**

The rest of the hyperparameters are the ones chosen by default by the application and that can be seen in the screenshot below:



Params to make a search

Added Square Grid Size

Start: 20

Stop: 45

Step: 5

Added Distance Function

Values to search with: Euclidean, Cosine, Manhattan, Chebyshev

Hiperparámetros pque se utilizarán en todos los entrenamientos de la búsque

Todas las posibles de combinaciones serán probadas por la búsqueda. Se dará la que mejor resultado produzca

Weights Initialization: PCA: Principal Compon...

Learning Rate: 0,5

Map Topology: Hexagonal

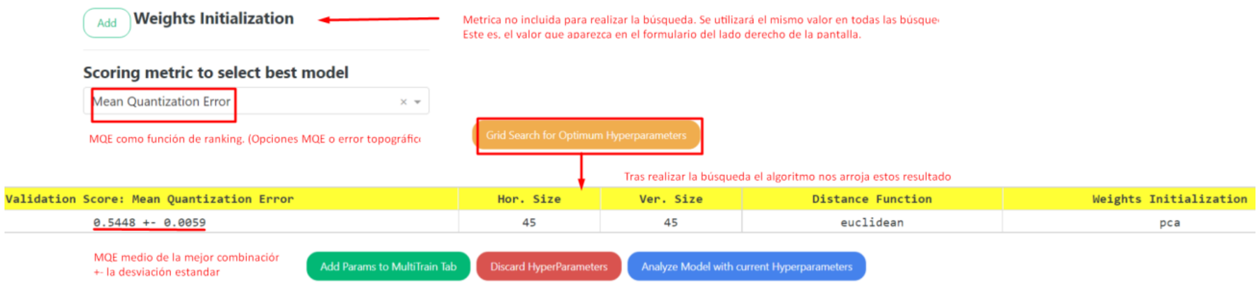
Neighborhood Function: Gaussian

Gaussian Sigma: 1,5

Max Iterations: 17000

Seed: Select Seed

After performing the search, using **the Mean Quantization Error (MQE) as a ranki**ng function, to choose as the best combination the model than a lower **result**


Add
Grid Search for Optimum Hyperparameters
Add Params to MultiTrain Tab
Discard HyperParameters
Analyze Model with current Hyperparameters

Weights Initialization

Metrica no incluida para realizar la búsqueda. Se utilizará el mismo valor en todas las búsque
Este es. el valor que aparezca en el formulario del lado derecho de la pantalla.

Scoring metric to select best model: Mean Quantization Error

MQE como función de ranking. (Opciones MQE o error topográfic

Tras realizar la búsqueda el algoritmo nos arroja estos resultado

Validation Score: Mean Quantization Error	Hor. Size	Ver. Size	Distance Function	Weights Initialization
0.5448 +- 0.0059	45	45	euclidean	pca

MQE medio de la mejor combinación
+- la desviación estandar

The search gives us **optimal hyperparameters of 45 neurons** for size (vertical and horizontal) **and Euclidean distance** as a function of distance.

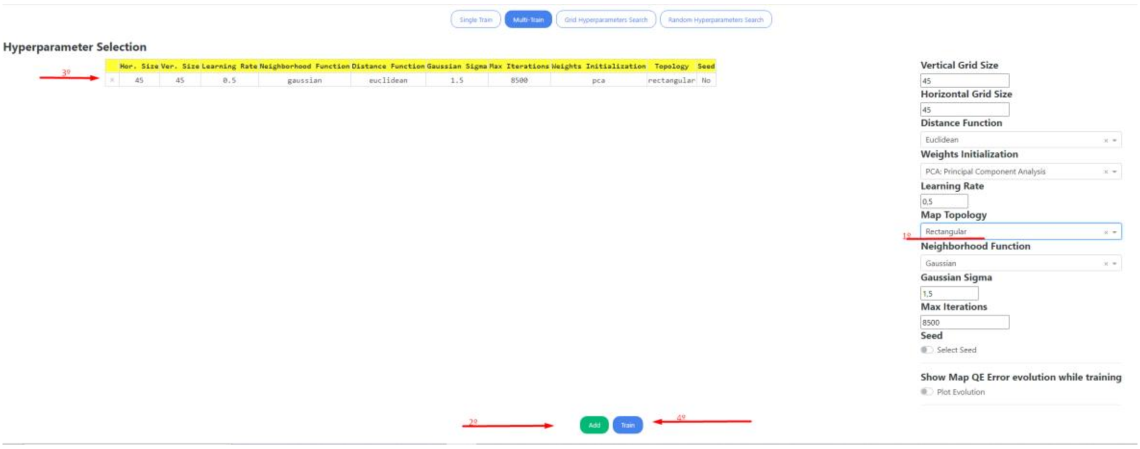
The application now gives us 3 options (3 colored buttons):

1. *Add these parameters to the TAB: Multi-Train* Since we want to compare these hyperparameters with a rectangular topology to make the comparison we choose this

option. Our model will appear in the next data analysis window, after performing a training from the Multi-Train tab.

2. *Discard such results.* If we have chosen some search ranges for the hyperparameters and these have given us a very high average MQE, we may want to discard them and perform another search by changing them.
3. *Analyze the model data directly.* Since a re-training of the model is done with the best parameters before finishing the search, we can go to the data analysis window directly, to analyze the model with said hyperparameters.

In our case, as we want a comparison with a model with other hyperparameters(rectangulartopology), we choose the first **button: Add Params to MultiTrain Tab (Green color),** and at the top of the interface, we select the Multi-Train Tab.

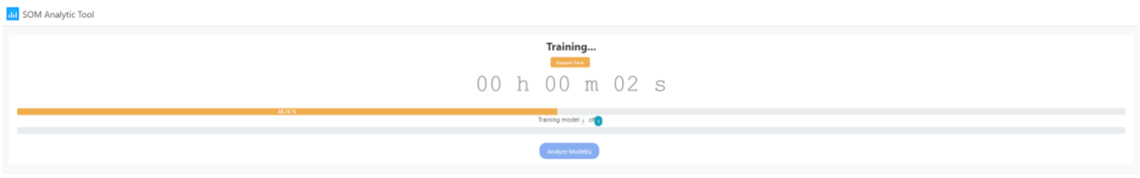


Hyperparameter Selection

	Hor. Size	Ver. Size	Learning Rate	Neighborhood Function	Distance Function	Gaussian Sigma	Max Iterations	Weights Initialization	Topology	Seed
×	45	45	0.5	gaussian	euclidean	1.5	8500	pca	rectangular	No

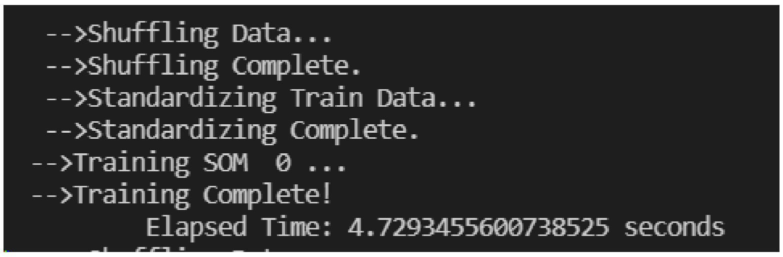
Vertical Grid Size: 45
Horizontal Grid Size: 45
Distance Function: Euclidean
Weights Initialization: PCA: Principal Component Analysis
Learning Rate: 0,5
Map Topology: Rectangular
Neighborhood Function: Gaussian
Gaussian Sigma: 1,5
Max Iterations: 8500
Seed: Select Seed
Show Map QE Error evolution while training: Plot Evolution

By default we see the hyperparameters already found in the search. We change the topology, add it to the training table with the **Add** button and then press the **Train** button.



After finishing the training we analyzed both models.

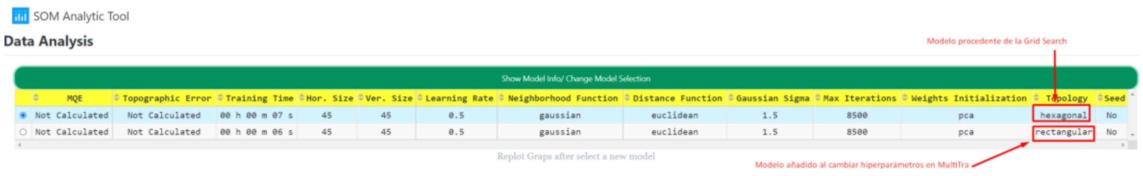
NOTE: In the terminal running that application you can see the process in which the application is located, useful for large datasets that require a model drawing time of a few minutes. In this way we can verify that the application has not been frozen.



```
  -->Shuffling Data...
  -->Shuffling Complete.
  -->Standardizing Train Data...
  -->Standardizing Complete.
 -->Training SOM  0 ...
 -->Training Complete!
        Elapsed Time: 4.7293455600738525 seconds
```

4. Trained models. Analysis.

On the model analysis page, if we press the "Show Model Info/Change Model Selection" button, we can see a list of all available trainings. We can change the selection of them in the radius that appears at the beginning of the table.

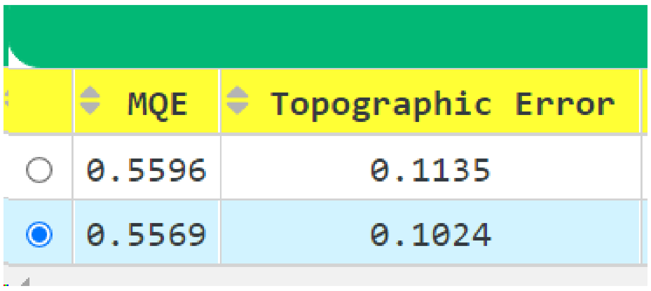

SOM Analytic Tool
Data Analysis
Show Model Info/ Change Model Selection
Replot Graps after select a new model
Modelo procedente de la Grid Search
Modelo añadido al cambiar hiperparámetros en MultiTra

	MQE	Topographic Error	Training Time	Hor. Size	Ver. Size	Learning Rate	Neighborhood Function	Distance Function	Gaussian Sigma	Max Iterations	Weights Initialization	Topology	Seed
●	Not Calculated	Not Calculated	00 h 00 m 07 s	45	45	0.5	gaussian	euclidean	1.5	8500	pca	hexagonal	No
○	Not Calculated	Not Calculated	00 h 00 m 06 s	45	45	0.5	gaussian	euclidean	1.5	8500	pca	rectangular	No

In our case, we found the hexagonal model with which we did the search (Grid Search)of optimal parameters, as well as the model added in MultiTrain. We can differentiate them because we chose first with hexagonal topology, and the second with rectangular.

4.1.Statistics

We calculate the MQE and the topographic erro for both models:



	MQE	Topographic Error
○	0.5596	0.1135
●	0.5569	0.1024

We observe how both models achieve similar results.

In both cases a rather **small** **MQE,** which means that the **datos are well represented** on average by each neuron, having been trained the model with a sufficiently representative amount of variety of **examples.**

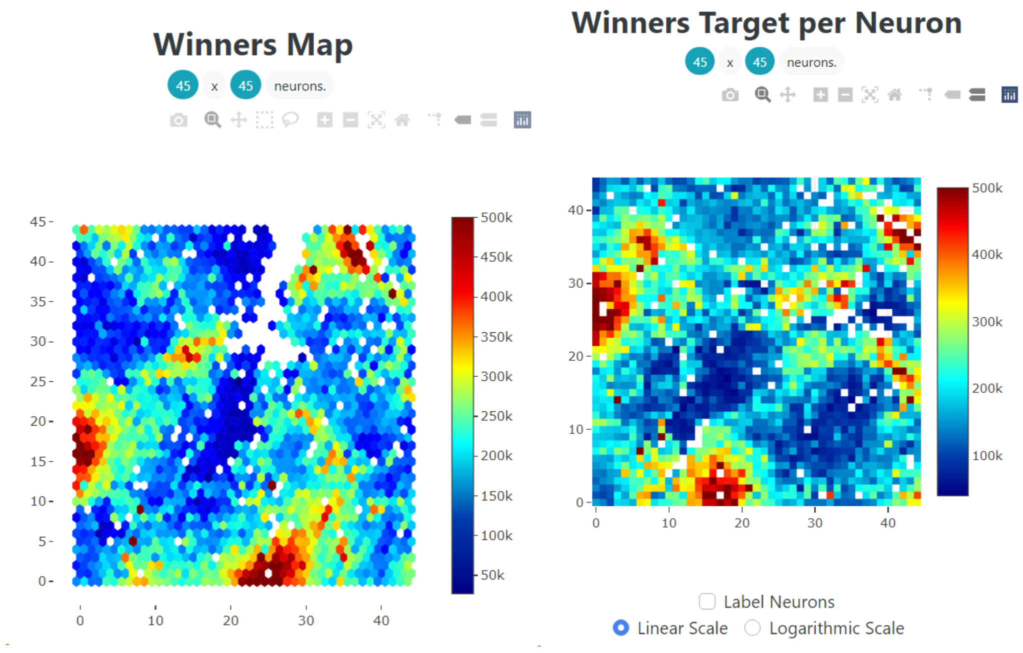
On the other hand we see how the **topographic error** is also **small.** Not being 0, we can say that the topology has not been **respected for all**examples, that is, the relationship set of similar characteristics – zone on the map, is not respected for all data. However, despite being able to have a smaller value, it is considered **acceptable.**

4.2. Map of Winning Neurons.

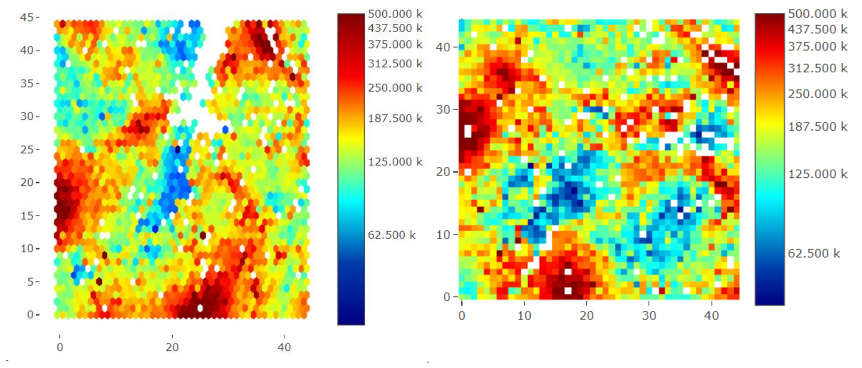
The map of winning neurons is calculated with the target(class to be predicted). To do this, the input data is taken, and for each of them its BMU is calculated. At the end of the process, each neuron will have several data (or none) of which it is its BMU. If a neuron has no data, it will be painted white in the application.

A well-trained model will organize similar or related targets in nearby positions, and different targets in distant positions. A BMU will represent only similar data, and therefore most likely a single target.

We draw the map of winning neurons. On the **left** the model with **hexagonal** neurons, on the **right** with **square neurons.**



Winning Neuron Maps: Linear Scale

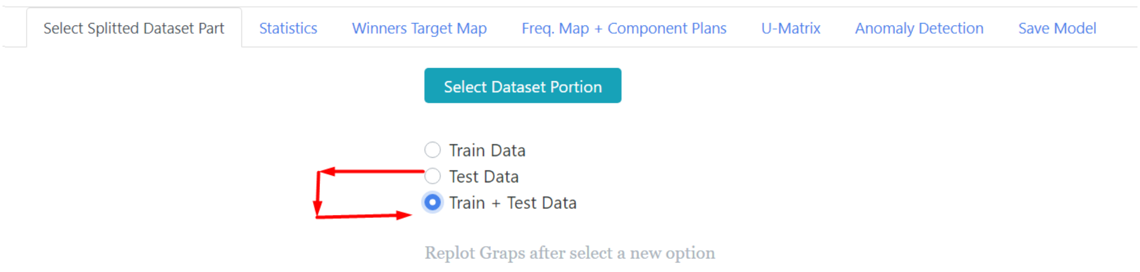


Winning Neuron Maps: Logarithmic Scale

As we can see, both topologies follow the same spatial patterns,since they have been trained with the same hyperparameters. **<u>However, the display on the hexagonal grid results in a more defined and natural perception, as if it were an image with higher resolution in pixels.</u>**

Since the dataset is split, we can choose the Train and Test sets to paint the **map** of winning neurons: with more examples the areas will be visually more represented, and probably white spaces will be painted, as a result of at least one piece of data from the set being chosen BMU from that neuron.

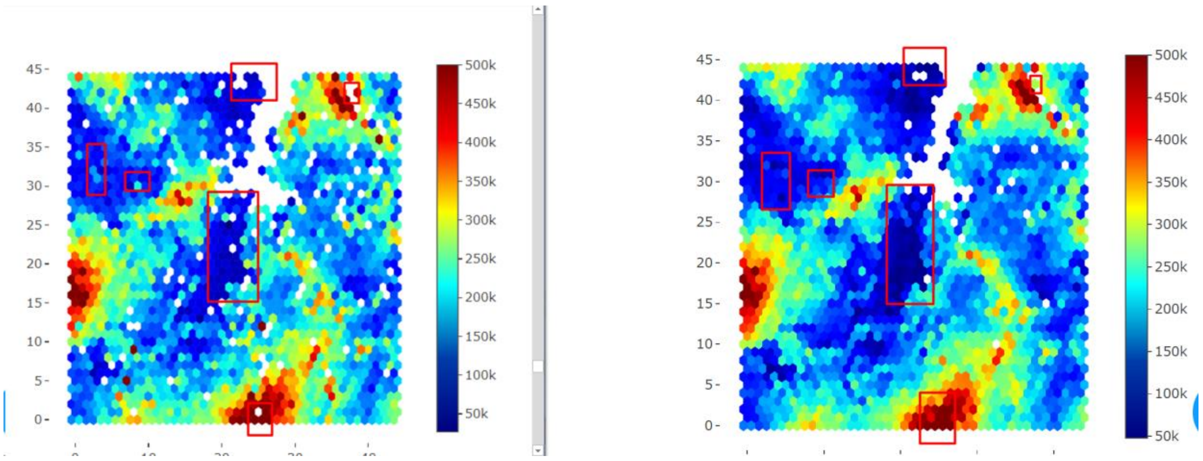
NOTE: Changing the datasets for the graphs does not change the model, only the more or less accurate representation of the data in the analysis.



Train Data
Test Data
[✓ Train + Test Data]

Replot Graps after select a new option

We redraw the graph(pro space motifs in this example only the hexagonal) with the "Plot"button:



Left: Using only training set data/

Right/Using: the entire dataset(settraining and testing)

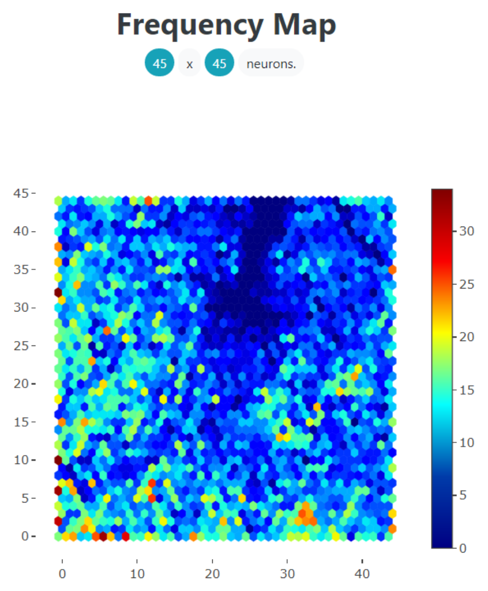
We can observe slight differences, not many, since the 8500 data added for the graphing are those used in the training. Even so we can observe **neurons that did not have a class to represent for lack of examples, and not weights,**since all neurons have weights that are related to their neighborhood and before an even more complete dataset, it is very likely that they end up representing intermediate data of those of their neighbors.

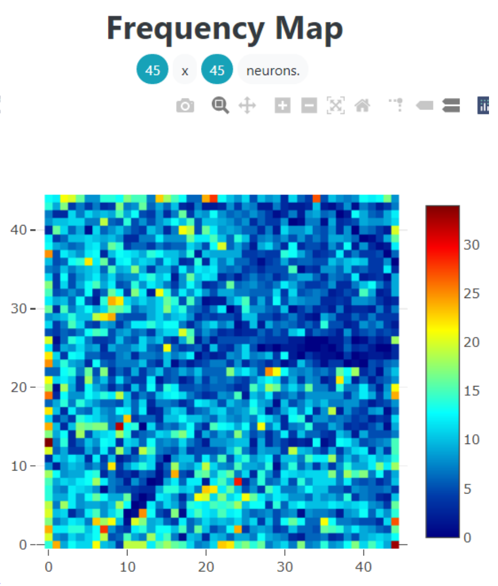
We can also observe **areas where classes intensify their value/color.** See in larger central rectangle larger and the small square above it.

It can be seen how these neurons represent low-value dwellings (around 50-100k) With a larger dataset, such as the one used on the right, it can be observed that this is because the central neurons concentrate the lowest values 50k, and their neighbors are slightly and gradually increasing the value to which they represent (100k the closest and 150k the farthest.) **That is to say, that area of the map has specialized in representing low-value housing,** which although it is not the only one, is the one that concentrates a neighborhood of larger representatives.

4.3. Frequency map.

This map represents the number of times a neuron has been chosen as the BMU of an input data. The winning neuron map is a frequency map that then takes into account the target of the data.





The frequency map gives us **a measure of reliability in the representation of the data in** several ways.

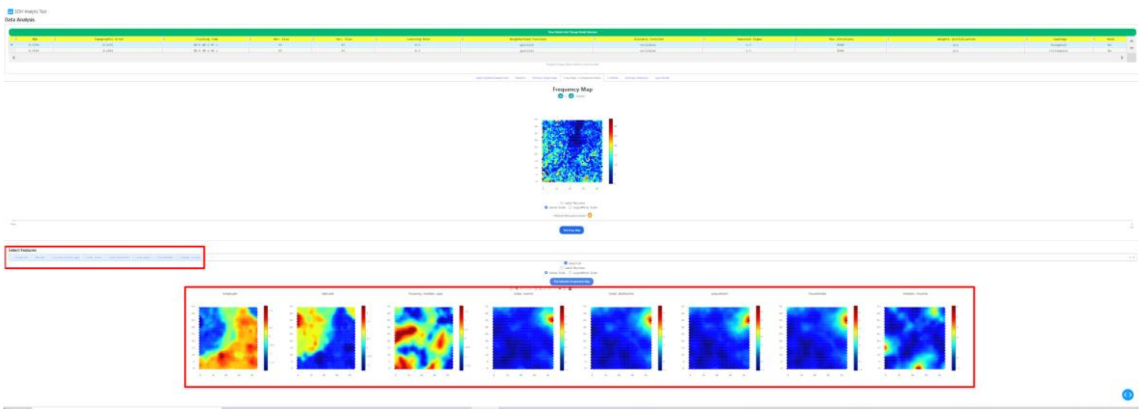
One of them is, in the case of neurons that have not been chosen once as BMU(Frequency 0). If these neurons are strut cases,itis likely that, although the map represents the data well and has a low mqe and typo, it is because more examples would be needed to better appreciate the gradients of relationships on the map. However if those neurons form groups of neurons on the map, and it is not for lack of examples, the model will have a high mqe and topographic error. This means that the topology is not preserved and that the data is not well represented, either due to lack of sufficient examples or a poor choice of hyperparameters during training.

In the case of isolated neurons with a number of 'hit' or very high frequency,with contiguous neurons, whose frequency is considerably much lower, is symptom of misrepresentation of the data. In this case such a neuron represents many examples, which should be distributed by more neurons on a larger map. This phenomenon can be seen in the U-Matrix in parallel, where you can see on that map a very dark area in the same coordinates of that neuron. Symptom that the weights of the erratic neuron and its neighbors are very different and make an abrupt and not gradual jump on the map.

Contiguous neurons with a very high frequency number but formingn regions on the map,   can be translated either in that the number of examples to be represented are many and very similar, or again to a bad choice of the size of the map during training, among other hyperparameters.

4.4. Component map.

The component maps represent for each neuron, the weights of the model trained for no attribute. In our case the model has been trained with 8 of the original 9 attributes of the dataset,and that is why we see 8 component maps.

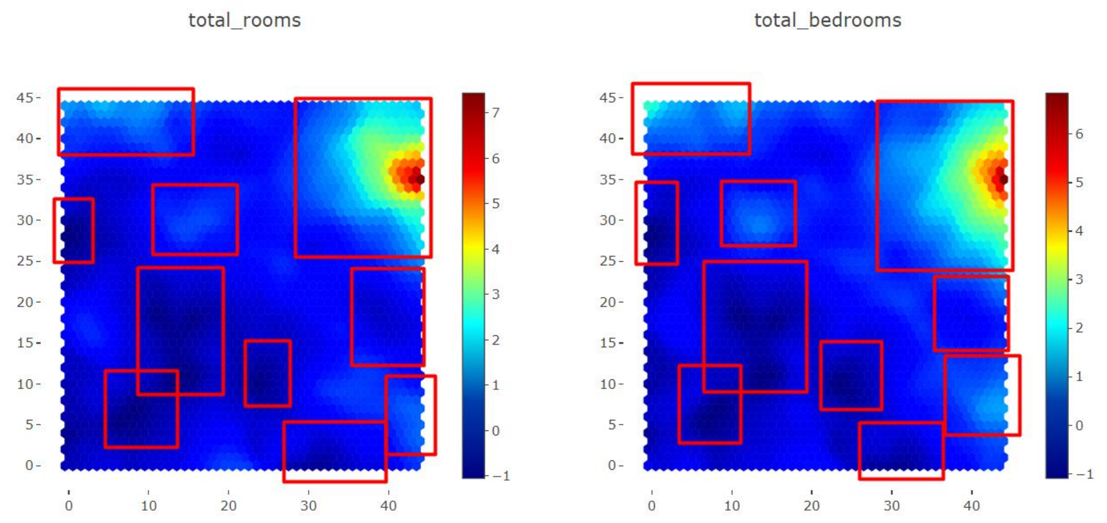


That is, each neuron in the model entering is a vector of dimension number of characteristics.

Component maps are used to visually discover relationships and patterns between attributes and attributes—and class to be predicted.

Let's look at it with 3 examples.

If we look at the attributes total bedrooms and total rooms:

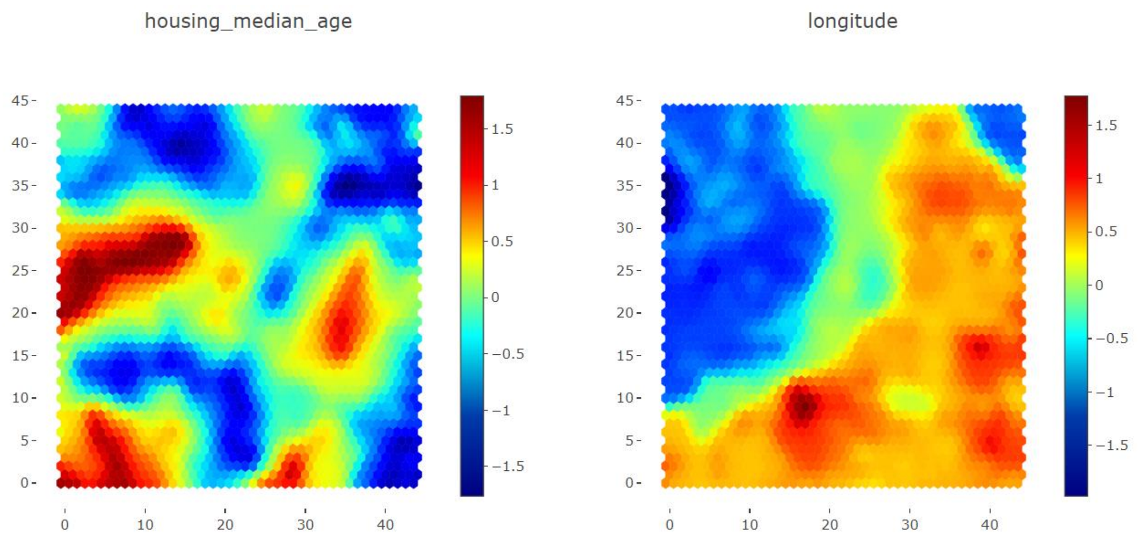


We can clearly see how any change in value in one of the features is reflected almost equally in all aspects. That is, **<u>they have</u>** **<u>a lot</u>** of common behavior **<u>patterns.</u>** In this particular case study, since we know the meaning of the attributes, we know that implicitly one implies the other, and that the relationship is of direct proportionality.

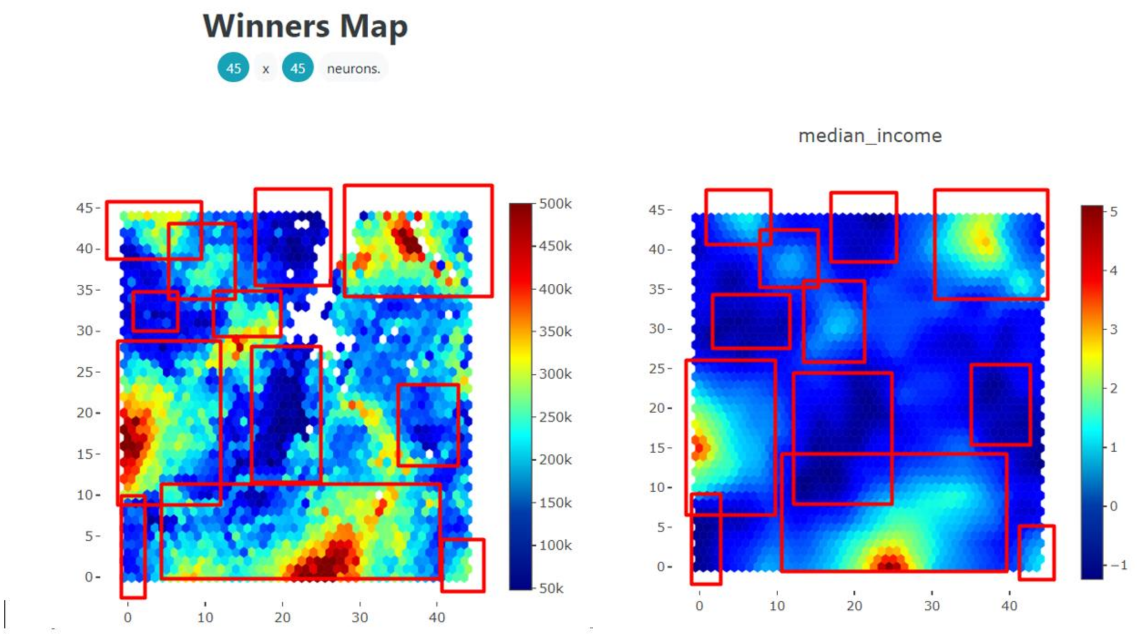
This **<u>gives us a lot of information</u>**, to beginwith, given the direct relationship of this example, we could eliminate one of the attributes of the model by being redundant, and thus obtain a smaller model, computationally saving memory and computing time.

It could be the case that the joint sum of several patterns in several characteristics translated into a third characteristic. That is, **we would find patterns of behavior.** This could be compared together with the winning neuron map and determine patterns between component maps and the latter.

With attributes that we hope do not save in real life any kind of relationship, we see how it translates very graphically in the maps of components, as is the example of the average age of a house and the length of its geographical coordinates.



Finally, we can compare the map of winning neurons with a map of components:



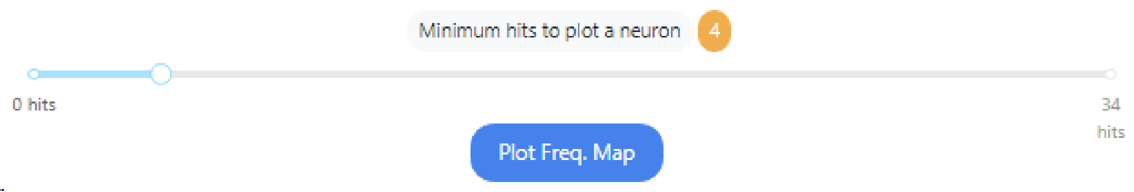
As seen in the image, the set of patterns is translated in the same way from the component map to the winning neurons map. Only the main ones have been highlighted in the image. However, we see how they are completely related, as we would expect in real life, since the income of families has a direct impact on the price they pay for their homes. Regions that concentrate higher incomes, have more expensive housing. The same goes for low and medium home rents and**prices. From this dataset, we can conclude that the determining component of housing price choice is the income of families.** Thus, if we had not known by the nature of the data the relationship between both characteristics, we would have discovered common behavior **patterns, in addition to a direct relationship.**

In more complex data, such as medical data, we could draw conclusions from patterns of behavior between various characteristics and the class to be predicted, which we might not otherwise have appreciated.

4.5. Frequencymapas a quality filter incomponent maps.

As we saw in section 4.3, sometimes certain neurons do not represent any data, since they have never been chosen as BMU of any input vector. As we also saw in point 4.4, component maps help us see behaviors, patterns, and anomalies common to the characteristics of the data.

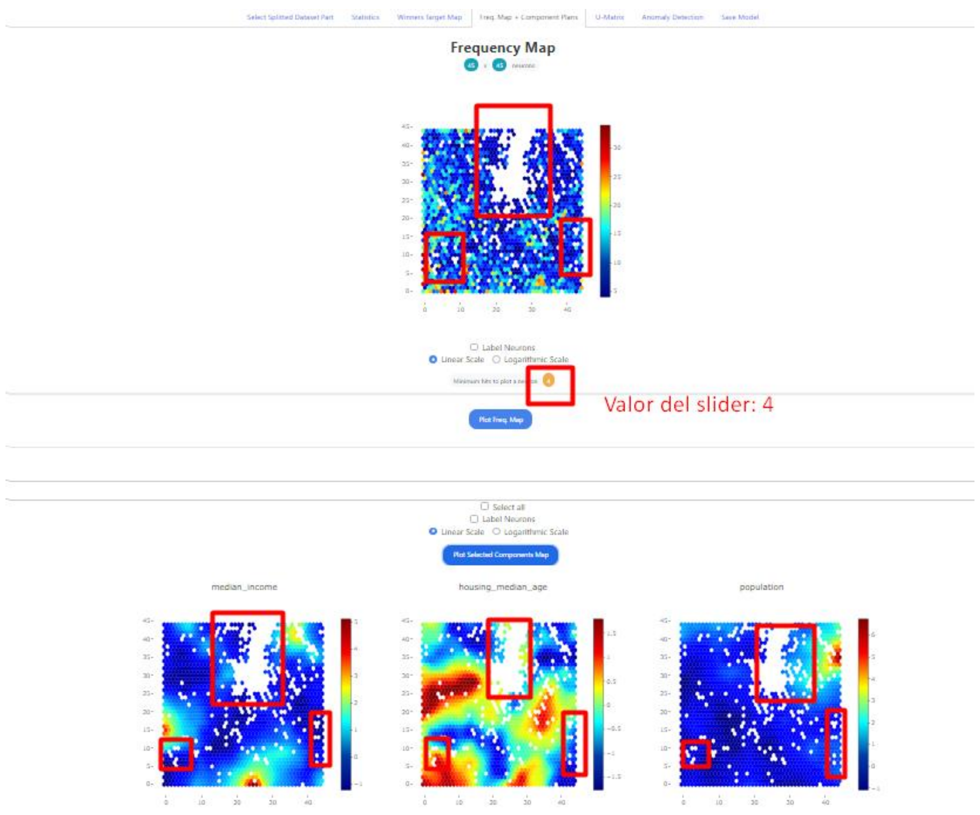
That is why, below the frequency map is a slider,which allows us to select the minimum number of times a neuron has had to be chosen BMU. This ensures that the painted data represents real data, and that they do so with a minimum number of examples to ensure that they represent actual data in a dataset and not point anomalies.



By using this slider, we remove from the graphs neurons that do not have a minimum frequency number, and this is done both in the frequency map and in all the maps of components drawn.

In this way, **we validate** in some way, that the patterns we visualize are not due to random or point **data, but that they are backed by real data.**

In the previous example, if we apply a minimum of 4, we are left with the following frequency map and the following component maps chosen:

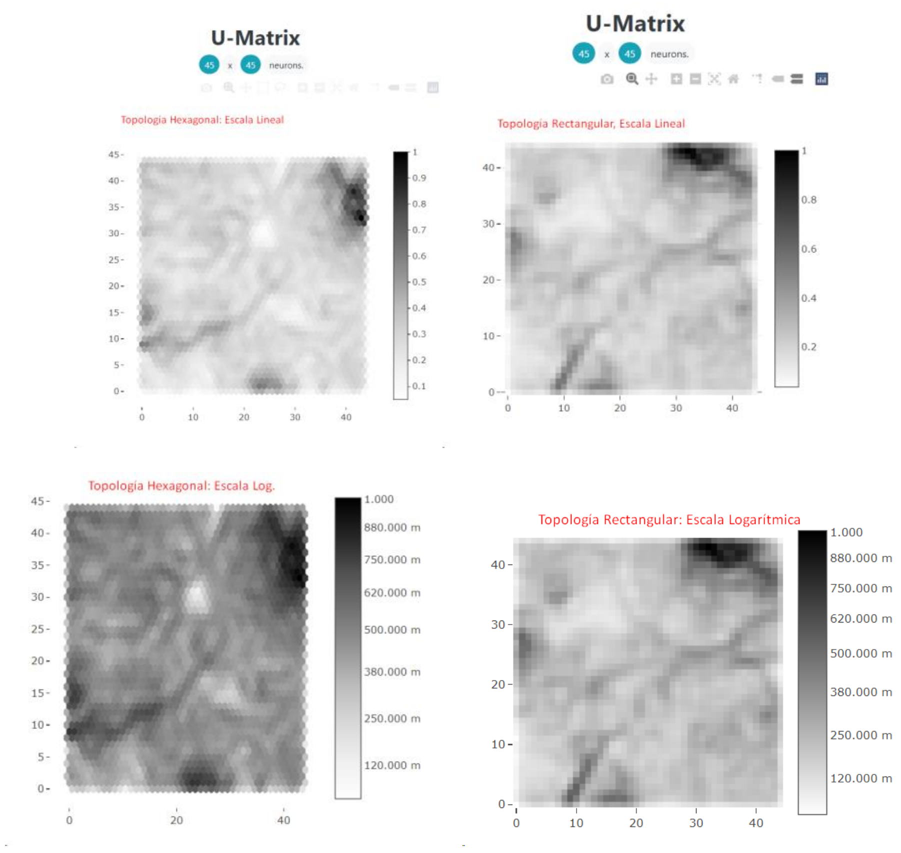


Where they all stop drawing the same neurons.

4.6. U-Matriz: Unified distance matrix

The unified distance matrix provides us with diverse information.

In the first instance, it shows us the "abrupt difference between the weights of contiguous neurons". That is, neurons whose weight vectors are much further away than from the rest of the neighborhood will have very dark values in the U-Matrix. This means that **<u>if the topology of the map is not preserved,</u>** as we mentioned in point 4.3, **<u>this will be reflected in the U-Matrix.</u>**



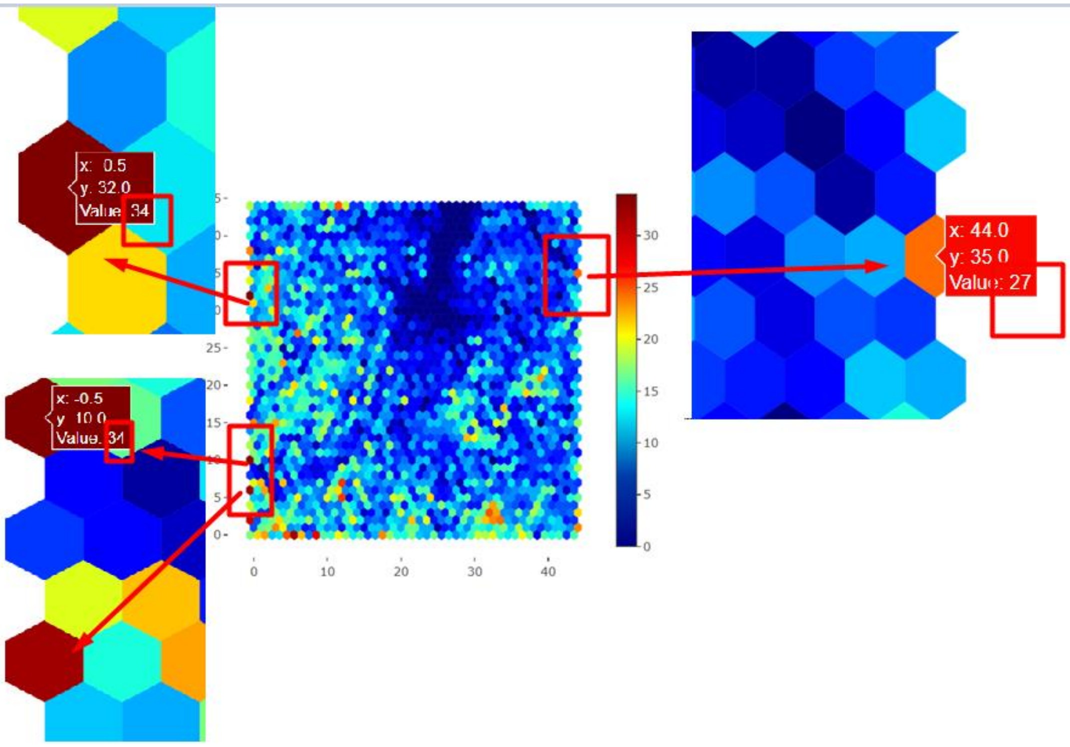
In contrast, clear zones will indicate that these neurons represent changes in the data gradually and proportionally to the size ofthe map, as expected from a well-trained and data-true model.

However dark areas do not mean that necessarily the model is poorly trained or poorly represented. In the case of **anomalous data,**these are usually reflected very well in such arrays.

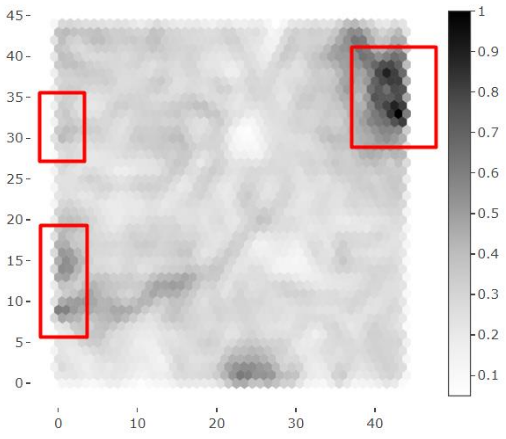
Normally when the map has very white and gray areas, the logarithmic scale is usually used, to better appreciate the behaviors described above.

In the example above we can compare the misrepresentation of certain data to the edges of the map. If we look at the frequency map again(of the hexagonal topology to simplify the explanation)

We see how there are **neurons** that have a much higher frequency index than their neighbors, which translates into a **reddish color around bluish colors,**behavior that we explained in the section of the frequency map.



If we look again at its Matrix U, we see how the big jumps translate into blacker colors:



In the neuron of the rectangle on the right, we see a very **black** color in the matrix u. This is because the jump in the frequency map is very high. The value of the anomalous neuron is 27 in the frequency map(which is directly related to the value of the weight vector when mapping examples),while that of the adjacent neuron on the left is 10. This is because the **weight vectors of neurons** are **far apart,more than normal in contiguous neurons.** Geographically it could be seen as the peak of a mountain in a valley. The valley is sloping, but the mountain protrudes.

On the other hand, in the rectangles on the right, we see how the color is more grayish, because the distances between the vectors of weights are high, but not so much. On the frequency map you can see how this is so: now the jump of the upper left neuron is 34 to 17. This means that a smaller number of data in difference from one neuron to another have been mapped into those neurons, and the weight updates of the neighbors have been equaling that difference, so now the color in the Matrix U is more grayish.

4. Conclusions

We have seen a practical case study on the use of the tool and seen the possible information that the different graphs can give us.

During this case study, using the application, we have detected behavior patterns and relationships between the characteristics of the data to be analyzed.

We have also seen that despite having obtained relatively low quantization and topographic errors, there are points on the map, specifically 3 neurons, where the representation on the map of the data could have been slightly better. This could have been achieved using other values the optimal search of hyperparameters "Random Search"or "Grid Search" included inthe tool, to achieve a better model. Since it was not the purpose of this case study, no attempt has been made to refine that representation.