IoTwins
BIG DATA OPTIMIZE INDUSTRY AND SERVICES

Grant Agreement N°857191

Distributed Digital Twins for industrial SMEs: a big-data platform

DELIVERABLE 3.4 – TOOLS FOR EXPLAINABILITY AND VISUALIZATION

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement № 857191

Document Identification

Project	IoTwins
Project Full Title	Distributed Digital Twins for industrial SMEs: a big-data platform
Project Number	857191
Starting Date	September 1st, 2019
Duration	3 years
H2020 Programme	H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)
Topic	ICT-11-2018-2019 - HPC and Big Data enabled Large-scale Test-beds and Applications
Call for proposal	H2020-ICT-2018-3
Type of Action	IA-Innovation Action
Website	iotwins.eu
Work Package	WP3 - AI services for distributed digital twins
WP Leader	UNIBO
Responsible Partner(s)	ESI
Contributing Partner(s)	THALES, UNIBO, ENSAM
Author(s)	Jean Christian Ahouangonou (ESI)
Contributor(s)	Vincent Thouvenot (THALES), Andrea Borghesi (UNIBO), Michel Quentin (ESI), Justin Teliet (ESI), Morgan Cameron (ESI)
Reviewer(s)	Francesco Giovannini (GCL), Thomas Courtat (THALES)
File Name	D 3.4 – TOOLS FOR EXPLAINABILITY AND VISUALIZATION
Contractual delivery date	M30 – 28 February 2022
Actual delivery date	M34 – 17 June 2022
Version	1.2
Status	Final
Type	DEM: Demonstrator, pilot, prototype
Dissemination level	Public
Contact details of the coordinator	Francesco Millo, francesco.millo@bonfiglioli.com

Document log

Version	Date	Description of change
V0.1	January 20th, 2022	First draft.
V0.1	January 24th, 2022	Update with dash xai component information (Vincent Thouvenot).
V0.2		Update with the Instrospector & visualization tool (Michel Quentin).
V0.3		Update with ESI Inspector tool description (Justin Teliet).
V0.4	February 28th, 2022	Review and completion of missing sections (J. Christian Ahouangonou).
V0.5	March 21st, 2022	Ready for the first review cycle.
V06	April 25th, 2022	Prepare the document for the second review cycle. The 1rst review feedbacks are taken into account.
V 1.0	May 16th, 2022	Document prepared for consortium review.
V 1.1	June 8th, 2022	Update after consortium review of the version 1.0. Document prepared for EU submission.
V 1.2	June 17th, 2022	Final version.

Executive summary

This document is the D3.4 dedicated to the results of the Task 3.6 of IoTwins. This task is dedicated to the tools for the AI and ML model visualization and model introspection. Due to their complexity, machine learning models tends to be prototypical black boxes, whose inner mechanisms are difficult to comprehend. These tools are mandatory to machine learning and simulation toolbox of the IoTwins platform to obtain a better machine learning acceptance by operational, help to improve machine learning model, improve the transparency of models, etc..

In this deliverable, we present three services. The *Dash xAI* component provides a web application with a toolbox that allows to interpret a machine learning model, both at a local level (i.e. explain the prediction made for one given instance) and at a global level (i.e. explain the general behaviours of the model). *ESI inspector* tool is a web application that allows to represent the high dimensionality data in a two-dimensional space using linear and non-linear techniques such as PCA (Principal Component Analysis), LLE (Locally Linear Embedding) and t-SNE (t-distributed Stochastic Neighbor Embedding). *Inendi Inspector* allows any user to perform interactive explorations over very large amounts of data. Thanks to its innovative set of visualizations, the user is freed from the classical burden of query-based interactions and can then use all their logical and deductive power to discover valuable insights from the data.

The deliverable contains two main parts: the first one is dedicated to the service development methodology and the second one to the description of the developed services. For each service we provide a technical description of the background and explain its usage on a use case.

Table of Contents

1 Introduction

This document presents the services developed as results of task 3.6 of the IoTwins project. The objective of this task is to provide tools for big data visualization technology, as well as deep inspection of ML and AI models. Since "ethics" and "safety" are gaining more and more interest from users, it is important to avoid the current "Black Box" effect of trained models. This will help the user improving the model explainability and interpretability during the model validation. This task will produce tools based on high-dimensional advanced big data visualization.

The official repository hosting the code of the services, together with the documentation, remains in its original address: https://gitlab.hpc.cineca.it/iotwins/ai-services. This repository is hosted in a CINECA-controlled area (CINECA is one of the partners of the consortium), so access must be granted according to CINECA policy.

And the users can access to the discussed services at this link: https://iotwins-paas.cloud.cnaf.infn.it/

2 Service development methodology

All the services proposed in the current deliverable are GUI-based applications. They are self-contained Docker containers. As already stated in the section 2 of the delivery D3.3, the Docker framework aims at the creation of compact services using OS-level virtualization, and the delivery of said software (SW) services as self-contained packages, endowed with all the required companion libraries and SW tools. In this way the AI-services can be used as standalone modules, while at the same time they can be deployed on the IoTwins platform produced as outcome of WP2.

We will describe below the three (3) services for data visualization and introspection implemented as demonstrators. The services can be fully tested by running them on the IoTwins platform, which relies on the PaaS orchestrator provided by INFN (INFN is one of the partners of the consortium),

The data types required for each service are different from one service to the another. So, each service will describe the kinds of data formats they are supporting as entry.

Common open sources tools, such as the programming language Python, the Machine Learning (ML) library scikit-learn[1] as, the web based GUI library Dash, and Plotly[2] for the data visualization are the main tools used to implement the different services.

The documentation is available in the online repository itself, as it is closely related to the code implementation and the services interface.

[1] https://scikit-learn.org/stable/
[2] https://plotly.com/

3 List and description of developed services

This section presents the different services developed for and integrated in the IoTwins platform to visualize, explain, and explore as much as possible ML & AI model results.

3.1 List of developed services

Three different services are provided and described in this section:

- Section 3.2 presents s **Dash xAI** component: global and local post-hoc machine learning explanation,
- Section3.3 illustrates the **Introspector-tool** that combines several advanced visualization techniques and algorithms to help the user introspect his data,
- Section 3.4 presents **Inspector**, a tool to manage large amount of data and provide tools for efficient visualization and clusterization.

Each section will contain a short theoretical introduction followed by a presentation of the tool supported by visual material. The two first ones will bring an illustration base a use case

3.2 Dash xAI component

3.2.1 Introduction

Machine Learning models are used for various applications with already successful results. Unfortunately, a common criticism is the lack of transparency associated with these algorithm decisions. This is mainly due to a greater interest in performance (measurable nonspecific tasks) at the expense of a complete understanding of the model. There are two kinds of interpretability methods: global methods that aim at explaining the general behavior of a model, and local methods that focus on each decision of a model. In another point of view, interpretability methods also split into agnostic and non-agnostic categories. Agnostic methods (also called post-hoc explanation) consider the model as a black box. On the other hand, inherent or non-agnostic methods can modify the structure of a model or the learning process to create intrinsically transparent algorithms. Local explanation focuses on a single instance and examines what the model predicts for this input and explains why.

We present in this report a component developed by Thales for the IoTwins project. The component consists in providing tools for black box machine learning model inspection. We provide in the Table below the approaches that are available in the component.

Explanation type	Objective	Tools
Global	Plot the effect of the features on the model outputs	Partial Dependence plot
		ALE Plot
		Influence plot (based on ethik AI)
	Plot the effect of the features on the model errors	Performance plot (based on ethik AI)
	Rank the features according their importance on the model outputs	Ranking based on influence plot (based on ethik AI)
	Rank the features according their importance on the model errors	Ranking based on performance plot (based on ethik AI)
		Permutation features importance
Local	Rank the features according their importance on the prediction made	Shapley Values (based on ShapKit)
	Find what are the minimal perturbation to do in the instance to change its prediction to a targeted prediction	Counterfactual Exemples

The component can be used to explain and interpret both regression and classification models.

3.2.2 Technical description

The reader can find the results of the plots described in the technical description can be find with an interpretation in the section dedicated to the example of use of the component.

3.2.2.1 Partial Dependence Plot

Partial Dependence Plot (PDP) (Friedman, 2001) plots the marginal effect of one or two features on the model output. With this plot, we can see for instance whether the relationship between a feature and the output is linear, quadratic or more complex. These plots present several advantages: they are easy to compute and implement in case of uncorrelated features and the interpretation is clear. However, they have several disadvantages: as we consider averaging marginal effects, some heterogeneous effects can be hidden and we assume the independence between the features, which is often unrealistic.

3.2.2.2 Accumulated Local Effect Plot

To avoid correlation issues of PDP, Accumulated Local Effect Plot (ALE) (Apley & Zhu, 2016) are appropriate. ALE describes how features influence the model output on average. While PDP computes the average of the model prediction when each instance has a given value, the ALE plot computes the change in the model prediction in a small window of the feature around a given value for the instances in that window. ALE deals with correlated features. It is fast to compute and easy to interpret, but it depends on the number of intervals chosen, and is harder to implement than PDP.

3.2.2.3 Ethik AI

(Bachoc, Gamboa, Halford, Loubes, & Risser, 2020) consider an information-theoretic framework in which they modify the distribution of the original testing set by stressing the mean value of a function of its variables (or simply by stressing the mean value of given variables), while minimizing the Kullback-Leibler information between a modified distribution and the original distribution.

3.2.2.4 Influence Plot

With Influence Plot, we stress the mean of a feature and study the impact on the average model prediction. For a given feature, with a mean equal to m1, we wonder what happening whether we stress the feature mean to m2. We could sample from the feature distribution and observe how the model average prediction changes with the new feature distribution. The problem here is that we only stressed the marginal distribution of the feature. However, this feature can be correlated to some others features of the model and it can lead to unrealistic instances. To address this issue, the key idea behind this function is to stress the joint distribution of the features set, the true label set and the model's outputs while minimizing the distance to the original distribution according the Kullback Leibler divergence. With influence plot, we represent the average prediction of the ML model when the mean of given feature is stressed according the process just described.

3.2.2.4.1 Performance Plot

With Influence Plot, we stress the mean of a feature and study the impact on the model error, that can be computed according different criteria, depending whether it is regression or a classification model (RMSE, MAPE, AUC, accuracy). It works in similar way that Influence Plot, except that we consider the evoluation of the error when the mean of the given feature is stressed instead of the model's average prediction.

3.2.2.4.2 Ranking based on Influence Plot

From the Influence Plot computed for all features, we can deduce a feature importance measure by computing, for a specific feature, the mean absolute difference between their influence curve and a horizontal curve equal to the original average prediction.

3.2.2.4.3 Ranking based on Performance Plot

From Performance Plot, we can deduce some feature importance measures. For all the features, both the minimum and maximum of the error criteria after features distribution stressing are retained.

3.2.2.5 Permutation feature importance

(Fisher, Rudin, & and Dominici, 2019) introduce the permutation feature importance measure, based on a similar idea to the classical MDA criterion used for Random Forest. Here, the feature importance is measured by computing the increase in the model prediction error after permuting the feature. By permuting a feature, we mean that a single feature value is randomly shuffled. This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature. Permutation feature importance measure represents the increase in model error when a single feature value is randomly shuffled. Feature importance provides a highly compressed, global insight into the model's behavior.

Note that when the error ratio is used instead of the difference-based one, the result has no unit. Thus, results obtained for two different problems are comparable. That takes into account both the main feature effect and the interaction effects on model performance. Another advantage is that permutation feature importance measure does not require to retrain the model. On the other side, permutation feature importance measure is linked to the model error. In some cases, one can seek out to measuring how much the model output change without considering the performance. Moreover, one needs some labelled data to

compute feature importance measures. In the case of correlated features, a bias in the estimation appears. We use the implementation provided by scikit-learn[3].

3.2.2.6 Shapley Values

In Collaborative Game Theory, Shapley Values (Shapley, 1953) allows to distribute a reward fairly among players according to their contribution to the win in a cooperative. Shapley Values offer a solution of local explanation from additive feature importance measure class ensuring desirable theoretical properties. We can make the following correspondence between Game Theory and model interpretability:

- The features are the players;
- The model is the game;
- The feature attribution is the gain attribution.

A prediction can be explained by assuming that each feature value of the instance is a “player” in a game where the prediction is the payout. The objective is to fairly distribute the payout around all features to obtain the prediction.

A major challenge in computing Shapley Values is the number of calculations to perform: the potential coalition, and so this number, grows exponentially as a function of the number of features. Some approaches to approximate this calculation have been proposed (Štrumbelj & Kononenko, Explaining prediction models and individual predictions with feature contribution, 2014), (Štrumbelj & Kononenko, 2010), (Lundberg & Lee, 2017), (Aas, Jullum, & Løland, 2019), (Grah & Thouvenot, 2020).

Shapkit [4] is a Python module to use Shapley Values as local explanation of Machine Learning model, based both on Monte Carlo approximation and weighted least square optimization problem. The method is a post-hoc explanation, so the user does not have to change her usual pipeline. In the application, we use ShapKit to compute Shapley Values in this context.

3.2.2.7 Counterfactual examples

A counterfactual explanation (Wachter, Mittelstadt, & Russell, 2018) describes a causal situation where we can say something like ``If this event does not occur, then that other event would not occur''. This type of reasoning can be applied to provide local explanation. A typical example is to put in light that if two features, A and B are increased by a% and b% then the prediction of the considered model would change to a target value. One first advantage of local interpretability with counterfactual examples is that it allows some contrastive explanations, which are well adapted to humans. There are several desirable properties to obtain a ”good” counterfactual examples:

- The prediction of the counterfactual example should be as close as possible to the target prediction;
- The counterfactual example should be as similar as possible to the original instance;
- Between the counterfactual example and the original instance, as few features as possible should be different;
- The counterfactual examples should have feature values that are valid and likely;

Multiple diverse counterfactual explanations should be generated to take into account the ``Rashomon effect'' i.e. the situation in which an event is given contradictory interpretations or descriptions by the individuals involved.

[3] https://scikit-learn.org/stable/
[4] Shapkit | shapkit (thalesgroup.github.io)

Counterfactual example computation approaches can be divided into four groups:

- Those which directly find counterfactual examples in the dataset.
- Independence-based methods, in which we assume that the input features of the model are independent. Amongst others approaches, some techniques use combinatorial solvers or evolutionary algorithms, and others use gradient-based optimization to minimize a loss with feasibility and diversity constraints. The main issues of these approaches are that they ignore the correlation between features, which is unrealistic in the majority of use cases where machine learning models are used;
- Causality-based methods, in which a Pearl's causal modeling is used. These approaches usually require knowledge of the system of causal equations or the causal graph. In practical cases, knowledge of the true causal models or graph is a very (too?) strong assumption;
- Dependence-based approaches, in which we use some generative models. These approaches allow to take into account the dependencies between the features.

In this component, we directly find the counterfactual examples in the dataset. The instances which are nearest to the original instance in the target prediction area are selected. The distance between instances is computed with a Gower-based distance.

3.2.3 Example of component use

3.2.3.1 General description of the component

3.2.3.1.1 Dependencies

The component is dedicated to post-hoc global and local explanation of machine learning models. These models can be used for regression or classification tasks and are based on tabular data. The component is implemented in Python 3 and uses the module *dash* for the implementation of the application.

The models can be trained with *scikit-learn*, some format similar that *scikit-learn* (i.e. *catboost*) or *tensorflow*. We do not adapt the implementation for PyTorch models. There is no difficulty to extend the component to PyTorch models. The requirements of the component are given by the list below.

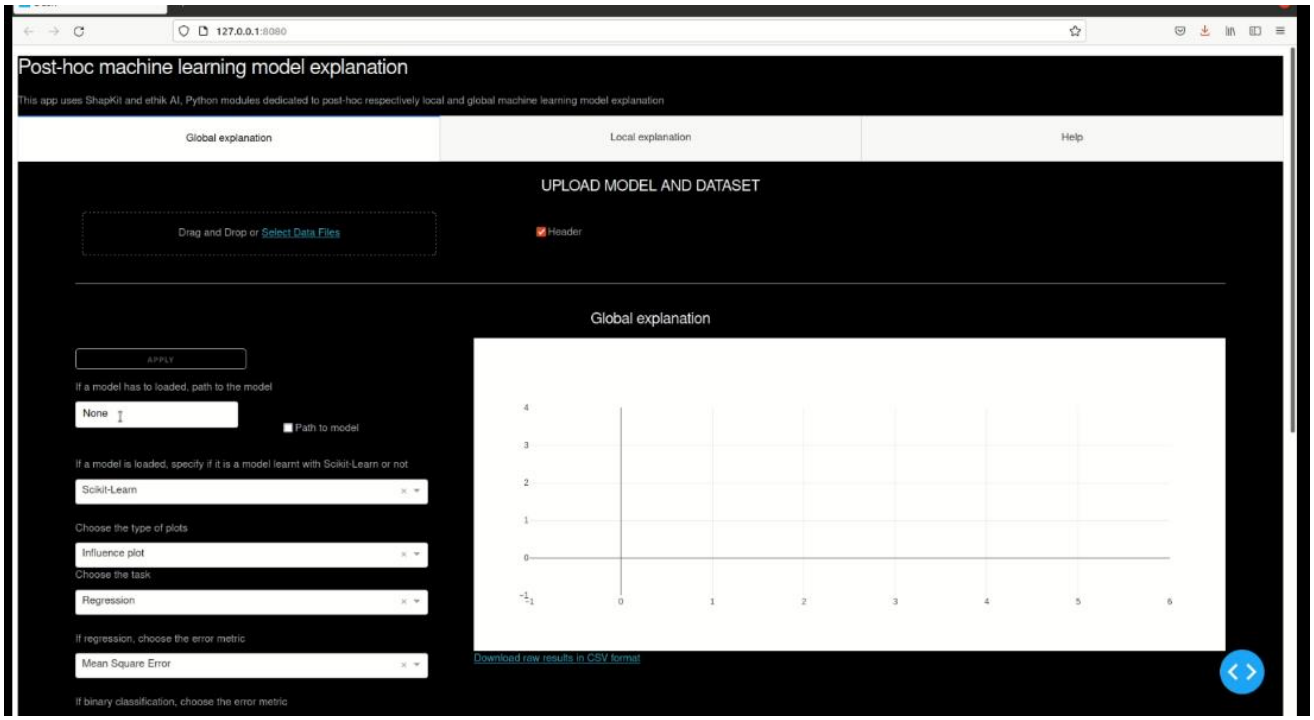
- Python 3.8
- Python module:
    - numpy==1.19.5
    - pandas==1.1.0
    - shapkit==0.0.4
    - plotly==4.9.0
    - fire==0.3.1
    - scikit-learn==1.0.1
    - ethik==0.0.4
    - PyALE==1.1.1
    - dash==1.14.0
    - dash-core-components==1.10.2
    - dash-html-components==1.0.3
    - Flask==1.1.2
    - tensorflow==2.6.2
    - catboost==0.24.3

3.2.3.1.2 Visual overview

The component is divided into three panels.

3.2.3.1.2.1 Global explanation panel

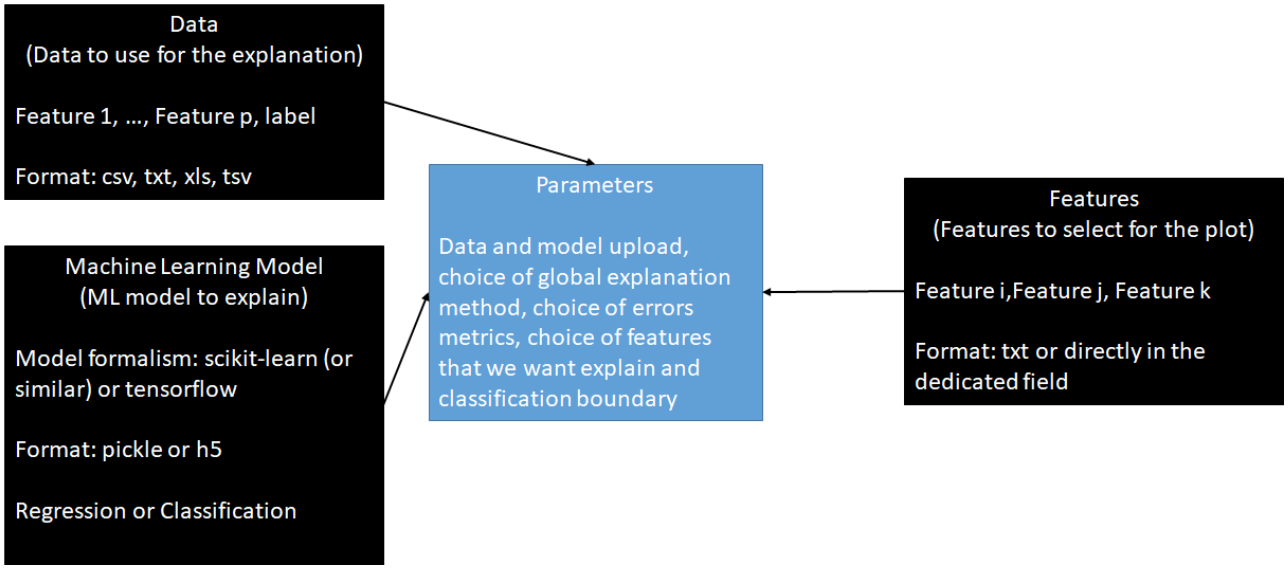
The first panel is dedicated to global explanation. The Figure below shows the given panel.



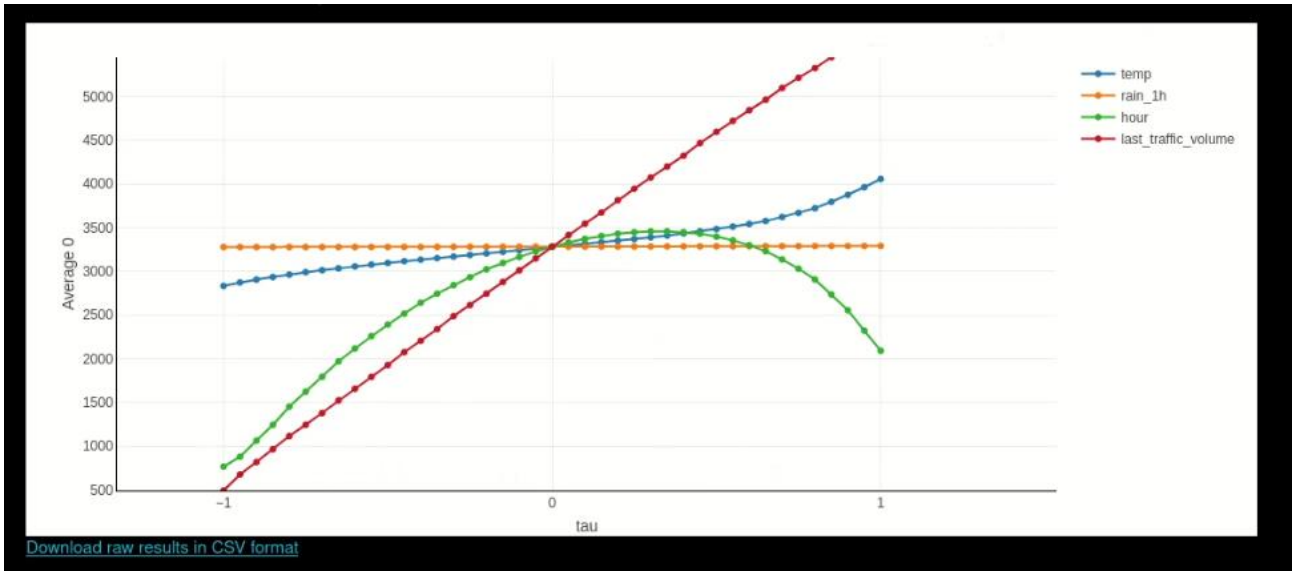
At the top level of the component, there is a button to upload data. The data can be a csv file, a xls file, a txt file or a tsv file. The last column contains the label of each instance, all the others the features used by the machine learning model. On the left column, there are the hyperparameters of the different global methods. For some methods, it is not mandatory to load the machine learning, whose we want to have an explanation and that has been learned and saved beforehand. Only the model predictions have to be provided. In this case, the model prediction is given by the last column of the dataset uploaded. If a model has to uploaded, the path has to be provided. The user can choose between scikit-learn and other format of model, according the format of the machine learning provided. The model uploaded has to be saved in pickle or in h5 format.

Then, the user can choose the method used and parametrize the method. In particular, the user can specify the features that will be represented. In some cases, considering all features is not optimal: the resulting plot will be unreadable. Moreover, when plotting the effect of a feature on the outputs or error, we can be more interested by focusing on a small set of features. For that, the user can use the element below “If a subset of features is considered, path to the selected features”. The user can directly specify the features to study in the field or she can indicate the path to a txt file with the list of features. In the two cases, if the user is interested by two features: features_1 and features_2, they will indicate features_1, features_2, without space and with a “,” as separator, either in the field or in the file. If they use the path to the text file, they have to check the button path to feature. In the case of the permutation feature importance criterion, the user can specify how many features to plot. If they indicate a number, i.e. n, then the n more important features will be represented. With the last field, the user can specify the classification boundary, i.e. in binary classification what score predicted lead to the class predicted by the model. If the score is greater than this boundary then the class predicted is the class 1, else the class predicted is the class 0.

We summarize the functioning of the left column and the data uploading in the Figure below.



The right column of the panel is dedicated to the resulting plot. The Figure below shows an example of resulting curves.



3.2.3.1.2.2 Local explanation panel

The second panel is dedicated to local explanation, i.e. the explanation of a prediction made by the model for one given instance. For instance, for one instance, the model has predicted a critical temperature equal to 4, and we want to understand why. On the top the user can provide the data with Drag and drop or Select Data Files field. The file can be csv, xls, txt or tsv format. On the left column of the panel, the user finds the following elements, among others:

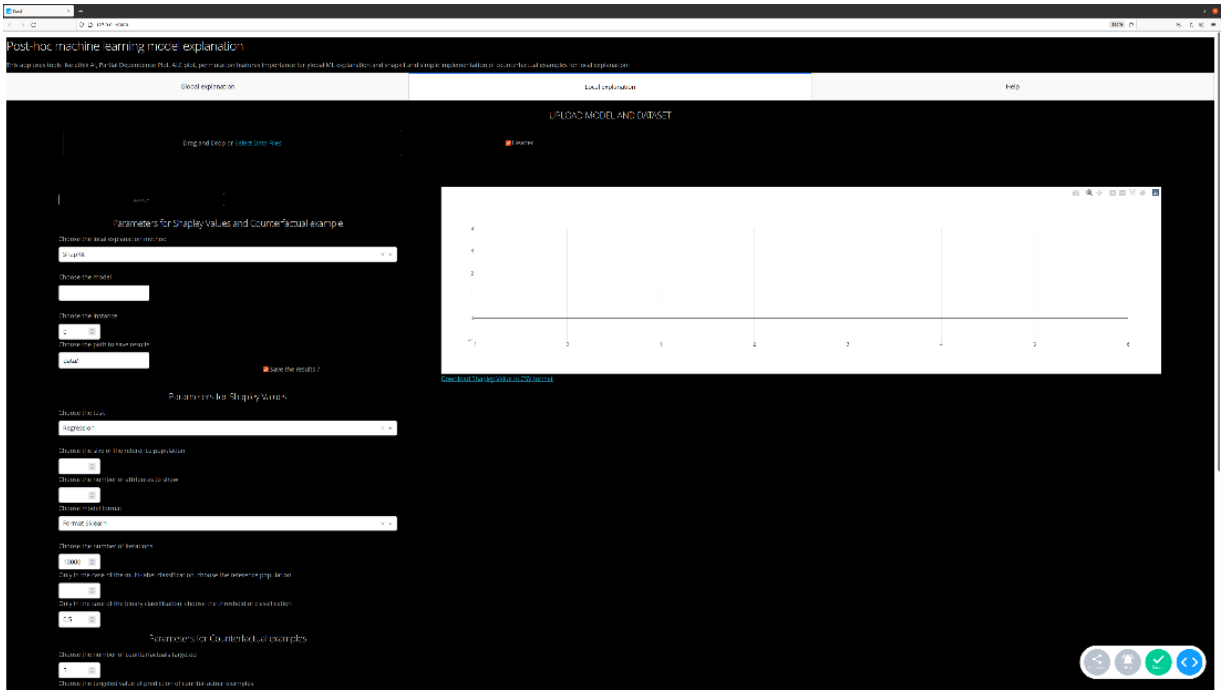
- Choose the local explanation method: choose between ShapKit (Shapley Value) and counterfactual example
- Choose the model: Path to the model to explain. Can be a pickle (for sklearn model format) or a h5 (for Keras model format) file
- Choose the task: Model task, can be regression, binary classification and multi-label classification
- Choose the instance: Index of the instance to explain

The other elements are dedicated to the use of shapkit or to the use of counterfactual examples. Concerning shapkit, We propose an application dedicated to local explanation for three supervised machine learning tasks: regression, binary classification and multi-label classification. For regression, the reference population is the whole population. In this case, the Shapley Value computes some kind of a deviation between the prediction for a typical instance and the prediction for the instance under consideration. We are interested here to study the difference between the prediction for the opposite class and the instance of interest. We look at the features that contribute the most to change the class. For multi-label classification, we compute the Shapley value according the maximum probability of the instance of interest and we consider as reference population the population predicted as another class that the instance to explain or one class chosen by the user. Moreover, two models formats can be used: Sklearn format (including module like catboost) and Tensorflow/Keras format. The parameters are given below:

- Choose the size of the reference population: size of the reference population. By default, the whole reference population is used.
- Choose the number of attributes to show: Number of features to plot on the graph. By default, all features are represented.
- Choose model format: Choose the format of the model, can be Sklearn format or Keras format.
- Choose the number of iterations: Number of Monte Carlo iterations for the approximation of the Shapley Values.
- Only in the case of the multi-label classification, choose the reference population: By default, the reference population is the instance predicted in another class that the instance of interest. If indicating an integer, the reference population is the individuals predicted in the class corresponding to the integer.

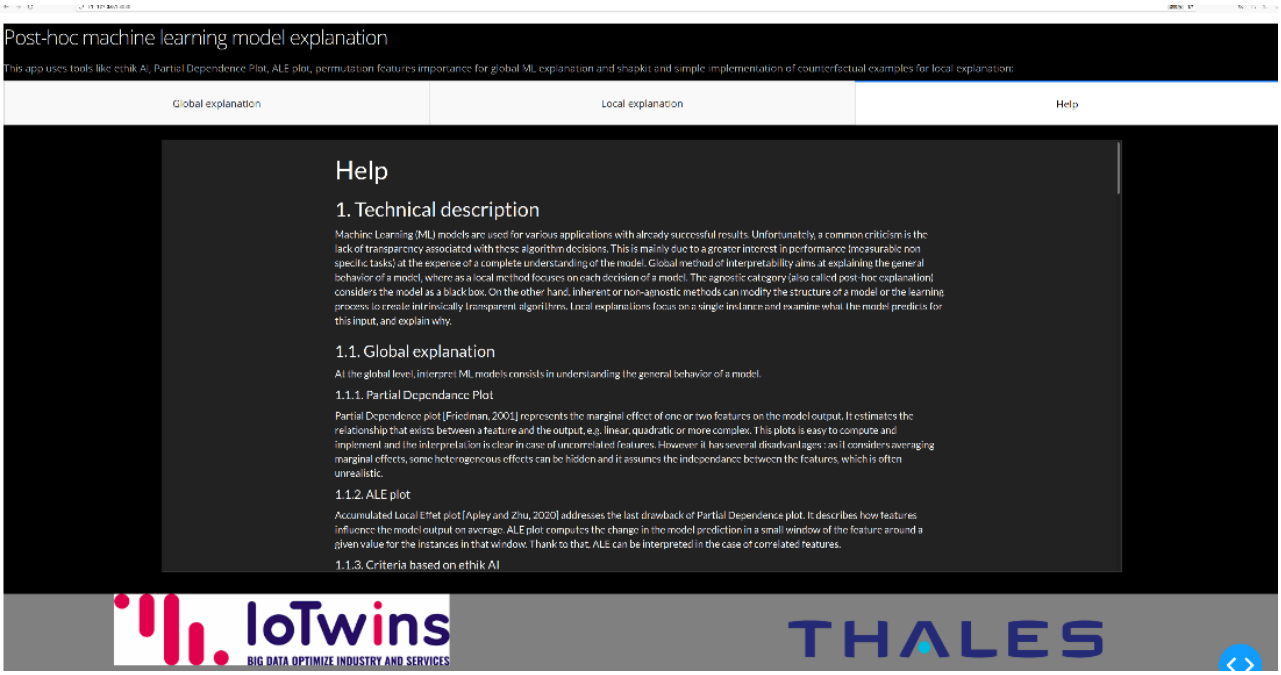
Concerning counterfactual example, the parameters are given below:

- Choose the number of counterfactuals targeted: select the number of counterfactual examples to compute and show
- Choose the target value for prediction of counterfactual examples: choose the target values outcome of the ML model. If the opposite option is enabled, we will search for a classification model a counterfactual example in the opposite class to the one predicted for the instance of interest. If we indicate a float, then we search counterfactual examples whose prediction is around this target. The definition of around is obtained with another parameters
- If categorical features, path to a csv with the names of categorical features
- If group of categorical features (i.e. when if we use one hot encoder for one feature), path to a pickle with a list of lists of features by group
- Choose if the prediction of the candidate should reach exactly (=0) or not the targeted value
- If previous item = 1, choose the relation between the prediction of counterfactual examples and the target
- If previous item is area, choose the size of the area around the targeted value
- Choose the minimum value of MAD (avoid dividing by 0)
- Only in the case of the binary classification, choose the threshold of classification: Threshold of the score for the final prediction.
- Download Shapley Value in CSV format: Download the computed Shapley Values in a csv file.



3.2.3.1.2.3 Help panel

The third panel is a help panel. It gives a brief description of the methods involved in the component, a description of the component, the important references, the credits to IoTwins project and the license of the component.



3.2.3.2 Use case description

The use case consists in predicting superconducting critical temperature based on the features extracted from the superconductor chemical formula. A superconducting material is a material that conducts current with zero resistance. It can be applied in many fields: Magnetic Resonance Imaging systems, superconducting coils used to maintain high magnetic fields, transport of electricity, etc. Although this large possibility of use, there are two major drawbacks. First, the superconductor conducts current with zero resistance only at or

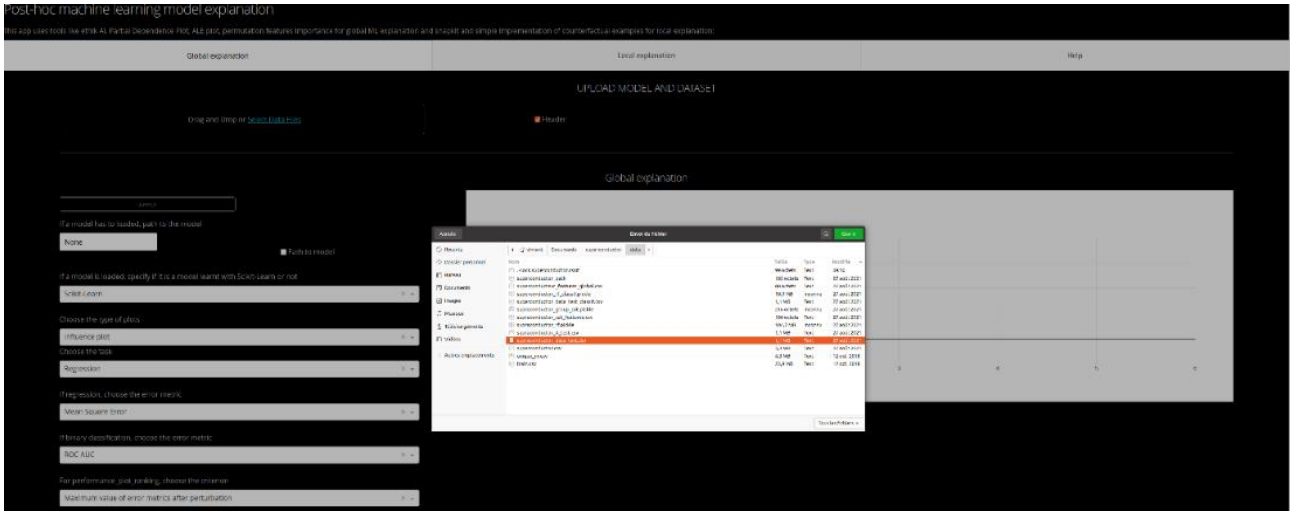
below its critical temperature and second it is very hard to predict the critical temperature for a superconductor.

We use the data described in (Hamidieh & Kam, 2018) that provides 21 263 superconductors features based on thermal conductivity, atomic radius, valence, electron affinity, and atomic mass, and the presence or not of mercury and osmium, a class according the quantity of iron and the quantity of calcium. These features are summarized in the Table below.

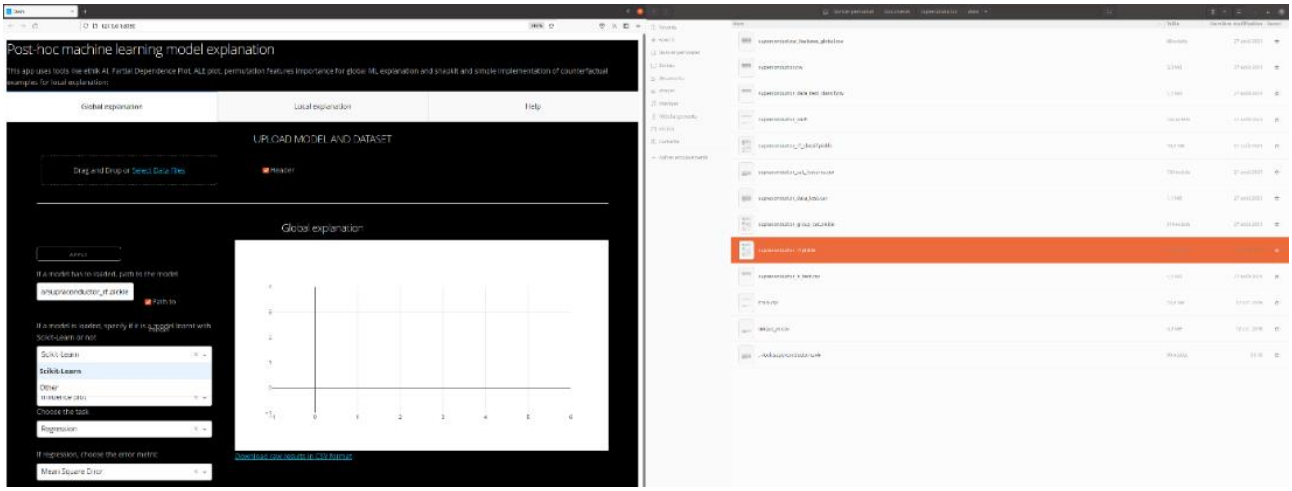
Type	Unit	Description
Atomic Mass	AMU	total proton and neutron rest masses
First Ionization	kJ/mol)	Energy energy required to remove a valence electron
Atomic Radius	pm	Calculated atomic radius
Density	kg/m3	Density at standard temperature and pressure
Electron Affinity	kJ/mol	Energy required to add an electron to a neutral atom
Fusion Heat	kJ/mol	Energy to change from solid to liquid without temperature change
Thermal Conductivity	W/(m × K)	Thermal conductivity coefficient κ Valence no units typical number of chemical bonds formed by the element

3.2.3.3 Illustration of global interpretation of machine learning model

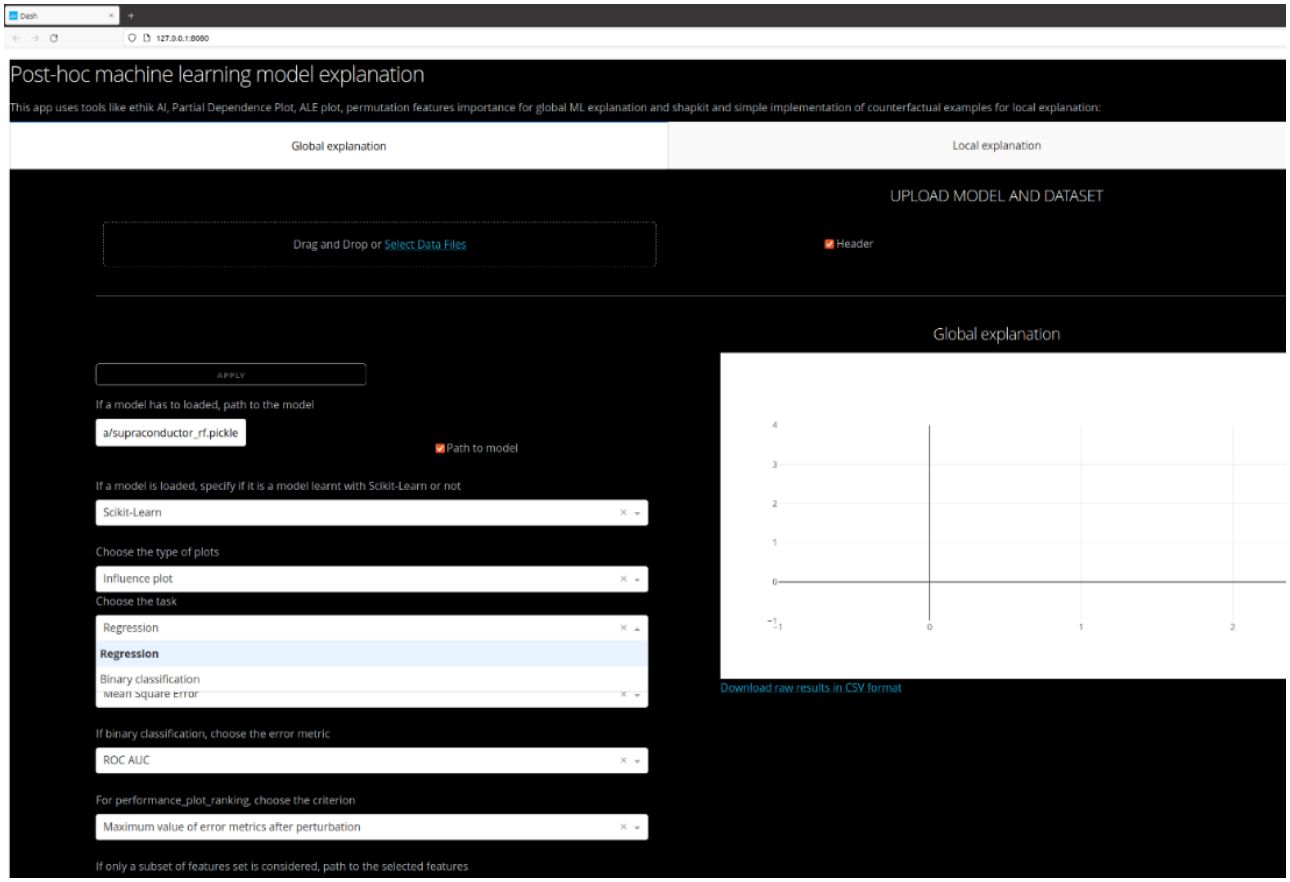
As shown on the Figure below, when the user clicks on “select dataf”, it opens a window where the user can select the path to the data. In the file, the first columns are the features used by the machine learning model and the last columns is the label.



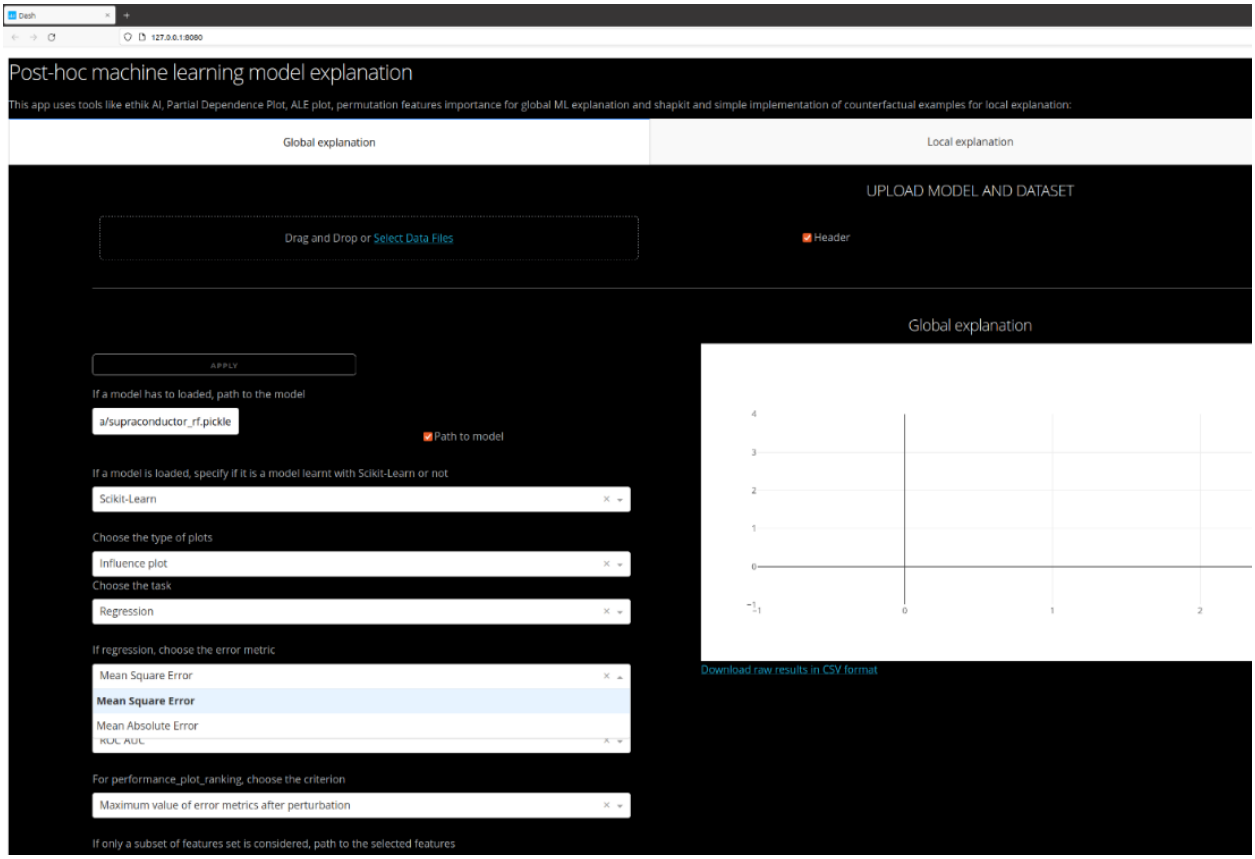
Then the user can provide the path to the model. It has to be a pickle file or a h5 file. The user indicates if the model has been trained with scikit-learn or with another machine learning library. On our example, the model has been trained with scikit-learn and is a random forest.



The user specifies whether the machine learning achieves regression or binary classification. For our example, we consider a regression task.

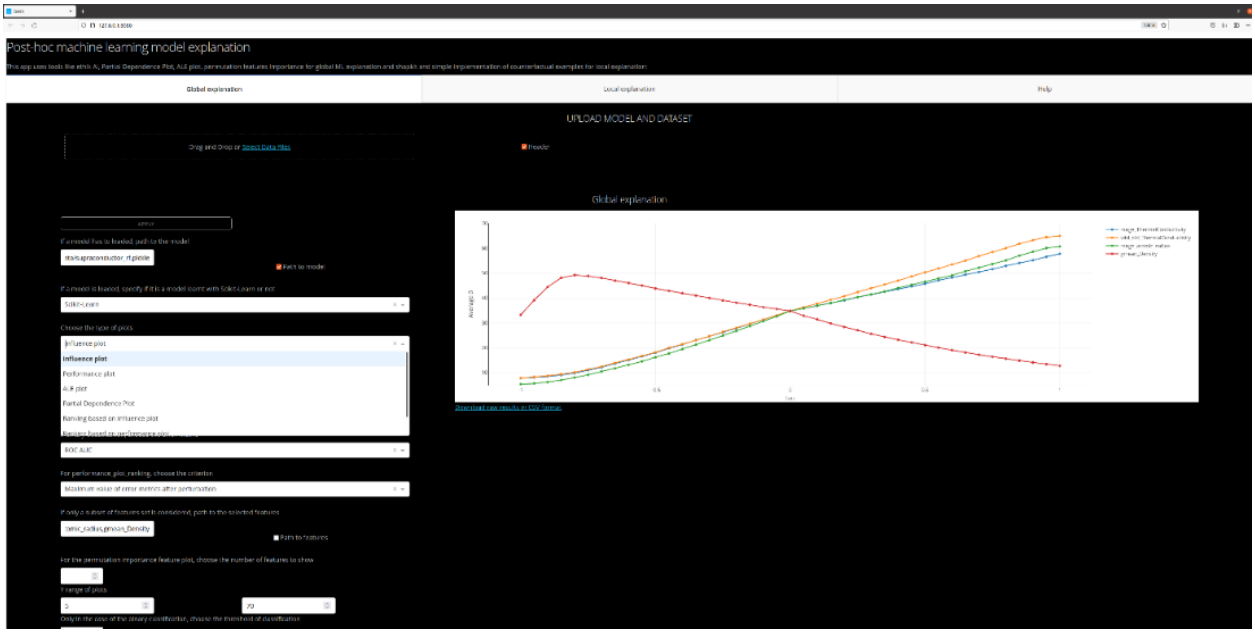


For the global explanation based on model performance, for regression task, the user can choose between two metrics: the means square error (MSE) and the Mean absolute Error, the first one penalizing more strongly the high errors. We choose to use the MSE.

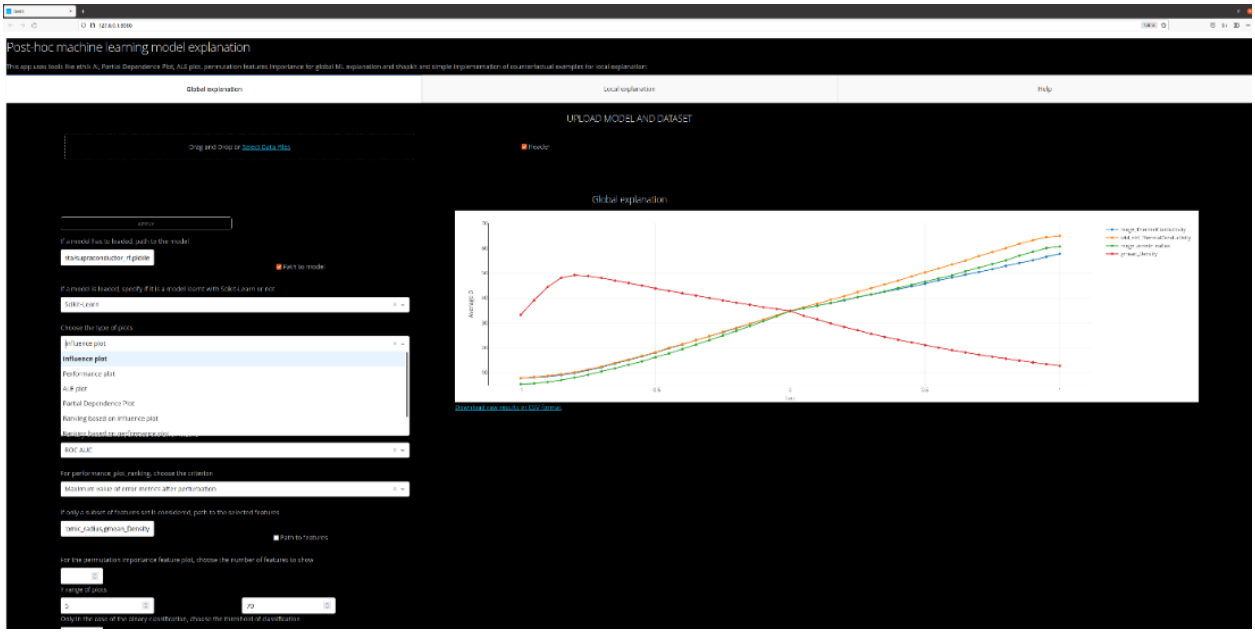


Then, we choose to use the influence plot. As explained in the technical part, Influence plot consists in representing the evolution of the average prediction when the average value of a feature is stressed. The stressing of the feature average value is performed according an information-theoretic framework. We are interested in plotting the influence plot for four features: the range of the thermal conductivity, the weighted standard deviation of the thermal conductivity, the range of the atomic radius and the geometric mean of the density.

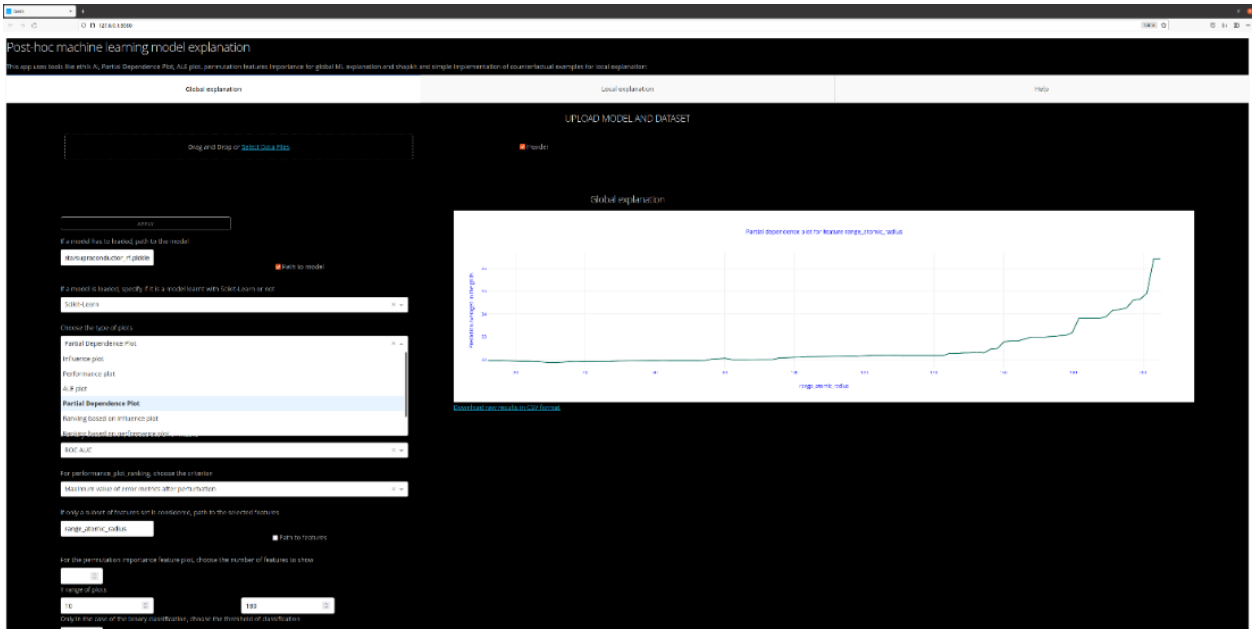
The plot given by Figure below shows that, if we stress the average of the three first features to increase them, then the average critical temperature predicted by the model increases. Concerning the geometric mean of the density, the effect is different: except if we strongly stress the mean of the variable to obtain a low value, when the mean of the geometric density increases, the average model prediction decreases.



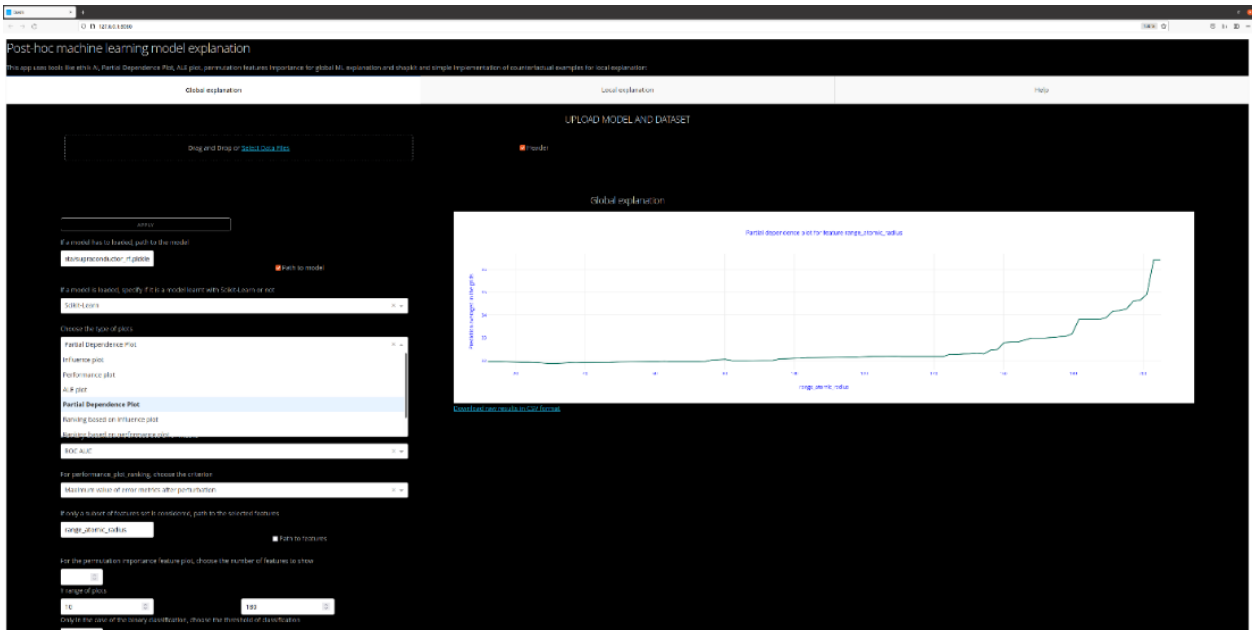
Then we use performance plot. In this plot, we stress the mean of each feature in similar way that for the influence plot and study the impact on the MSE. Except for the extreme values, where uncertainties are more important, this performance plot follows similar patterns that the influence plot.



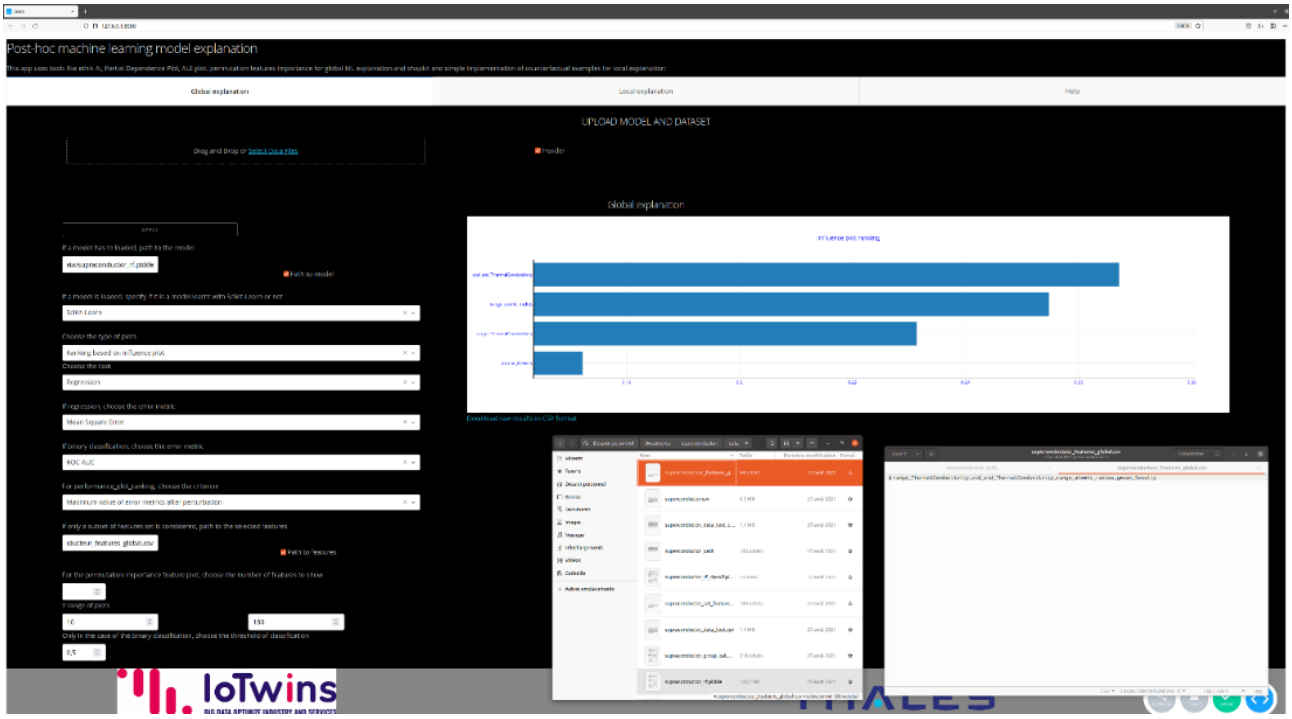
Another global explanation approach is based on partial dependence plot (PDP). The PDP shows the marginal effect one or two features have on the predicted outcome of a machine learning mode. We give in the Figure below the PDP for the range of atomic radius. According to this plot, the critical temperature predicted by the model grows when the range of atomic radius increases. The growth is stronger for high value of range of atomic radius.



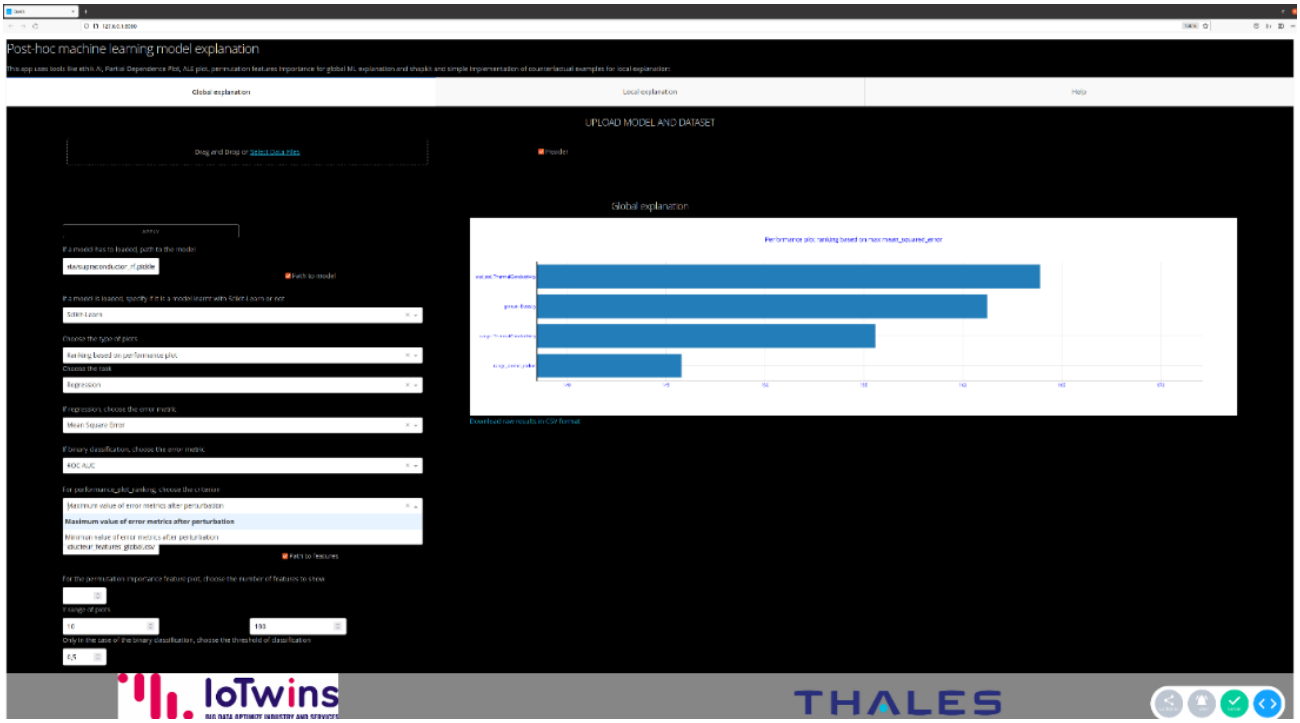
The user can use the ALE plot too. This plot describes how features influence the prediction of a machine learning model on average, and unlike the PDP, are unbiased. We provide the ALE plot for the range of atomic radius in the Figure below. The plot is similar to the PDP, with a change point around 200. However the ALE plot is smoother than the PDP. The user should prefer the ALE Plot to perform the model inspection.



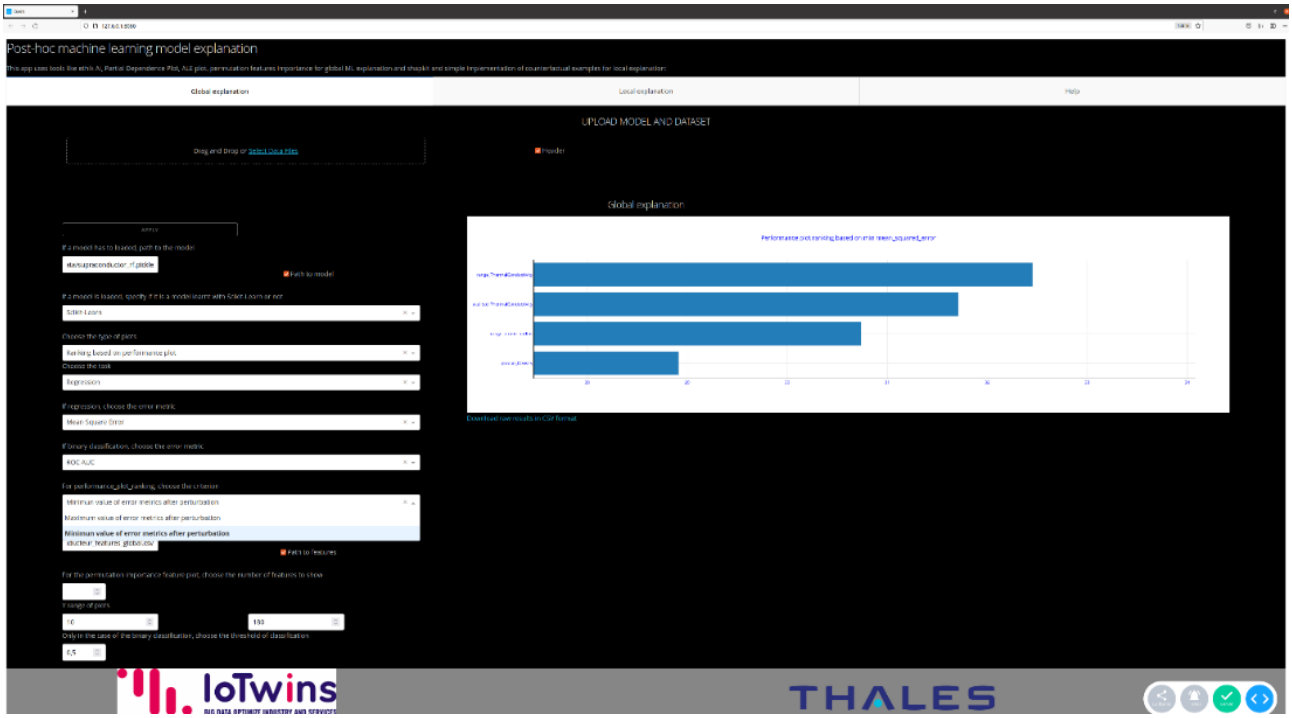
Based on the influence plot, the user can compute a ranking of features. It deduces a feature importance measure by computing, for a specific feature, the mean absolute difference between their influence curve and a horizontal curve equal to the original average prediction. We rank the range of the thermal conductivity, the weighted standard deviation of the thermal conductivity, the range of the atomic radius and the geometric mean of the density on the Figure below. According this criterion, among these four features, the most important is the weighted standard deviation of the thermal conductivity. For this example, to focus on the four features, we use a csv field to name the features and check the button Path to feature. We show the file with the features in the Figure below. It lists the four features, separated by a comma, without space, on one row.



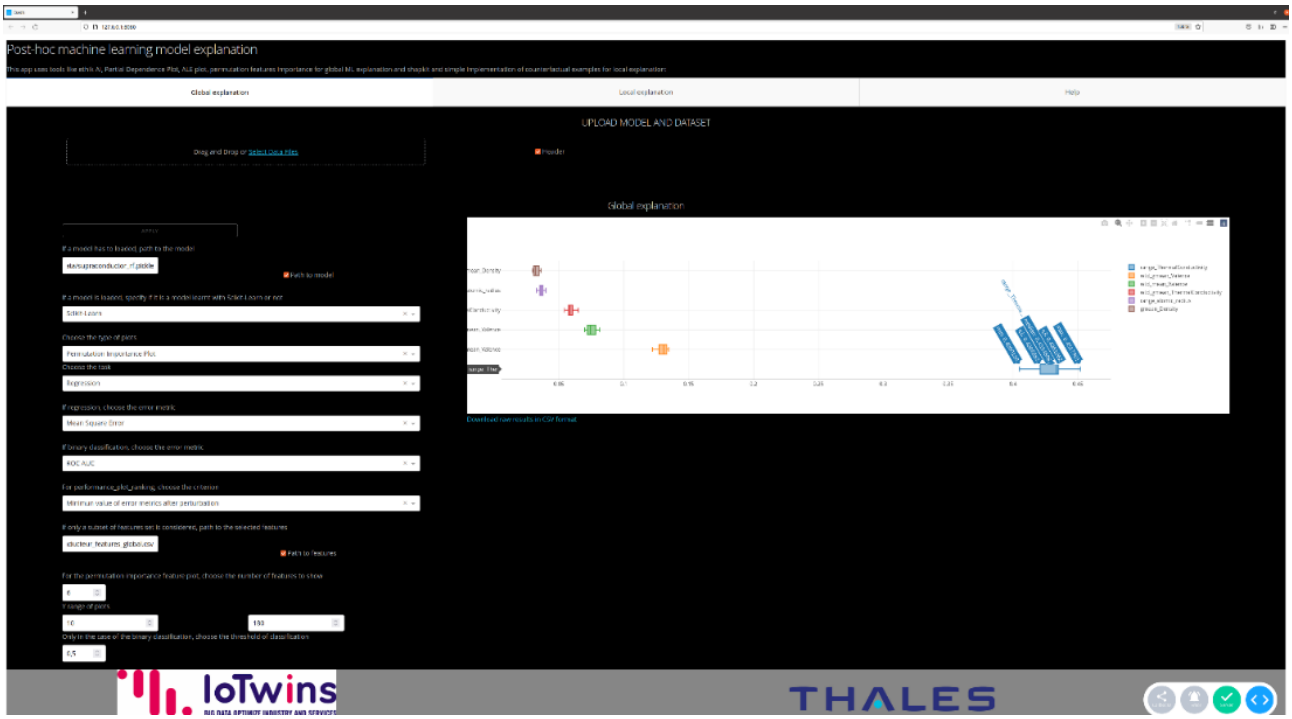
From the Performance plot, the user can deduce a ranking of the features. Both the minimum and maximum of MSE after features distribution stressing can be retained. If the maximum of MSE after stressing is chosen, a strong value of the criterion indicates that the feature is important. If the maximum of MSE after stressing is chosen, a weak value of the criterion indicates that the feature is important. In Figure below, we rank the four features according the maximum MSE after feature stressing. Again, according to this criterion, the weighted standard deviation of the thermal conductivity is the more important. However, the geometric density of atomic radius, that was ranked as the least important of the four features according the ranking based on the Influence plot, is ranked second by this criterion.



In the Figure below, we plot the ranking based on performance plot based on the minimum MSE after stressing. According this criteria, the most important feature is the geometric mean of the density.



We use the permutation features importance to rank all the features. We only show on Figure below the 6 most important. To compute the criterion, we compute several times the increase in the model's prediction error after permuting the features. The more a feature is important, the more this value should be high. According permutation feature importance, the features the most important is the range of thermal conductivity. It is worth noting that as the plot are interactive and generated by the Plotly library.

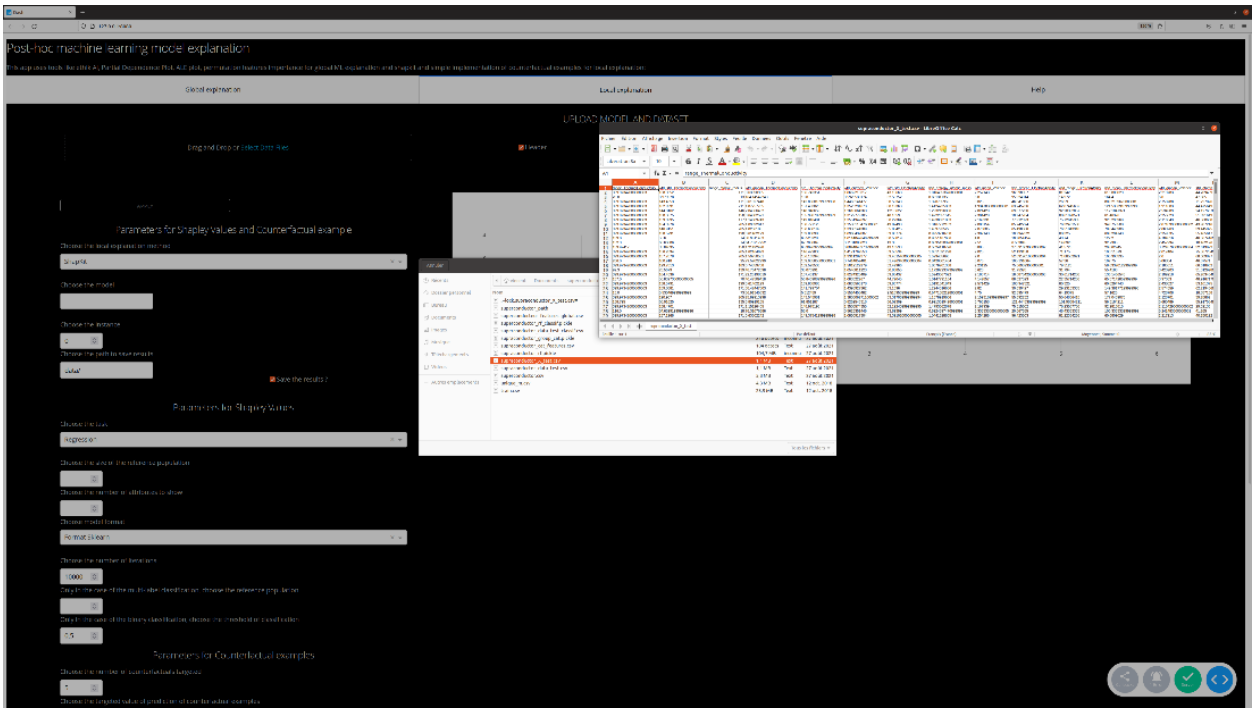


To summarize this panel, it allows to inspect the model in different ways:

- We can study the impact of one feature on the outputs of models. For instance, we see that when the range of atomic radius increases, the critical temperature predicted by the model increases and there is a gap around a value of range of atomic radius around 190, where the critical temperature predicted increases strongly
- We can study the impact of one feature on the models errors. For instance, when the range of atomic radius increases, then the model errors increase (partly due the increase of critical temperature predicted)
- We can rank the features according their importance. For instance, according permutation feature importance, the most important feature is the range of thermal conductivity

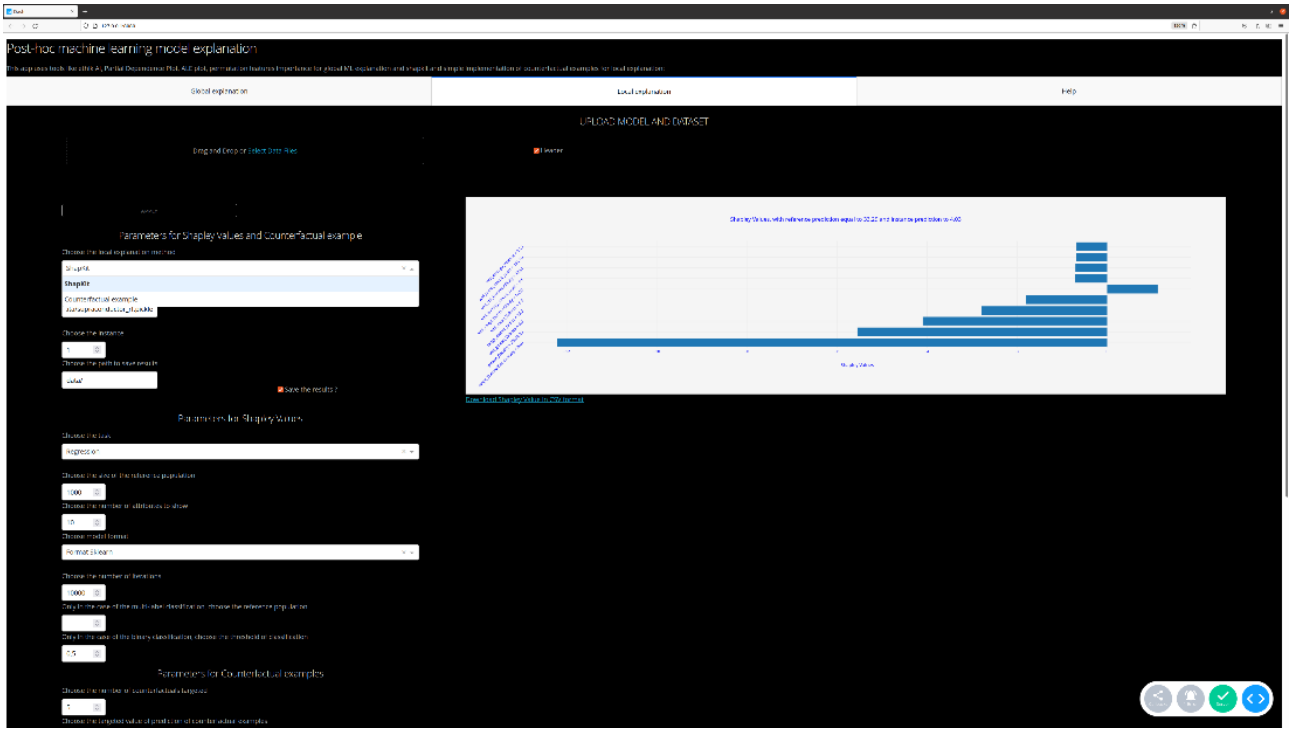
3.2.3.4 Illustration of local interpretation of machine learning model

The first stage in using the local interpretation panel consists in uploading the data. The data has to be in csv, xls, txt or tsv format. It contains the value of the features (the presence of the label is not mandatory).

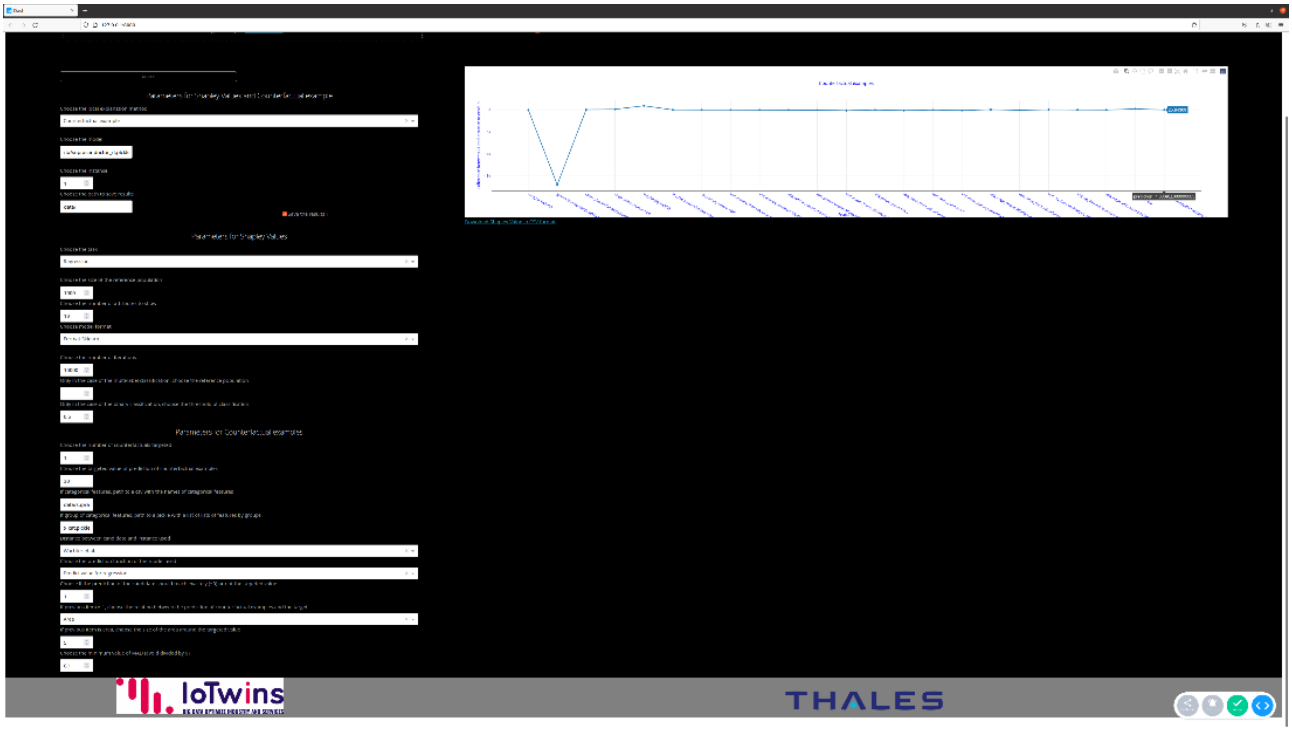


The critical temperature prediction for the second instance of the dataset is equal to 4,03. It is small compared to the average prediction of the critical temperature, which is around 33. We will use shapkit and the counterfactual examples to explain how the features impact this prediction.

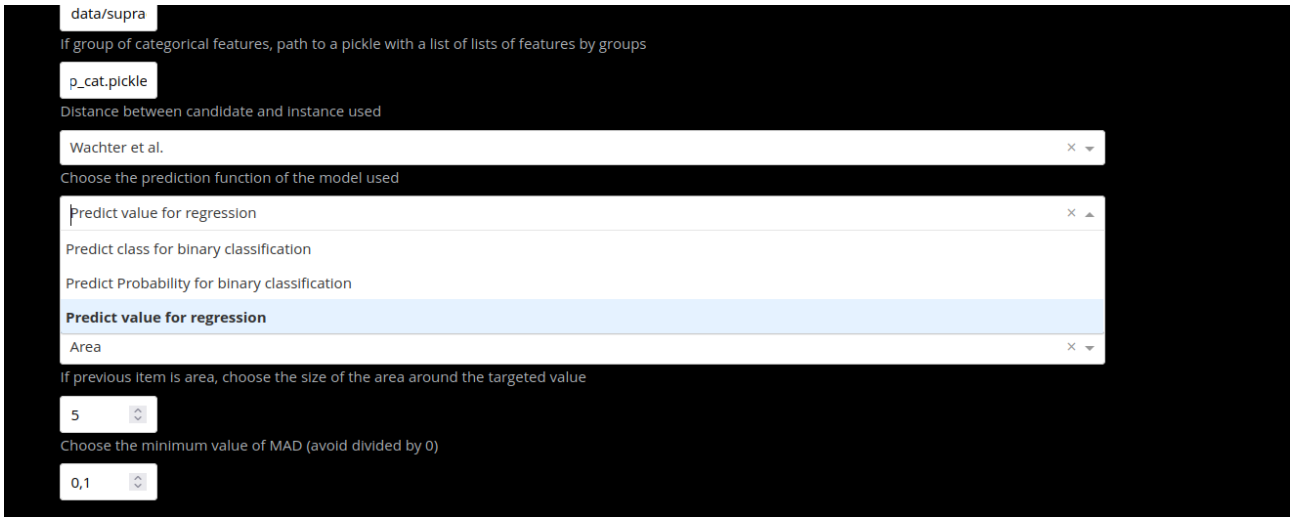
First, we use shapkit. The reference population size is 1000, that means that 1000 instances are randomly selected in the dataset. We represent only the features with the strongest absolute value of shapley value. If we sum all the shapley values with the average prediction of the reference population 33,29, we obtain the prediction made for the original instance. The features that influence the most to the decreasing of the prediction are the range of thermal conductivity, which is equal to 20 for the original instance, the geometric mean of density, which is strong and equal to around 20839 and the weighted geometric mean of the valence. One feature pushes up the predictive value for the original instance: the weighted range of the electron affinity.



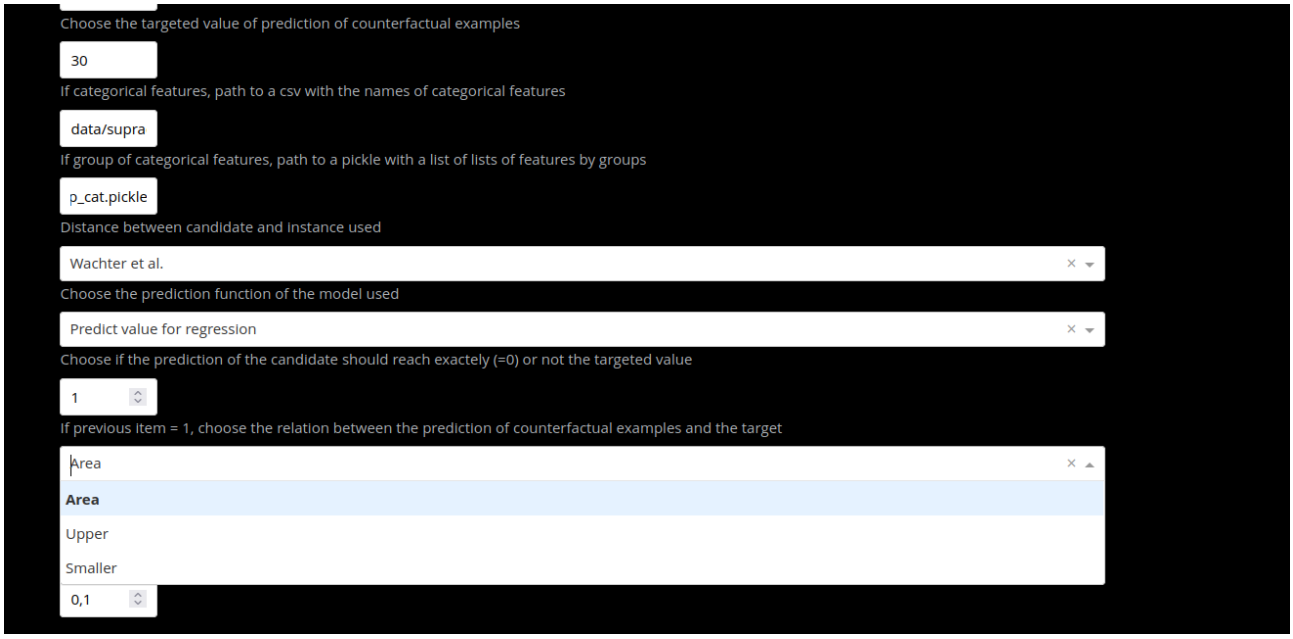
The second local method consists in finding counterfactual examples. For a regression task, we provide a target prediction. For our original instance, the target prediction is 30 plus or minus 5. For a classification task, the user can choose the opposite class or target a score. We provide in the Figure below the counterfactual example obtained for the original instance whose the critical temperature prediction is 4. All the features that change are represented on the plot. It represents all the differences between the feature values of the original instance and the counterfactual example for all features that has be changed. As the plot is interactive, the user can see the numeric value of the change by hovering over the plot. For instance, we see that the original instance geometry mean of density is significantly higher than the one of the counterfactual examples.



According the model task, we can choose between up to three strategies to characterize the counterfactual examples. In our example, we work on a regression model. We can adopt only one strategy: we search one instance whose prediction is in a range of values that we determine. For instance we target prediction around 30 for our original instance whose prediction is 4. For classification, we can choose between two strategies: we can search observation predict in a given class (basically in the opposite class) or we can search the counterfactual examples in the instances whose predict score is in a range of value that we determine. It corresponds to the Figure below.



As in the Figure below, we can choose that the target prediction is 30. We can choose if we want to reach this exact value or a neighboring of this value with the field “Choose if the prediction of the candidate…”. If the value of the field is 0, we target the exact value, else it is equal to 1, we target an interval near the target prediction. In our example, we target an interval around 30. The interval is between 25 and 35. We can choose three types of intervals: if we choose area, it creates an interval centered on the target prediction. If we choose Upper (resp. smaller), the target area will be the data greater than the target prediction (resp. smaller).



3.2.4 References

Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values.

Apley, & Zhu. (2016). Visualizing the effects of predictor variables in black box.

Bachoc, F., Gamboa, F., Halford, M., Loubes, J.-M., & Risser. (2020). Explaining Machine Learning Models using Entropic Variable Projection.

Fisher, A., Rudin, C., & and Dominici, F. (2019). All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously.

Friedman. (2001). Greedy function approximation: a gradient boosting machine.

Grah, S., & Thouvenot, V. (2020). A projected stochastic gradient algorithm for estimating Shapley value applied in attribute importance.

Lundberg, S. M., & Lee, S.-I. A. (2017). A unified approach to interpreting model predictions.

Shapley. (1953). A value for n-person games.

Štrumbelj, E., & Kononenko, I. (2010). An Efficient Explanation of Individual Classifications using Game Theory.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contribution.

Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR.

3.3 Description of ESI Introspector tool

3.3.1 Introduction

"The greatest value of a picture is when it forces us to notice what we never expected to see." John Tukey

Data visualization in data analysis is the key to understanding vast amounts of data. Data visualization helps the user by organizing data into an easier way to quickly understand the information and find relation between them.

The component described below, developed by ESI-Group, consists of a web application to visualize, analyze multi-dimensional datasets and identify the main data that could lead to a fault in the monitored system.

3.3.2 Technical description

As for **reducing the space dimensionality** for easier visualization, various clustering algorithms can be relevant. We propose here to represent high dimension data in a two-dimensional space using linear and non-linear techniques such as PCA (Principal Component Analysis), LLE (Locally Linear Embedding) and t-SNE (t-distributed Stochastic Neighbor Embedding) described below.

PCA decomposes the high-dimension space into successive orthogonal components that explain a maximum amount of the variance. It is based on a Singular Value Decomposition (SVD) to get the axes of a new space with maximum information. The new space is therefore composed with linear combinations from the initial

space axes for a given dimension equal to or lower than the initial space. The PCA method was developed by Karl Pearson (1901) and Harold Hotelling (1933) [5].

LLE looks for a lower dimensional projection of the data that preserves the distances between local neighbors. It can be interpreted as multiple local PCA that create a non-linear embedding. It was proposed by Sam T. Roweis and Laurence K. Saul in 2000.

t-SNE converts similarities of data points into probabilities. In the initial space, the metric is a Gaussian probability transformed into a Student's t-distribution in the target space. This allows t-SNE to spotlight local structures. The non-linear Kullback-Leibler (KL) distance is used to joint probabilities and minimized by a gradient descent. The t-SNE algorithm can be better fitted for reducing higher dimensionalities with data lying in multiple manifolds or clusters but is computationally more expensive. It is based on Stochastic Neighbor Embedding originally developed by Sam Roweis and Geoffrey Hinton, where Laurens van der Maaten proposed the t-distributed variant.

Machine Learning algorithms allow us to identify the most important data. Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It shows better results in classification tasks. The first algorithm for random decision forests was created in 1995 by Tin Kam Ho. An extension of the algorithm was developed by Leo Breiman [9] and Adele Cutler who registered "Random Forests" as a trademark in 2006.

3.3.3 Example of component use

3.3.3.1 General description of the component

3.3.3.1.1 Dependencies

The tool is dedicated to visualizing high-dimensional data in a two-dimensional space using PCA (*Principal Component Analysis)*, LLE (Locally Linear Embedding), or t-SNE (t-distributed Stochastic Neighbor Embedding) algorithms. It is also possible to sort by importance these data with a “random forest” algorithm and visualize them in a parallel coordinates plot.

These data are based on tabular files like .csv ascii files or .parquet binary files.

The tool is a web application embedded in a Docker container that can be deployed on any server.

The tool is implemented in Python3. It uses the following modules:

- Dash - framework for developing web applications for data visualization.
- plotly - open-source library that can be used for visualizing and understanding data simply and easily.
- pandas - library that allows easy manipulation of data for various analyses.
- numpy - library that includes functions for manipulating multidimensional matrices or arrays.
- scikit-learn - machine learning framework. It includes functions for estimating based on random forests, logistic regressions, classification, and support vector machines algorithms

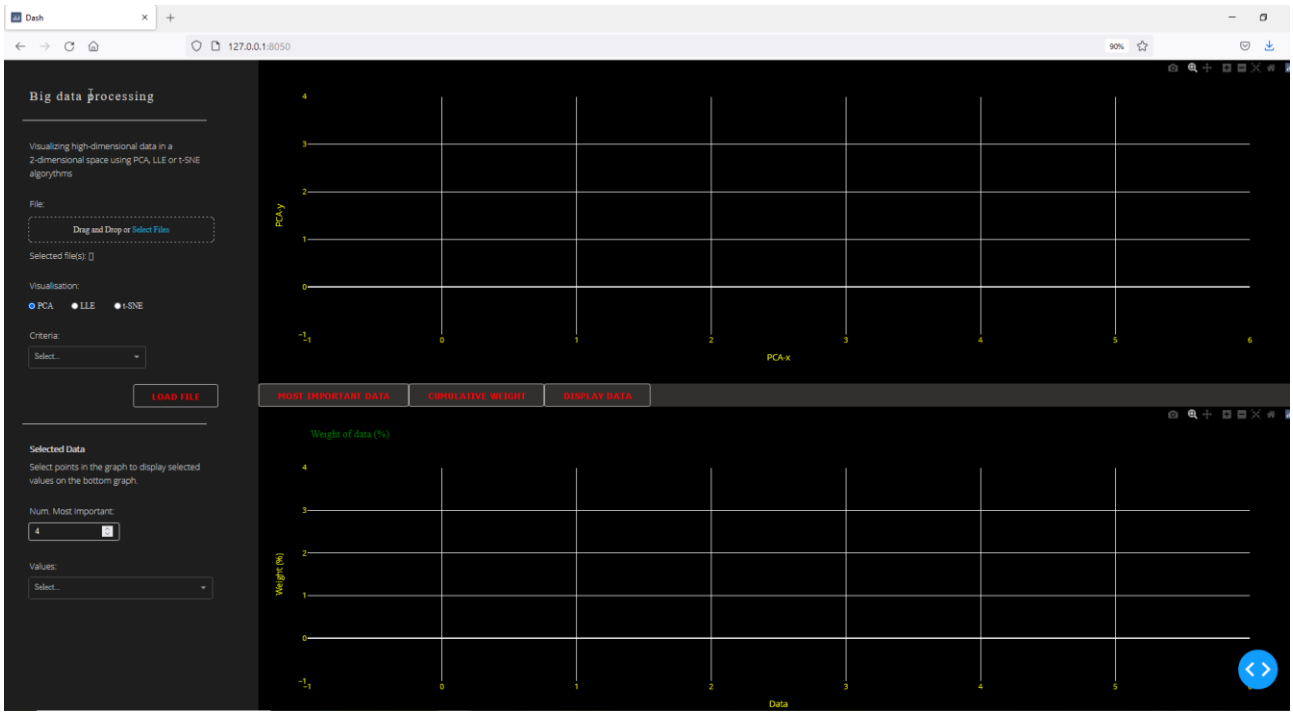
[5] https://en.wikipedia.org/wiki/Principal_component_analysis#History

3.3.3.1.2 Visual overview

The tool is divided in two main panels:

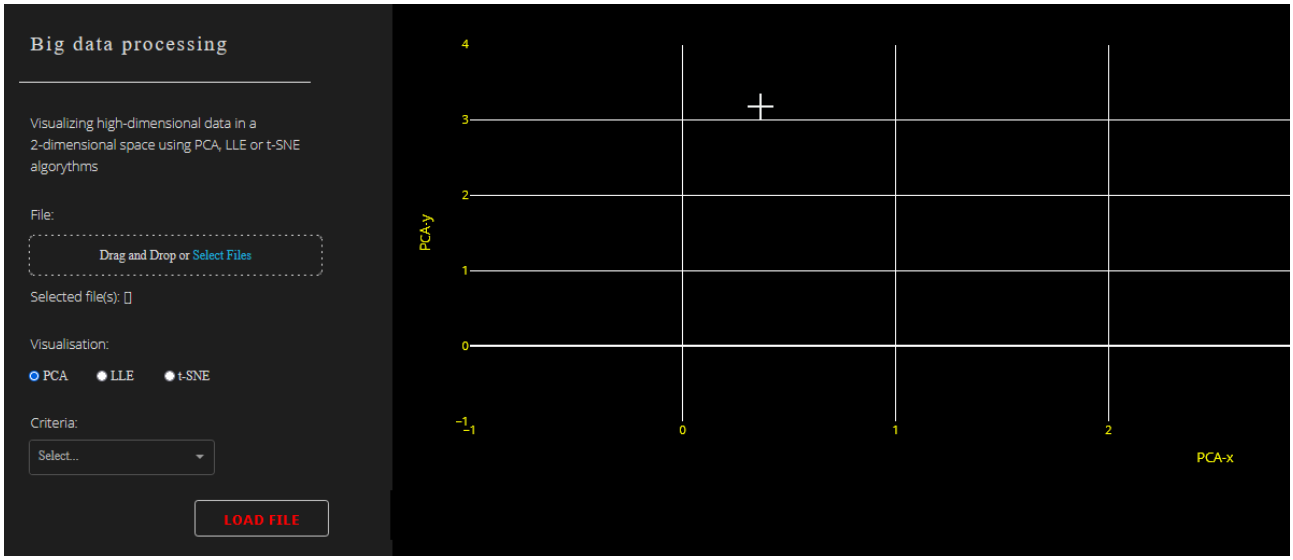
1. Two-dimensional reduction display
2. Most important data and data values display

The figure below shows an overview of the full tool.



3.3.3.1.2.1 Two-dimensional reduction

The first panel is dedicated to the two-dimensional reduction using different algorithms: PCA (*Principal Component Analysis)*, LLE (Locally Linear Embedding), t-SNE (t-distributed Stochastic Neighbor Embedding). The figure below shows the given panel.



At the top left level of the component, there is the button “Drag and Drop or Select Files” to upload one (or several) data files. Once the file is read, the user can choose the type of algorithm and the criteria in a

dropdown list that will be used to compute the dimensional reduction. The data can be a .csv file, a .parquet file or a .txt file. The first column contains the timestamp, all the others are the data used by the algorithm. We give in the figure below an example of data that can be provided to the component.

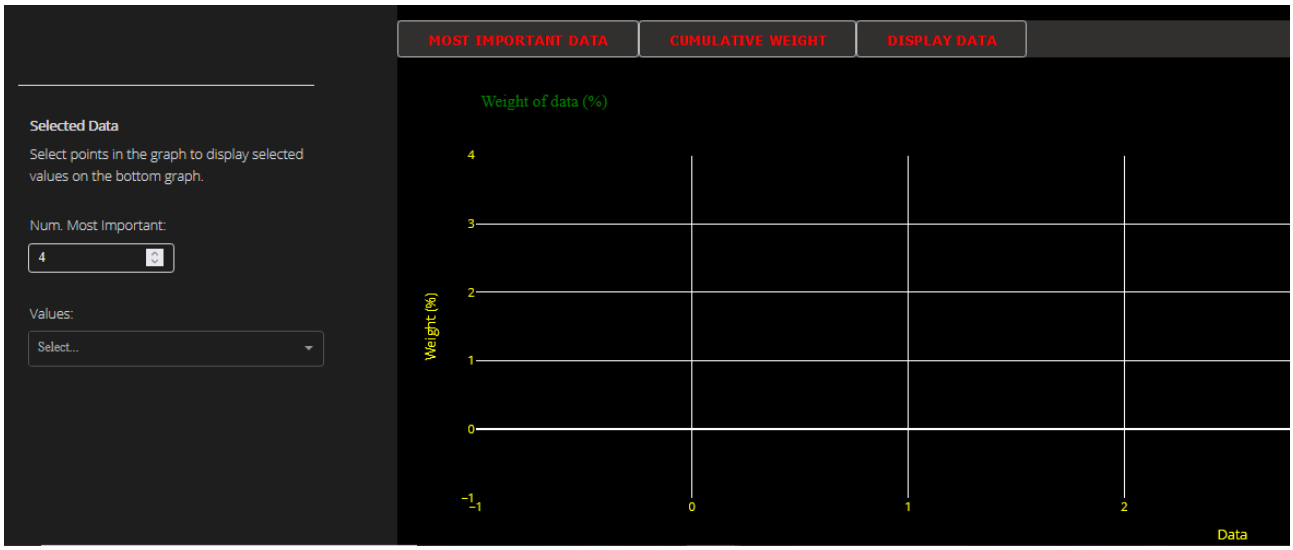
Supercomputer Monitored Data

timestamp	avg:ambient	var:ambient	max:ambient	min:ambient	avg:dimm0_temp	var:dimm0_temp	max:dimm0_temp	min:dimm0_temp	avg:dimm10_temp	var:dimm10_temp	max:dimm10_temp	min:dimm10_temp	avg:dimm11_temp	var:dimm11_temp	max:dimm11_t
2020-06-01T00:00:00	25.574999999999996	0.005000000000000065	25.6	25.4	32.25	0.21428571428571494	33	32	34	0	34	34	33	0	33
2020-06-01T00:15:00	25.560000000000006	0.006857142857143055	25.6	25.4	32.333333333333336	0.23809523809523672	33	32	34	0	34	34	33	0	33
2020-06-01T00:30:00	25.600000000000005	0	25.6	25.6	32.53333333333333	0.26666666666666694	33	32	34	0	34	34	33	0	33
2020-06-01T00:45:00	25.6	0	25.6	25.6	32.07142857142857	0.07142857142857144	33	32	34	0	34	34	33	0	33
2020-06-01T01:00:00	25.571428571428566	0.005274725274725395	25.6	25.4	32.07142857142857	0.07142857142857144	33	32	34	0	34	34	33	0	33
2020-06-01T01:15:00	25.6	0	25.6	25.6	32	0	32	32	34	0	34	34	33	0	33
2020-06-01T01:30:00	25.600000000000005	0	25.6	25.6	32.2	0.17142857142857135	33	32	34	0	34	34	33	0	33
2020-06-01T01:45:00	25.519999999999992	0.010285714285714603	25.6	25.4	32	0	32	32	34	0	34	34	33	0	33
2020-06-01T02:00:00	25.199999999999992	0	25.2	25.2	32	0	32	32	33.2	0.17142857142857146	34	33	33	0	33
2020-06-01T02:15:00	25.085714285714282	0.010549450549450508	25.2	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T02:30:00	25	0	25	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T02:45:00	25	0	25	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T03:00:00	25	0	25	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T03:15:00	25	0	25	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T03:30:00	25.185714285714428	0.0028571428571428628	25.2	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T03:45:00	25.199999999999992	0	25.2	25.2	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T04:00:00	25.199999999999992	0	25.2	25.2	32	0	32	32	33.06666666666667	0.06666666666666643	34	33	33	0	33
2020-06-01T04:15:00	24.8923076923077	0.017435897435897394	25.2	24.8	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T04:30:00	25.066666666666663	0.009523809523809352	25.2	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T04:45:00	24.920000000000005	0.010285714285714283	25	24.8	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T05:00:00	25	0	25	25	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T05:15:00	24.973333333333336	0.004952380952380877	25	24.8	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T05:30:00	24.92	0.010285714285714202	25	24.8	32	0	32	32	33	0	33	33	33	0	33
2020-06-01T05:45:00	24.866666666666667	0.009523809523809499	25	24.8	32	0	32	32	33	0	33	33	33	0	33

3.3.3.1.2.2 Most important data and data values

The second panel is dedicated to data visualization. The user has to select the points he wants to see in the data from the previous plot. All the points can be selected or just a part of them. In our use case, by “most important” data we mean the data that could be identified to explain the success or the failure of the computer in a datacenter. It could also be the quality of workpieces in case of machining. On the left, the user can choose how many “most important” data he wants to compute. According to the amount of data in the file, the default value will be either 4 or 10. The user can also select the data he wants to plot the values. By default, it will be the “most important”, but any data can be plotted by choosing the name in the dropdown list. On the right, first tab is the “most important” data computation using a “random forest” method. Second tab is the same result using a cumulative curve plot. The last tab is the display of the values of “most important” or selected data.

The figure below shows the given panel.



3.3.3.2 Use case description

The use case objective consists in identifying in a datacenter the critical data and the level of each of these data that leads a node of a rack of machines to fail. Each machine is equipped with 460 sensors that report data such as:

- Total free disk space
- CPU Speed
- Core Temperature
- Total node power consumption
- Total number of running processes
- …

The record starts on 2020/06/01 and stops on 2021-04-30. Each 15 minutes the information given by the sensors on a computer is stored in a parquet file.

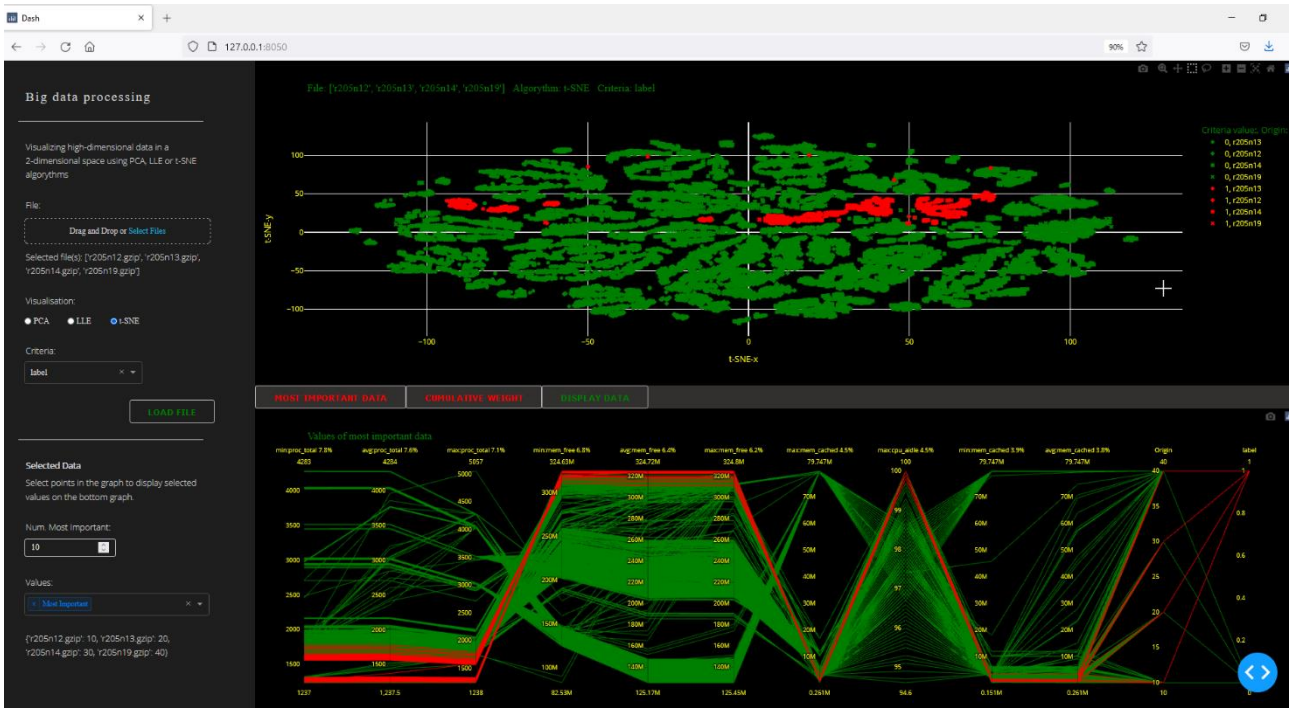
The data are given in a set of parquet files. The naming convention is this one: r205n12.parquet where “r205” is the id of the rack, “n12” is the id of the node.

3.3.3.3 An overview of the tool

The figure below shows a global view of the tool. The top view shows a two-dimensional reduction of the data. They are clustered according to the criteria selected by the user. Here is the result for the “Label” criteria. In red the timestamp has failed, in green it succeeded.

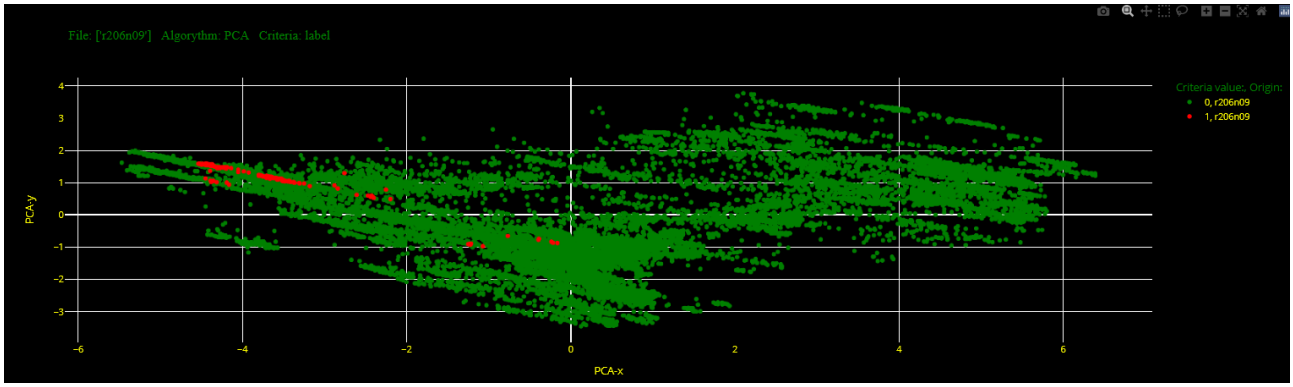
In the bottom view, the user can see the detail of data sorted by importance for each set of selected timestamps. Here we can see that “failed” timestamps in red have more or less the same data compared to the rest in green.

So, the sensors reporting the “most important” data should be examined further. Red flags can be put in place to monitor these data and avoid the failure

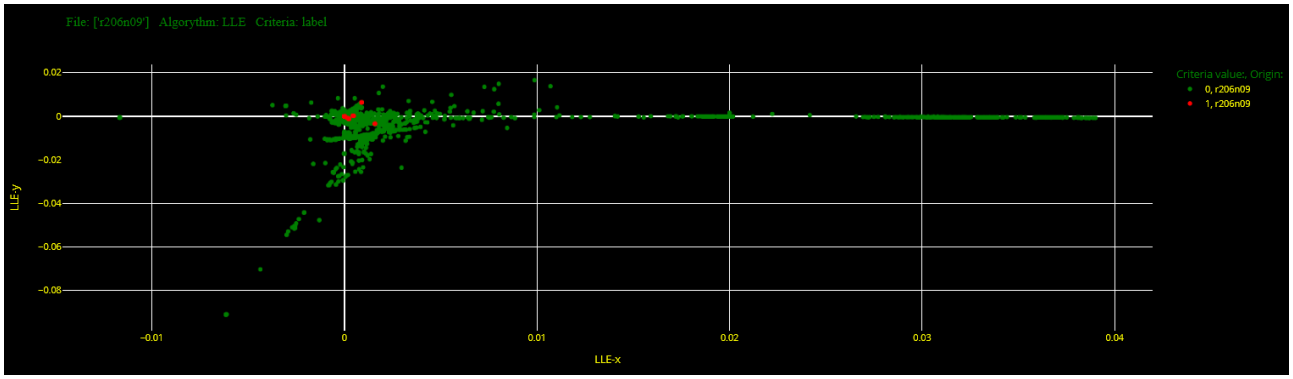


3.3.3.4 Illustration of the two-dimensional reduction

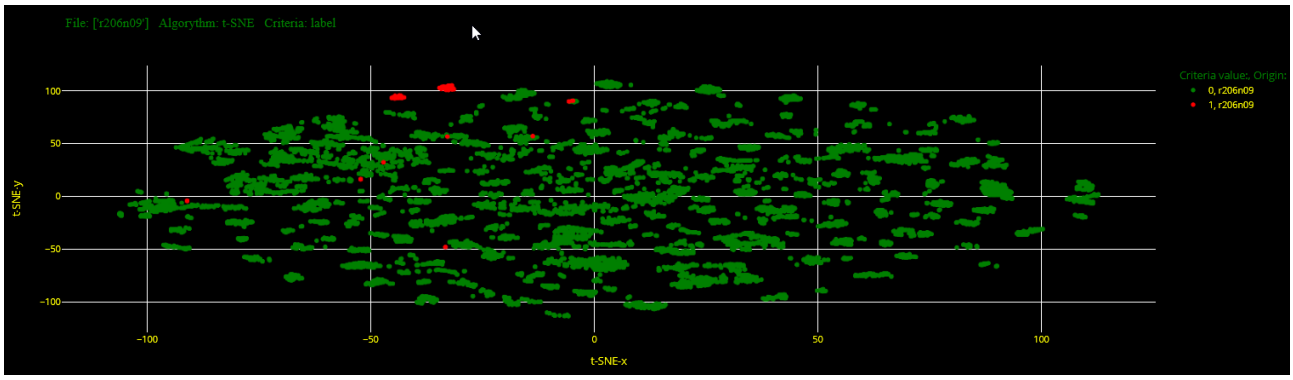
In the figure below we can see the results using the PCA method for one node. Each green point of the cloud is a timestamp which has succeeded. Red points correspond to timestamps which failed.



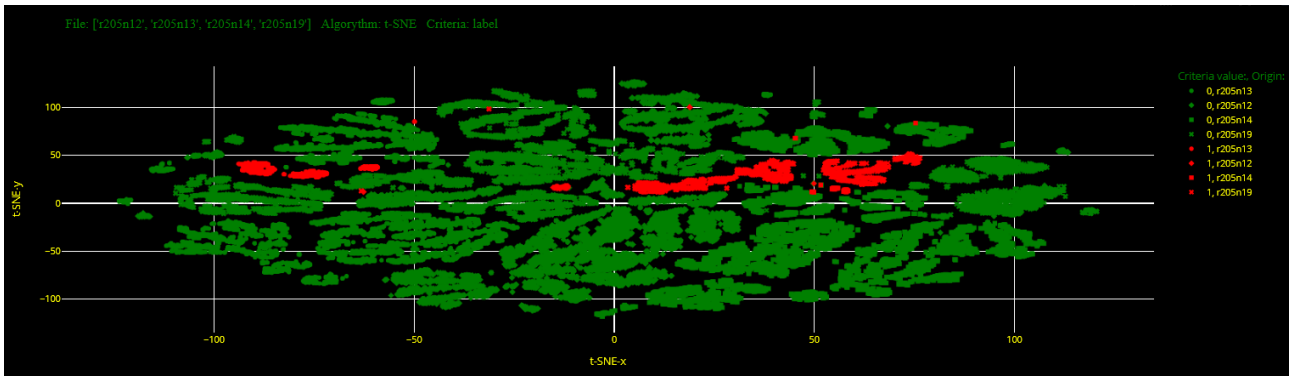
Same input with the LLE method:



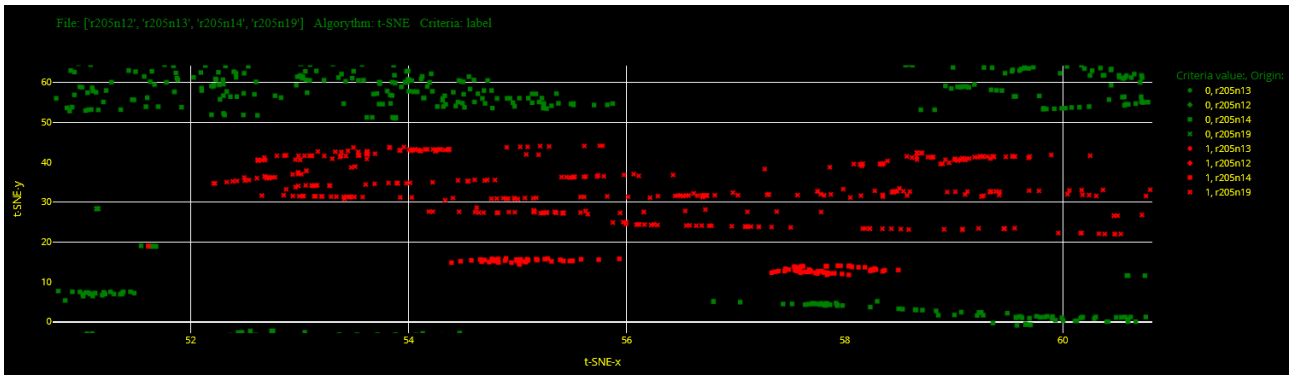
Same input with t-SNE method:



The figure below is the same computation as the t-SNE method but applied to a rack (set of five nodes).



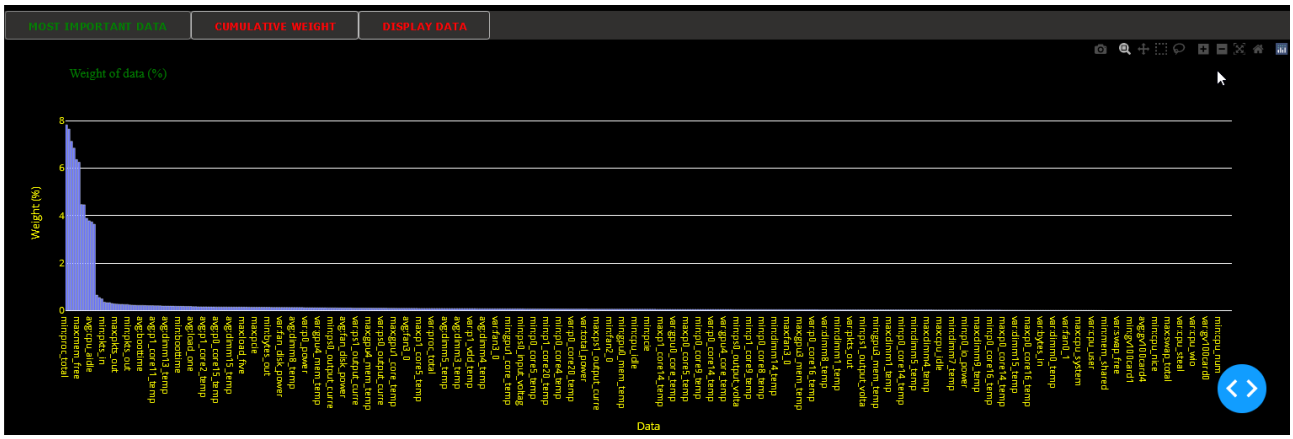
A zoom on a part of the plot:



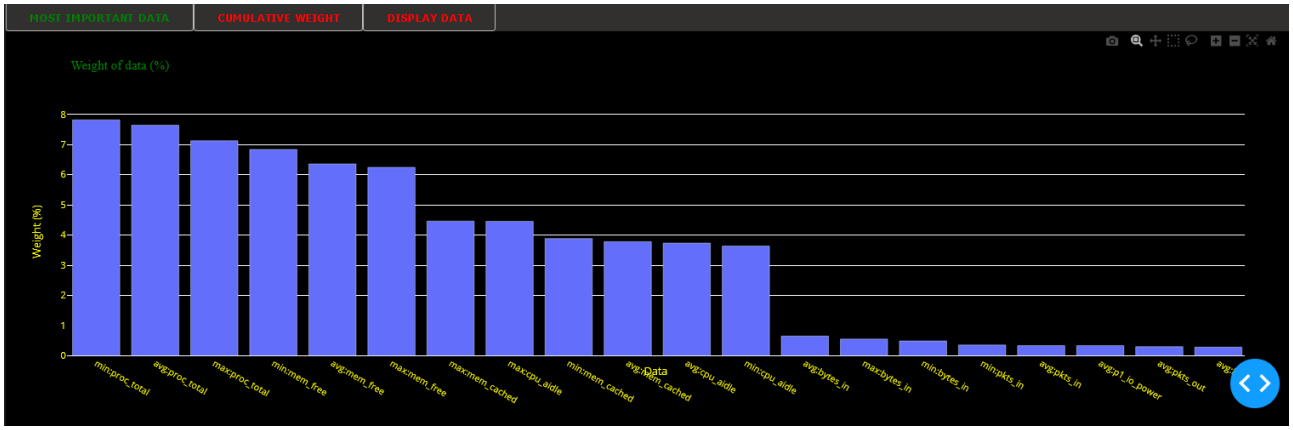
From the figures above, we can clearly see some red clusters. This indicates that there is a link between the timestamps that failed but we cannot yet draw any clear conclusions. Further study of the “most important” data and the values of these data in the second view could lead to more meaningful conclusions.

3.3.3.5 Illustration of the most important data identification

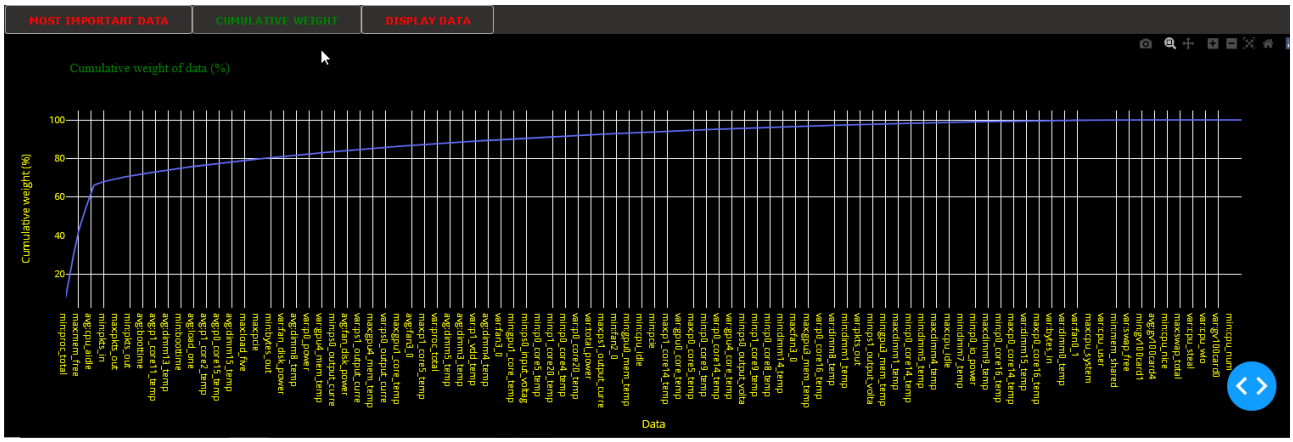
The figure below shows the result of the computation of the classification of the “most important” data. The relative weight of each data is given as a percentage. We can see in this example that a small group of data assumes most of the weight, in this case suggesting that among the 460 data reported by the sensors, only 12 are really important.



Same input with a zoom on the main data.

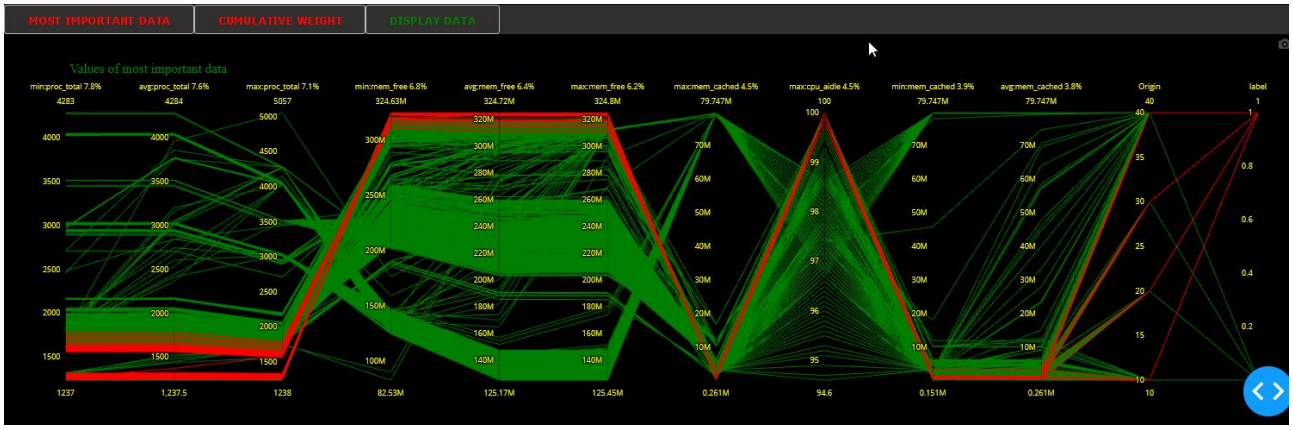


The result above can also be displayed with a cumulative curve. We can see that the first twelve data represent 80% of the weight of the 460 parameters.

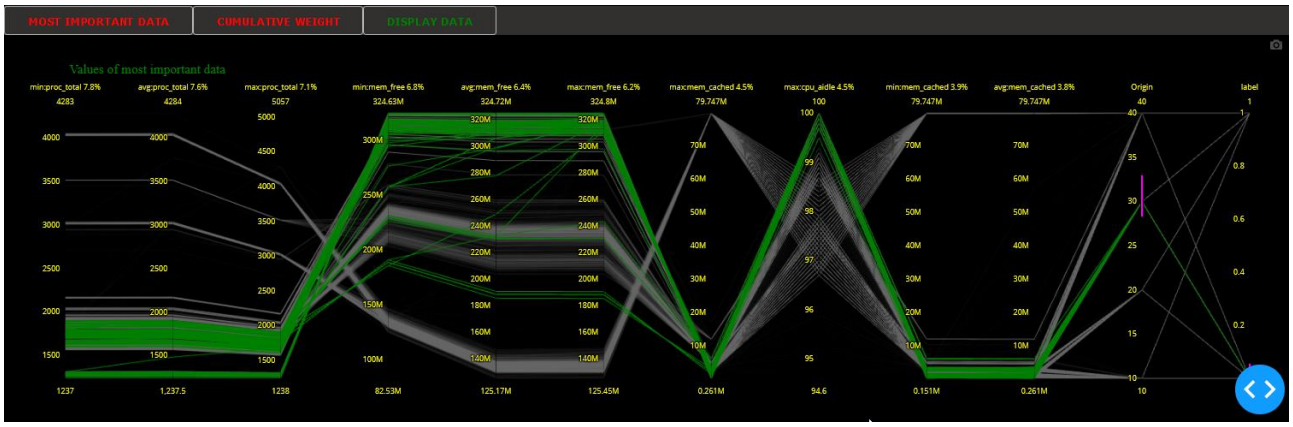


3.3.3.6 Illustration of the plot of the values of data

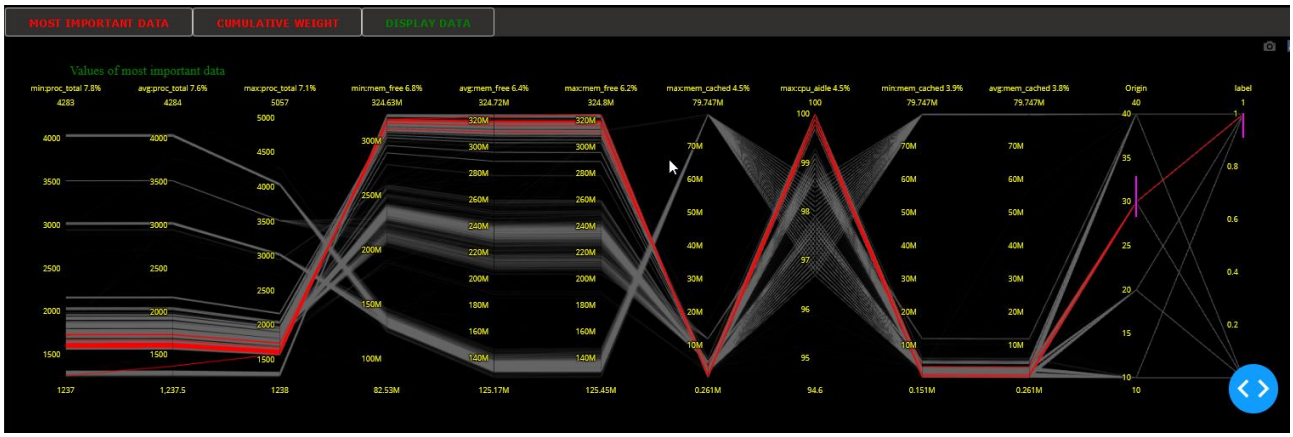
It is possible to visualize the values of the data. The figure below is the plot in a parallel coordinate's representation of the first 10 most important data computed previously for a full rack. Each (green or red) line is a timestamp of one node.



It is possible to separate the results for the different nodes. Figure below shows the "success" results for all the timestamps of one node.



Same node in the rack but “failed” timestamps:



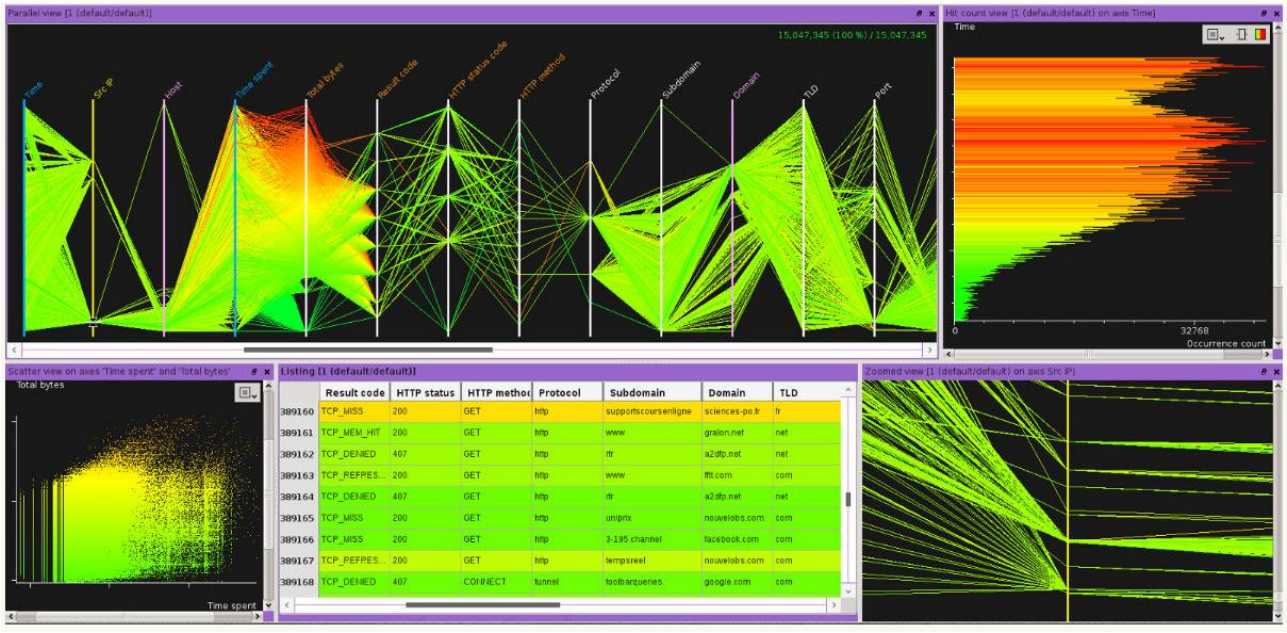
3.4 Description of Inendi Inspector

3.4.1 Introduction

INENDI Inspector allows the user to perform interactive explorations of very large amounts of data. Thanks to its innovative set of visualizations, the user is freed from the classical burden of query-based interactions and can then use all their logical and deductive power to discover valuable insights from the data. It is particularly suitable for analyses involving data of very high dimensions (number of columns), where visualisations such as the Global Parallel Coordinates view allow the user to identify the most important features and trends in the data. Furthermore, the ability to correlate several datasets with different structures allows the user to intuitively understand the causal chains over the course of the whole life cycle of the studied product.

It can be used for different purposes:

- Initial understanding and filtering of large and complex datasets
- Isolating weak signals very efficiently
- Controlling and improving Machine Learning algorithms



3.4.2 Technical description

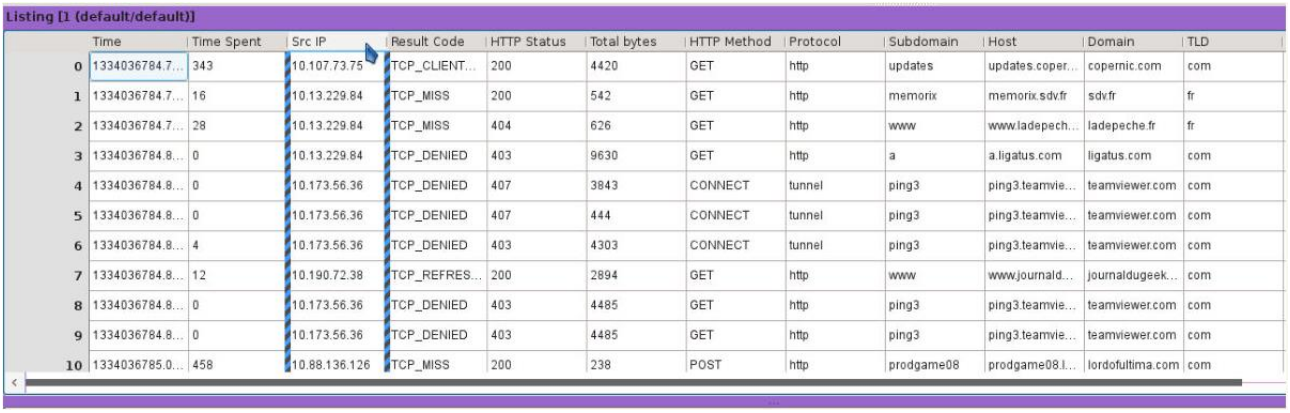
3.4.2.1 General description of the component

Inspector is a visualization tool developed in C++, with dependencies on Nvidia CUDA, Qt5 andOpenGL for the UI components. It was recently released as open-source software[6]. Inspector is developed as a linux application, while Flatpak[7] is used as a packaging mechanism to ensure a consistent (and sandboxed environment) for the application. The application can be executed under WSL2 on Windows. To enable its deployment on the IoTwins platform, Inspector was packaged as a Docker image, with novnc, x11vnc, xvfb embedded to ensure that the visualization could be embedded e.g. within client websites.

3.4.2.2 Listing

The purpose of the Listing View is to present the data in its textual representation. Therefore, it is a tabular representation of the original data with one line of the table dedicated to one Event of the dataset, and one Column of the table dedicated to one Field of the dataset. Each value of a Field in a given Event has a textual representation (i.e. as a string) that is shown as a listing in a given cell of a table.

The Listing View has the following aspect:



	Time	Time Spent	Src IP	Result Code	HTTP Status	Total bytes	HTTP Method	Protocol	Subdomain	Host	Domain	TLD
0	1334036784.7...	343	10.107.73.75	TCP_CLIENT...	200	4420	GET	http	updates	updates.coper...	copernic.com	com
1	1334036784.7...	16	10.13.229.84	TCP_MISS	200	542	GET	http	memorix	memorix.sdv.fr	sdv.fr	fr
2	1334036784.7...	28	10.13.229.84	TCP_MISS	404	626	GET	http	www	www.ladepech...	ladepeche.fr	fr
3	1334036784.8...	0	10.13.229.84	TCP_DENIED	403	9630	GET	http	a	a.ligatus.com	ligatus.com	com
4	1334036784.8...	0	10.173.56.36	TCP_DENIED	407	3843	CONNECT	tunnel	ping3	ping3.teamvie...	teamviewer.com	com
5	1334036784.8...	0	10.173.56.36	TCP_DENIED	407	444	CONNECT	tunnel	ping3	ping3.teamvie...	teamviewer.com	com
6	1334036784.8...	4	10.173.56.36	TCP_DENIED	403	4303	CONNECT	tunnel	ping3	ping3.teamvie...	teamviewer.com	com
7	1334036784.8...	12	10.190.72.38	TCP_REFRES...	200	2894	GET	http	www	www.journald...	journaldugeek...	com
8	1334036784.8...	0	10.173.56.36	TCP_DENIED	403	4485	GET	http	ping3	ping3.teamvie...	teamviewer.com	com
9	1334036784.8...	0	10.173.56.36	TCP_DENIED	403	4485	GET	http	ping3	ping3.teamvie...	teamviewer.com	com
10	1334036785.0...	458	10.88.136.126	TCP_MISS	200	238	POST	http	prodgame08	prodgame08.l...	lordofultima.com	com

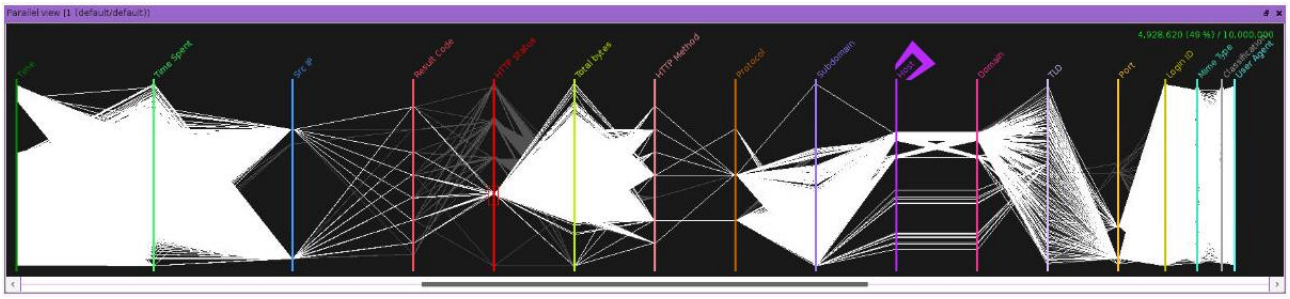
3.4.2.3 Global Parallel Coordinates

This is usually the main and central Graphical View of an investigation done with INENDI Inspector. It represents large, multidimensional datasets in a visually intuitive way that allow the user to identify global trends and home in one area of interest for detailed exploration.

The Global Parallel Coordinates View represents the *n* columns in the Listing View as *n* vertical, equally spaced lines/axes. A scale for each axis is automatically determined based on the values contained in the global dataset. For numerical quantities, minima and maxima are determined (minima at the bottom of the vertical axes, maxima at the top), while for discrete data, e.g., strings, the different values are equally spaced along the axes. Each row in the Listing View is then represented as a polyline, connecting points on the individual axes. The plot below shows the Global Parallel Coordinates view of the data seen in the Listing View above.

[6] https://www.esi-group.com/news/esi-group-in-collaboration-with-ensam-deploys-in-open-source-its-inspector-software

[7] https://flatpak.org/

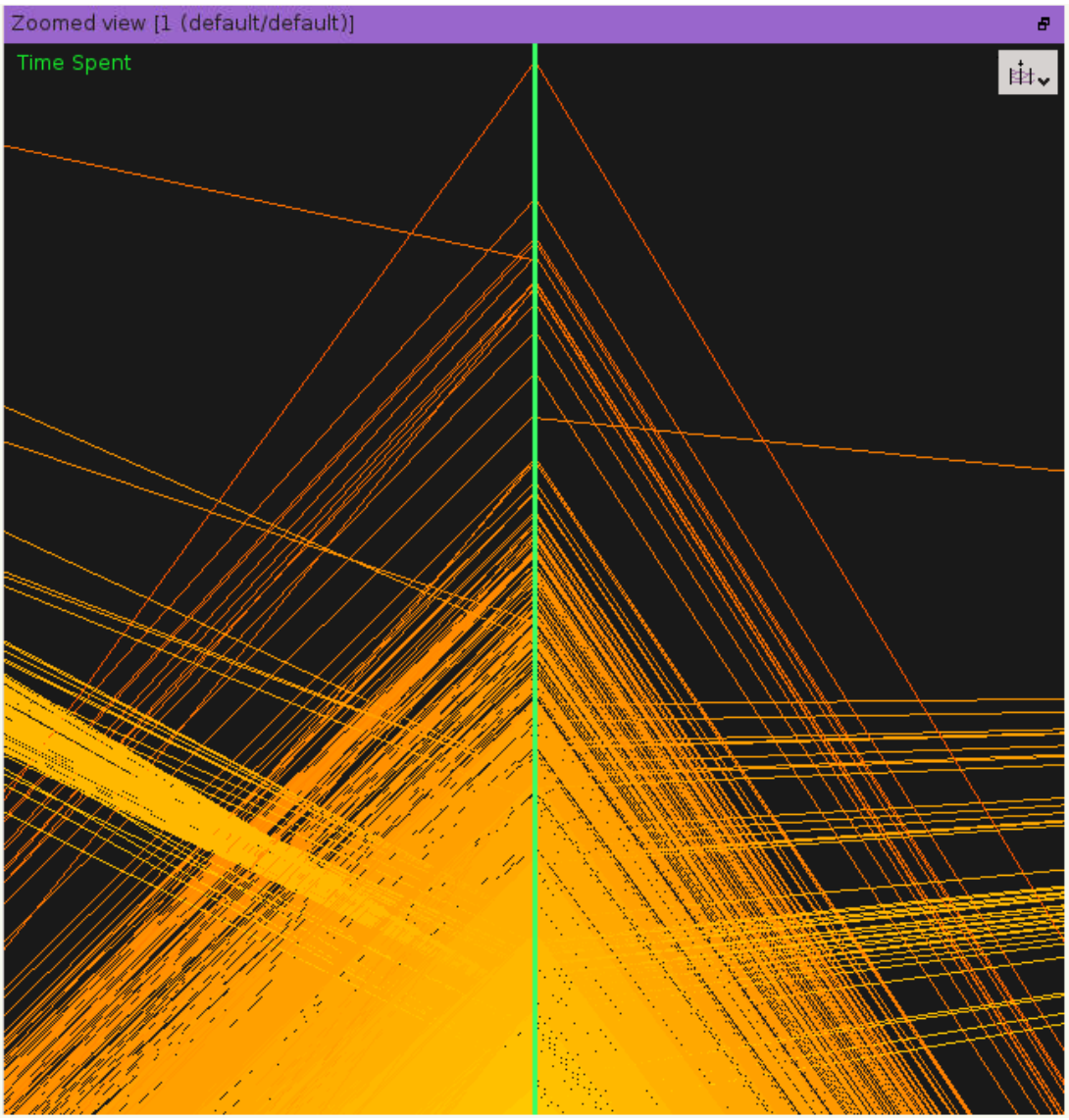


3.4.2.4 Zoomed Parallel Coordinates

The attentive user has probably noticed, from the previous section, that the Global Parallel Coordinates View offers no possibility to zoom vertically. The main reason to this limitation lies in the fact that the Global Parallel Coordinates View is aimed at providing a global representation: zooming on a detail of this View will remove from the user's eyes the panorama provided by the Global Parallel Coordinates View.

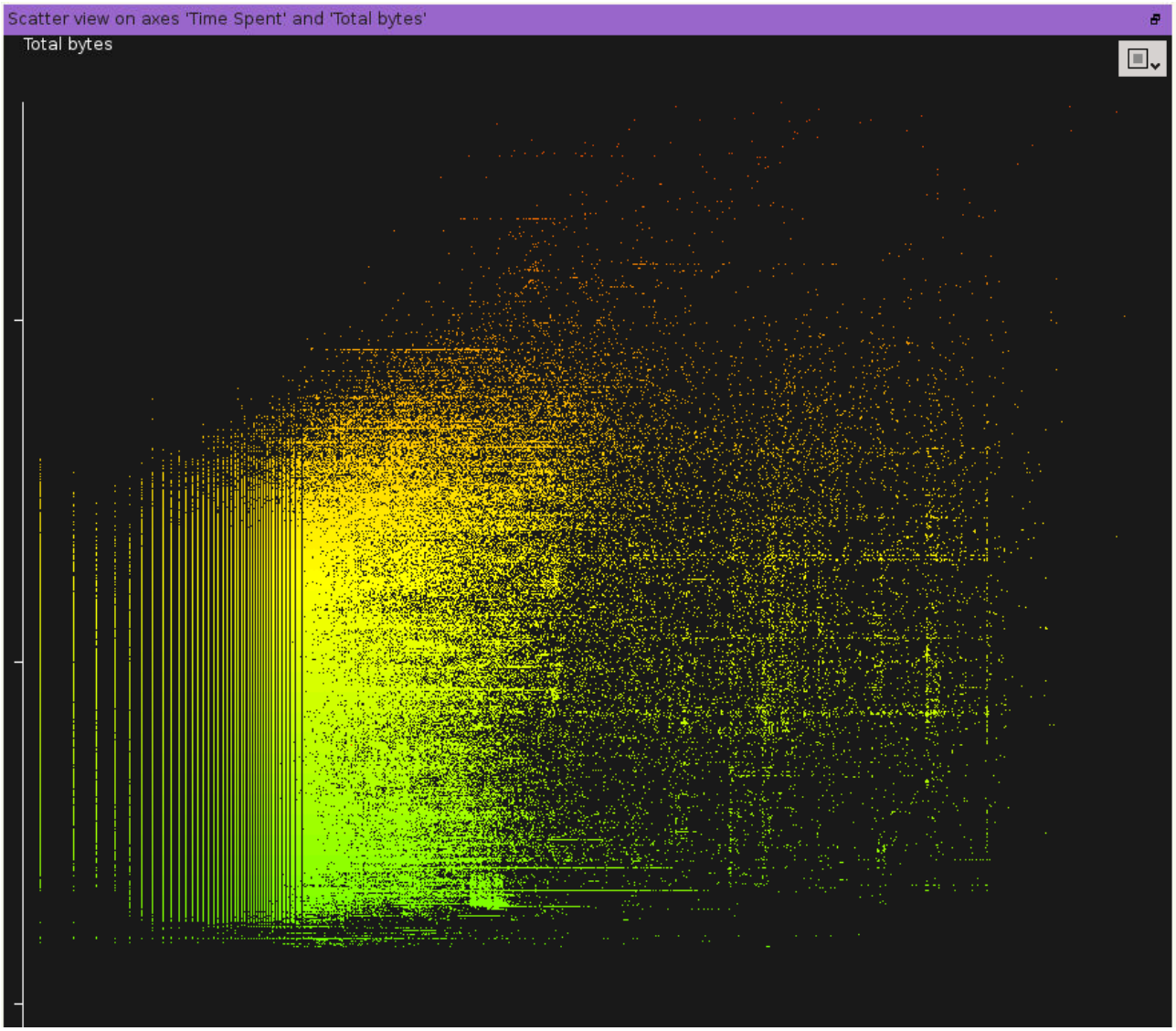
However, as far as it is very important to get an idea of the exact positions of values on a given Axis, INENDI Inspector supports the Zoomed Parallel Coordinates View. This View will provide the user the possibility to easily zoom (on or out) of an Axis.

A typical Zoomed Parallel Coordinates View looks like this:



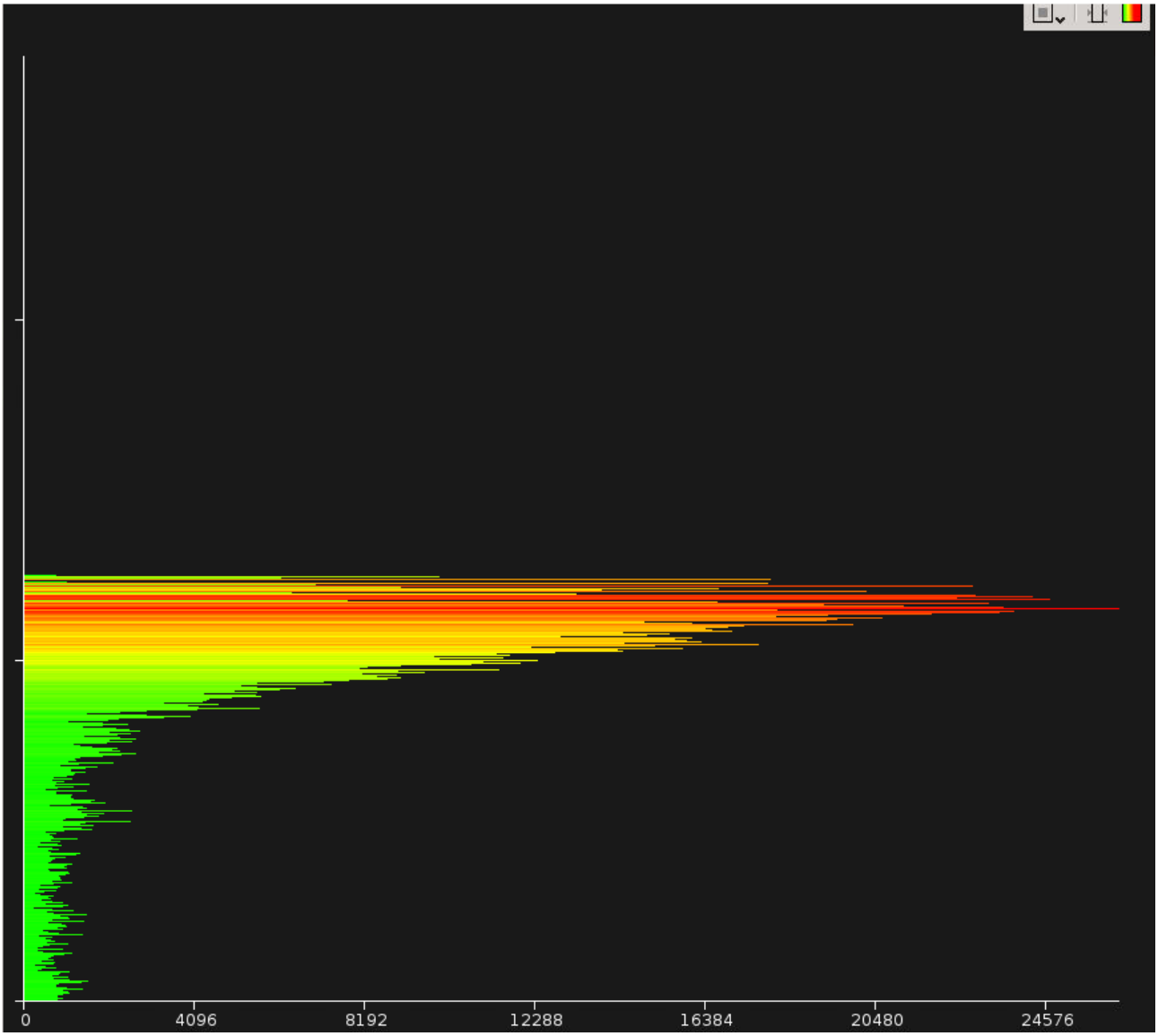
3.4.2.5 Scatter Plot

This graphic View provides an alternative zone representation using cartesian coordinates instead of parallel coordinates.



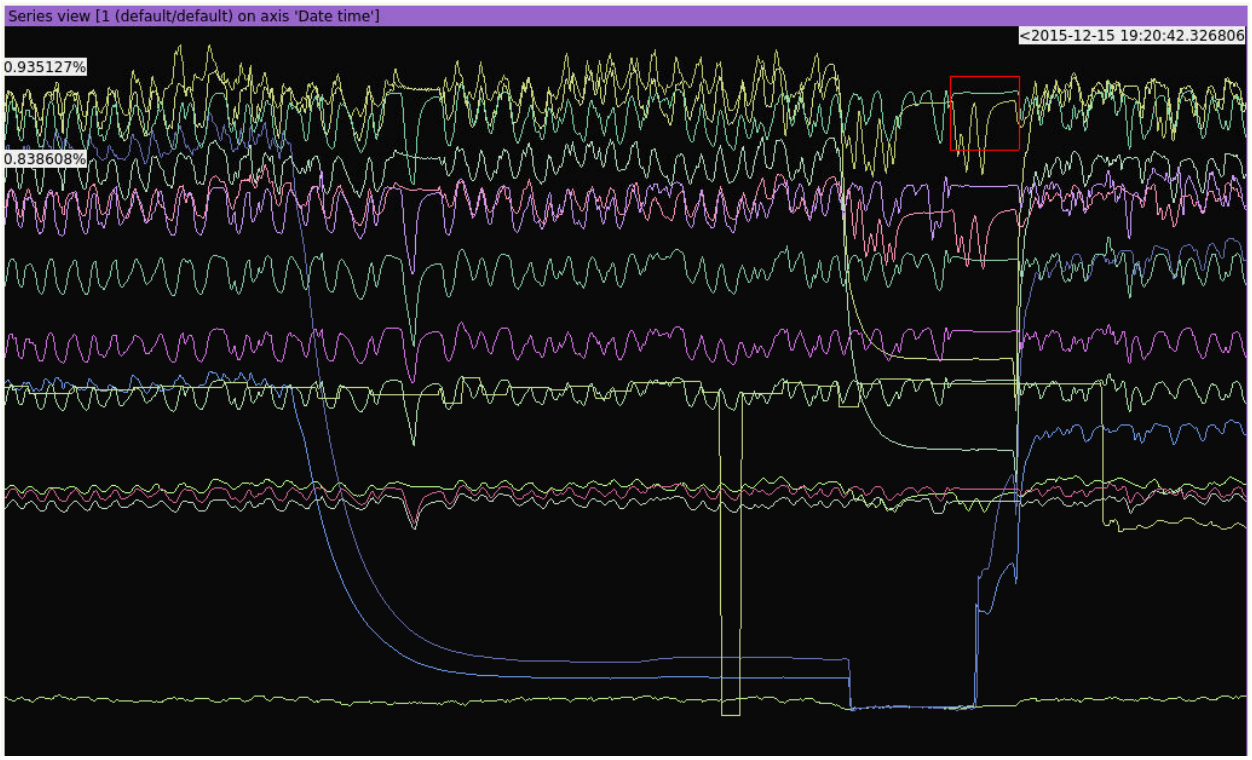
3.4.2.6 Hit Count

This View allows the user to visualize the density of events along a specific axis.



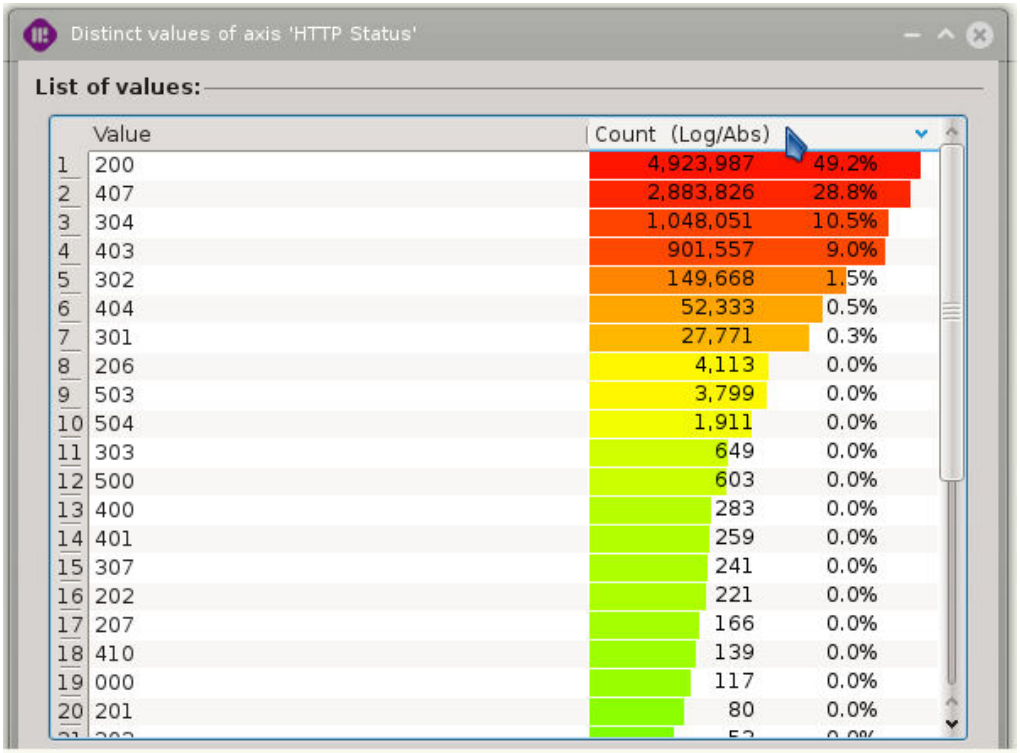
3.4.2.7 Time Series

This visualization displays different sets of metrics evolving over a common dimension. Each numeric axis of the parallel coordinates view can therefore be displayed as a series. Thereby, a series is dependent of the Plotting of its axis.



3.4.2.8 Statistics

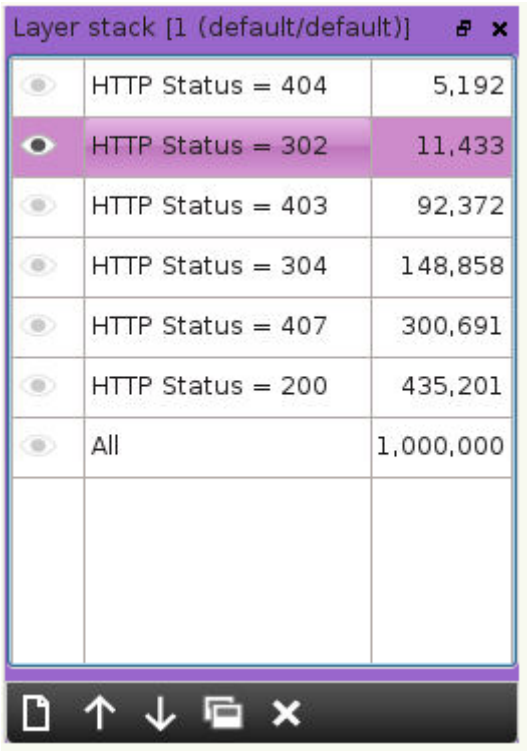
Statistics views are visually enhanced representations of specific requests especially tailored for optimal performances allowing powerful interactive manipulation. They are indispensable tools aimed at being used in conjunction with all other graphical views in order to progress meaningfully through the investigation.



	Value	Count (Log/Abs)	
1	200	4,923,987	49.2%
2	407	2,883,826	28.8%
3	304	1,048,051	10.5%
4	403	901,557	9.0%
5	302	149,668	1.5%
6	404	52,333	0.5%
7	301	27,771	0.3%
8	206	4,113	0.0%
9	503	3,799	0.0%
10	504	1,911	0.0%
11	303	649	0.0%
12	500	603	0.0%
13	400	283	0.0%
14	401	259	0.0%
15	307	241	0.0%
16	202	221	0.0%
17	207	166	0.0%
18	410	139	0.0%
19	000	117	0.0%
20	201	80	0.0%

3.4.2.9 Layer Stack

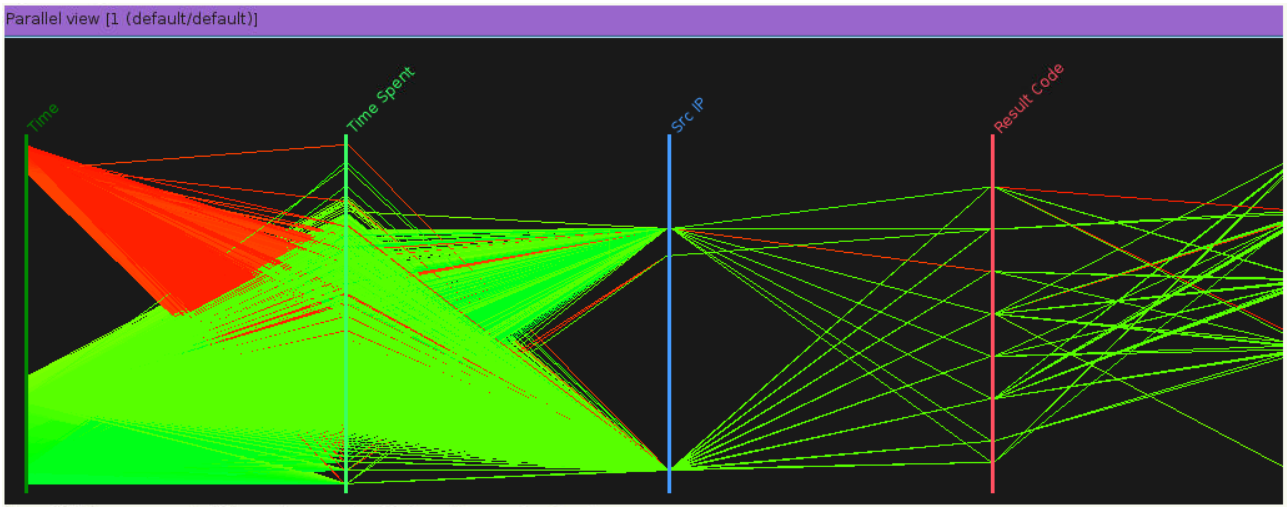
The Layer Stack View is the widget that manages all existing Layers of a View. By default, it is located at the bottom-right corner of the application's window.



3.4.2.10 Filters

In INENDI Inspector, Filter is a generic term that refers to a computation or a filtering process. It applies to a part or to all the data being investigated and might produce a change on the two following items:

- the current Selection of selected Events.
- the color of selected Events in the current active Layer.



3.4.2.11 Mapping and Plotting data values

INENDI Inspector is all about visualizing data in Parallel Coordinates views and about speeding-up the investigation and discovery processes by use of:

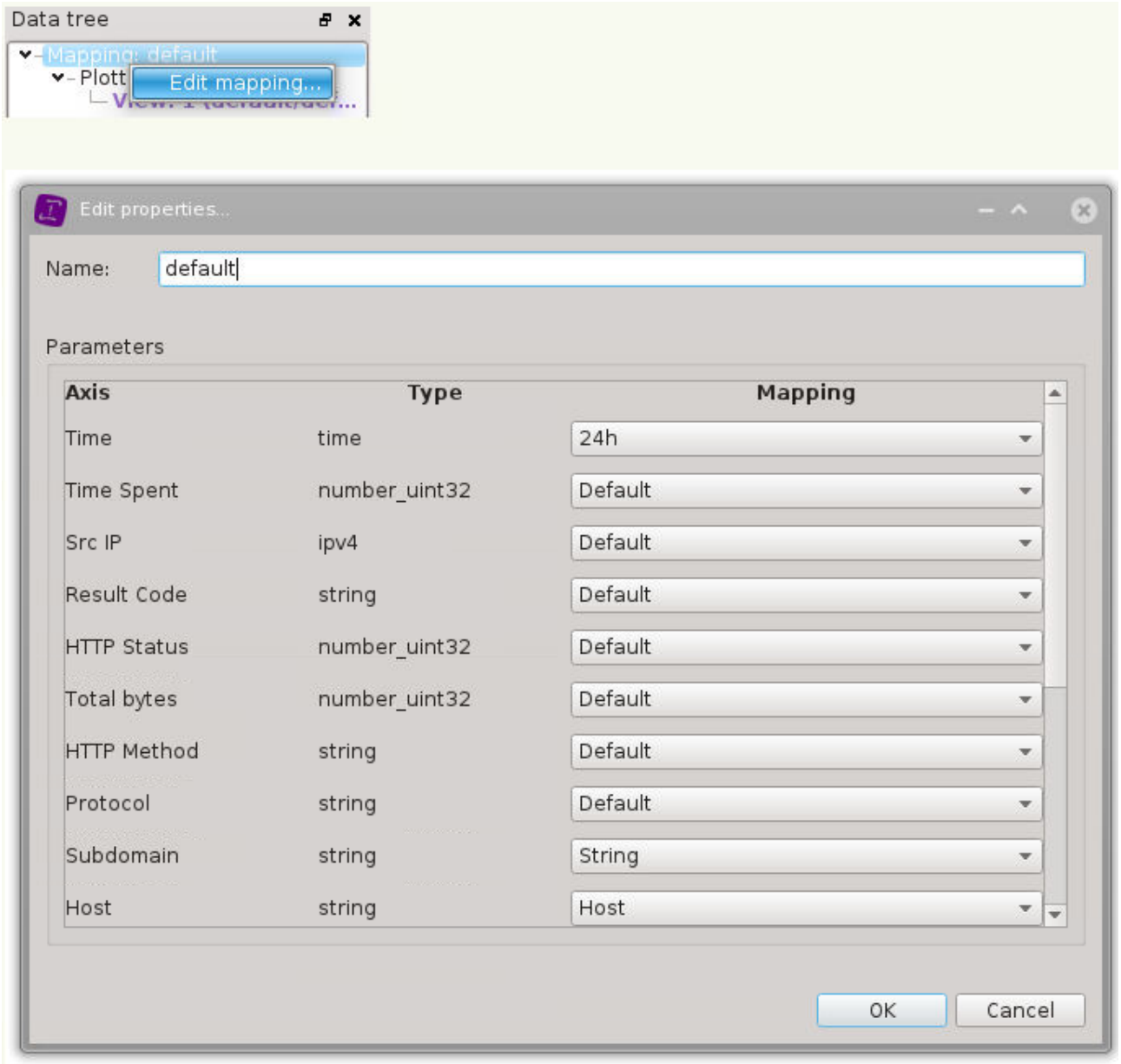
- the central Global Parallel Coordinates View

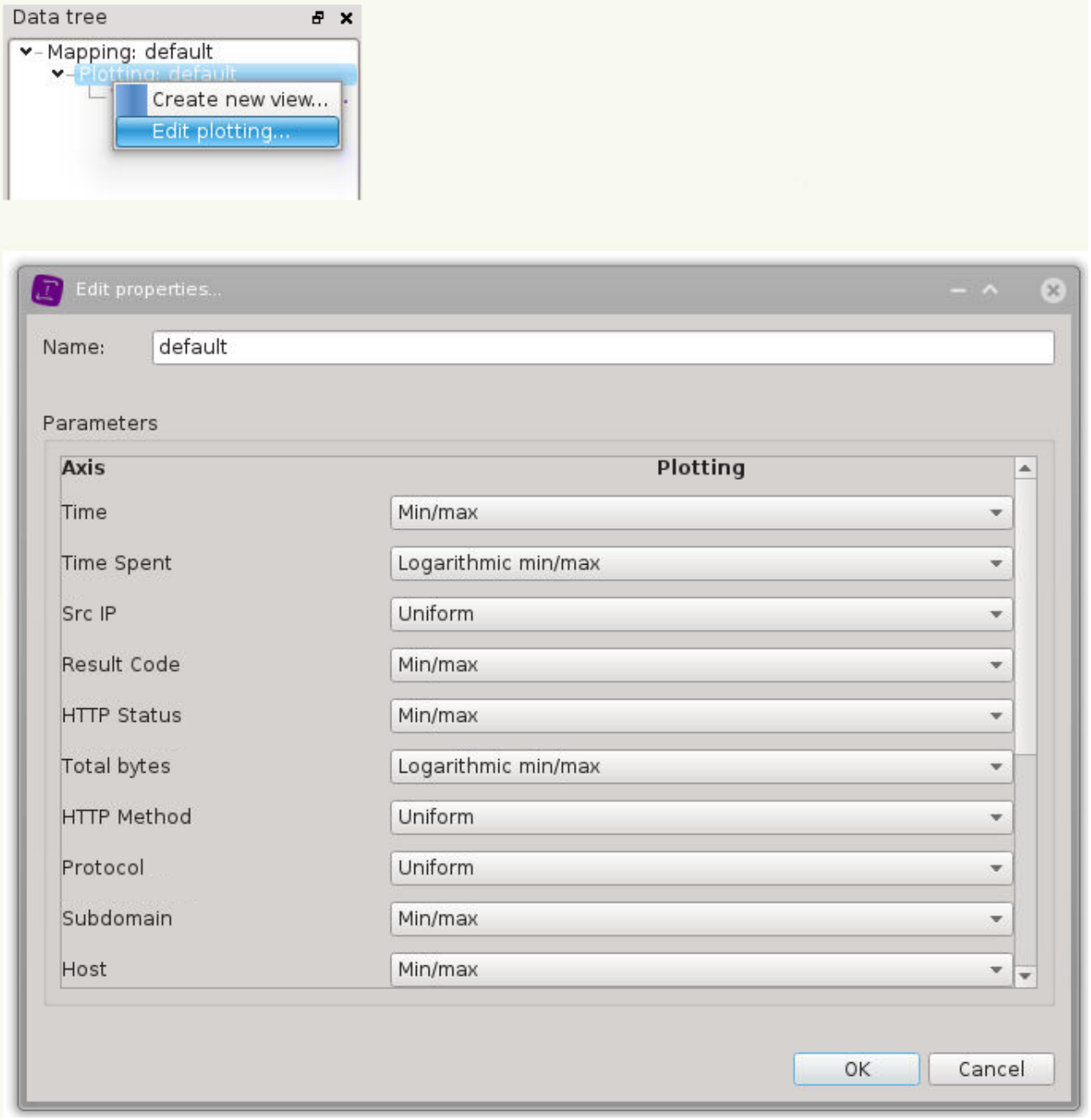
- all the satellite Views.

Such an investigation usually starts with a set of data that is parsed and cut in small pieces that end up filling a big table of values. This stage of the process requires a Format, that gives a description of how to cut the data, line by line.

But some more settings are required to turn this table of data into something that can be plotted on vertical Axes: namely, one needs to give two settings per Axis (i.e. column of the data table):

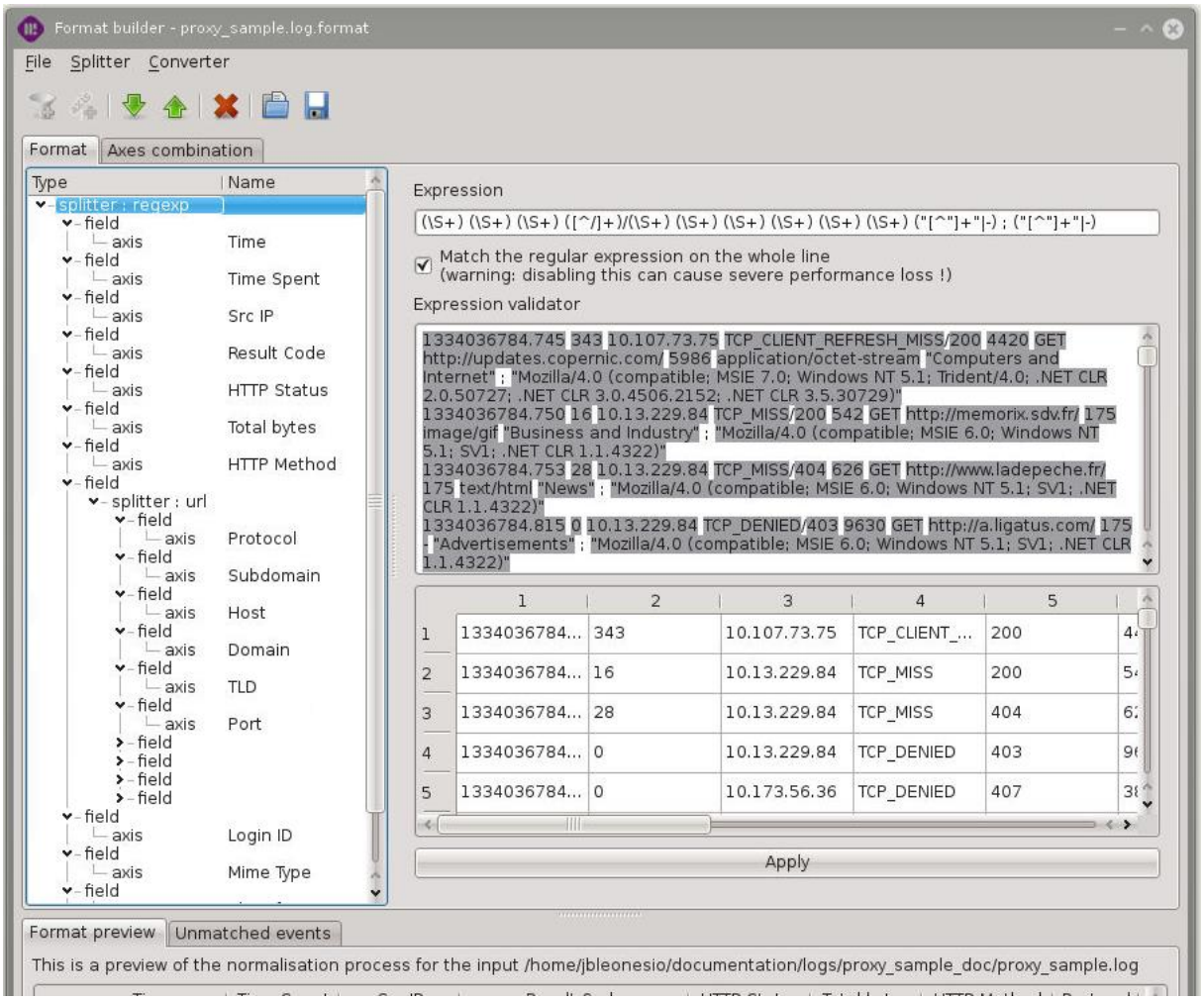
- a Mapping: this tells INENDI Inspector how it should map the values of that Axis (i.e. column) to mathematical values (real numbers or integer numbers) that naturally sit on a mathematical Axis.
- a Plotting: this setting tells INENDI Inspector how it should map the preceding mathematical values to the segment of the vertical Axis visible in the Global Parallel Coordinates View.





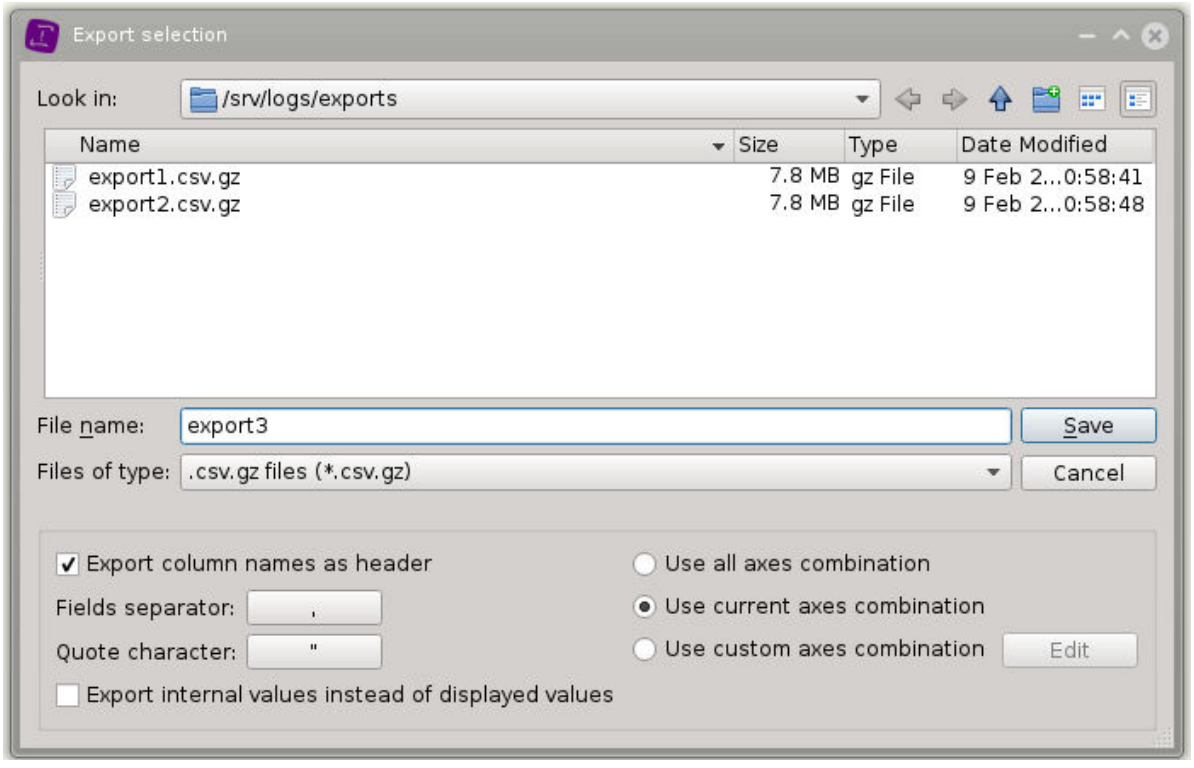
3.4.2.12 Format builder

The Format Builder is a main part of INENDI Inspector that is fully dedicated to the creation and management of Formats. Creating a Format can sometimes require some patience and skills but once the Format is ready and properly working, it can be used for a long time. It is usually quickly amortized. The format builder is able to automatically detect the most commonly used data types. It can for example automatically detect CSV files with the proper separator and quoting characters, a potential header and use it as axes name, but also all the supported columns types!



3.4.2.13 Exporting data

One can export various kinds of data out of INENDI Inspector.

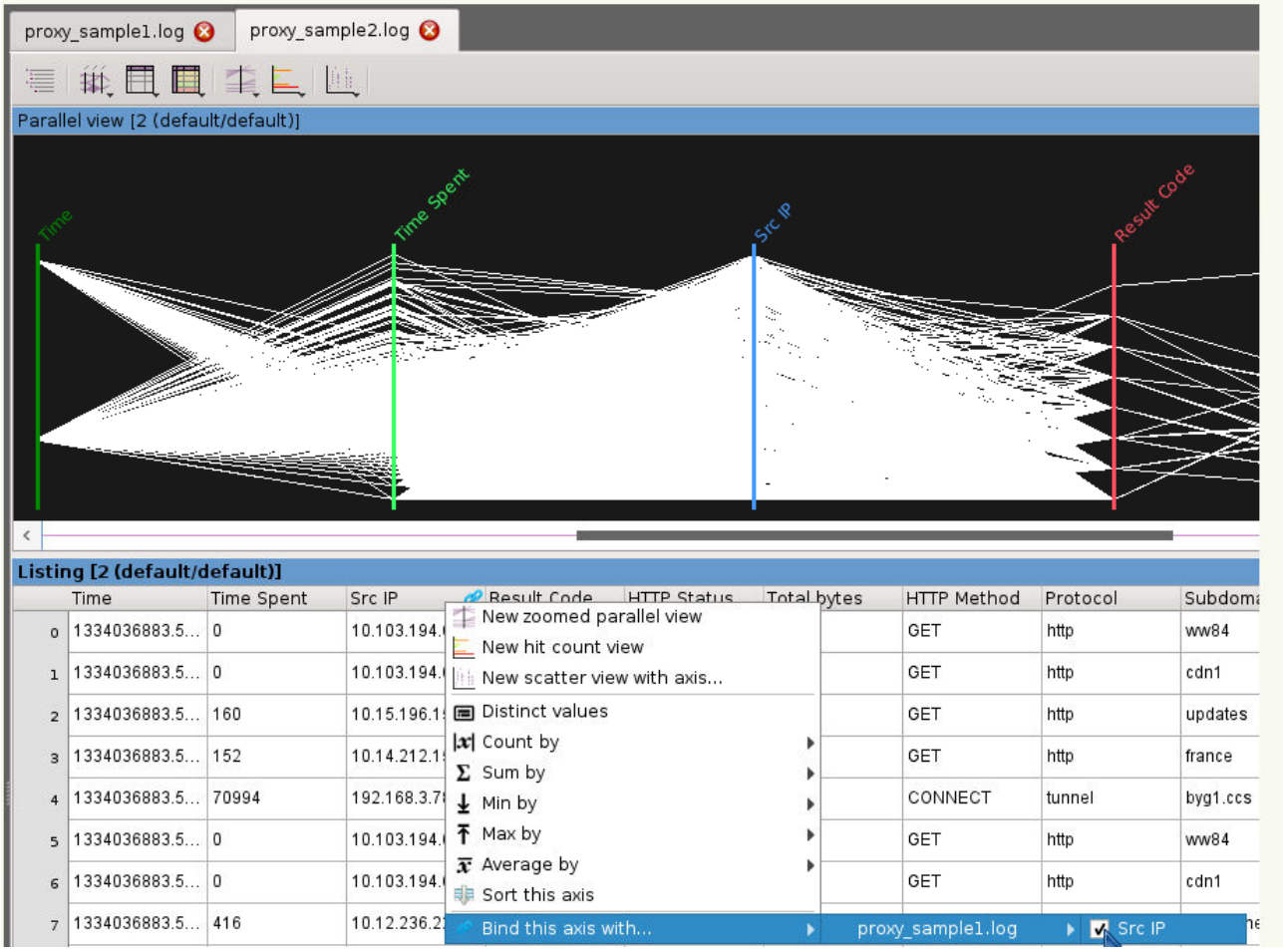


3.4.2.14 Correlation

The correlation module provides a way to propagate selections between sources and views using basic rules. It can be used to work seamlessly with different log types.

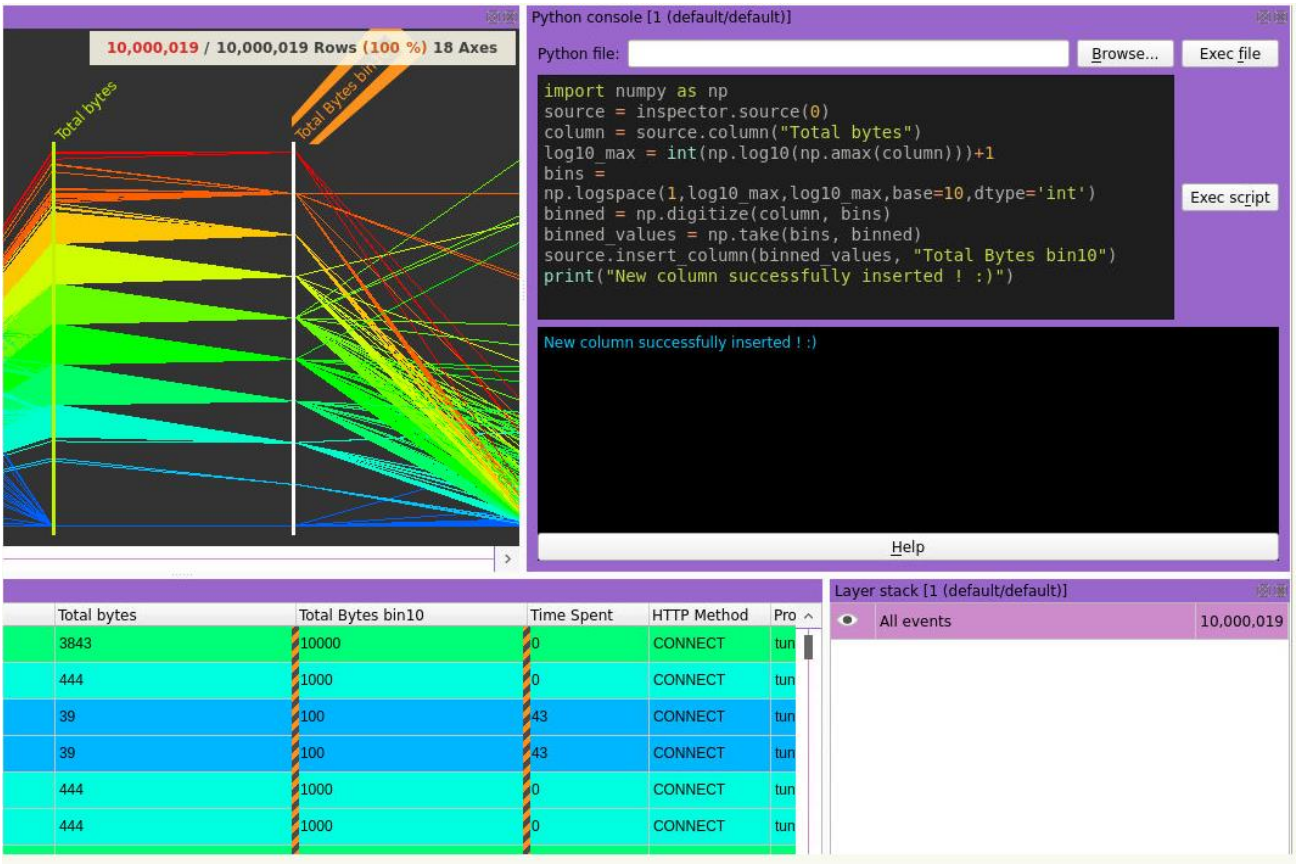
When an axis from a source is bound to a compatible axis from another source, every selection made on the origin source is propagated on the current layer of the destination source. As an example, if the binding is done on ipv4 axes, all the ipv4 selected on the current layer will automatically be selected on the current layer of the destination source.

A correlation is enabled when two axes are bound together:



3.4.2.15 Python scripting

This offers the possibility to create columns, layers and edit selection from python script.



== in for version, not package description

4 Conclusions

In this deliverable, we have provided the results of the Task 3.6 of IoTwins dedicated to the tools for explainability and visualization. We have presented three components developed for this task: a web-application that provides a toolbox for machine learning inspection, a web-application dedicated to high dimensional data representation in a smaller dimension, and an application dedicated to interactive explorations over very large amounts of data. For each component, we explain the technical background and illustrate its usage on a use case.