Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Master Degree in Data Science

Master Sc. Final Project

Machine Learning Implementations on Neurosuites Software

Author: Hugo Ernesto Nugra Madero
Tutors: Concha Bielza, Pedro Larrañaga

Madrid, October 2020

Master's final project
Master's degree in Data Science

Title: Machine Learning Implementations on Neurosuites Software

October 2020

Author:	Hugo Ernesto Nugra Madero
Tutors:	Concha Bielza, Pedro Larrañaga
Department:	Artificial Intelligence ETSI Informáticos Universidad Politécnica de Madrid

Acknowledgment

This project has been an invaluable opportunity for me for growing professionally because has given me the possibility of putting in practice my new knowledge acquired during my studies at Universidad Politécnica de Madrid. I would like to give special thanks to professors Concha Bielza and Pedro Larrañaga for their guidance, teachings and patience during the development of the present project. Without your support and ideas this whole project would not have been completed or existed. Additionally, I want to give thanks to my colleagues inside the Computer Intelligence Group for their advices and experiences shared inside the group.

I take the moment to express my gratitude to my parents Hugo and Anita who have allowed me to achieve this new goal with their support since the beginning of my life. To all my family for their unfailing messages of support in each conversation. Finally, I thank Madelyn for her love and help during this Master's Thesis

Abstract

Machine Learning has been one of the fastest-growing areas in the Computer Science field. Several public and private entities are making significant efforts to take the approach of this new set of tools. Today it is possible to see the effect of Artificial Intelligence through Machine Learning on recommendation systems, facial recognition, or natural language processing.
On the other hand, a new technology produces a gap between the people with expertise in the domain and those who do not. In this case, the use of Machine Learning algorithms is relegated to people who know how to program software. Already this skill is not as spread as needed and now even more with the upcoming wave of innovations brought by the advances on Artificial Intelligence. Perhaps there are some tools for performing Machine Learning through a user interface without writing code, a complete web suite for performing machine learning seems necessary for reducing this gap.
NeuroSuites is a project created at the UPM (Universidad Politécnica de Madrid) as part of the Horizon 2020 Fet Flagship Human Brain Project and of a Spanish Ministry-funded project (TIN2016 -P). It includes a set of tools for working with neuroscience. There are two modules made for a general-purpose: data statistics and machine learning. With the idea of making machine learning available to more users, The Computer Intelligence Group has included the Machine Learning module inside NeuroSuites, which allows the society to use these new techniques without the necessity of coding software.
This study is composed of the implementation of four parts: data pre-processing, non-probabilistic clustering, multi-label classification and visual changes. The first one is adding pre-processing to the already implemented supervised classification module. With it, the user will have many options to make changes and configurations to prepare the dataset that will be passed to the classification algorithms. The non-probabilistic clustering functionality allows the use of many algorithms to create and analyze clusters from a dataset, following the flow suggested by the application. The multi-label classification is implemented over the supervised classification model. The way NeuroSuites works within it depends on the number of target variables selected. In the case of multi-label classification, users have a list of options for working with this type of dataset. Finally, visual changes have been made to the application for giving it a fresher and more harmonious image.

Resumen

El aprendizaje automático ha sido una de las áreas de más rápido crecimiento en el campo de la informática. Varias entidades públicas y privadas están realizando importantes esfuerzos tomar ventaja de este nuevo conjunto de herramientas. Hoy es posible ver el efecto de la inteligencia artificial a través del aprendizaje automático en los sistemas de recomendación, el reconocimiento facial o el procesamiento del lenguaje natural.
Por otro lado, una nueva tecnología produce una brecha entre las personas con experiencia en el dominio y las que no. En este caso, el uso de algoritmos de Machine Learning queda relegado a personas que tienen la capacidad de programar software. Esta habilidad hoy no está tan extendida como se necesita y ahora aún más con la próxima ola de innovaciones traídas por los avances en Inteligencia Artificial. Quizás existen algunas herramientas para realizar Machine Learning a través de una interfaz de usuario sin escribir código, sin embargo, una suite web completa para realizar Machine Learning parece necesaria para reducir esta brecha.
NeuroSuites es un proyecto creado en la UPM (Universidad Politécnica de Madrid) como parte del Proyecto Horizonte 2020 Fet Flagship Human Brain Project y de un proyecto financiado por el Ministerio de España (TIN2016 -P). Incluye un conjunto de herramientas para trabajar con neurociencia. Hay dos módulos hechos para un propósito general: estadística de datos y aprendizaje automático. Con la idea de hacer que el aprendizaje automático esté disponible para más usuarios, El Grupo de Inteligencia Computacional (CIG por sus siglas en inglés) ha incluido el módulo de aprendizaje automático dentro de NeuroSuites, que permite a la sociedad utilizar estas nuevas técnicas sin el requisito de saber escribir código de software.
Este estudio se compone de la implementación de cuatro partes: preprocesamiento de datos, agrupamiento no probabilístico, clasificación multi-etiqueta y cambios visuales. El primero es agregar preprocesamiento al módulo de clasificación supervisada ya implementado. Con él, el usuario tendrá un conjunto de opciones para realizar cambios y configuraciones para preparar el conjunto de datos que se pasará a los algoritmos de clasificación. La funcionalidad de agrupamiento no probabilístico permite el uso de varios algoritmos para crear y analizar grupos a partir de un conjunto de datos, siguiendo el flujo sugerido por la aplicación. La clasificación de etiquetas múltiples se implementa sobre el módulo de clasificación supervisado. La forma en la que este módulo funciona depende del número de variables objetivo seleccionadas por el usuario. En el caso de la clasificación de etiquetas múltiples, los usuarios tienen una lista de distintos algoritmos para trabajar con este tipo de datos. Finalmente, se han realizado cambios visuales en la aplicación para darle una imagen más actual y armoniosa.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Machine Learning growth

Today machine learning is a hot topic by itself, but at the same time is part of two big subjects: Data Science and Artificial Intelligence. It is often possible to find news about new advances in this field, and its importance increases due to its use in different areas like disease diagnosis, self-driving vehicles, natural language processing, and many other application fields that are present in several platforms.

The academic environment shows evidence of this wave of publications. Figure 1.1 shows that there are 100 new machine learning papers being published per day just on arXiv in 2018, representing about 33000 papers per year. Additionally, this trend does not seem to have changed, and the growth rate is even higher than Moore's law, that is of 2x every two years.

This expansion is appreciable in Figure 1.2 but this time in the European Machine Learning market size. This extension exhibits a non-linear curve describing the number of challenges being solved, and that many more are coming. Furthermore, the economic impact if this technology is getting more significant, and several industries are making it part of their internal processes. Figure 1.3 displays the end-use share adoption by sector, where banking, financial services & insurance and advertising & media are receiving primarily the solutions provided by Machine Learning. Moreover, healthcare, automotive & transportation, and retail are fields with an important and similar market share. Activities like agriculture or law seem to have room for innovation, acquiring the newly available features given by Machine Learning.

The application of Artificial Intelligence powered by Machine Learning appears even on services related to creativity for writing like narrative science[1], which generates storytelling taking data as a source, replacing it in this way to the dashboards.

Notably, artificial intelligence is turning a decisive factor, and that is why organizations like open.ai[2] were founded to give insights about how this technology should be driven. Another sign of it is the AI ethics meeting sponsored by the Vatican with Microsoft and IBM[3] to develop a plan to collaborate and manage the possible coming

[1]https://narrativescience.com/
[2]https://openai.com/about/
[3]https://www.bbc.com/news/technology-51673296

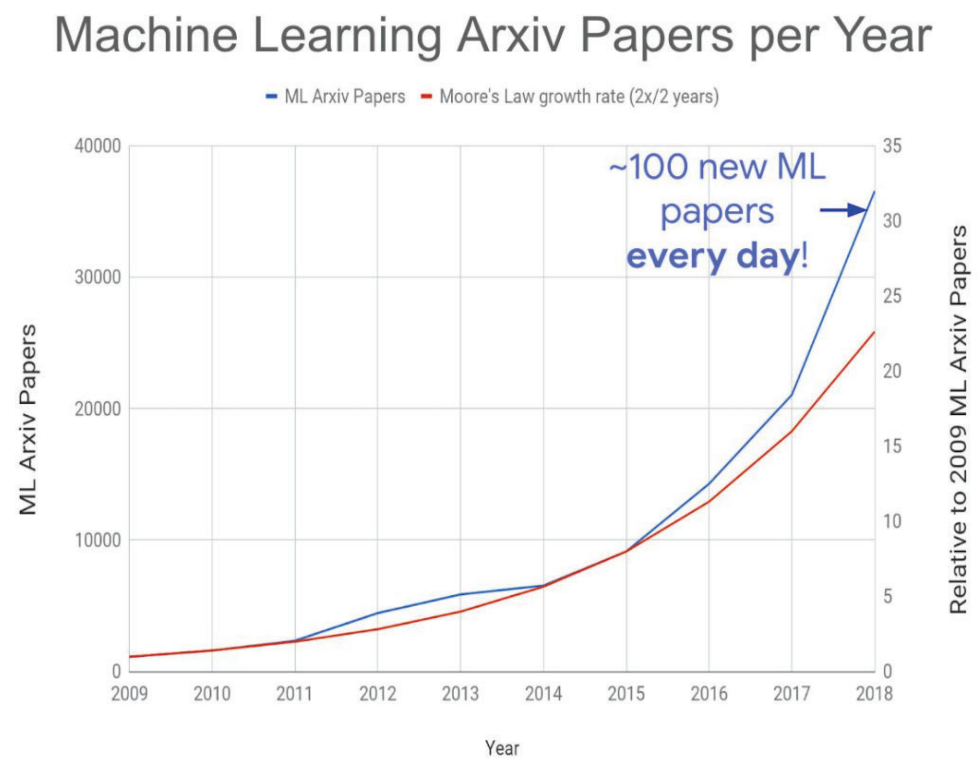


Figure 1.1: Machine learning papers per year
Chart obtained from https://ieeexplore.ieee.org/stamp

implications. Finally other initiatives like the Future of Life Institute [4] are currently focused on Artificial Intelligence by trying to keep it beneficial tu humanity. All these facts show the weight of this technology in the actual context.

1.2 NeuroSuites project

NeuroSuites is a web application that offers tools for mainly working with neuroscience through a set of sub-applications with a defined use. The Computational Intelligence Group at Universidad Politécnica de Madrid develops it starting by 2017. This project has received funds from the Human Brain Project (SGA2 and SQA3) and the Spanish government (LTIN2016). The project is accessible through the URL: https://neurosuites.com/.

NeuroSuites includes two general-purpose modules: data statistics and machine learning. It means that the same dataset can be used for performing statistical analyses or/and machine learning. The data statistics module offers the possibility of performing tests, relations among variables and visualizing plots about the selected dataset.

The machine learning module is a tool that helps to perform classification or clustering processes, but as mentioned, it can work with any other domain. In fact, for the case study for clustering (Section 4.5), the source was data from astronomy about pulsar stars, and for multi-label classification (Section 5.4), an emotions dataset.

[4]https://futureoflife.org/team/

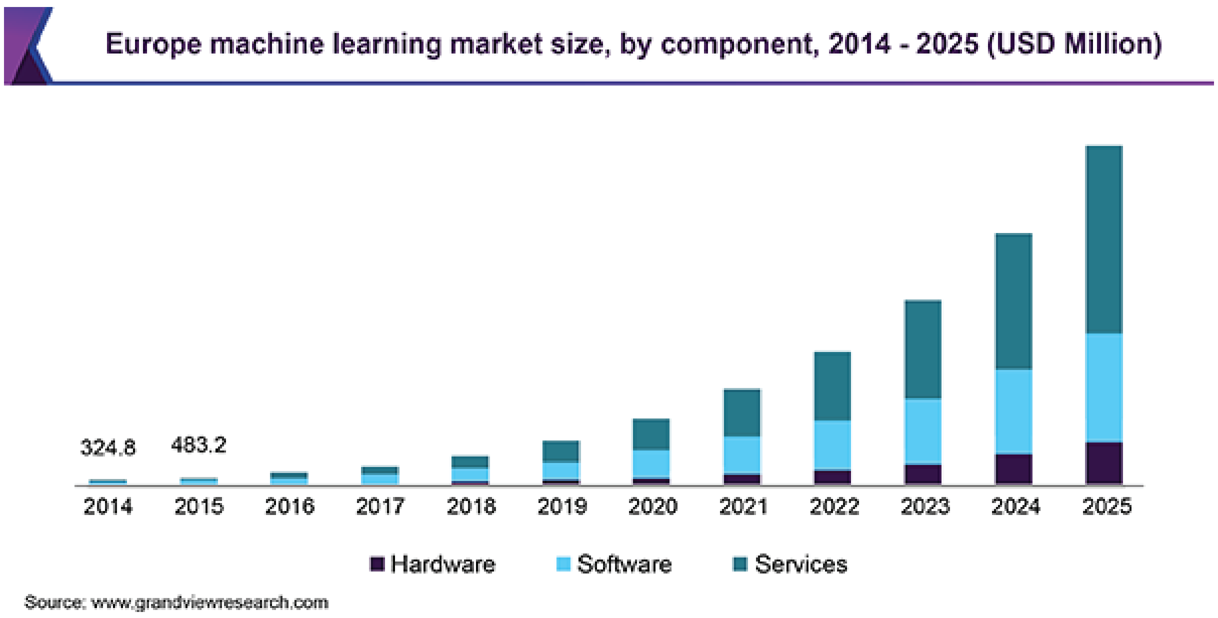


Figure 1.2: Global machine learning market size, by component, 2014 - 2025
Chart obtained from
https://www.grandviewresearch.com/industry-analysis/machine-learning-market

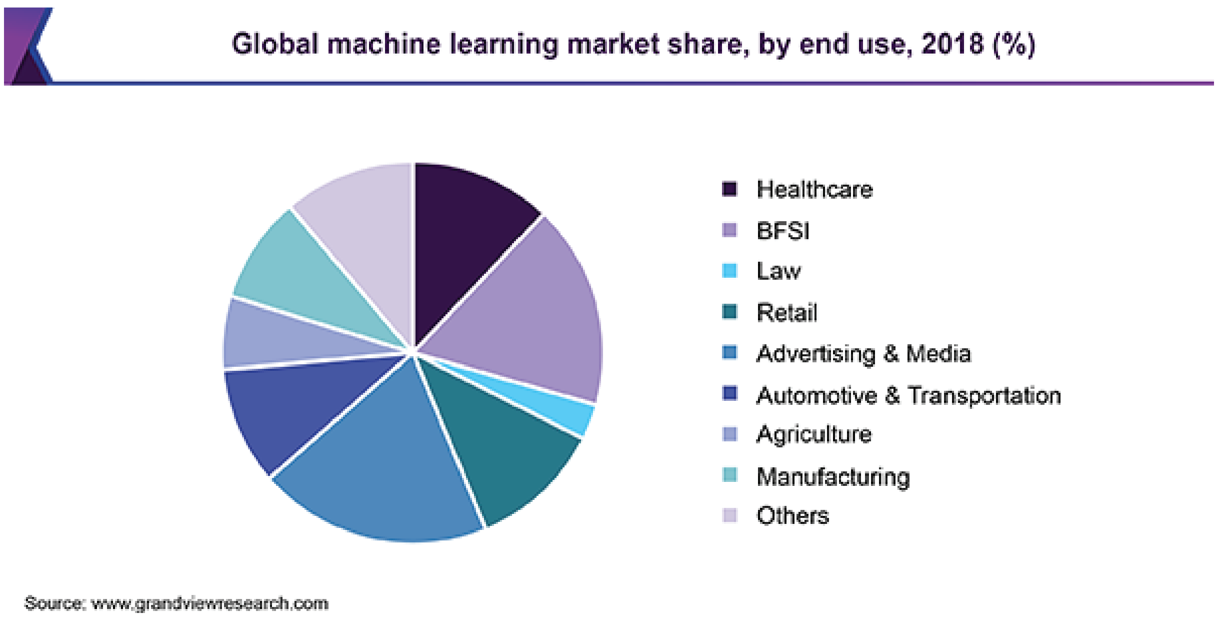


Figure 1.3: Global machine learning market share, by end use, 2018
Chart obtained from
https://www.grandviewresearch.com/industry-analysis/machine-learning-market

1.3 State of the art in no coding machine learning platforms

Machine learning nowadays is a hot topic in Computer Science and Artificial Intelligence; the advances are continuously growing on each sub-category of this domain. There are several resources for learning machine learning foundations or practicing like Scikit-learn for machine learning, Tensorflow for deep learning, or many other options. However, there is a requirement that makes the learning process harder, and it is the skill of coding software. That is why NeuroSuites, with its module for performing machine learning, was created because users will be able to apply these new techniques through a graphical user interface that is hosted online without programming skills. The application is available on any operating system and any device without installing it. Some tools aim at the same purpose, but there are differences, mainly on how to approach the goal, of letting new users use several algorithms without coding.

In this review four platforms with functions related to machine learning were taken into account. These are Weka, BigML, IBM SPSS and RapidMiner Studio. The resources that can be found on the Internet are mainly associated with coding scripts but there are no tools for this goal of working with machine learning without coding. As a comparison between the platforms Table 1.1 relates them according to their main implementation characteristics, and Table 1.2 displays them by their Machine Learning features.

Weka[5] (Waikato Environment for Knowledge Analysis) is an open-source application that allows access through a graphical interface, a terminal or a Java API. It was developed by the Waikato University and was written on Java. It includes a collection of tools for visualizing and analyzing data, predictive modeling, and data mining. This software can be installed on any operating system, but this is the restriction at the same time because there is not a web interface for accessing the tool.

BigML[6] differs from Weka, IBM SPSS and RapidMiner Studio because it is an online platform that allows applying a variety of machine learning resources for training models. It provides access from a graphical interface or through an API. This is not an open-source software and provides a free access tier for small datasets up to 16 megabytes. In addition BigML is the same company that develops this software.

IBM SPSS[7] is a desktop application that needs to be installed, but it can be on any operating system. IBM SPSS offers an advanced statistical analysis that integrates machine learning, and due to its easy use and flexibility, it lets access to users without significant expert knowledge. This platform does not offer a free access layer nor is open-source.

RapidMiner[8] Studio is an informatics program for data analysis and data mining. Its original version was developed at the Dortmund University, but the actual versions are paid. It is a multi-platform desktop application and offers integration with other programming languages like Python or R.

[5]https://www.cs.waikato.ac.nz/ml/weka
[6]https://bigml.com
[7]https://www.ibm.com/analytics/spss-statistics-software
[8]https://rapidminer.com/

Application	Hosting platform	Free	Saves on cloud	Open-source
NeuroSuites	Web	Yes	No	Yes
Weka	Desktop	Yes	No	Yes
BigML	Web	Yes (up to 16MB)	Yes	No
IBM SPSS	Desktop	No	No	No
RapidMiner Studio	Desktop	Yes (for old versions)	No	Yes (for old versions)

Table 1.1: Machine learning platforms comparison by implementation characteristics

Application	Classification	Regression	Clustering	Multi-label classification
NeuroSuites	Yes	No	Yes	Yes
Weka	Yes	Yes	Yes	No
BigML	Yes	Yes	Yes	No
IBM SPSS	Yes	Yes	Yes	No
RapidMiner Studio	Yes	Yes	Yes	No

Table 1.2: Machine learning platforms comparison by Machine Learning features

1.4 Contributions

The main objective in this project is adding more capabilities to the machine learning module inside NeuroSuites. This work represents six months of work within the Computational Intelligence Group CIG at Universidad Politécnica de Madrid. During this period, the principal machine learning module has incorporated three new functionalities. The list of contributions has been divided into four chapters in this report.

1. **Feature pre-processing** is part of the clustering and classification and allows users to make transformations to the data before or later it be used for training the models. The input options and elements for this purpose were previously defined, but the code implementation was not done. Additionally, feedback on specific steps is shown to the user for giving information about the internal changes.

2. **Non-probabilistic clustering** was wholly designed and implemented during this project. The results observation step shows graphical information about clusters and samples all projected in a PCA of two components. The algorithms permit the input of several hyperparameters, the obtained groups can be downloaded and in special cases additional tables or plots are available.

3. **Multi-dimensional classification** was built over the same visual elements of the single class classification functionality; the reason was to give an intuitive and standardized workflow to the system. This contribution is a distinctive point in comparison with other platforms that do not include it. It allows users to choose more than one classification algorithm and includes problem transformation and algorithm adaptation methods for processing the task.

4. **Visual changes** are present in the application with a brand new logo, a new color palette, and renovated menus colors and organization. Data stats with its two sub-functionalities received a layout organized by tabs representing progressive steps. In the same way, the previous single-class classification capacity was changed to have the tab layout. All these changes aim to make the use of the application more friendly in an attractive visual environment.

Chapter 2

NeuroSuites platform

NeuroSuites is a system composed of a set of several modules that are mainly focused on the neuroscience field. However, there are some other modules like machine learning, which is the central part of work during this project. Table 2.1 presents the list of applications organized by their characteristics.

2.1 Project architecture

NeuroSuites follows the MVC architecture, and the back-end is built using **Python 3**[1] as the programming language and **Django**[2] as the framework. The front-end is implemented with the Django templating system, which uses **HTML 5**[3], **CSS 3**[4] with **Bootstrap 3**[5], and **Javascript**[6] with **Jquery**[7]. The application performs several asynchronous calls to the server, and **Ajax**[8] is the tool used for this goal.

The project is contained over **Docker**[9]. For interaction with **R**[10] packages, **R2py**[11] is used. Finally, a **PostgreSQL**[12] database stores the session data, and the message broker is **RabbitMQ**[13] for long time requests.

The version control system is **Git**[14], that allows traceability over the code changes, and this is hosted on **Gitlab**[15]. The code is available to anyone and, at the same time, is protected through the URL: https://gitlab.com/mmichiels/neurosuite.

[1]https://www.python.org/download/releases/3.0/
[2]https://www.djangoproject.com/
[3]https://dev.w3.org/html5/html-author/
[4]https://developer.mozilla.org/en-US/docs/Archive/CSS3
[5]https://getbootstrap.com/docs/3.3/
[6]https://developer.mozilla.org/en-US/docs/Web/JavaScript
[7]https://api.jquery.com/
[8]https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
[9]https://www.docker.com/
[10]https://www.r-project.org/
[11]https://rpy2.github.io/
[12]https://www.postgresql.org/
[13]https://www.rabbitmq.com/
[14]https://git-scm.com/
[15]https://about.gitlab.com/

Module	Field of application	Need of a dataset
L-Measure - Extract morphological measurements	Neuroscience	Yes
NeuroViewer - 3D Neuron reconstruction	Neuroscience	Yes
NeuroSTR - Validator, format converter	Neuroscience	Yes
3DBasalRM - Repair cut-points in the basal arborization	Neuroscience	Yes
GabaClassifier - Interneuron classifier	Neuroscience	Yes
Statistics engine	General-purpose	Yes
Machine learning	General-purpose	Yes
3DspineS - Dendritic spine simulation	Neuroscience	No
3DSomaMS - Delimit the neuronal soma	Neuroscience	No
3DSynapsesSA - Analyze spatial distribution of cortical synapses	Neuroscience	No
Dendrite arborization simulation - Generate synthetic dendrite arborization	Neuroscience	No

Table 2.1: List of modules that are part of NeuroSuites

Another element is the server that hosts the application, It is a server running over **Ubuntu Server**[16] with 64 GB of RAM memory and an i7-6800k processor with six cores and a frequency of 3.40 GHz. These characteristics let to manage heavy load jobs. The installed Docker container includes all the project images letting it run without major configuration. A general view of the system's structure is available at Figure 2.1

2.2 Machine learning in NeuroSuites before this project

The machine learning module was composed of a set of three functionalities: Bayesian networks, probabilistic clustering, and supervised classification. Bayesian networks and probabilistic clustering are options that are beyond the scope of this study. On the other hand, supervised classification was a non-complete functionality due to the lack of a pre-processing step for data. It produces errors on algorithms which can not deal with empty or null values. There are other missing functions like features normalization or data discretization. The visual structure of data pre-processing was already built, but the code implementation was not provided.

Supervised classification was working just for classification problems with one target variable. However, for multi-label classification problems with more target variables, it was not possible.

2.2.1 Supervised classification

The flow for supervised classification follows four steps:

[16]https://ubuntu.com/server

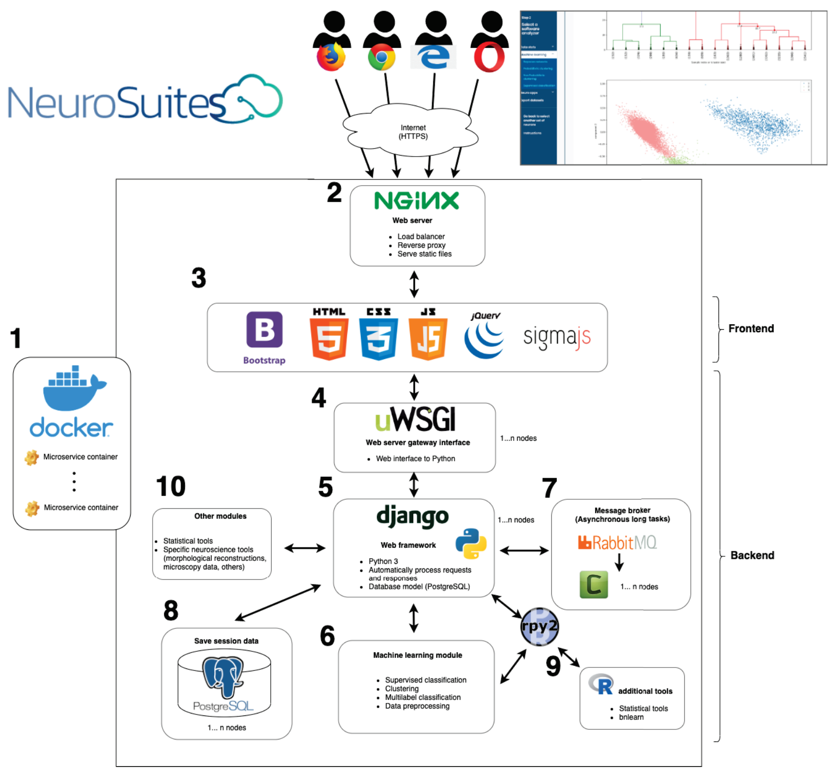
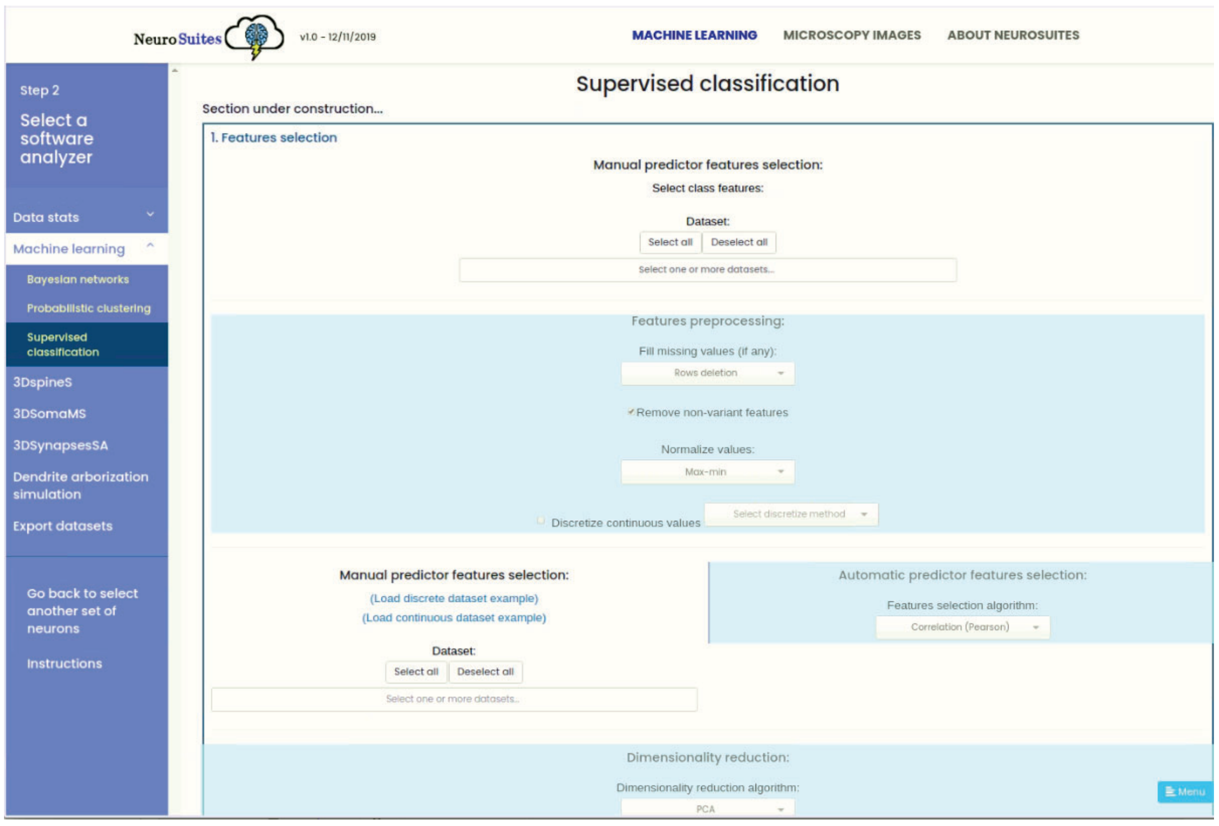


Figure 2.1: NeuroSuites architecture.
Chart obtained from CIG documentation



Supervised classification

Section under construction...

1. Features selection

Manual predictor features selection:

Select class features:

Dataset:

Select all | Deselect all

Select one or more datasets...

Features preprocessing:

Fill missing values (if any):

Rows deletion

✓ Remove non-variant features

Normalize values:

Max-min

Discretize continuous values | Select discretize method

Manual predictor features selection:

(Load discrete dataset example)

(Load continuous dataset example)

Dataset:

Select all | Deselect all

Select one or more datasets...

Automatic predictor features selection:

Features selection algorithm:

Correlation (Pearson)

Dimensionality reduction:

Dimensionality reduction algorithm:

PCA

Figure 2.2: Supervised classification before this project (screenshot).

1. Feature pre-processing (incomplete)
2. Validation schemes: training-testing, k-fold cross-validation
3. Algorithm selection
4. Performance evaluation

On feature pre-processing (step 1), the user selects the class variable and predicting features. There was wrongly available the option of choosing continuous variables as target variables. All this falls into the regression category, beyond to date the scope of NeuroSuites. All the other transformation steps were not implemented. Fig reffig:supervised$_beforeshowsascreenshotofthefunctionalitybeforetheappliedchanges.$

The second step corresponds to selecting the k value for the k-fold cross-validation process for the learning phase. Establishing the percentage of instances that will be used for training and testing sets is also decided. There is the possibility of shuffling the samples.

For the learning process (step 3), the user has the possibility of selecting one or more supervised classification algorithms. The options are: k-nearest neighbors, rule induction, decision tree, random forest, support vector machines, neural network, linear discriminant analysis, quadratic discriminant analysis, logistic regression, Naive Bayes, tree augmented Naive Bayes, bagging meta-classifier, boosting meta-classifier and stacking meta-classifier. Each model receives its hyper-parameters that are set by the users.

In step four, after the learning step, users can download the predictions generated by each model on CSV or Parquet gzip format, see the hyper-parameters and download wrong predictions. Users can also check performance evaluation metrics like training time, accuracy, F-score, precision, recall, and Brier score.

Chapter 3

Implementation of data pre-processing techniques in NeuroSuites

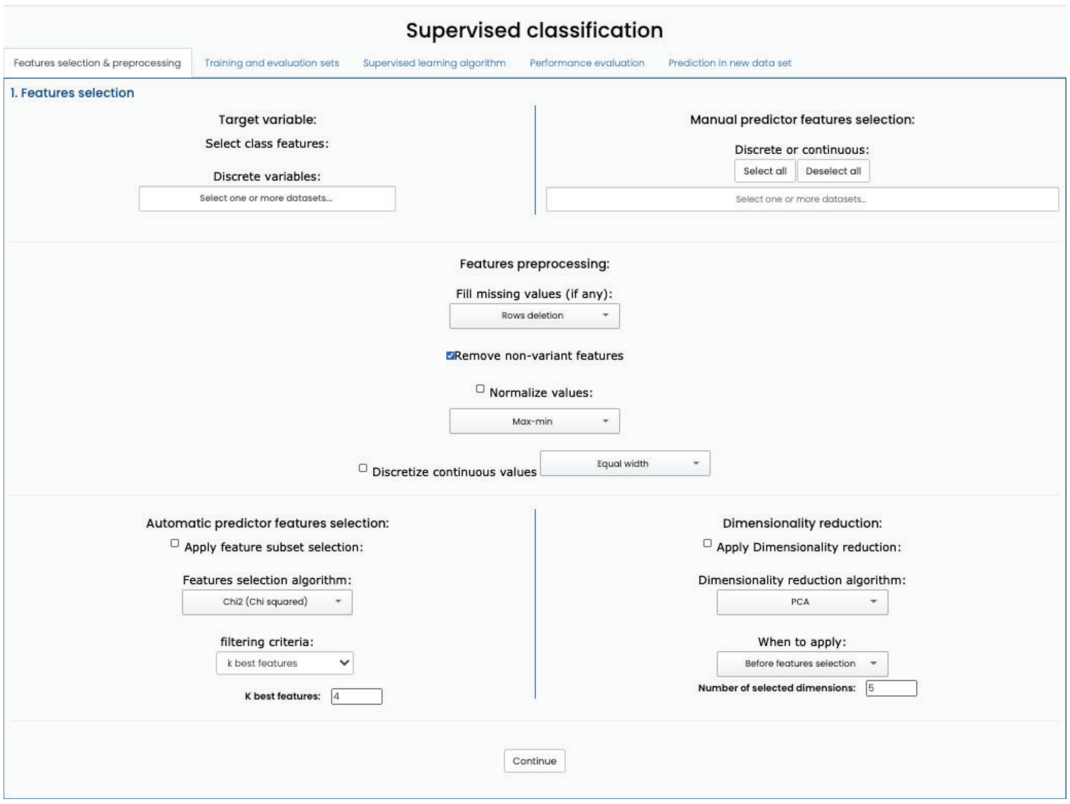
Several algorithms are sensitive to not standardized data or empty values; consequently, a previous step before feeding models is necessary for fixing all possible errors or biases inside the dataset. Data pre-processing is present in supervised classification entirely

This chapter presents the options added in supervised classification and non-probabilistic clustering. The internal implementation is built using javascript and AJAX for sending the parameters to the server asynchronously. Once the server has received the request, it transforms the uploaded dataset with the options selected by the user for pre-processing the data and returns to the client a status response and some information depending on the method, e.g. a plot of the data after dimensionality reduction.

3.1 Functionality flow

The dataset follows a set of steps almost lineal. Each part helps to prepare the source and receive the parameters defined by the user a screenshot of this module is presented in Figure 3.1. The list that conforms this functionality in NeuroSuites is:

1. Fill missing values
2. Remove constant features
3. Values normalization
4. Continuous values discretization
5. Feature subset selection
6. Dimensionality reduction (can be applied before, after, or before and after feature subset selection)



Supervised classification

Features selection & preprocessing | Training and evaluation sets | Supervised learning algorithm | Performance evaluation | Prediction in new data set

1. Features selection

Target variable:
Select class features:
Discrete variables: Select one or more datasets...

Manual predictor features selection:
Discrete or continuous: Select all | Deselect all
Select one or more datasets...

Features preprocessing:
Fill missing values (if any): Rows deletion
☑ Remove non-variant features
☐ Normalize values: Max-min
☐ Discretize continuous values: Equal width

Automatic predictor features selection:
☐ Apply feature subset selection:
Features selection algorithm: Chi2 (Chi squared)
filtering criteria: k best features
K best features: 4

Dimensionality reduction:
☐ Apply Dimensionality reduction:
Dimensionality reduction algorithm: PCA
When to apply: Before features selection
Number of selected dimensions: 5

Continue

Figure 3.1: Feature pre-processing flow

3.2 Implementation design

In order to achieve pre-processing, were implemented two classes: Preprocessing and MachineLearningHelpers. The first one is responsible of applying the different steps to data in the mentioned flow and processing those steps which do not need Machine Learning. For the other cases where Machine Learning must be used, Feature selection and dimensionality reduction the class MachineLearningHelper lets this functions and the first class has access to these methods. Figure 3.2 displays the UML class diagram with the details of methods implemented by each class.

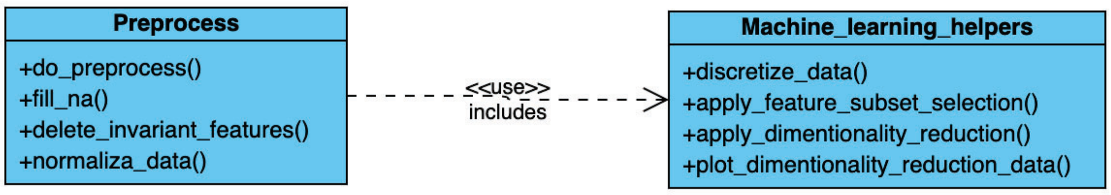
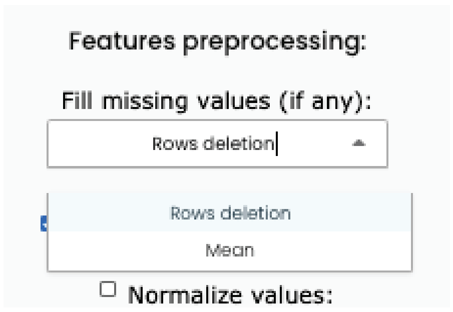

Preprocess
+do_preprocess()
+fill_na()
+delete_invariant_features()
+normaliza_data()
<<use>>
includes
Machine_learning_helpers
+discretize_data()
+apply_feature_subset_selection()
+apply_dimentionality_reduction()
+plot_dimentionality_reduction_data()

Figure 3.2: Pre-processing UML class diagram


Features preprocessing:

Fill missing values (if any):

Rows deletion

Rows deletion
Mean

☐ Normalize values:

Figure 3.3: Missing values

☑Remove non-variant features

Figure 3.4: Constant features

3.3 Missing values

A missing value is a null or empty value that corresponds to a feature in an instance. Missing values may produce errors for specific algorithms; that is the reason for cleaning the dataset. NeuroSuites includes two options for handling missing values: rows deletion and mean imputation. The element is shown in Figure 3.3

3.3.1 Rows deletion

This method eliminates each instance with an empty or null value on any feature.

3.3.2 Mean imputation

This option fills an empty or null value with the mean of the corresponding feature. It means that the value depends on the population, avoiding to lose information as with rows deletion.

3.4 Constant features

A feature with no variation among each instance shows no interaction between the other features. With this premise, the feature will pass noise to the classification algorithm. Therefore this method removes this kind of features from the dataset. This element is visible in FIgure 3.4

3.5 Values normalization

Features with higher values scales tend to have more impact over the process. With the intention of reducing this bias normalization or scaling aims to set the same degree of importance to all features using a defined rule (Huang et al., 2015).

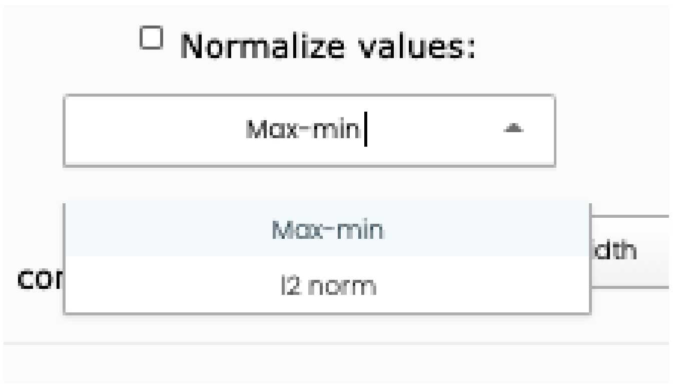


Figure 3.5: Values normalization

Method	Description
Equal width	Each bin has the same range of values
Equal frequency	Each bin has the same amount of values
Fayyad & Irani MDL	Uses mutual information to recursively get the best bins (Fayyad and Irani, 1993)

Table 3.1: Discretization options

3.5.1 Max-min

This method scales the values in all features to range from 0 to 1. The way his process works is by applying the formula:

$$y = x - min(x)/max(x) - min(x)$$

The code implementation used in this project was the once available from the Library Scikit-Learn:

3.5.2 l2 norm

This approach calculates the vector's length of features in a Euclidean space. The definition is given by the square root of the total sum of the squares of the values on each feature. Finally each feature is divided by the calculated distance and the result will be obtained for each feature.(Li and Jain, 2009). The results are from -1 to 1.

3.6 Continuous values discretization

Some classification algorithms, like trees or rule induction, work better with discrete values. Discretization reduces complexity in the data althought some information is lost. The result is a set of intervals called bins, which contains grouped values with labels for each interval. In order to add discretization was used the library Scikit-Learn as in all the functions in this chapter. In NeuroSuites were implemented three different options of discretization detailed in Table 4.2 and FIgure 3.6

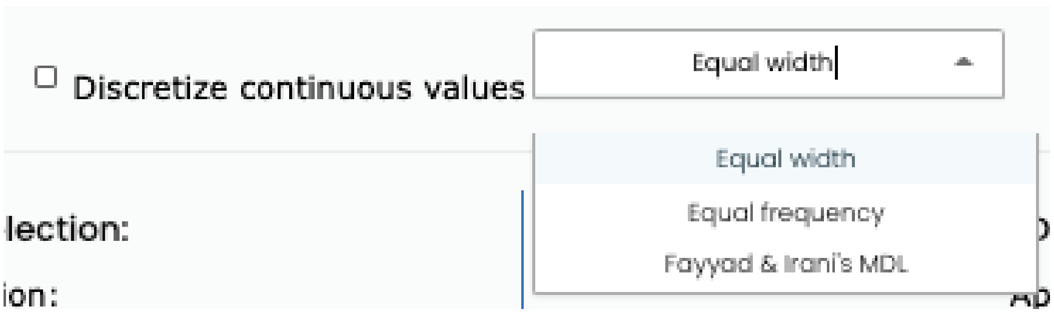


Figure 3.6: Values discretization

Parameter	Option	Description
Method	Chi squared	Applies the chi squared test to each feature
	Mutual information	Evaluates mutual information of each feature with the target
	ANOVA F score	Evaluates ANOVA F score of each feature with the target
Filter criteria	k best	Selects the k best features according the score or p-value
	Percentile	Selects the selected percentile of features
	Family wise error test	Selects the features that pass the test

Table 3.2: Feature subset selection parameters

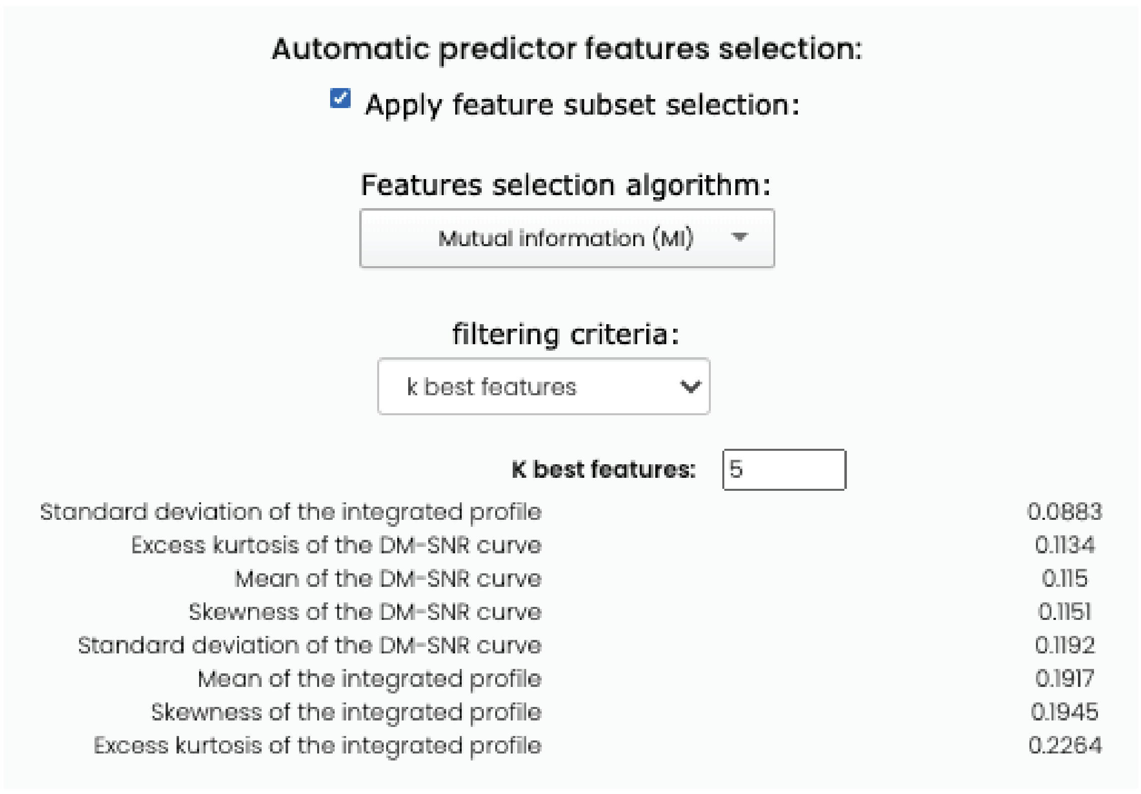
3.7 Feature subset selection

Predictor features can be relevant, redundant, or noisy in a dataset. From this idea, what features subset selection does is selecting a subset with the features that have a better impact than using the whole set of variables (Huang et al., 2015).

In NeuroSuites univariate filtering algorithms were implemented. These algorithms are part of Scikit-Learn, and like all the previous steps, it transforms the dataset and passes the result to the next step. This process receives two different parameters. The first one is the filtering method, and the second one is the filtering criterion. It could provide the best k variables or the n percentage of features ordered by the method. Additionally, this step shows the list of features ordered by its score according to the filtering method. The user can then have an idea of the impact of variables in the future model. Figure 3.7 shows the result of applying Feature Selection and Table 3.2 present the available parameters.

3.8 Dimensionality reduction

This method reduces the number of dimensions in a dataset. This reduction tries to keep the geometry or variation depending on the method used (**?**). With the application of this step, the source for learning algorithms tends to be simpler, and perhaps losing some information, the results may be better. A different use of dimensionality reduction is for analyzing the shape of the points in a lower dimensional space. An example of this is checking if the data is is clustered or mixed among categories according to its features. In NeuroSuites five algorithms were implemented, all of them


Automatic predictor features selection:

☑ Apply feature subset selection:

Features selection algorithm: Mutual information (MI)

filtering criteria: k best features

K best features: 5

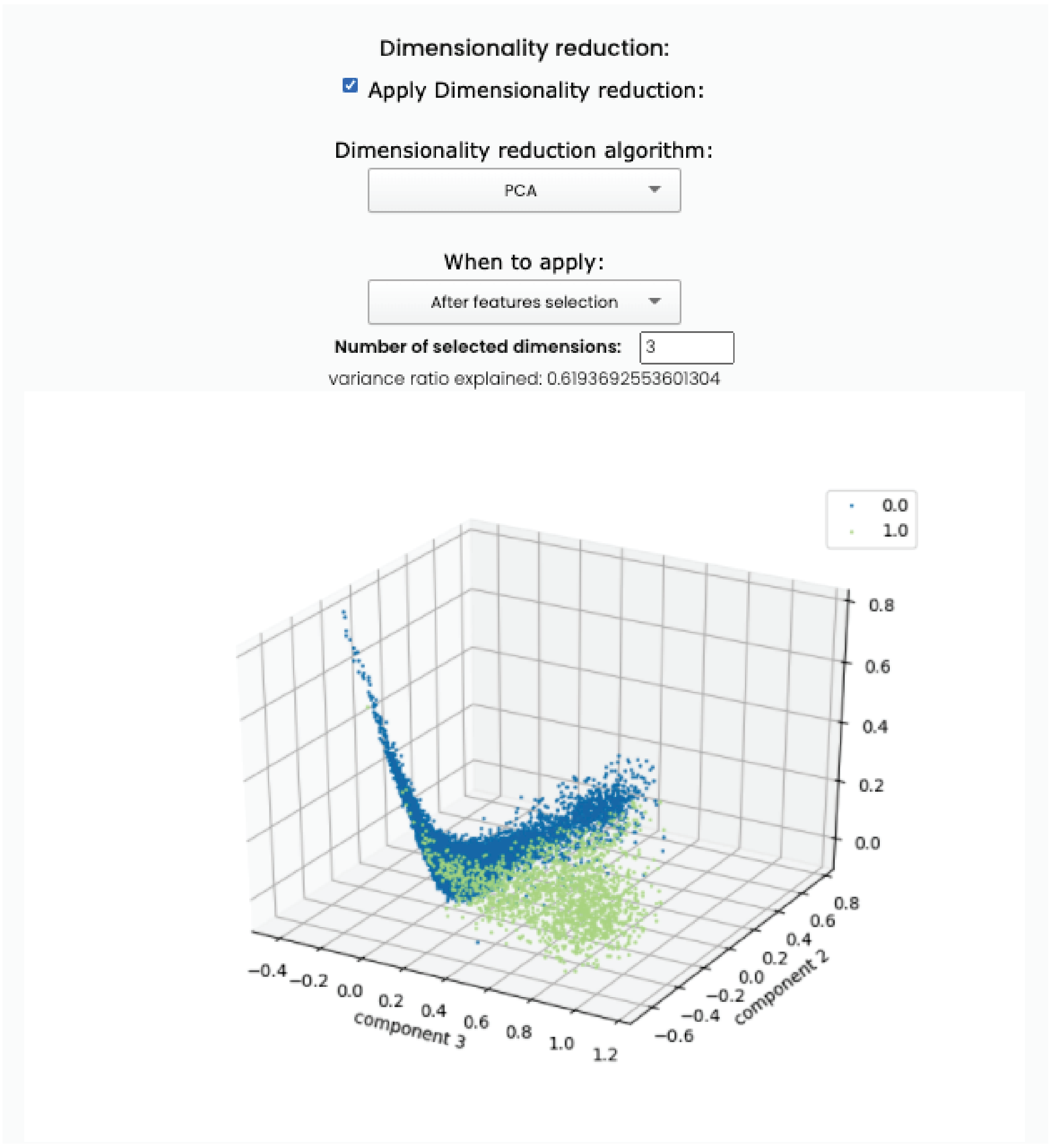
Standard deviation of the integrated profile	0.0883
Excess kurtosis of the DM-SNR curve	0.1134
Mean of the DM-SNR curve	0.115
Skewness of the DM-SNR curve	0.1151
Standard deviation of the DM-SNR curve	0.1192
Mean of the integrated profile	0.1917
Skewness of the integrated profile	0.1945
Excess kurtosis of the integrated profile	0.2264

Figure 3.7: Feature subset selection

available on Scikit-Learn. This process in NeuroSuites gives three results depending on the input parameters. Table 3.3 shows the parameters accepted and Figure 3.8 exposes the result of applying dimensionality reduction with PCA.

1. Lower dimension dataset
2. A plot of the dataset if 2 or 3 dimensions are selected.
3. Variance ratio explained if PCA is selected.

Additionally, this step is the only in NeuroSuites, which lets the user define when to be applied. Together with feature selection, these steps are active just for supervised machine learning with one target variable; otherwise with multi-dimensional data both are disabled.



Dimensionality reduction:

☑ Apply Dimensionality reduction:

Dimensionality reduction algorithm: PCA

When to apply: After features selection

Number of selected dimensions: 3

variance ratio explained: 0.6193692553601304

Figure 3.8: Dimensionality reduction

Parameter	**Option**
Algorithm	PCA
	Kernel PCA
	ICA
	t-SNE
	Multi-dimensional scaling
When	Before features selection
	After features selection
	Before and after features selection
Number of features	An integer number with the number of selected dimensions

Table 3.3: Dimensionality reduction parameters

Chapter 4

Implementation of non-probabilistic clustering in NeuroSuites

Clustering is the process of grouping data according to their similarities among features. It is part of unsupervised learning, which means that the learning process does not need a label or category. This family of algorithms helps to find patterns in the organization of samples in a dataset (Daelemans and Morik, 2008).

NeuroSuites included a module for applying probabilistic clustering, but there was a lack of an option for performing non-probabilistic clustering. This new functionality applies the previous pre-processing step from chapter 3 and a similar flow of data used in supervised classification. Nevertheless, this module is a completely new implementation because it includes the user interface design, the steps to be followed, and the whole algorithm implementations using Scikit-Learn.

4.1 Functionality flow

The flow is composed of three steps; each one has a set of options and parameters introduced by the user to achieve a result and analyze it. The list of actions associated to different tabs is:

1. Feature selection and pre-processing
2. Clustering algorithm
3. Clustering result

4.2 Implementation design

In the development process a non-probabilistc-clustering class was created that contains all the general functions available for the methods like as plotting a PCA graph of two dimensions. Then each algorithm has its own specific class implementation. NeuroSuites includes six different algorithms to choose. All the algorithms were taken from the Scikit-Learn library and embedded in the NeuroSuites classes.

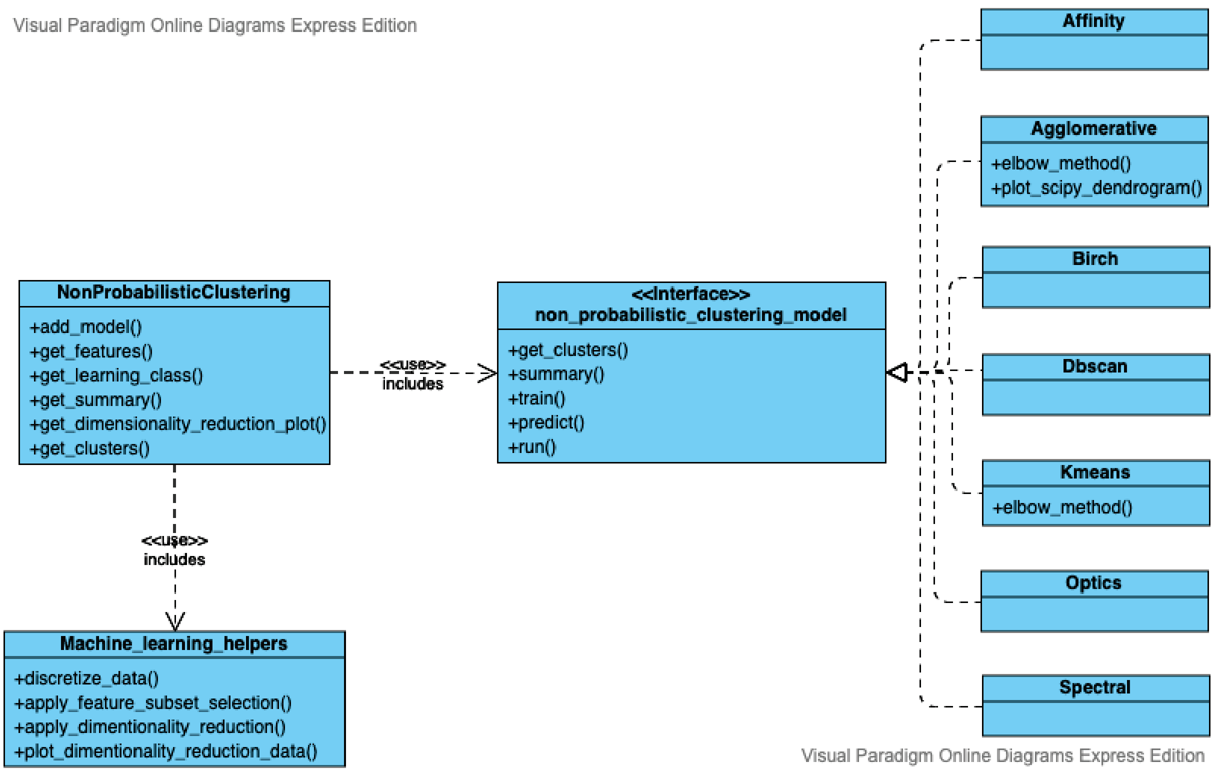


Figure 4.1: Non-probabilistic clustering UML class diagram

The principal class related to this objective and is responsible of the communication between the controller and the models is NonProbabilisticClass. This class access to the embedded methods through the interface non-probabilistic-clustering-model. With it that methods implemented by each specific algorithm are available to the embedding class. The mentioned interface is shown but not built because the Python syntax dealing with similar structures is not mandatory.

Additionally the Machine-learning-helpers class was reused in this context because it provides the PCA plot and is accessed from the NonProbabilisticClass class. Figure 4.1 shows the UML class diagrams the methods and relationships among the different classes related to this functionality.

Furthermore, is possible to see how Agglomerative clustering and Kmeans more functions because these two method offers more plots, the elbow-method for both and and dendrogram just for the first algorithm.

4.3 Feature selection and pre-processing

Regarding feature selection, a list of the dataset features is presented, and the user manually chooses those that will be part of the study. If a dataset was not added previously, a message would appear showing the link for uploading it to the platform.

Once the variables have been selected, four steps for pre-processing the data are available, all of them coming from the previous data-preprocessing functionality added in supervised classification:

1. Fill missing values
2. Remove constant features
3. Values normalization
4. Continuous values discretization

4.4 Algorithm selection

The principal tab inside this functionality is the selection of the learning algorithm. Each algorithm has its hyper-parameters and can be introduced by the user. One significant difference compared to the supervised classification module is that the user can select just one clustering algorithm. In contrast, on supervised classification, it is possible to select more than one. The available methods are listed:

1. Agglomerative clustering
2. K-means
3. DBSCAN
4. Spectral clustering
5. BIRCH
6. OPTICS

It is important to mention that the affinity propagation method was implemented, but due to its time-consuming solution, it was removed from the list of available options.

4.4.1 Agglomerative clustering

Agglomerative clustering is part of the category of hierarchical clustering, which means that it creates a hierarchical tree of clustered samples. With the obtained tree, a line (cutoff) divides the branches and with the upper side are taken the clusters that represent the groups (**?**). Figure 4.2 shows a screenshot with the corresponding parameters and Table 4.1 presents the available options.

4.4.2 K-means

K-means is part of the partitional clustering category of algorithms. It means that it does not generate a hierarchical tree. Instead, it aims to find partitions in the dataset that differentiates each cluster.

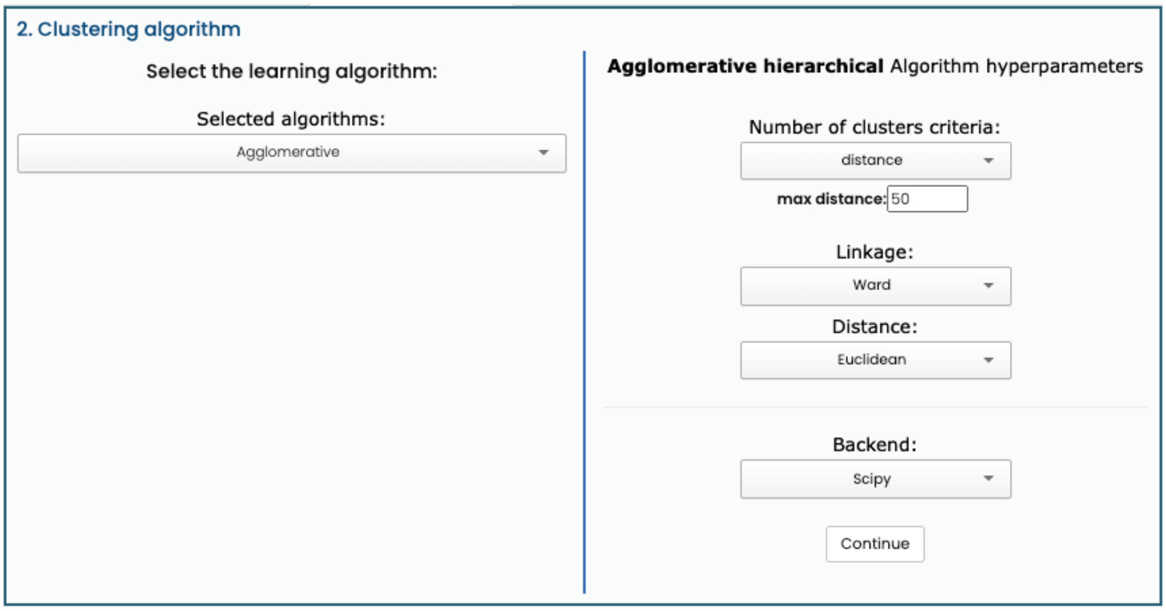
Therefore this clustering method creates partitions which are different according to each class variance, and the number of clusters is a parameter set by the user (MacQueen, 1967). The corresponding layout for this method is seen in Figure 4.3 and the once parameter in Table 4.1

4.4.3 DBSCAN

Density-Based Spatial Clustering (DBSCAN) tries to find groups with significant similarity. Basically what it does is counting the number of samples in a region within a

Parameter criterion	Option
Number of clusters	Distance
	K
Max distance	Input integer number representing the cutoff height over the dendrogram
Linkage	Ward's method
	Complete
	Average
	Centroid
	Single
Distance	Euclidean
	Hamming
	Manhattan
	Cosine

Table 4.1: Hierarchical agglomerative parameters
More information at: `https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html`


2. Clustering algorithm

Select the learning algorithm:

Selected algorithms: Agglomerative

Agglomerative hierarchical Algorithm hyperparameters

Number of clusters criteria: distance

max distance: 50

Linkage: Ward

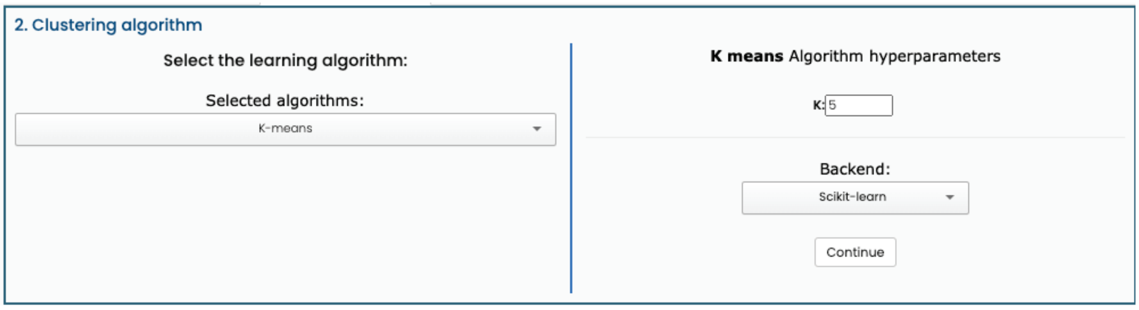
Distance: Euclidean

Backend: Scipy

Figure 4.2: Screenshot with the input of the agglomerative clustering parameters input

Parameter	Option
K	Input integer number representing the number of clusters

Table 4.2: K-means parameters
More information at: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`


2. Clustering algorithm

Select the learning algorithm:

Selected algorithms: K-means

K means Algorithm hyperparameters

K: 5

Backend: Scikit-learn

Figure 4.3: K-means clustering parameters input screenshot

Parameter	Option
Epsilon	The maximum distance between two samples
Min samples	The number of samples for considering a core point
Distance	Euclidean
	Manhattan
	Cosine

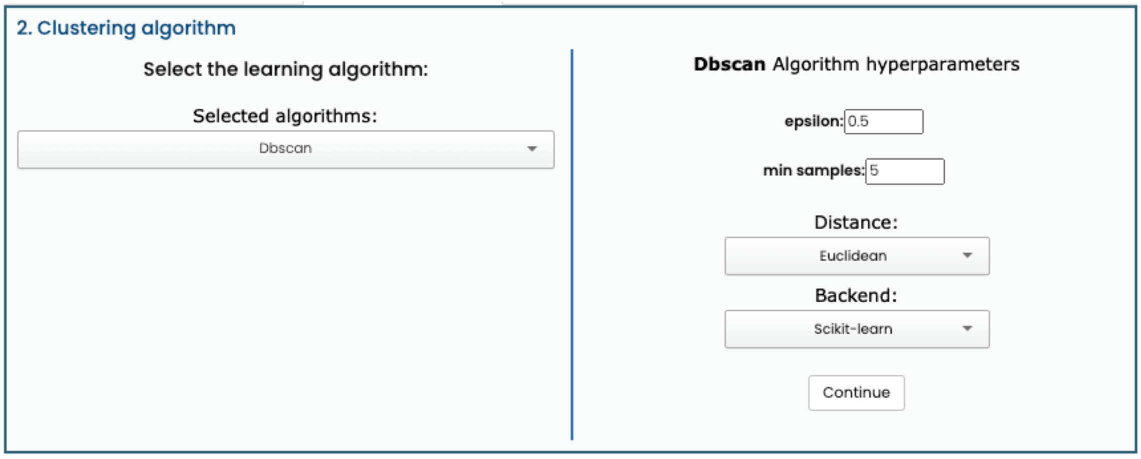
Table 4.3: DBSCAN parameters

More information at: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`

defined radius; those points with a density higher than a threshold are selected like clusters (Birant and Kut, 2007).

4.4.4 Spectral clustering

The spectral method is an option that appeared after K-means and offered a better performance depending on the problem. The process is based on using the top eigenvectors that come from a matrix that is derived from the existing distance among points. There are several approaches and discussions on how to decide which are the best list of eigenvectors (Ng et al., 2002). Figure 4.5 presents the visual elements related to this algorithm and Table 4.4.4 the corresponding parameters.


2. Clustering algorithm

Select the learning algorithm:

Selected algorithms: Dbscan

Dbscan Algorithm hyperparameters

epsilon: 0.5

min samples: 5

Distance: Euclidean

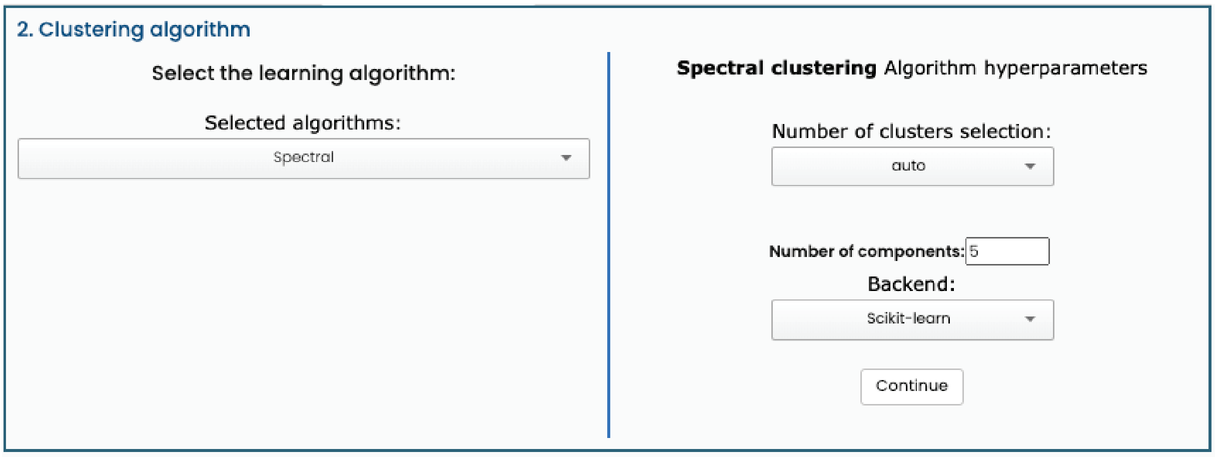
Backend: Scikit-learn

Figure 4.4: Screenshot with the input of the DBSCAN clustering parameters

Parameter	**Option**
Number of cluster selection	Automatic
	Manual
Number of clusters	Input integer number representing the number of clusters
Number of components	Number of eigenvectors to be used

Table 4.4: Spectral clustering parameters

More information at: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html`


2. Clustering algorithm

Select the learning algorithm:

Selected algorithms: Spectral

Spectral clustering Algorithm hyperparameters

Number of clusters selection: auto

Number of components: 5

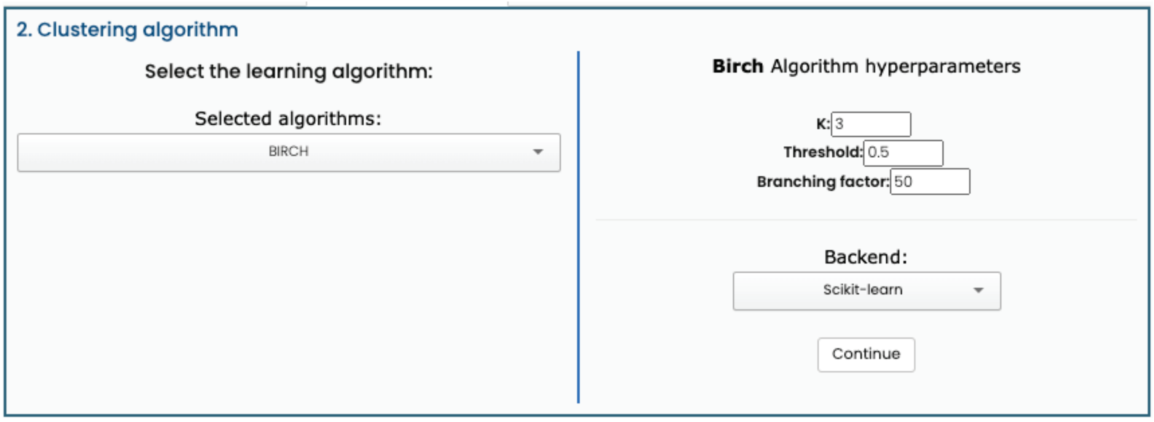
Backend: Scikit-learn

Figure 4.5: Screenshot with the input of the Spectral clustering parameters

Parameter	**Option**
K	Number of clusters
Threshold	The radius of the subcluster obtained by merging a new sample and the closest subcluster should be smaller than the threshold
Branching factor	Maximum number of subclusters in each node

Table 4.5: BIRCH parameters

More information at: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html`


2. Clustering algorithm

Select the learning algorithm:

Selected algorithms: BIRCH

Birch Algorithm hyperparameters

K: 3

Threshold: 0.5

Branching factor: 50

Backend: Scikit-learn

Figure 4.6: Screenshot with the input of the BIRCH clustering parameters

4.4.5 BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm was presented as a solution for processing large datasets in a more efficient way than other approaches. It is a good option for applying clustering with low computational resources. The internal process generates a more compact summary of the dataset containing the largest amount of distribution information that is possible. From this reduction comes the efficiency in the method that turns the problem cost into linear (Zhang et al., 1997). Figure 4.6 exposes a screenshot of the visual layout of this method and Table 4.4.5 the available parameters.

4.4.6 OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) provides an augmented ordering of the dataset clustering structure. Subsequently, this information will give as result a cluster analysis and, also, will show the intrinsic clustering structure. One crucial difference with other methods is that it does not need the number of clusters as an input parameter, being useful because usually this number changes in a significant way the final clusters, and this value is not always clear (Ankerst et al., 1999). Figure 4.7 shows the visual elements related and Table 4.6 details the parameters.

Parameter	Option
Min samples	The number of samples in a neighborhood for a point to be considered as a core point
Distance	Euclidean
	Manhattan
	Cosine

Table 4.6: OPTICS parameters

More information at: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html`

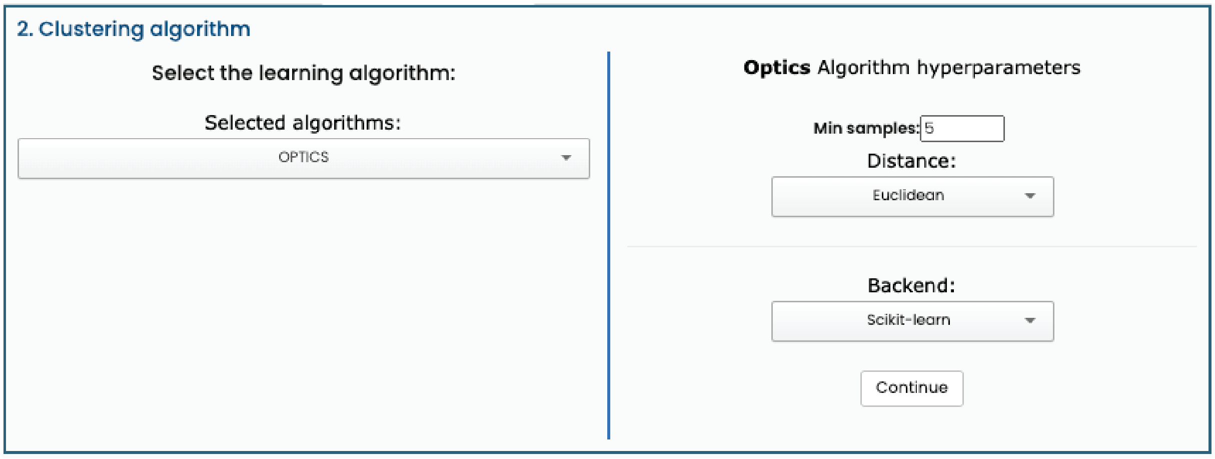


Figure 4.7: Screenshot with the input of the OPTICS clustering parameters

4.5 Results observation

Finally, the last tab is devoted to visualizing the resulting clusters. The layout contains three to five components depending on the selected clustering method. The main idea behind this last step is to make use of the obtained model through a set of resources for this purpose. The elements that compose this layout are presented below:

1. Cluster instances table
2. Elbow method plot (K-means and agglomerative hierarchical method)
3. Download button
4. Dendrogram (K-means)
5. Dendrogram (agglomerative hierarchical method)
6. PCA plot of two components

In the first place, the clusters instances table has three columns representing the number of the cluster, the number of instances, and the corresponding percentage according to the population. This table size depends on the elbow method plot. If it is present, the width will be half of the page; otherwise, if it is absent, the width will be the total page width.

Then, the elbow method represents the explained variance as a function of the number of clusters. The idea is that the best number is located in the "elbow" of the curve, helping to determine the best parameter for the method. As mentioned before this plot is available only for K-means and hierarchical agglomerative methods.

In all the methods there exists the option of downloading the resulting dataset in CSV format through the download button. The structure of the CSV file contains all the selected features and a final column with the label that corresponds to the instance's label. Internally it generates an AJAX request to the server, and the answer is the CSV file to the client for its download.

In the case of agglomerative hierarchical clustering, the algorithm generates a dendrogram representing the previously mentioned tree. when visualizing results, the corresponding dendrogram is shown as part of the plots. Due to a significant number of samples that could be part of the data, the dendrogram is truncated to a maximum of 16 branches for representing the clusters. The value line will be drawn over the graph if the cutting line is selected as an option.

In addition, the K-means method has a result centroids, which means the point located in the center of each cluster, so they act as the average description of the cluster that they belong. In NeuroSuites only K-means is possible to watch a table with the list of the centroids and the value over each variable.

Visualization of samples with more than three dimensions is impossible for humans. Therefore, the approach followed in NeuroSuites for this problem is plotting a graph of two dimensions. Both dimensions result from PCA application over the whole dataset, letting users observe the obtained clusters comprehensively. Each point represents a sample, and the color symbolizes the belonging cluster. This plot is available for all the clustering algorithms.

In conclusion, all these resources let the user receive more information than the resulting data by itself. Thus, the decisions over the parameters introduced in the model can be edited or even change the algorithm for a better performance with the uploaded data.

4.6 Case study

With the intention of comparing the results among all the options in algorithms implemented in NeuroSuites, in this section will be presented the obtained clusters table, the PCA plot for each method, and the time needed for the learning process.

The dataset used is Pulsar stars obtained from (Lyon et al., 2016) This data is the result of watching pulsar stars, a particular type of stars made of neutrons that produce radio emissions from each pole that can be detected from the Earth planet. The data has nine attributes and 17898 samples and is useful for classification or clustering purposes.

As pre-processing, three steps were applied, rows deletion for samples with null values, then removing constant features and finally the process of values standardization with max-min scaling.

To decide the number of clusters for the algorithms two tools were employed, One is the dendrogram provided by the hierarchical agglomerative method (Figure 4.8) being a methodology that is exposed as a recommendation in (Bielza and Larrañaga, 2020).

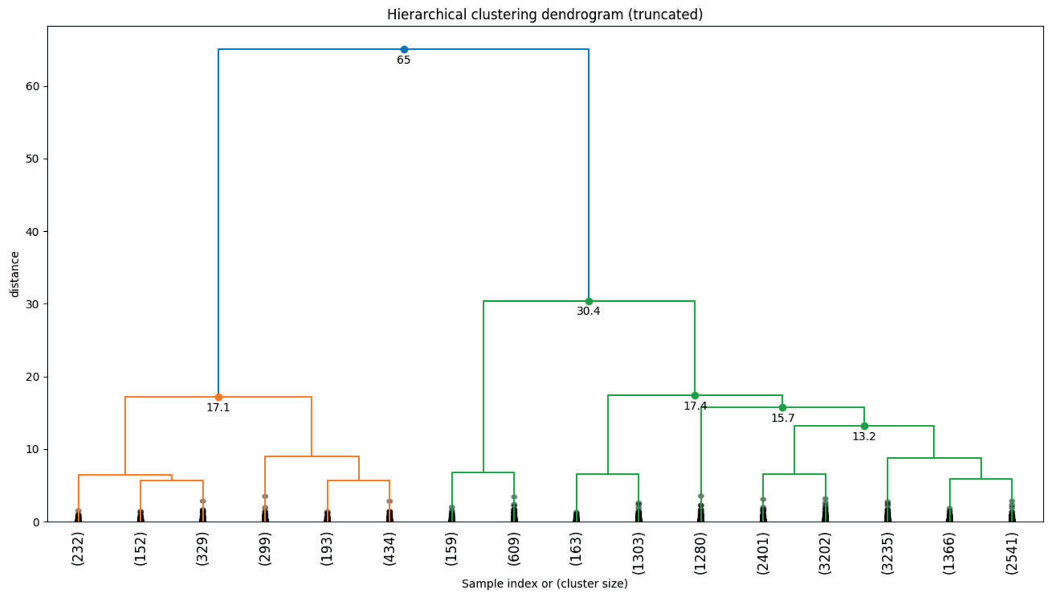


Figure 4.8: Obtained dendrogram from agglomerative hierarchical clustering

The second approach was using the elbow method taken from the K-means model displayed in Figure 4.9. Perhaps the best option in both cases is two clusters. In this case, three groups were selected in order to visualize the differences of each algorithm in a more detailed way. The hyperparameters introduced are visible in Table 4.7 and are the default ones except for spectral clustering, where minimum samples changed from five to twenty because otherwise, it was producing 315 clusters.

Figure 4.10 exhibits the resulting clusters generated by each model. Agglomerative hierarchical and K-means originated similar organizations with a cluster that is the same in both cases, and this group is present on DBSCAN and BIRCH as well. For two clusters, then the resulting output would be the same as found by the BIRCH model. DBSCAN generated three clusters, with one them composed by one instance; after reviewing this cluster graphically in fact, this sample is located far from the two principal groups at the top of the plot. In the case of spectral, the eight clusters are well separated, but the differences among them are not considerable. On the other hand, OPTICS delivered three clusters with one big group representing 99.8 percent of the samples; at the same time, the other two are not well separated and have 21 instances. A initial idea is that the location of these two sets inside the big set is because of the PCA projection, but if it was the case all the other algorithms should have shown similar problems, so in this case the OPTICS method with the given parameters is not accomplishing an acceptable result. The tables of the obtained clusters are available from Figure 4.11 to Figure 4.16.

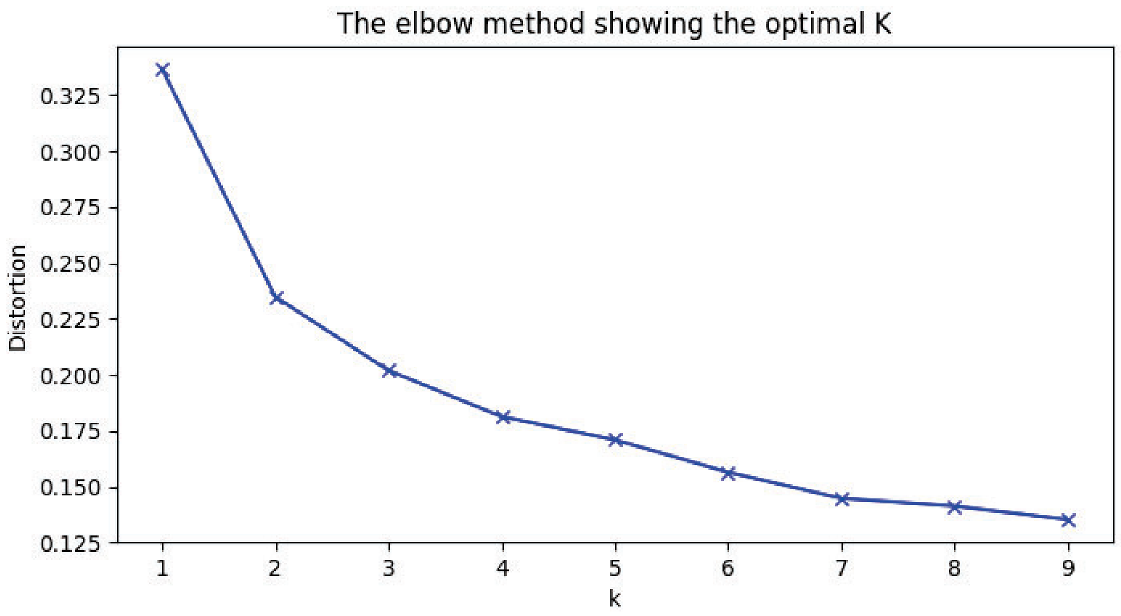
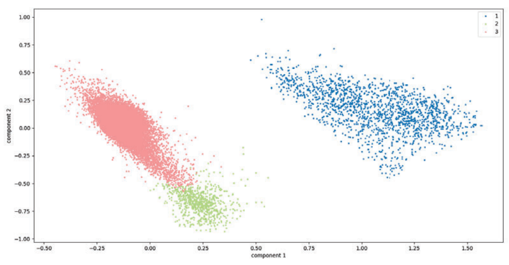


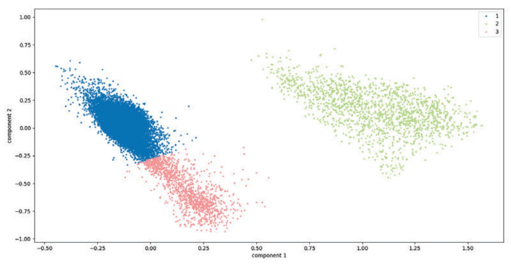
Figure 4.9: Elbow method applied to the K-means output

Method	**Parameters**	**Learning time (seconds)**	**Number of clusters**
Agglomerative clustering	Criteria: K, Number of clusters (K): 3, Linkage: Ward, Distance: Euclidean	16.20	3
K-means	K: 3	10.48	3
DBSCAN	Epsilon: 0.5, min samples: 5, Distance: Euclidean	5.66	3
Spectral clustering	Number of clusters selection: automatic, Number of components: 5	36.41	8
BIRCH	K: 3, Threshold: 0.5, Branching factor: 50	1.26	2
OPTICS	Min samples: 20, Distance: Euclidean	39.55	3

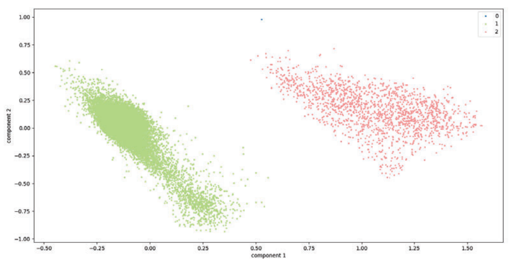
Table 4.7: Input parameters and learning times



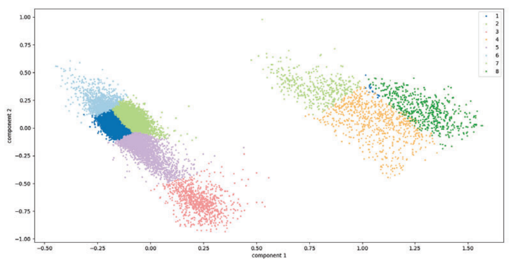
(a) Agglomerative hierarchical



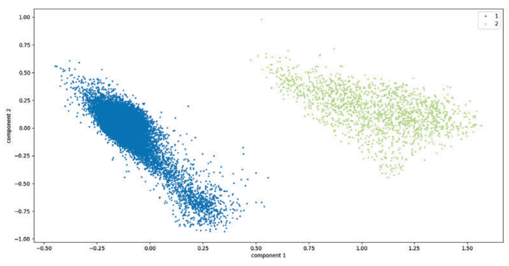
(b) K-means



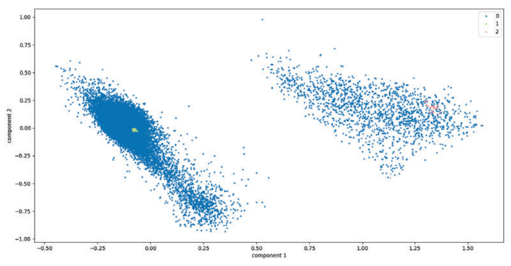
(c) DBSCAN



(d) Spectral

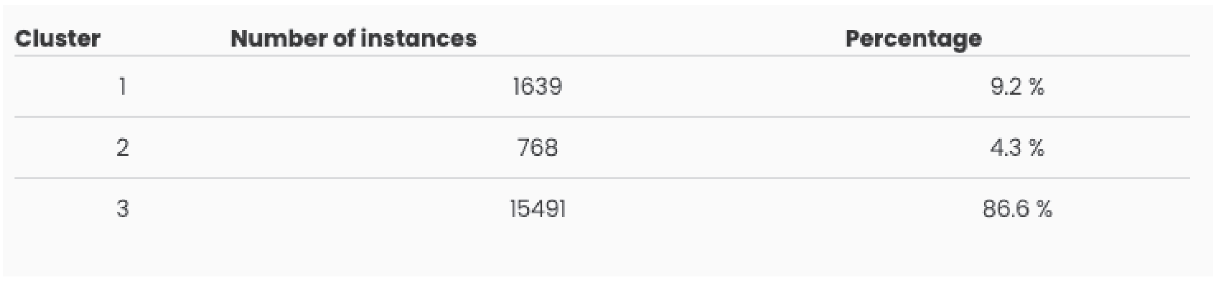


(e) BIRCH



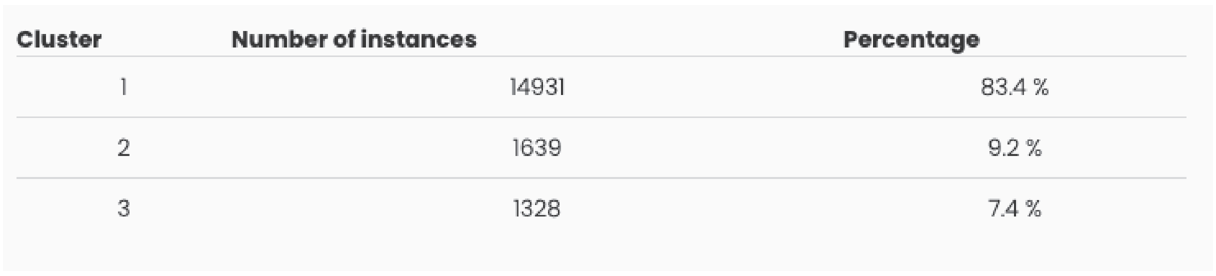
(f) OPTICS

Figure 4.10: Plots of the obtained clusters by method



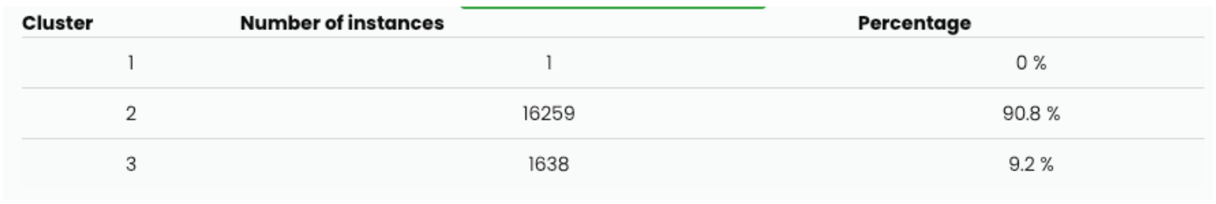
Cluster	Number of instances	Percentage
1	1639	9.2 %
2	768	4.3 %
3	15491	86.6 %

Figure 4.11: Agglomerative hierarchical clustering table result



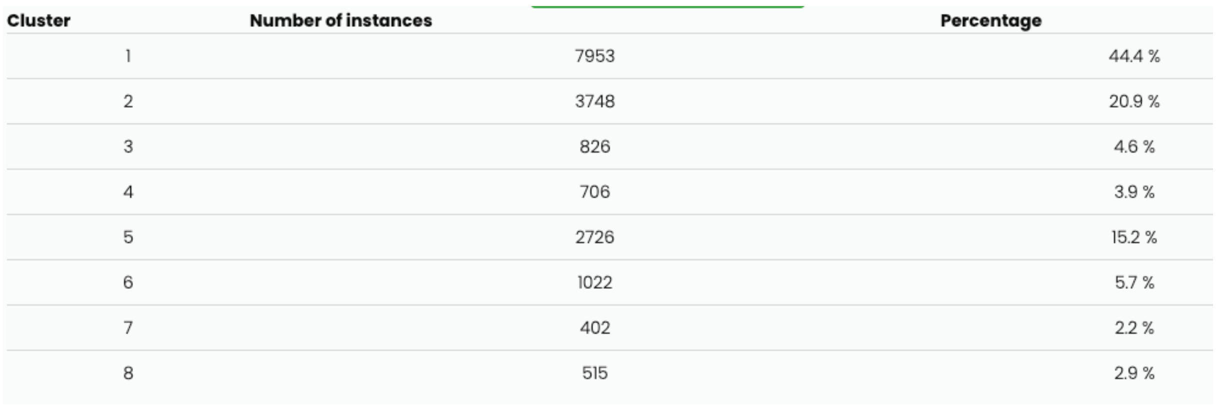
Cluster	Number of instances	Percentage
1	14931	83.4 %
2	1639	9.2 %
3	1328	7.4 %

Figure 4.12: K-means table result



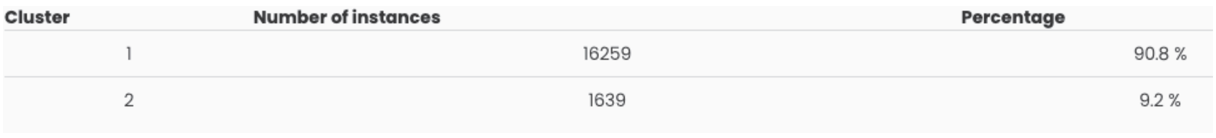
Cluster	Number of instances	Percentage
1	1	0 %
2	16259	90.8 %
3	1638	9.2 %

Figure 4.13: DBSCAN table result



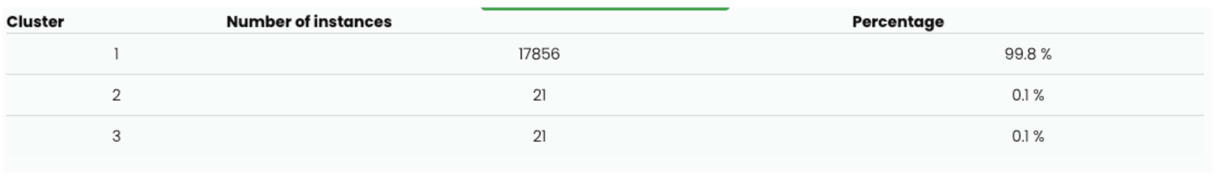
Cluster	Number of instances	Percentage
1	7953	44.4 %
2	3748	20.9 %
3	826	4.6 %
4	706	3.9 %
5	2726	15.2 %
6	1022	5.7 %
7	402	2.2 %
8	515	2.9 %

Figure 4.14: Spectral clustering table result



Cluster	Number of instances	Percentage
1	16259	90.8 %
2	1639	9.2 %

Figure 4.15: BIRCH table result



Cluster	Number of instances	Percentage
1	17856	99.8 %
2	21	0.1 %
3	21	0.1 %

Figure 4.16: OPTICS table result

Chapter 5

Implementation of multidimensional classification in NeuroSuites

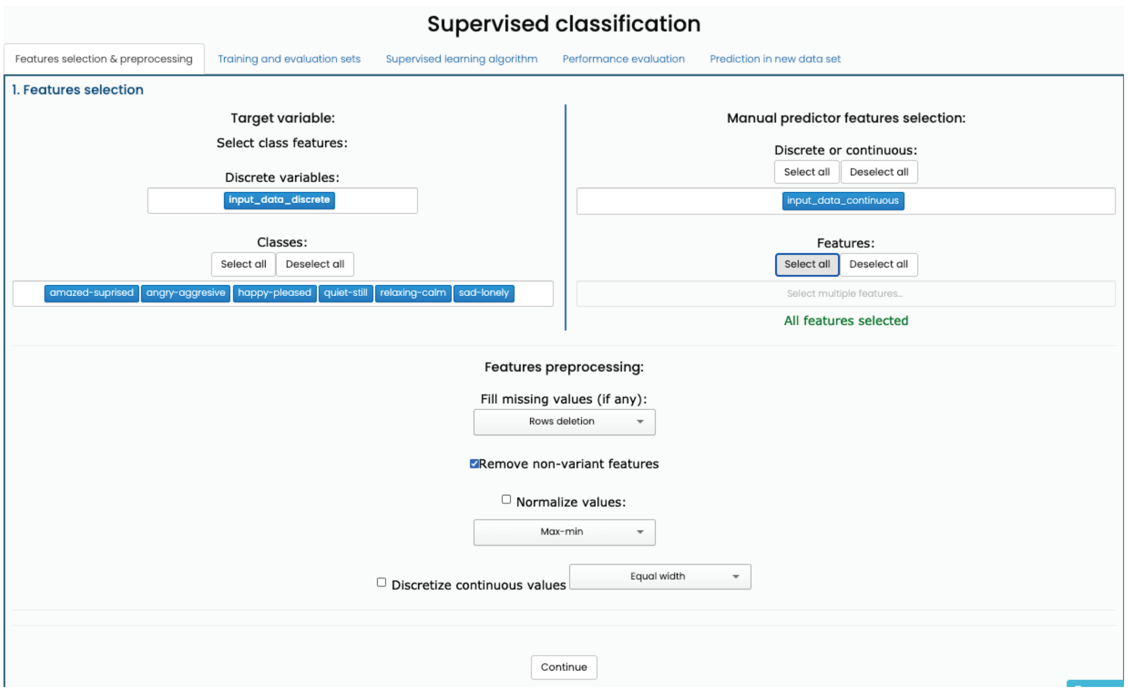
This feature is uncommon among machine learning platforms, being this a particular functionality inside NeuroSuites. This problem belongs to the supervised classification family, but instead of the usual prediction of one class variable, in this case, one or more target classes are simultaneously predicted.

There are two approaches to solving these problems. The first one is problem transformation that aims to transform a multi-label problem into one or more single-label problems (Bielza and Larrañaga, 2020). With this approach is possible to pass to data to a known classifier independently from the previous process. In Neurosuites four different problem transformation methods are available, all of them coming from the Scikit.ml. Finally is possible to embed some of the usual classification algorithms, but not all of them:

1. K nearest neighbors
2. Decision tree
3. Support vector machine
4. Linear discriminant analysis
5. Quadratic discriminant analysis
6. Logistic regression
7. Naive Bayes

The list of problem transformation methods is:

1. Binary relevance
2. Classifier chains
3. Label powerset
4. RAkEL



Supervised classification

Features selection & preprocessing
Training and evaluation sets
Supervised learning algorithm
Performance evaluation
Prediction in new data set

1. Features selection

Target variable:

Select class features:

Discrete variables: input_data_discrete

Classes: amazed-suprised, angry-aggresive, happy-pleased, quiet-still, relaxing-calm, sad-lonely

Manual predictor features selection:

Discrete or continuous: input_data_continuous

Features: Select multiple features...

All features selected

Features preprocessing:

Fill missing values (if any): Rows deletion

☑ Remove non-variant features

☐ Normalize values: Max-min

☐ Discretize continuous values: Equal width

Figure 5.1: Pre-processing step for multi-label classification (screenshot)

Secondly, the other approach is algorithm adaptation methods. It refers to the adaptation of an algorithm for working with multi-label data. NeuroSuites has two options for solving this kind of problem: k nearest neighbors and support vector machines; these both are part of the Scikit.ml package.

For solving this challenge, Scikit.mll was the solution used for adding this functionality. Scikit.mll is a Python Package created for handling multi-label problems. It has the code implementation of methods and metrics.

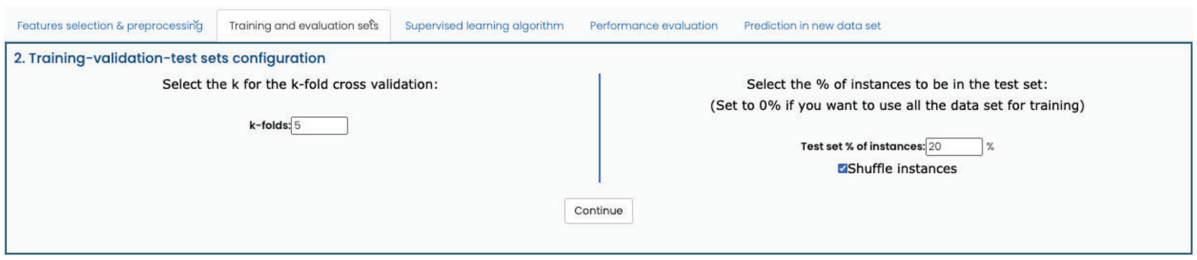
5.1 Functionality flow

The process for working with multi-label data is almost the same used in supervised classification. In fact, single-class and multi-label share the same option and screens. To work with common single-class classification the user should select just one target class. However, selecting choose more than one target class activates the multi-label classification functionality. Subsequently, the composing steps are the same:

1. Feature selection and preprocessing: The first step is the same explained in Chapter 3 and is able to accept more than one target class. The main difference with the usual process is that feature selection and dimensionality reduction are not present. Figure 5.1 shows a screenshot of the corresponding layout to the preprocessing step.

2. Training and testing sets: On this tab, the user selects the percentage of data used for training and testing the model. It also lets define the number of k-folds for cross-validation. In Figure 5.2 is shown the tab for setting the parameters for this step.

3. Supervised learning algorithm: On this step, the user selects the learning algo-


Features selection & preprocessing
Training and evaluation sets
Supervised learning algorithm
Performance evaluation
Prediction in new data set
Continue

2. Training-validation-test sets configuration

Select the k for the k-fold cross validation:

k-folds: 5

Select the % of instances to be in the test set:
(Set to 0% if you want to use all the data set for training)

Test set % of instances: 20 %

☑ Shuffle instances

Figure 5.2: Training and validation sets screenshot

rithm and passes the hyper-parameters needed by the model.

4. Performance evaluation: Once the model or models have been trained, this layout shows the evaluation metrics to the user. It also allows to download the predicted samples in two formats: CSV or Parquet gzip.

In conclusion, the main idea of this functionality is to make the use of these functions easier, taking advantage of the already developed solutions. In this case, the flow is the same used for normal classification problems.

5.2 Implementation design

In terms of composition this is the most complex module related to Machine Learning inside NeuroSuites 17 classes in total are related in way of accomplishing this functionality. The central and most important is SupervisedClassification, a class that can manage single labeled and multi labeled datasets. This class is responsible of dealing with the controller and the models. Supervised-classification-model is an interface that is not entirely created because of the Python syntax but help to describe all the methods that must be implemented by each one of the Multi-Label specific models. This class uses Supervised-classification-metrics, which is class that can deal with single and multi label data as well and has the responsability of calculating the different metrics.

For the problem-transformation methods, a single-label classification algorithm is needed and will be embedded inside the parent class (transformation method). All of these methods uses again the Supervised-classification-model but in this case to use single-label algorithms and be accessed from the embedding class. Figure 5.3 exposes the UML class diagram used to build this functionality.

Finally it is important to mention that all the methods used in NeuroSuites related to this module are taken from Scikit-multilearn Python library created for performing Machine Learning problems with Multi-label problems.

5.3 Algorithm selection

This section lists the algorithms that have been implemented as part of NeuroSuites. The structure is a brief description, a table with the parameters, and a screenshot of the corresponding layout.

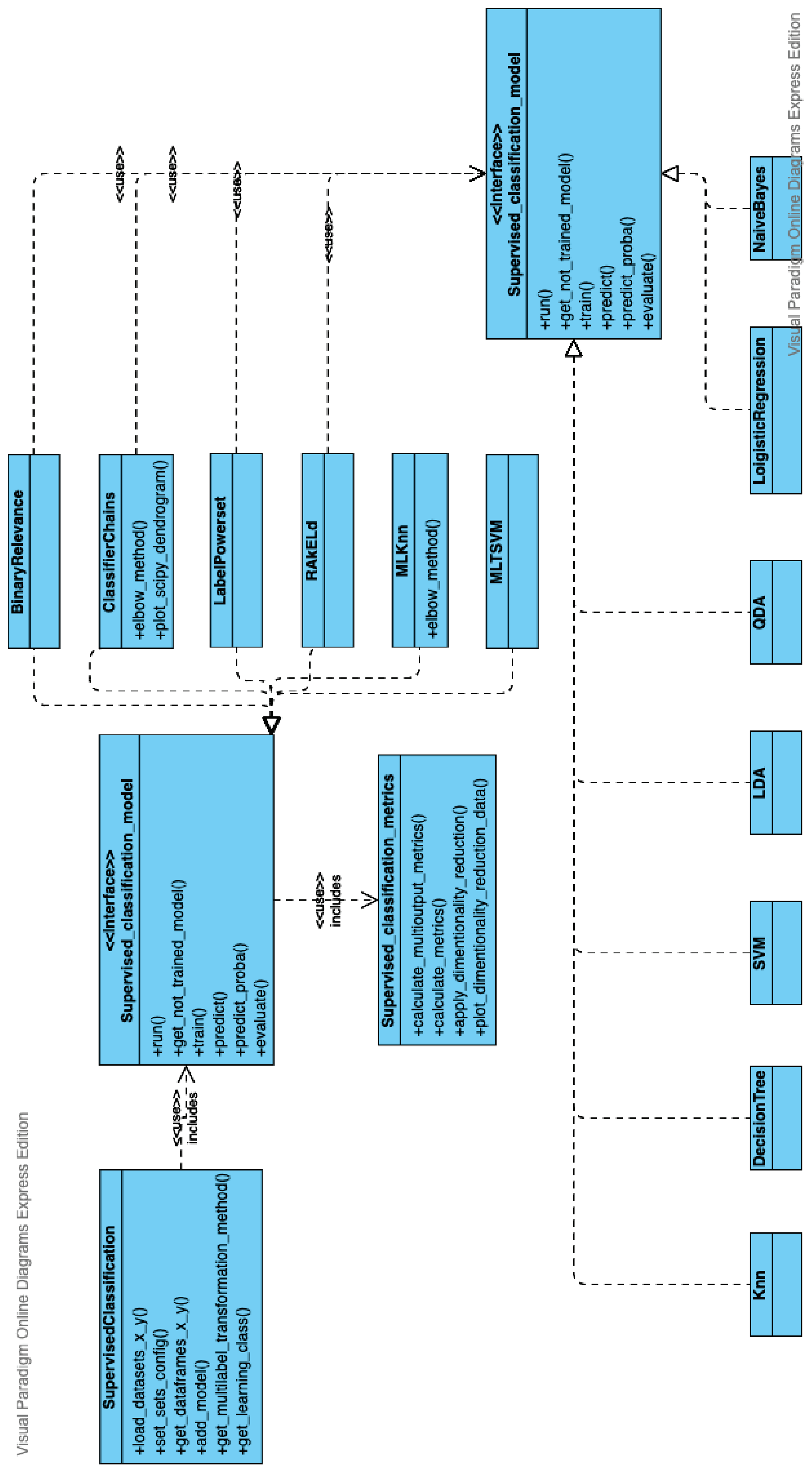
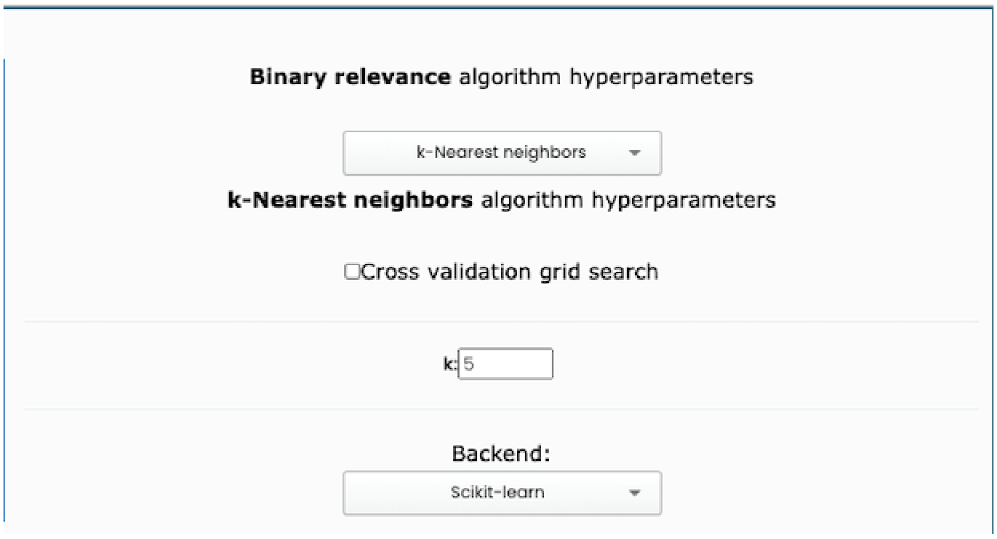


Figure 5.3: Multi-label performance evaluation screenshot (rotated image)

Parameter	**Option**
Classification algorithm	K nearest neighbors
	Decision tree
	Support vector machine
	Linear discriminant analysis
	Quadratic discriminant analysis
	Logistic regression
	Naive Bayes

Table 5.1: Binary relevance parameters

More information at: `http://scikit.ml/api/skmultilearn.problem_transform.br.html#skmultilearn.problem_transform.BinaryRelevance`



Binary relevance algorithm hyperparameters

k-Nearest neighbors

k-Nearest neighbors algorithm hyperparameters

☐Cross validation grid search

k: 5

Backend:

Scikit-learn

Figure 5.4: Screenshot with the input of the binary relevance parameters

5.3.1 Binary relevance

This method is usually known as the most intuitive approach since it decomposes the multi-label problem into several binary tasks. The inconvenience is the not to consider relationships between the labels (Zhang et al., 2018). Figure 5.4 displays the visual elements of this method, and Table 5.1 list the available options.

5.3.2 Classifier chain

This algorithm combines the computational efficiency of binary classifiers with the addition of relationships among the labels. The way it works is by creating a chain structure where, following a given order, for the class variables each predicted class variable turns into a prediction feature for the next target label (Read et al., 2009). Figure 5.5 shows the related layout to classifier chains, and Table 5.2 presents the list of accepted parameters.

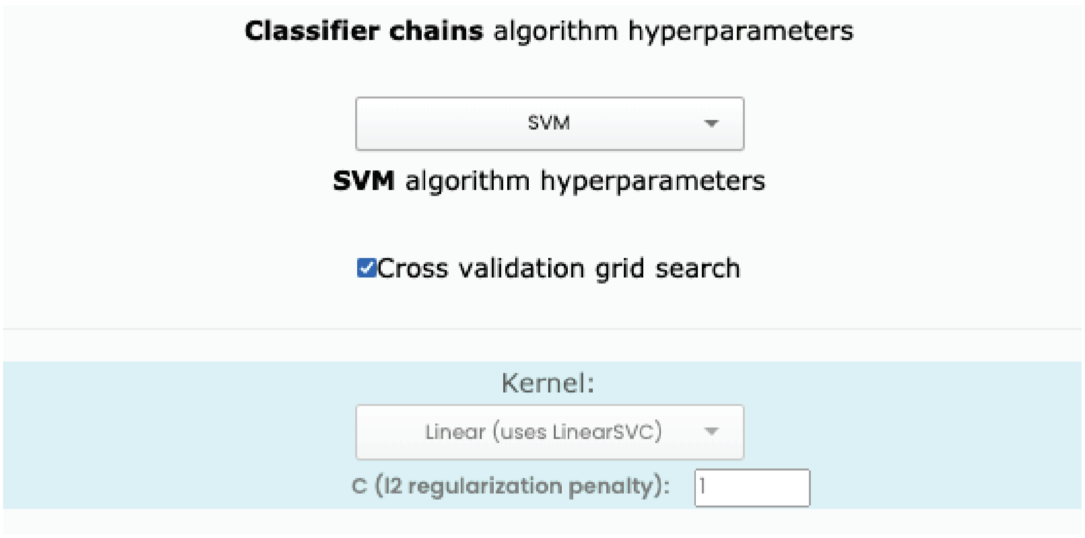
5.3.3 Label powerset

Label powerset transforms the problem into a multi-class classification task for training a model with all the combinations found in the dataset. One of the possible prob-

Parameter	**Option**
Classification algorithm	K nearest neighbors
	Decision tree
	Support vector machine
	Linear discriminant analysis
	Quadratic discriminant analysis
	Logistic regression
	Naive Bayes

Table 5.2: Classifier chain parameters

More information at: `http://scikit.ml/api/skmultilearn.problem_transform.cc.html#skmultilearn.problem_transform.ClassifierChain`



Classifier chains algorithm hyperparameters

SVM

SVM algorithm hyperparameters

☑ Cross validation grid search

Kernel: Linear (uses LinearSVC)

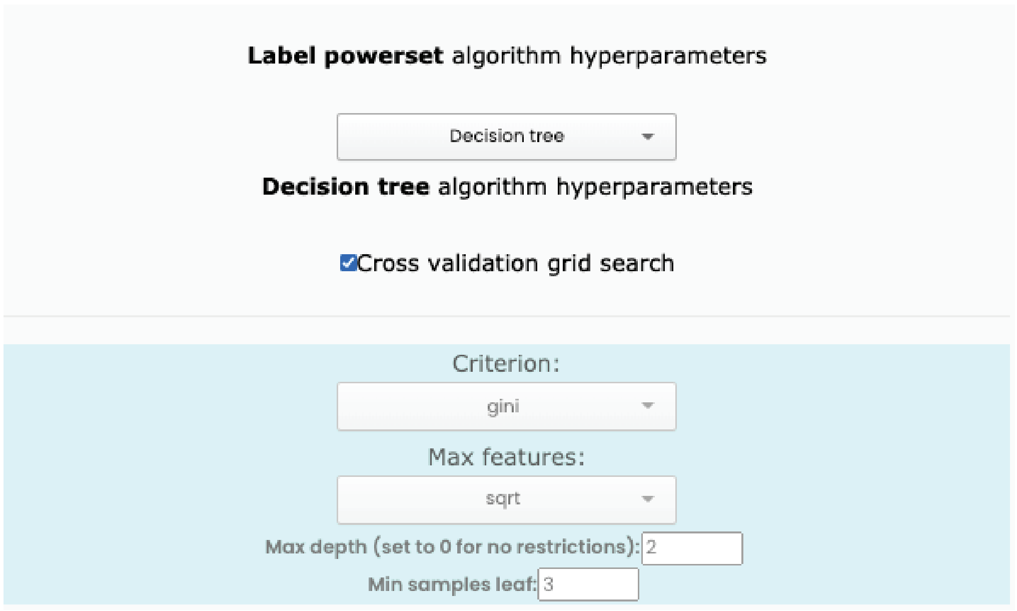
C (l2 regularization penalty): 1

Figure 5.5: Screenshot with the input of the classifier chain parameters

Parameter	**Option**
Classification algorithm	K nearest neighbors
	Decision tree
	Support vector machine
	Linear discriminant analysis
	Quadratic discriminant analysis
	Logistic regression
	Naive Bayes

Table 5.3: Label powerset parameters

More information at: `http://scikit.ml/api/skmultilearn.problem_transform.lp.html#skmultilearn.problem_transform.LabelPowerset`


Label powerset algorithm hyperparameters

Decision tree

Decision tree algorithm hyperparameters

☑ Cross validation grid search

Criterion: gini

Max features: sqrt

Max depth (set to 0 for no restrictions): 2

Min samples leaf: 3

Figure 5.6: Screenshot with the input of the label powerset parameters

lems detected is that if some combination is absent in the data, the model will not be able to predict this new class because each combination of labels becomes a unique label id (Szymański and Kajdanowicz, 2017). The visual layout is exposed in Figure 5.6 and the possible parameters are presented in Table 5.3.

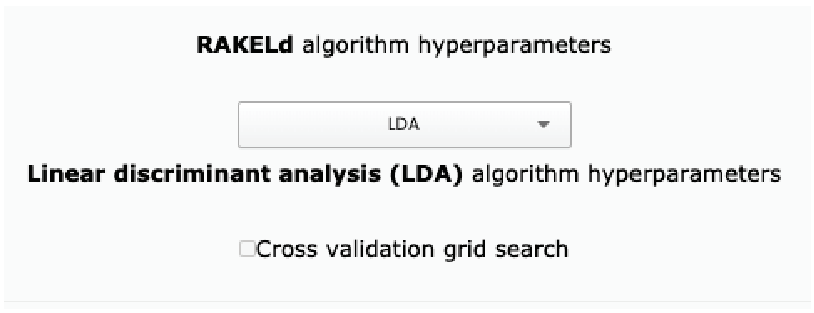
5.3.4 Random K-labEL sets

This method is based on label sets, but it creates k label set partitions in the dataset for generating a multi-class classifier per subset with all the combinations inside it. In the end, the prediction is derived from the votes of all the predicted values by each sub-model. This method has shown a better performance compared to other approaches. (Tsoumakas et al., 2011). A screenshot of the related visual elements is visible in Figure 5.7 and the available parameters are presented in Table 5.4

Parameter	Option
Classification algorithm	K nearest neighbors
	Decision tree
	Support vector machine
	Linear discriminant analysis
	Quadratic discriminant analysis
	Logistic regression
	Naive Bayes

Table 5.4: RAkEL parameters

More information at: `http://scikit.ml/api/skmultilearn.ensemble.rakeld.html#skmultilearn.ensemble.RakelD`


RAKELd algorithm hyperparameters

LDA

Linear discriminant analysis (LDA) algorithm hyperparameters

☐ Cross validation grid search

Figure 5.7: Screenshot with the input of the RAkEL parameters

5.3.5 Multi-label K-nearest neighbors

This method (MLKnn) and support vector machines belong to the model adaptation algorithms. MLKnn is based on the K-nearest neighbors algorithm, and the process is that for each unseen instance, the K nearest neighbors are identified; then Bayesian inference is applied in order to extract the assigned labels (Zhang and Zhou, 2007). This method only receives two parameters and are visually presented in Figure 5.8 and detailed in Table 5.5.

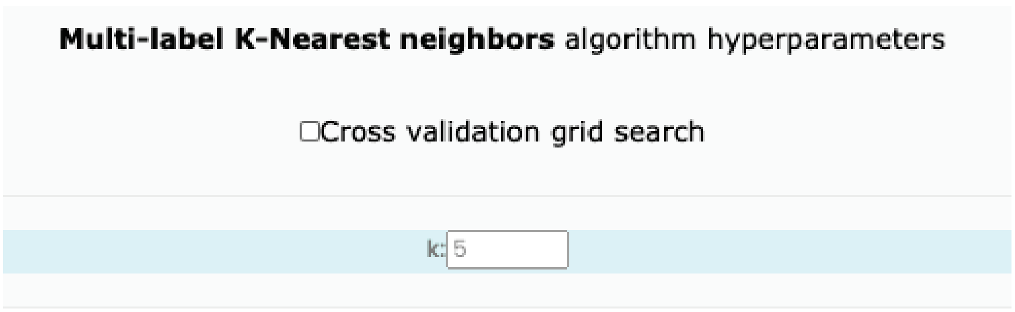
5.3.6 Multi-label Support vector machine

The internal process performed by multi-label support vector machines (MLTSVM) is to determine several nonparallel hyperplanes for capturing the multi-label information inside the data in order to maximize the margin or distance between two classes. The main goal with this method is that each pair of hyperplanes is closer to the corresponding class and as far as possible from the other one.(Chen et al., 2016). Details

Parameter	Option
Cross-validation grid search	yes
	no
K	integer that represents the number of neighbors

Table 5.5: Multi-label K nearest neighbors parameters

More information at: `http://scikit.ml/api/skmultilearn.adapt.mlknn.html#skmultilearn.adapt.MLkNN`


Multi-label K-Nearest neighbors algorithm hyperparameters

☐Cross validation grid search

k: 5

Figure 5.8: Screenshot with the input of the multi-label K nearest neighbors parameters

Parameter	**Option**
Cross-validation grid search	yes
	no
c	l2 regularization penalty: It represents the empirical risk penalty defined in order to determine a trade-off between the set of loss terms

Table 5.6: Multi-label Support vector machines

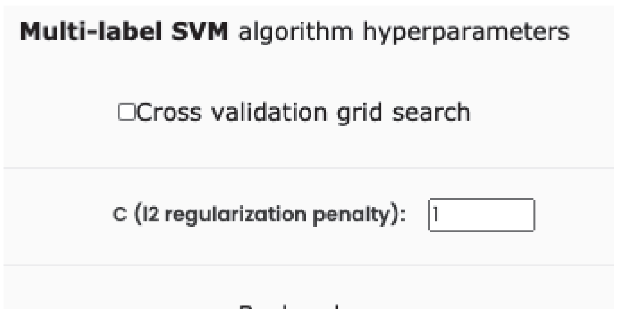
More information at: http://scikit.ml/api/skmultilearn.adapt.mltsvm.html#skmultilearn.adapt.MLTSVM

about the accepted parameters are available in Table 5.6 and the visual layout is shown in Figure 5.9.

5.4 Results observation

The corresponding tab for this functionality uses the same structure found with conventional single-class classification. In the software development process it included the creation of a new Python class that shares a similar behavior according to the internal methods for training and evaluating models between the existing and new processes. This new class has internally implemented the inherent methods in order to access to the data, functions and metrics from the selected algorithm.

The layout has two components, and each one is a table with the same columns and


Multi-label SVM algorithm hyperparameters

☐Cross validation grid search

C (l2 regularization penalty): 1

Figure 5.9: Screenshot with the input of the multi-label support vector machines parameters

rows. The first one shows information about the training set, whereas the second table corresponds to the testing set.

The rows correspond to each of the selected algorithms and the list of columns is:

1. Model
2. Hyperparameters
3. Download predictions in CSV format option
4. Download predictions in Parquet gzip format option
5. Model explanation option
6. Training time
7. Accuracy
8. Hamming loss
9. F-score
10. Precision
11. Recall
12. Jaccard similarity

The first two columns show the user's introduced options according to the model and parameters. The next two are buttons for downloading the resulting data with predictions, and the fifth column is a button for viewing a model explanation if is possible, depending on the model. For multi-label classification, there is not a model that can be explained graphically.

Then, from the sixth to the last option there are metrics that correspond with the performance of the algorithms, including the training time.

The tables have more possibilities, like ordering the samples by column or exporting the table to Excel format, CSV format, or pdf format, even copying the data to the clipboard. Another possible option is showing a statistical summary of the column values and four distinct plots:

1. Bar chart
2. Probability density functions using Gaussian kernel density estimation (KDE)
3. Box plot points
4. Scatter plot

5.5 Case study

For practically presenting the results, a case study for comparing all the six algorithms is shown in this section. In order to compare them, the tables generated by the application will be used including the metrics and training times.

The dataset used is Emotions obtained from Trochidis et al. (2008) and represents music and the associated emotions. Music uses to produce more than one feeling

Model	**Hyperparameters**
Binary relevance	'classification algorithm name': 'K-nearest neighbors'
Classifier chains	'classification algorithm name': 'logisticRegression'
Label powerset	'classification algorithm name': 'naiveBayes'
RAkEL	'classification algorithm name': 'svm'
MLKnn	'cross validation': True, 'K': 3
MLTSVM	'cross validation': True, 'c k': 0.125

Table 5.7: Parameters by model

Model	**Training time (s) (mean)**	**Accuracy (mean)**	**Hamming loss (mean)**	**F-score (mean)**	**Precision (mean)**	**Recall (mean)**	**Jaccard similarity (mean)**
Binary relevance	5.29	0.401	0.153	0.731	0.785	0.693	0.624
Classifier chains	15.217	0.409	0.175	0.733	0.726	0.757	0.639
Label powerset	0.008	0.435	0.185	0.698	0.715	0.705	0.614
RAkEL	12.552	0.481	0.142	0.771	0.769	0.787	0.691
MLKnn	10.725	0.511	0.117	0.806	0.824	0.791	0.718
MLTSVM	25.831	0.127	0.325	0.652	0.502	0.949	0.512

Table 5.8: Training metrics

making it an excellent example of multi-label classification. The descriptive features are related to the description of sounds, and all of them are continuous variables.

The pre-processing steps applied were a standardization through a max-min scale, which turns the vales from 0 to 1, removing constant features if any, and rows deletion in case of missing values inside a sample. The configuration for cross-validation, if applicable, is of K equal to 5, and the percentage of instances used for the test set 20 percent.

To decide the embedded algorithms for transformation models we previously tried all the options and the best model was selected for each one. For every embedded model and algorithm adaptation models cross-validation was used for automatically setting the best hyper-parameters. Table 5.7 exhibits the configurations applied.

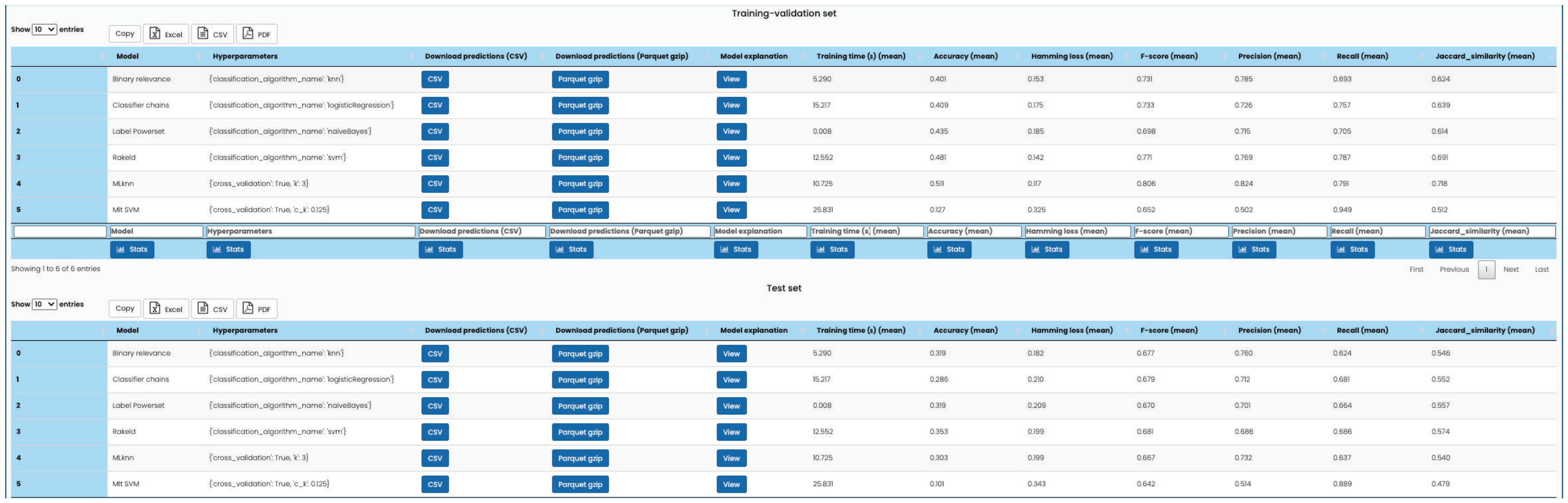
The training phase metrics are presented in Table 5.8. Note that MLKnn is the model with the best performance in terms of all the metrics and taking 10.725 seconds making it the third faster option. In terms of Hamming loss, Jaccard similarity and accuracy, binary relevance with an embedded K-nearest neighbor classifier together with random k-labelsets (Rakel) with a support vector machines algorithm are the second best options. Clearly MLTSVM is the model with the worst performing.

During the validation phase shown in Table 5.9 with the test dataset, the best performing model changed now. Instead of MLKnn, in terms of accuracy, F-score and

Model	**Training time (s) (mean)**	**Accuracy (mean)**	**Hamming loss (mean)**	**F-score (mean)**	**Precision (mean)**	**Recall (mean)**	**Jaccard similarity (mean)**
Binary relevance	5.29	0.319	0.182	0.677	0.76	0.624	0.546
Classifier chains	15.217	0.286	0.21	0.679	0.712	0.681	0.552
Label powerset	0.008	0.319	0.209	0.67	0.701	0.664	0.557
Random k-labelsets	12.552	0.353	0.199	0.681	0.686	0.686	0.574
MLKnn	10.725	0.303	0.199	0.667	0.732	0.637	0.54
MLTSVM	25.831	0.101	0.343	0.642	0.514	0.889	0.479

Table 5.9: Test metrics

Jaccard similarity random k-labelsets (RAkEL) is the best model. On the other hand, corresponding to Hamming loss and precision, binary relevance has the best results. MLTSVM has the highest score for recall, but together with the lowest accuracy and precision. It means that it is taking a significant number of cases as truth instead of filtering them correctly. The obtained tables are presented in the corresponding tab, and a screenshot has been made in order present it in the report in Figure 5.10.



Training-validation set

Show 10 entries

Copy | Excel | CSV | PDF

	Model	Hyperparameters	Download predictions (CSV)	Download predictions (Parquet gzip)	Model explanation	Training time (s) (mean)	Accuracy (mean)	Hamming loss (mean)	F-score (mean)	Precision (mean)	Recall (mean)	Jaccard_similarity (mean)
0	Binary relevance	{'classification_algorithm_name': 'knn'}	CSV	Parquet gzip	View	5.290	0.401	0.153	0.731	0.785	0.693	0.624
1	Classifier chains	{'classification_algorithm_name': 'logisticRegression'}	CSV	Parquet gzip	View	15.217	0.409	0.175	0.733	0.726	0.757	0.639
2	Label Powerset	{'classification_algorithm_name': 'naiveBayes'}	CSV	Parquet gzip	View	0.008	0.435	0.185	0.698	0.715	0.705	0.614
3	Rakeld	{'classification_algorithm_name': 'svm'}	CSV	Parquet gzip	View	12.552	0.481	0.142	0.771	0.769	0.787	0.691
4	MLknn	{'cross_validation': True, 'k': 3}	CSV	Parquet gzip	View	10.725	0.511	0.117	0.806	0.824	0.791	0.718
5	Mlt SVM	{'cross_validation': True, 'c_k': 0.125}	CSV	Parquet gzip	View	25.831	0.127	0.325	0.652	0.502	0.949	0.512
	Model	Hyperparameters	Download predictions (CSV)	Download predictions (Parquet gzip)	Model explanation	Training time (s) (mean)	Accuracy (mean)	Hamming loss (mean)	F-score (mean)	Precision (mean)	Recall (mean)	Jaccard_similarity (mean)
	Stats	Stats	Stats	Stats	Stats	Stats	Stats	Stats	Stats	Stats	Stats	Stats

Showing 1 to 6 of 6 entries

First Previous 1 Next Last

Test set

Show 10 entries

Copy | Excel | CSV | PDF

	Model	Hyperparameters	Download predictions (CSV)	Download predictions (Parquet gzip)	Model explanation	Training time (s) (mean)	Accuracy (mean)	Hamming loss (mean)	F-score (mean)	Precision (mean)	Recall (mean)	Jaccard_similarity (mean)
0	Binary relevance	{'classification_algorithm_name': 'knn'}	CSV	Parquet gzip	View	5.290	0.319	0.182	0.677	0.760	0.624	0.546
1	Classifier chains	{'classification_algorithm_name': 'logisticRegression'}	CSV	Parquet gzip	View	15.217	0.286	0.210	0.679	0.712	0.681	0.552
2	Label Powerset	{'classification_algorithm_name': 'naiveBayes'}	CSV	Parquet gzip	View	0.008	0.319	0.209	0.670	0.701	0.664	0.557
3	Rakeld	{'classification_algorithm_name': 'svm'}	CSV	Parquet gzip	View	12.552	0.353	0.199	0.681	0.686	0.686	0.574
4	MLknn	{'cross_validation': True, 'k': 3}	CSV	Parquet gzip	View	10.725	0.303	0.199	0.667	0.732	0.637	0.540
5	Mlt SVM	{'cross_validation': True, 'c_k': 0.125}	CSV	Parquet gzip	View	25.831	0.101	0.343	0.642	0.514	0.889	0.479

Figure 5.10: Multi-label performance evaluation screenshot (rotated image)

Chapter 6

Visual changes for NeuroSuites

One component that is part of every platform is visual design. This characteristic has the impact of attracting more or fewer users, depending on the ease of use or an attractive user interface. Another objective of this project was to give a better look to the application in terms of aesthetics and usability.

In NeuroSuites some existing modules were changed, and for the newly added ones the flow in terms of functionality was applied. Then, more changes were applied to the colors and logo of the application. For a better explanation, each change is described in a separate section in this chapter.

6.1 Logo

One of the first impressions received after watching a new application is the icon or logo. The previous logo had a descriptive but not friendly image. It included three overlapped components: a brain with a irregular pattern of blue color, a cloud without background, and yellow lightning. Due to the complexity, it was not giving a modern impression, that was the reason for designing a new logo with a fresher image for presenting the project.

The new logo only contains the name and a cloud linked to the last letter. The colors used were selected to be consequent with the new menu, in blue scales. Additionally a second option was created for dark backgrounds. The previous logo is visible in Figure 6.1 and the two new versions are in Figure 6.2.

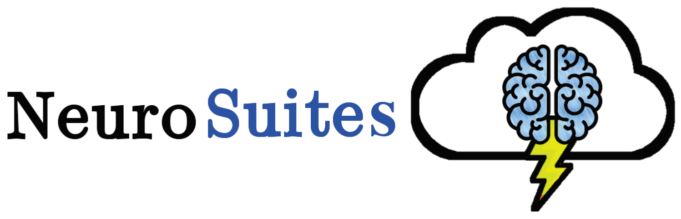
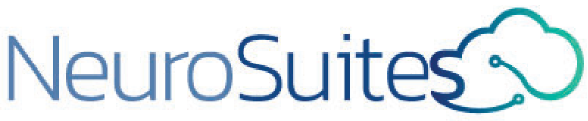


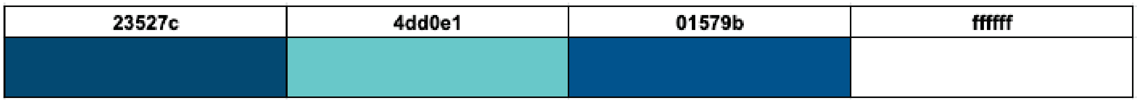
Figure 6.1: Previous logo



(a) Logo for light background

(b) Logo for dark background

Figure 6.2: New logo



23527c	4dd0e1	01579b	ffffff
🟦	🟦	🟦	⬜

Figure 6.3: New colors used for NeuroSuites

6.2 Menu and colors

With the previous palette, the logo and menu did not harmonize, as seen in the Figure 6.6. While, the logo was not including any purple color, the menu had purple as the main color. To obtain a harmonic combination the **Google Material Design color tool**[1] was used. The list of colors in hexadecimal notation decided after the recommendations were (see Figure 6.3):

1. 23527c
2. 4dd0e1
3. 01579b
4. ffffff

For the main menu the electric blue for the selected option was replaced by the new main blue color: 23527c at the same time that the logo was changed. On the other hand, the side menu purple background color was replaced by 23527c. Aditionally a new grouping organization was added according to the functionality that a module provides, being all the neuroscience related modules: 3DspineS, 3dSomaMS , 3DSynapsesSA and Dendrite arborization simulation grouped in the "Neuro Apps" item. The previous and new main menus are presented in Figure 6.4 and FIgure 6.5, and the previous and new side menus are shown in Figure 6.6.

6.3 Upload data page

Previously (see Figure 6.7), the page only included the instructions in textual format, which was not intuitive for a new user. Furthermore, the list of available modules was shown without filters or classifications, making it harder to categorize the modules.

[1]https://material.io/resources/color/

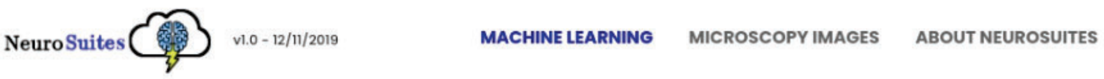


Figure 6.4: Previous main menu

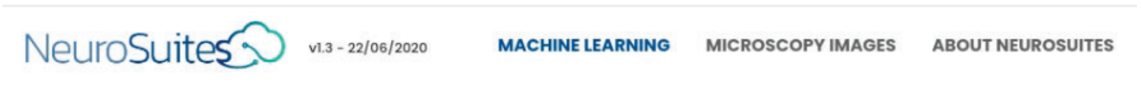
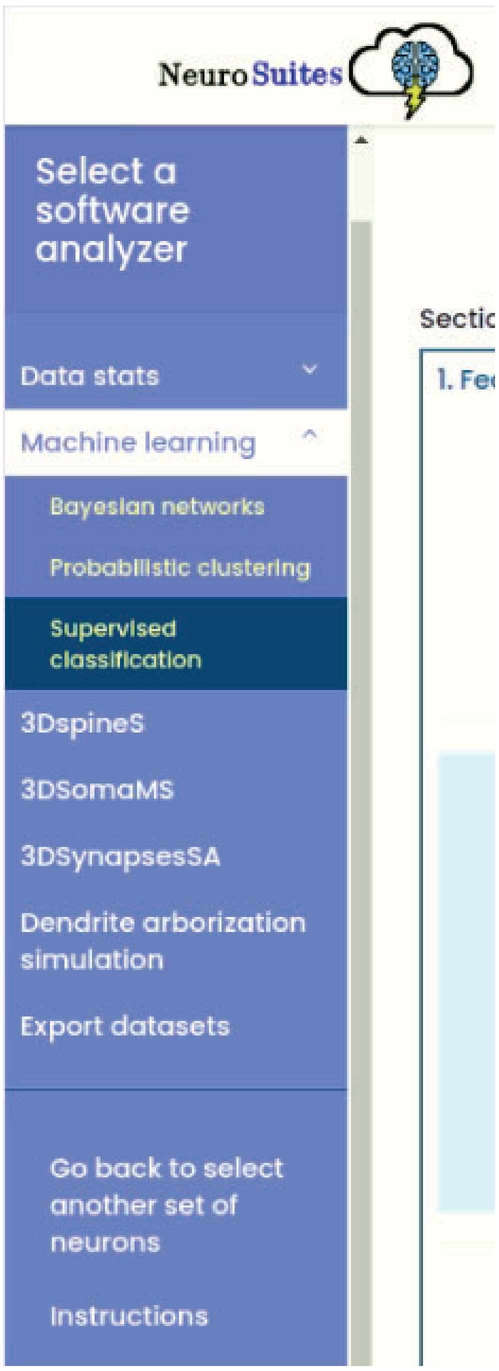
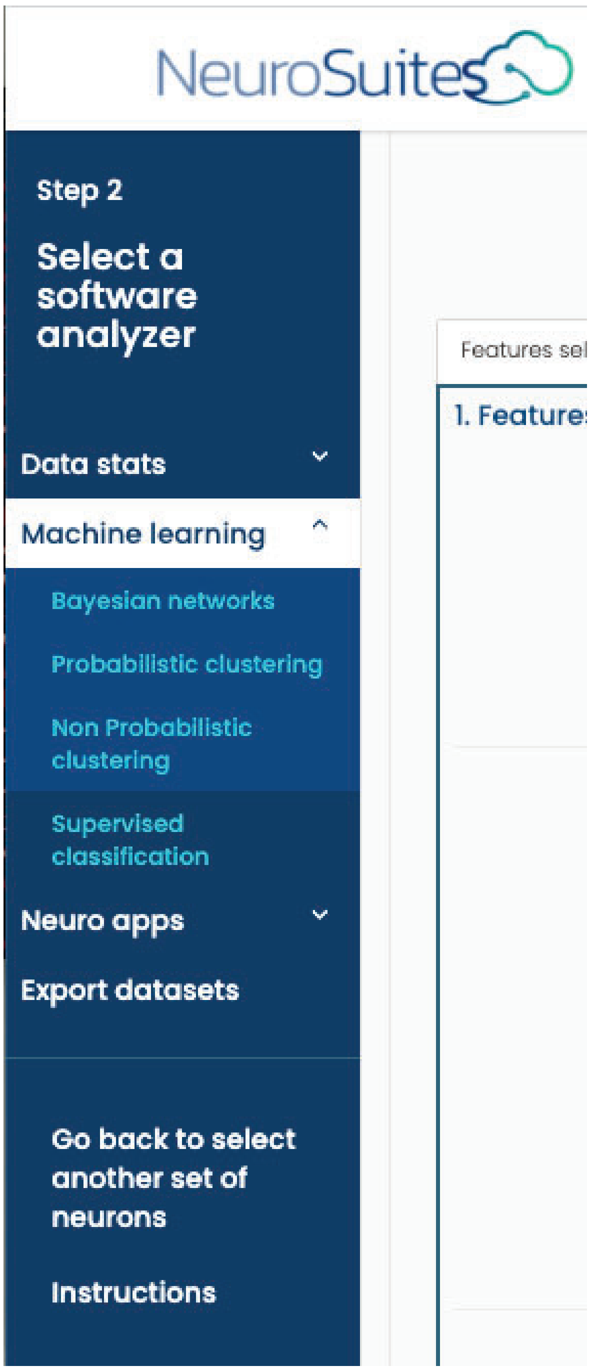


Figure 6.5: New main menu

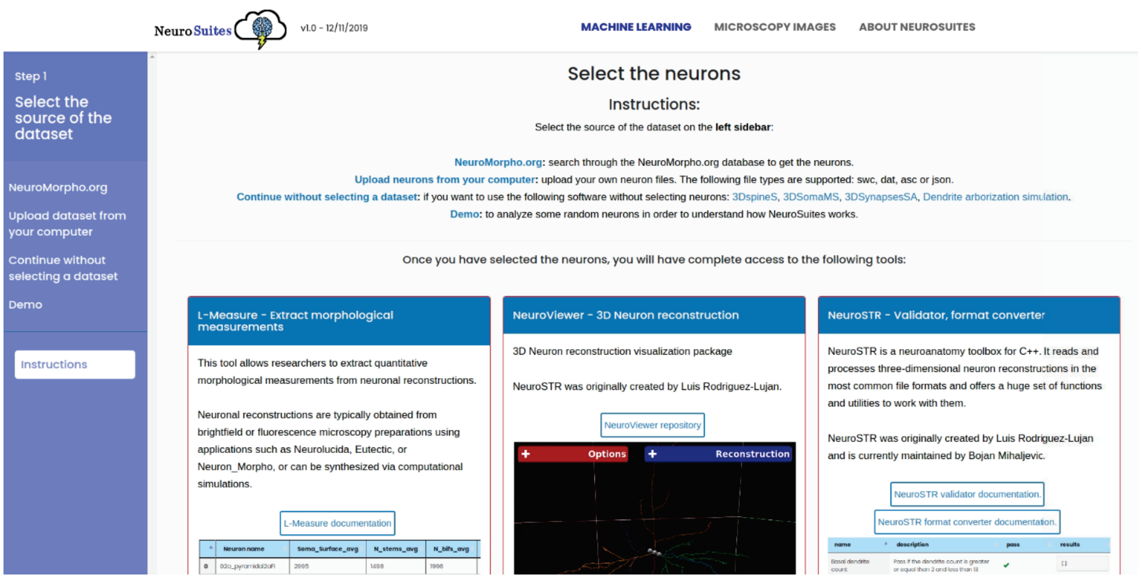


(a) Previous side menu



(b) New side menu

Figure 6.6: Previous and new side menu layouts



Select the neurons

Instructions:

Select the source of the dataset on the **left sidebar**:

NeuroMorpho.org: search through the NeuroMorpho.org database to get the neurons.
Upload neurons from your computer: upload your own neuron files. The following file types are supported: swc, dat, asc or json.
Continue without selecting a dataset: if you want to use the following software without selecting neurons: 3DspineS, 3DSomaMS, 3DSynapsesSA, Dendrite arborization simulation.
Demo: to analyze some random neurons in order to understand how NeuroSuites works.

Once you have selected the neurons, you will have complete access to the following tools:

L-Measure - Extract morphological measurements

This tool allows researchers to extract quantitative morphological measurements from neuronal reconstructions.

Neuronal reconstructions are typically obtained from brightfield or fluorescence microscopy preparations using applications such as Neurolucida, Eutectic, or Neuron_Morpho, or can be synthesized via computational simulations.

	Neuron name	Soma_Surface_avg	N_stems_avg	N_bifs_avg
0	02a_pyramidal2aFI	2995	1498	1996

NeuroViewer - 3D Neuron reconstruction

3D Neuron reconstruction visualization package

NeuroSTR was originally created by Luis Rodriguez-Lujan.

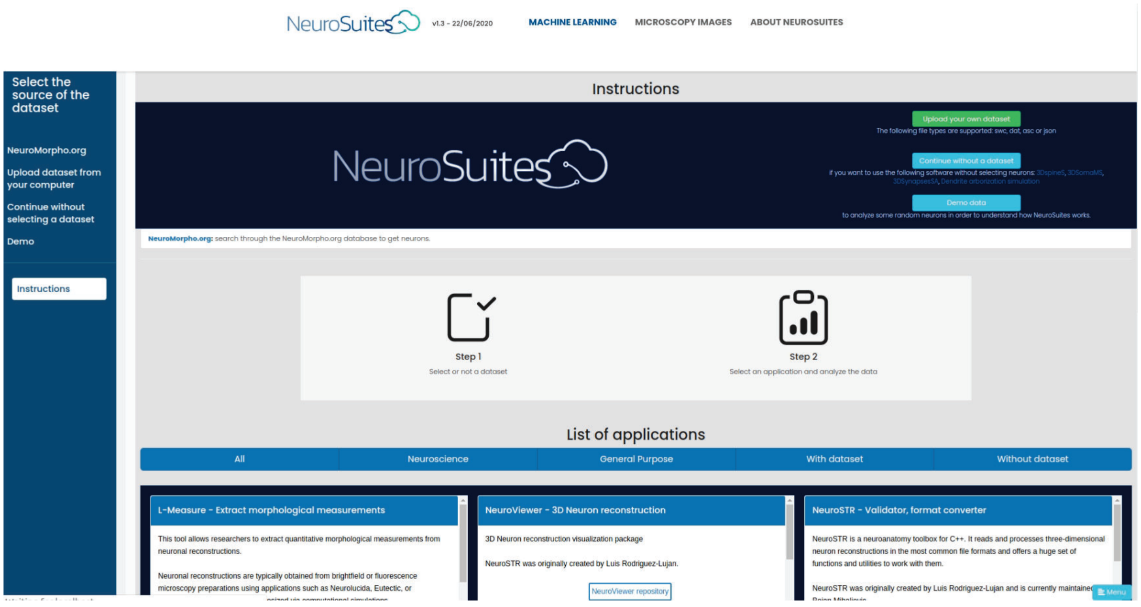
NeuroSTR - Validator, format converter

NeuroSTR is a neuroanatomy toolbox for C++. It reads and processes three-dimensional neuron reconstructions in the most common file formats and offers a huge set of functions and utilities to work with them.

NeuroSTR was originally created by Luis Rodriguez-Lujan and is currently maintained by Bojan Mihaljevic.

name	description	pass	results
Basal dendrite count	Pass if the dendrite count is greater or equal than 2 and less than 13	✓	[]

Figure 6.7: Previous upload page layout



Instructions

The following file types are supported: swc, dat, asc or json

if you want to use the following software without selecting neurons: 3DspineS, 3DSomaMS, 3DSynapsesSA, Dendrite arborization simulation

to analyze some random neurons in order to understand how NeuroSuites works.

NeuroMorpho.org: search through the NeuroMorpho.org database to get neurons.

Step 1
Select or not a dataset

Step 2
Select an application and analyze the data

List of applications

L-Measure - Extract morphological measurements

This tool allows researchers to extract quantitative morphological measurements from neuronal reconstructions.

Neuronal reconstructions are typically obtained from brightfield or fluorescence microscopy preparations using applications such as Neurolucida, Eutectic, or

NeuroViewer - 3D Neuron reconstruction

3D Neuron reconstruction visualization package

NeuroSTR was originally created by Luis Rodriguez-Lujan.

NeuroSTR - Validator, format converter

NeuroSTR is a neuroanatomy toolbox for C++. It reads and processes three-dimensional neuron reconstructions in the most common file formats and offers a huge set of functions and utilities to work with them.

NeuroSTR was originally created by Luis Rodriguez-Lujan and is currently maintained

Figure 6.8: New upload page layout

Alternatively, the new layout (see Figure 6.8) includes two steps graphically with descriptive text, and with the different options inside buttons. The list now contains a header with working filters for the applications according to five criteria depending on the functionality and the need of a dataset:

1. All
2. Neurosience
3. General purpose
4. With dataset
5. Without dataset

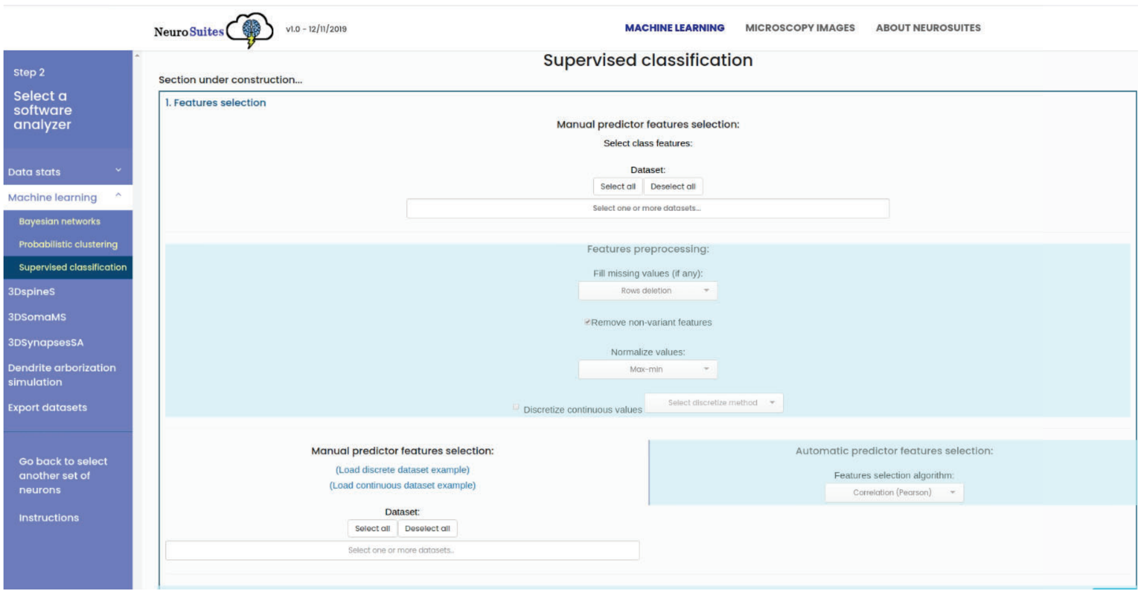
6.4 Supervised classification

The supervised classification layout was changed to add pre-processing and multi-label classification, in fact, the layout for showing the steps was intervened too. Before the new version, the page was spread in a single tab, being the scroll the only option for changing the current step of the process. It made it messy to reach a configuration parameter. In order to give more organization and information about the actual step, the layout was divided into four tabs ordered by the learning process, making it more intuitive and faster to navigate through the flow. These tabs represent each step to achieve the classification models that were described previously. A comparison between the previous and new Classification layouts is visible in Figures 6.9 and 6.10 respectively.

6.5 Statistics module

This module is a separated functionality from machine learning. At the same time, it shares the same uploaded dataset. Its functionality is complementary for the machine learning process, including several graphs and hypothesis tests. The problem found here is the same as in supervised classification; all the elements are distributed on the same single page with scrolling as the unique navigation option. The issue, in this case, is more noticeable because the space used by graphs and the number of options and combinations among variables are significant, making it easy for the user to get lost in the layout. The statistics module is composed of two different sub-modules: discrete statistics and continuous statistics, each one showing information about the corresponding kind of variables.

The solution implemented for both cases is dividing the whole layout into tabs according to its function inside the process. Each tab has sub-tabs in case of being necessary as the case of discrete descriptive statistics with univariate and bivariate statistics. The previous and new continuous statistics layouts are presented in Figures 6.11 and 6.12. Additionally the previous and new s discrete statistics layouts are available in Figures 6.13 and 6.14.



Supervised classification

Section under construction...

1. Features selection

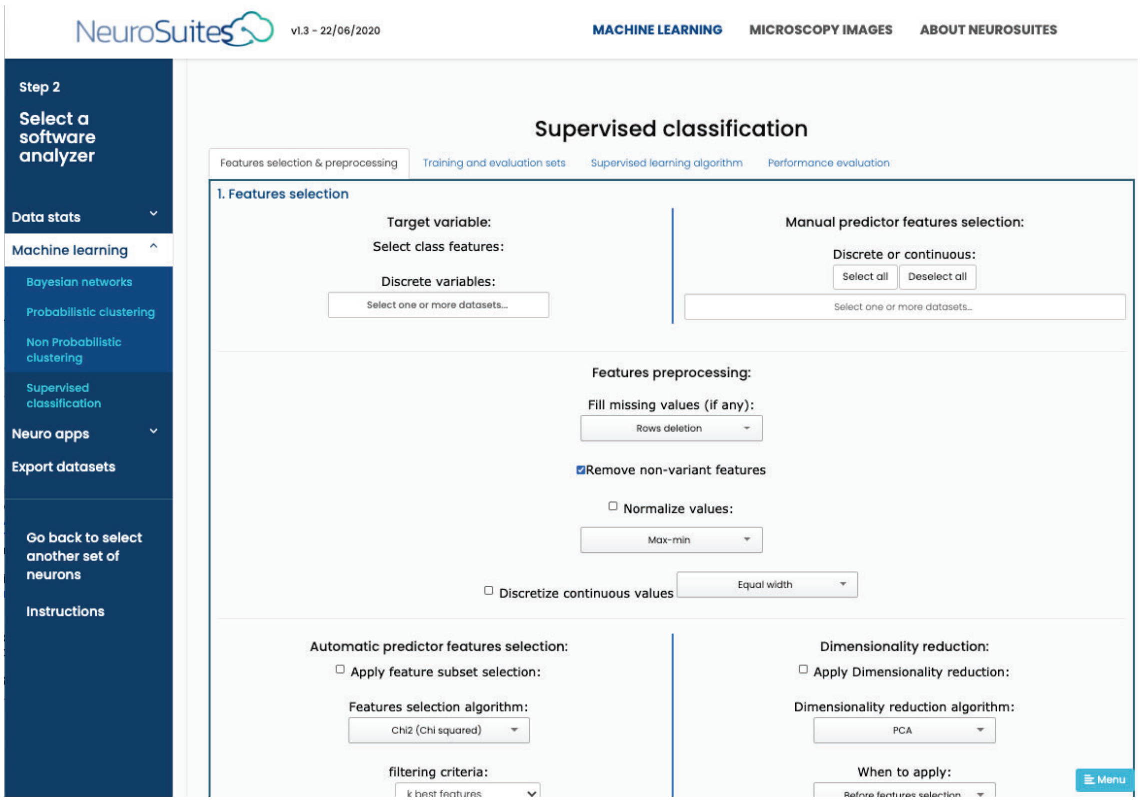
Manual predictor features selection:
Select class features:
Dataset:
Select one or more datasets...

Features preprocessing:
Fill missing values (if any): Rows deletion
☑ Remove non-variant features
Normalize values: Max-min
☐ Discretize continuous values: Select discretize method

Manual predictor features selection:
Dataset:
Select one or more datasets...

Automatic predictor features selection:
Features selection algorithm: Correlation (Pearson)

Figure 6.9: Previous supervised classification page layout



Supervised classification

1. Features selection

Target variable:
Select class features:
Discrete variables: Select one or more datasets...

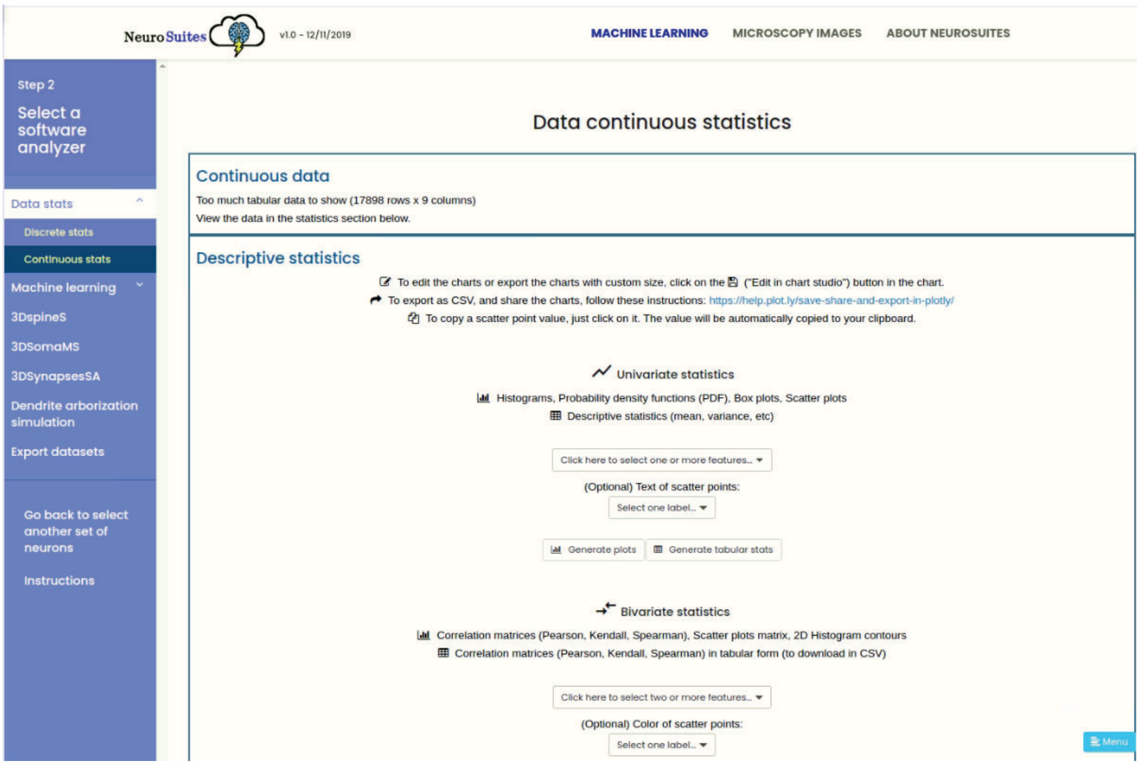
Manual predictor features selection:
Discrete or continuous:
Select one or more datasets...

Features preprocessing:
Fill missing values (if any): Rows deletion
☑ Remove non-variant features
☐ Normalize values: Max-min
☐ Discretize continuous values: Equal width

Automatic predictor features selection:
☐ Apply feature subset selection:
Features selection algorithm: Chi2 (Chi squared)
filtering criteria: k best features

Dimensionality reduction:
☐ Apply Dimensionality reduction:
Dimensionality reduction algorithm: PCA
When to apply: Before features selection

Figure 6.10: New supervised classification page layout



Data continuous statistics

Continuous data

Too much tabular data to show (17898 rows x 9 columns)
View the data in the statistics section below.

Descriptive statistics

To edit the charts or export the charts with custom size, click on the ("Edit in chart studio") button in the chart.
To export as CSV, and share the charts, follow these instructions: https://help.plot.ly/save-share-and-export-in-plotly/
To copy a scatter point value, just click on it. The value will be automatically copied to your clipboard.

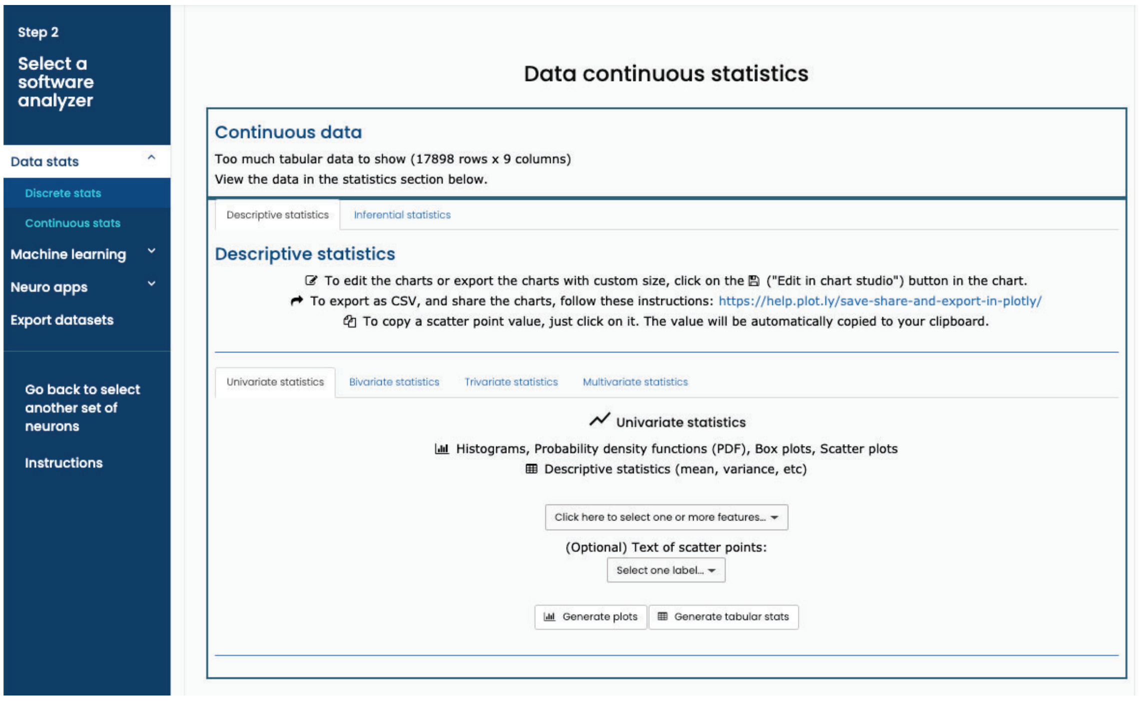
Univariate statistics
Histograms, Probability density functions (PDF), Box plots, Scatter plots
Descriptive statistics (mean, variance, etc)

Click here to select one or more features...
(Optional) Text of scatter points:
Select one label...
Generate plots
Generate tabular stats

Bivariate statistics
Correlation matrices (Pearson, Kendall, Spearman), Scatter plots matrix, 2D Histogram contours
Correlation matrices (Pearson, Kendall, Spearman) in tabular form (to download in CSV)

Click here to select two or more features...
(Optional) Color of scatter points:
Select one label...

Figure 6.11: Previous continuous statistics layout



Data continuous statistics

Continuous data

Too much tabular data to show (17898 rows x 9 columns)
View the data in the statistics section below.

Descriptive statistics | Inferential statistics

Descriptive statistics

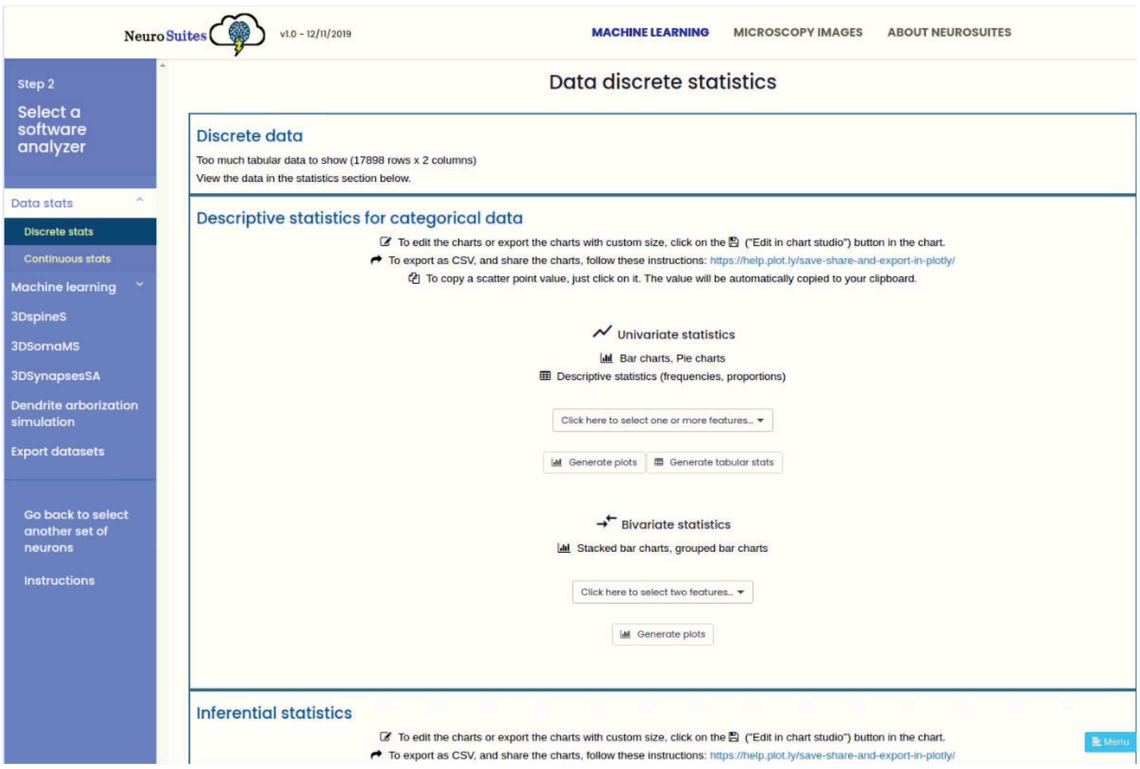
To edit the charts or export the charts with custom size, click on the ("Edit in chart studio") button in the chart.
To export as CSV, and share the charts, follow these instructions: https://help.plot.ly/save-share-and-export-in-plotly/
To copy a scatter point value, just click on it. The value will be automatically copied to your clipboard.

Univariate statistics | Bivariate statistics | Trivariate statistics | Multivariate statistics

Univariate statistics
Histograms, Probability density functions (PDF), Box plots, Scatter plots
Descriptive statistics (mean, variance, etc)

Click here to select one or more features...
(Optional) Text of scatter points:
Select one label...
Generate plots
Generate tabular stats

Figure 6.12: New continuous statistics layout



Data discrete statistics

Discrete data

Too much tabular data to show (17898 rows x 2 columns)
View the data in the statistics section below.

Descriptive statistics for categorical data

To edit the charts or export the charts with custom size, click on the ("Edit in chart studio") button in the chart.
To export as CSV, and share the charts, follow these instructions: https://help.plot.ly/save-share-and-export-in-plotly/
To copy a scatter point value, just click on it. The value will be automatically copied to your clipboard.

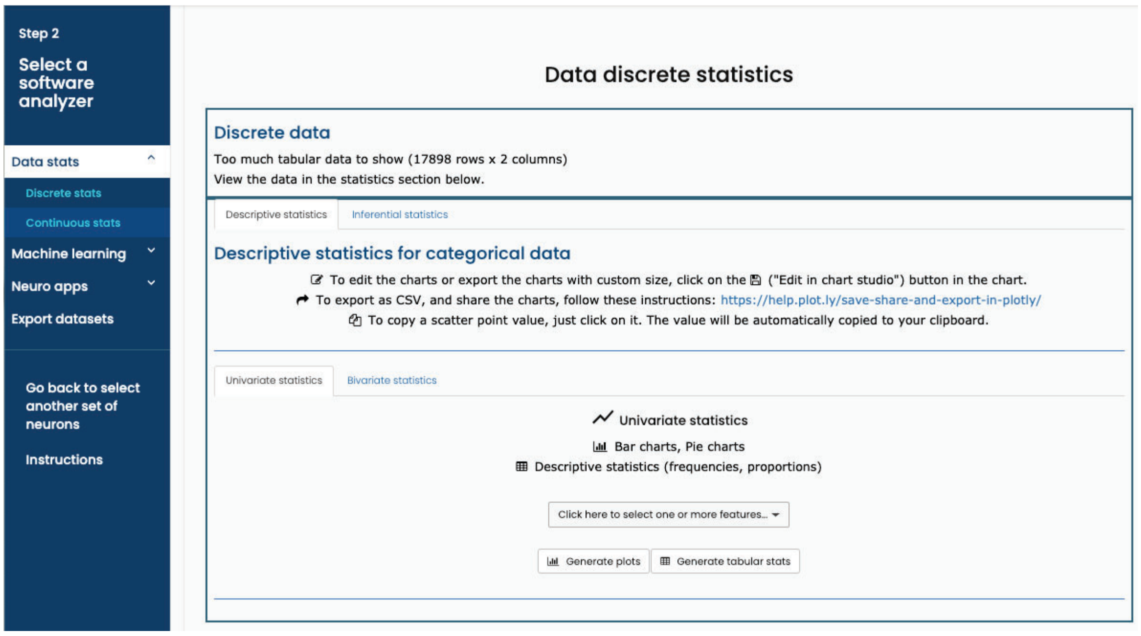
Univariate statistics
Bar charts, Pie charts
Descriptive statistics (frequencies, proportions)

Bivariate statistics
Stacked bar charts, grouped bar charts

Inferential statistics

To edit the charts or export the charts with custom size, click on the ("Edit in chart studio") button in the chart.
To export as CSV, and share the charts, follow these instructions: https://help.plot.ly/save-share-and-export-in-plotly/

Figure 6.13: Previous discrete statistics layout



Data discrete statistics

Discrete data

Too much tabular data to show (17898 rows x 2 columns)
View the data in the statistics section below.

Descriptive statistics for categorical data

To edit the charts or export the charts with custom size, click on the ("Edit in chart studio") button in the chart.
To export as CSV, and share the charts, follow these instructions: https://help.plot.ly/save-share-and-export-in-plotly/
To copy a scatter point value, just click on it. The value will be automatically copied to your clipboard.

Univariate statistics
Bar charts, Pie charts
Descriptive statistics (frequencies, proportions)

Figure 6.14: New discrete statistics layout

Chapter 7

Conclusions and future work

7.1 Conclusions

1. Machine learning is a hot topic with a growing trend for the next years. The importance that is receiving is noticeable even for governments and corporations, given the impact of this technology on society.
2. In the current context, NeuroSuites fills the necessity of a free online coding-less machine learning platform. The other web applications have a commercial objective whereas NeuroSuites has an academic goal and gives tools directly to users without expecting payments.
3. Supervised classification and non-probabilistic clustering are now included also with pre-processing facilities. They give users the flexibility to configure more than the models hyperparameters to achieve better results.
4. The flow, layouts, and results observation inside non-probabilistic clustering were completely designed and created in this project and now are part of the machine learning module.
5. Multi-label classification is a non-common functionality implemented and now offered as part of NeuroSuites. This extension was built over the one-class supervised classification functionality to maintain standardization in the platform.
6. The image of the platform was refreshed in order to have harmonic colors together with a new logo, which combines well with the new palette. At the same time, the menu grouping of items has changed to have context options according to the use needed. Then, the layout distribution for data-stats and supervised classification was improved in a more intuitive and clean way that gives an order to steps and saves time to users.

7.2 Future work

The machine learning module is a general-purpose service, and a critical feature offered by other platforms is regression. NeuroSuites would thus cover the two approaches of supervised learning. With this addition, the use-cases will increase and more users will employ the platform.

Additionally, Feature selection and dimensionality reduction are functionalities that can be implemented in probabilistic clustering, non-probabilistic clustering and multi-label classification because it will provide standarization to the system and new configuration possibilities.

In order to promote the use of NeuroSuites, search engine optimization for appearing in the searches for machine learning platforms should be applied. With more users and the feedback received from them would help to improve the application.

The general-purpose modules data-stats and machine learning, could be in a separate project or platform to avoid possible confusions about restrictions with neuroscience requirements for the datasets.

A functionality that could be added is the possibility of downloading the models. It would be useful for users who want to continue working with the model generated on NeuroSuites but in a programmatical way.

7.3 Personal remark

Carrying out this project was an opportunity to combine my previous experience with web development and my new knowledge gathered during the Data Science Master, so enhancing professional experience.

The challenge of taking the whole project with the administration of the server and learning about Docker technology was hard but exciting at the same time. At this moment, having finished the project gives me a satisfying feeling.

Additionally, for me, it is a priceless experience to have been part of a project that is mounted in the wave of machine learning, aiming to reduce the gap generated with new technologies. In particular, being part of an academic environment gave me a perspective of how the scientific and research community work from inside, by the daily share with the office coworkers and teachers.

Finally, I am feeling motivated because I resolved the question of how to put into production machine learning models. NeuroSuites gave this answer with a complete flow that passes from pre-processing, training, and testing models, and all this inside a web application. Now I want to solve many other problems mixing my two profiles of software developer and data scientist.

Bibliography

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2):49–60.

Bielza, C. and Larrañaga, P. (2020). *Data-Driven Computational Neuroscience: Machine Learning and Statistical Models*. Cambridge University Press.

Birant, D. and Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221.

Chen, W.-J., Shao, Y.-H., Li, C.-N., and Deng, N.-Y. (2016). MLTSVM: A novel twin support vector machine to multi-label learning. *Pattern Recognition*, 52:61–74.

Daelemans, W. and Morik, K. (2008). Clustering via local regresion. In *Machine Learning and Knowledge Discovery in Databases: European Conference, 2008, Proceedings*, volume 5212, pages 456–471. Springer.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In Bajcsy, R., editor, *IJCAI*, pages 1022–1029. Morgan Kaufmann.

Huang, J., Li, Y.-F., and Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, 67:108–127.

Li, S. Z. and Jain, A., editors (2009). *L2 norm*, pages 883–883. Springer.

Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., and Knowles, J. D. (2016). Fifty years of pulsar candidate selection: From simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

Szymański, P. and Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*.

Trochidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multi-label classification of music into emotions. In *ISMIR*, volume 2011, pages 325–330.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2011). Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089.

Zhang, M.-L., Li, Y.-K., Liu, X.-Y., and Geng, X. (2018). Binary relevance for multi-label learning: Overview. *Frontiers of Computer Science*, 12(2):191–202.

Zhang, M.-L. and Zhou, Z.-H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Zhang, T., Ramakrishnan, R., and Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182.