Data and text mining

ClassificaIO: machine learning for classification graphical user interface

Raeuf Roushangar[1,2], George Mias[1,2,*]

[1]Department of Biochemistry and Molecular Biology,

[2]Institute for Quantitative Health Science and Engineering,

Michigan State University, East Lansing MI 48824

*To whom correspondence should be addressed.

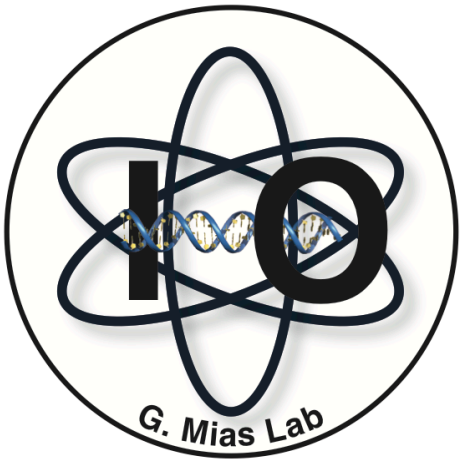
Abstract

Summary: ClassificaIO is an open-source Python graphical user interface (GUI) for machine learning classification for the scikit-learn module. ClassificaIO aims to provide an easy-to-use interactive way to train, validate, and test data on a range of classification algorithms. The GUI enables fast comparisons within and across classifiers, and facilitates uploading and exporting of trained models, and both validated, and tested data results.

Availability: ClassificaIO is implemented as a Python application and is available for download and installation through the Python Package Index (PyPI) (http://pypi.python.org/pypi/ClassificaIO) and it can be deployed using the "import" function once installed. The application is distributed under an MIT license and source code is available for download (for Mac OS X, Unix and Microsoft Windows) at: (http://github.com/gmiaslab/ClassificaIO).

Contact: gmias@msu.edu

SUPPLEMENTARY INFORMATION: ClassificaIO User Manual



ClassificalO 1.0.1

Machine Learning for Classification Graphical User Interface User Manual

Summary:

ClassificaIO is an open-source Python graphical user interface (GUI) for machine learning classification for the scikit-learn module. ClassificaIO aims to provide an easy-to-use interactive way to train, validate, and test data on a range of classification algorithms. The GUI enables fast comparisons within and across classifiers, and facilitates uploading and exporting of trained models, and both validated, and tested data results.

Prerequisites:

ClassificaIO is a Python library with the following external dependencies: nltk ≥ 3.2.5, Tcl/Tk ≥ 8.6.7, Pillow ≥ 4.3, pandas ≥ 0.21, numpy ≥ 1.13, scikit-learn ≥ 0.19.1. ClassificaIO requires Python version 3.5 or higher and we recommend using the Spyder integrated development environment (IDE) in Anaconda Navigator (https://www.anaconda.com/download/) on Mac OS High Sierra (10.13) and Microsoft Windows 10 or higher.

Download and installation:

ClassificaIO can be installed using pip (https://pypi.python.org/pypi/pip) in the terminal:

```
$ pip install --user -U ClassificaIO
```

You can also install it directly from the main GitHub repository using:

```
$ pip install git+https://github.com/gmiaslab/ClassificaIO.git
```

Or:

```
$ pip install git+git://github.com/gmiaslab/ClassificaIO.git
```

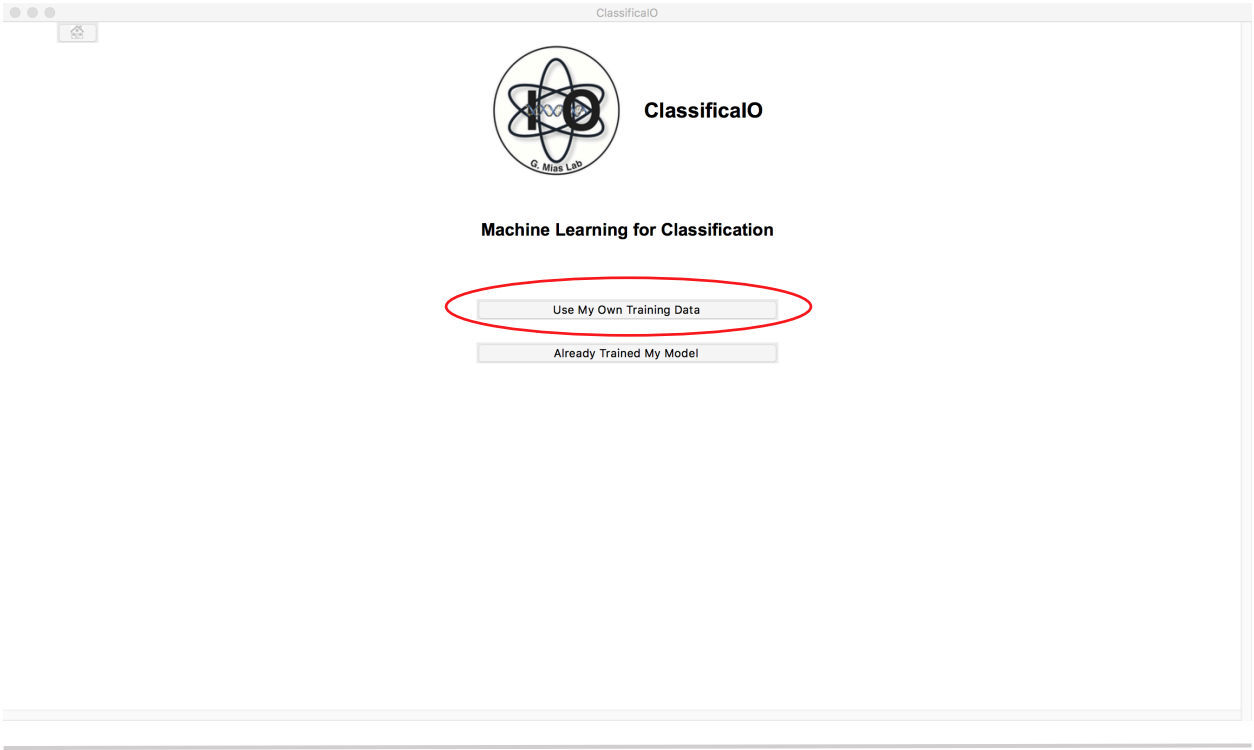
In case you do not have pip installed, you must install it first! Or obtaining and installing ClassificaIO by downloading or cloning ClassificaIO source code from ClassificaIO GitHub repository.

Getting started:

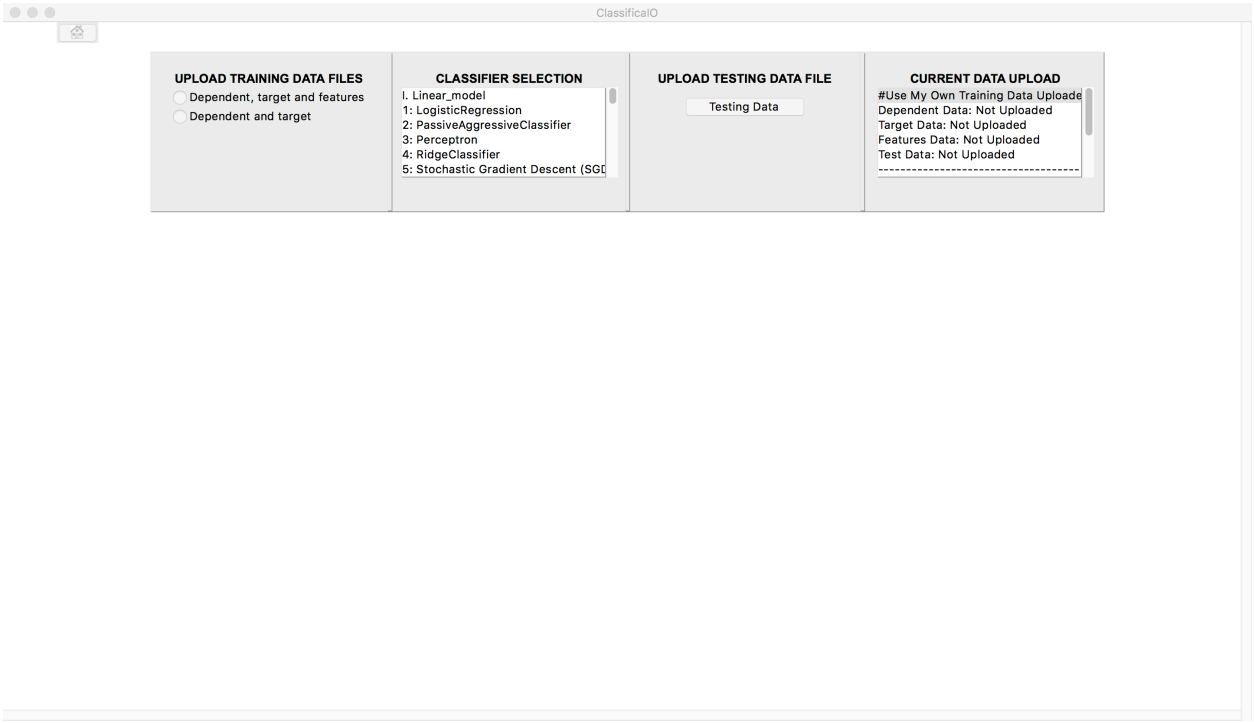
After installing ClassificaIO and all dependences, please deploy it using the “import” function in Python:

> import ClassificaIO

Once ClassificaIO main window appears on the screen, you can click on ‘Use My Own Training Data’ button and start your new supervised machine learning classification project.

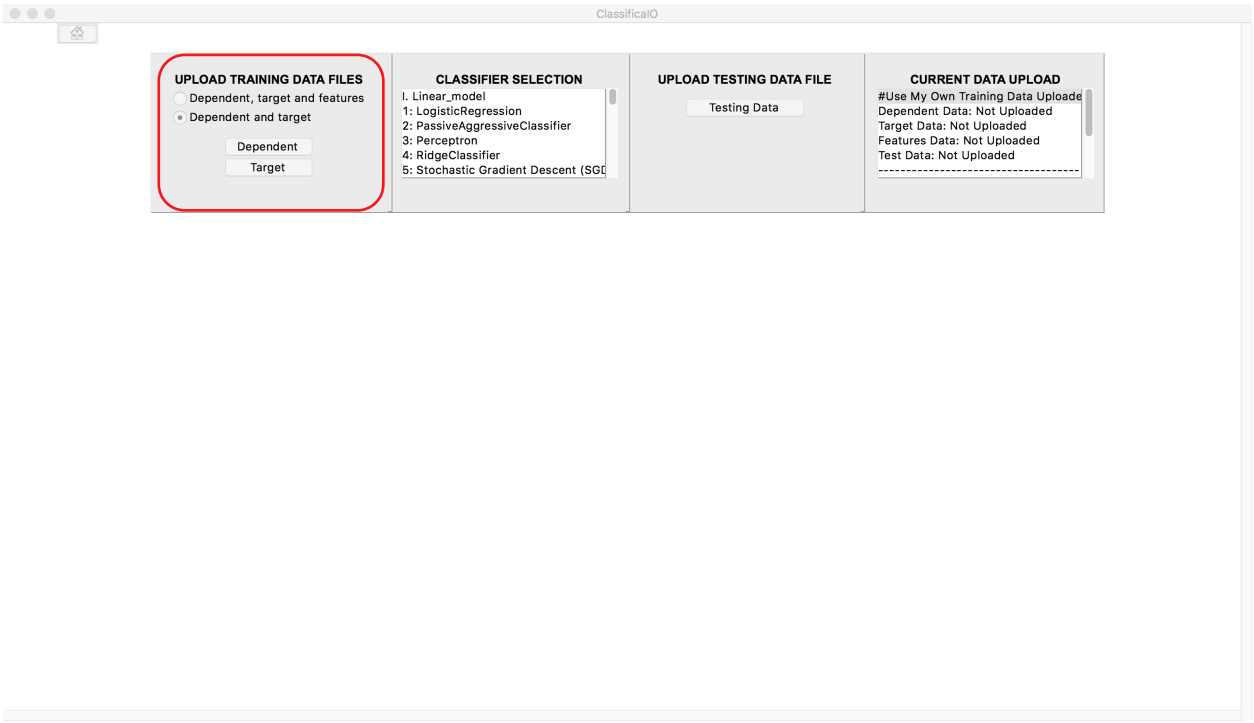


From here you can see ‘Use My Own Training Data’ window.

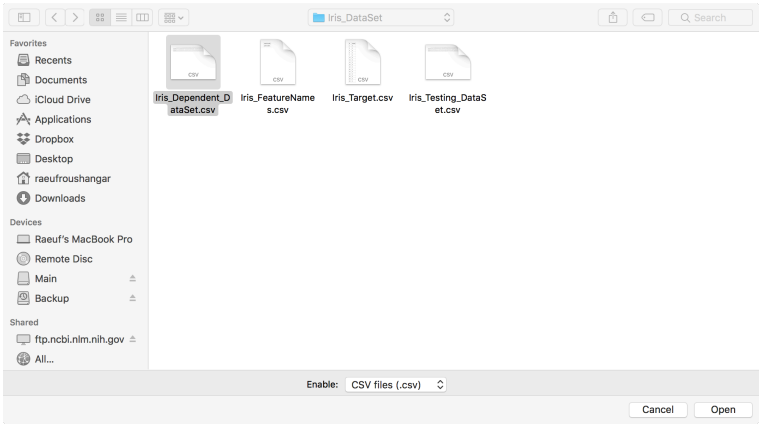


Training data input:

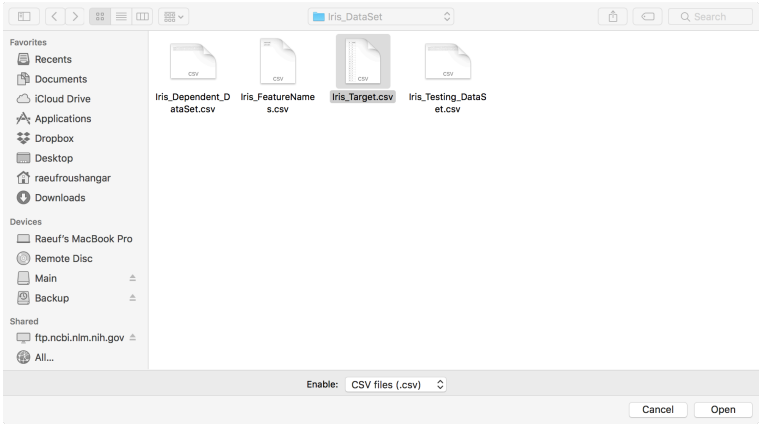
You first need to make a selection (‘Dependent and Target’ or ‘Dependent, Target and Features’) from the ‘UPLOAD TRAINING DATA FILES’ panel and upload training data files. For this example, we select ‘Dependent and Target’.



By clicking the corresponding buttons in the 'UPLOAD TRAINING DATA FILES' panels, a file selector directs you to upload dependent data file (**Supplementary Figure 1**) and target data file (**Supplementary Figure 2**).

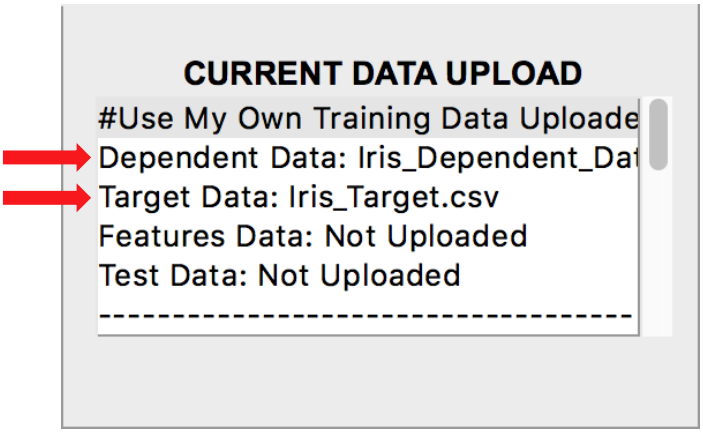


Supplementary Figure 1.a Dependent data file upload.



Supplementary Figure 1.b Target data file upload.

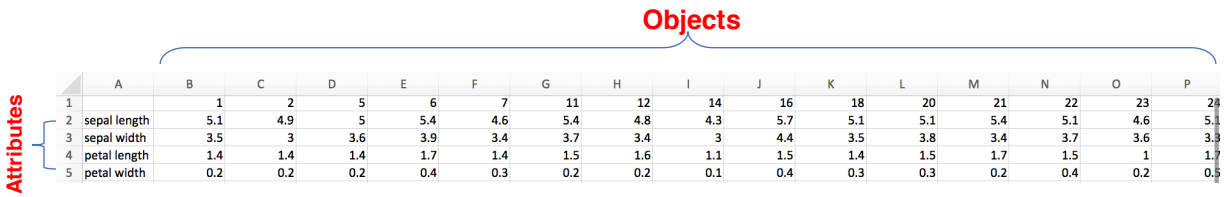
A history of all uploaded data files (file name and directory) is automatically saved in the 'CURRENT DATA UPLOAD' panel (**Supplementary Figure 2**).



Supplementary Figure 2. Uploaded dependent and target log. Data files (Red arrows point to files names). Scroll down for uploaded files directories.

Training data format:

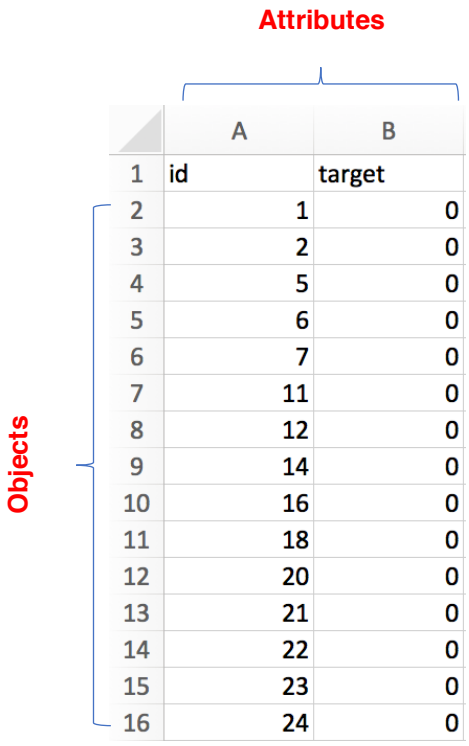
To start training your machine leaning classification algorithms, you must first provide supporting training data files (dependent, **Supplementary Figure 3.a**, and target, **Supplementary Figure 3.b**). The files must be comma-separated values (CSV) format.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		1	2	5	6	7	11	12	14	16	18	20	21	22	23	24
2	sepal length	5.1	4.9	5	5.4	4.6	5.4	4.8	4.3	5.7	5.1	5.1	5.4	5.1	4.6	5.1
3	sepal width	3.5	3	3.6	3.9	3.4	3.7	3.4	3	4.4	3.5	3.8	3.4	3.7	3.6	3.3
4	petal length	1.4	1.4	1.4	1.7	1.4	1.5	1.6	1.1	1.5	1.4	1.5	1.7	1.5	1	1.7
5	petal width	0.2	0.2	0.2	0.4	0.3	0.2	0.2	0.1	0.4	0.3	0.3	0.2	0.4	0.2	0.5

Supplementary Figure 3.a. Example of partial dependent data format.

- For both data files format and training, validating, and testing classification (below) demonstration, we used the machine learning Iris Dataset (Anderson, 1935; Fisher, 1936) with 70% training (105 data points) and 30% testing (45 data points) split.

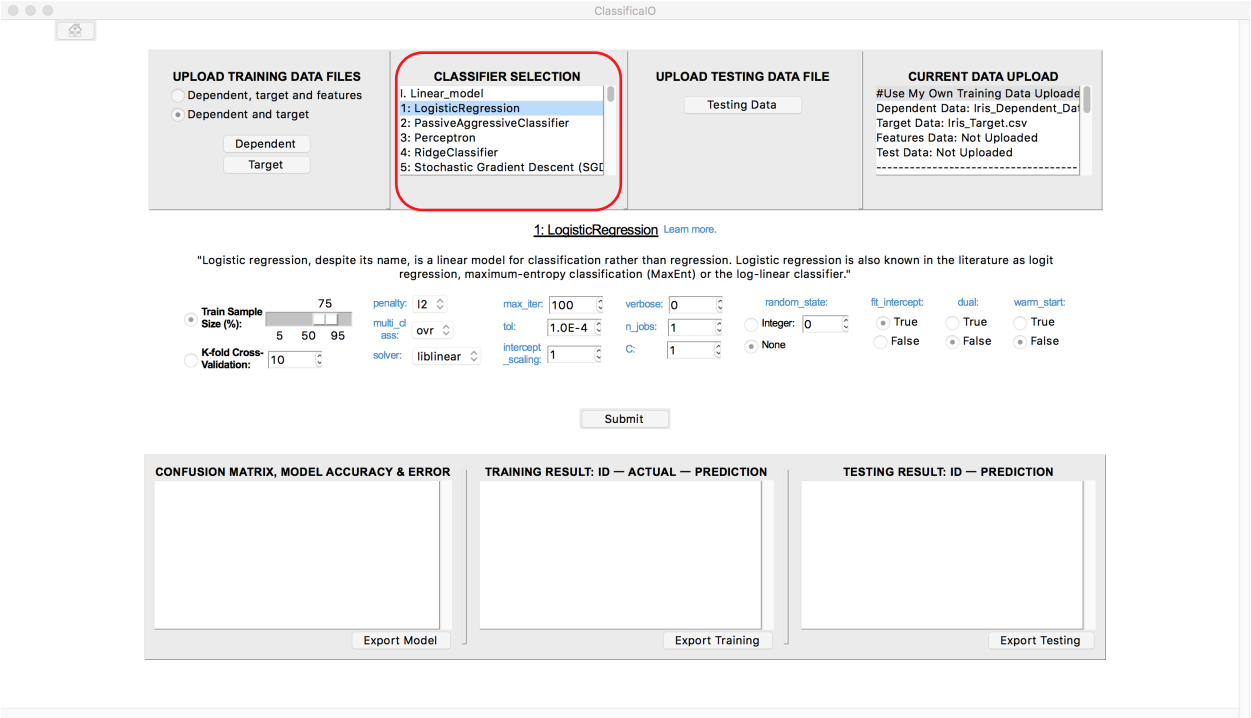


	A	B
1	id	target
2	1	0
3	2	0
4	5	0
5	6	0
6	7	0
7	11	0
8	12	0
9	14	0
10	16	0
11	18	0
12	20	0
13	21	0
14	22	0
15	23	0
16	24	0

Supplementary Figure 3.b. Example of partial target data file format.

Classifier selection:

Once you have uploaded all training data files, you can easily select between 25 different machine learning classification algorithms in the ‘CLASSIFER SELECTION’ panel.



Here are all 25 different classification algorithms in order of appearance in the panel:

I. Linear_model

1: LogisticRegression
2: PassiveAggressiveClassifier
3: Perceptron
4: RidgeClassifier
5: Stochastic Gradient Descent (SGDClassifier)

II. Discriminant_analysis

6: LinearDiscriminantAnalysis
7: QuadraticDiscriminantAnalysis

III. Support vector machines (SVMs)

8: LinearSVC
9: NuSVC
10: SVC

IV. Neighbors

11: KNeighborsClassifier
12: NearestCentroid
13: RadiusNeighborsClassifier

V. Gaussian_process

14: GaussianProcessClassifier

VI. **Naive_bayes**

15: BernoulliNB
16: GaussianNB
17: MultinomialNB

VII. **Trees**

18: DecisionTreeClassifier
19: ExtraTreeClassifier

VIII. **Ensemble**

20: AdaBoostClassifier
21: BaggingClassifier
22: ExtraTreesClassifier
23: RandomForestClassifier

IX. **Semi_supervised**

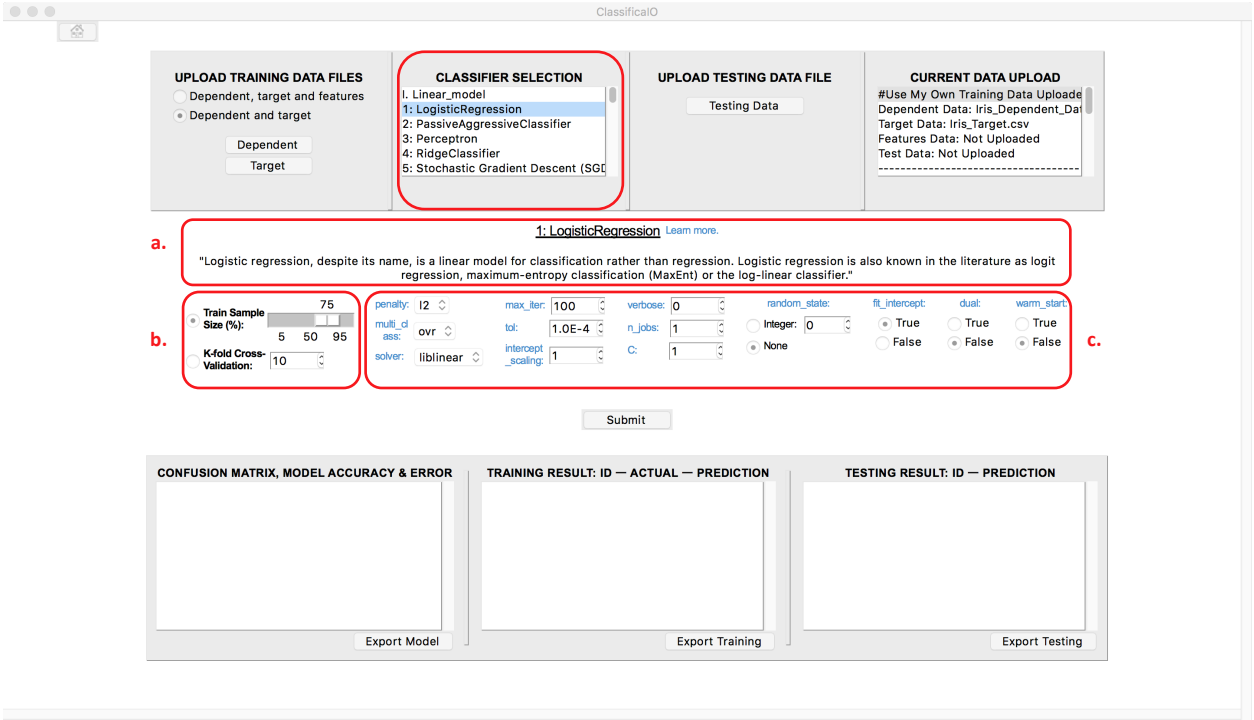
24: LabelPropagation

X. **Neural_network**

25: MLPClassifier

The following will populate once you make a classifier selection:

- **Supplementary Figure 4.a**: the classifier definition with a clickable hyperlink "learn more" in blue, which, once clicked, opens an external web-browser to the selected classifier scikit-learn documentation
- **Supplementary Figure 4.b**: easy interactive way to select between train-validate split and cross-validation methods, which are necessary to prevent/minimize training model overfitting
- **Supplementary Figure 4.c**: classifier parameters, to provide you with a point-and-click interface to set, modify, and test the influence of each parameter on your data



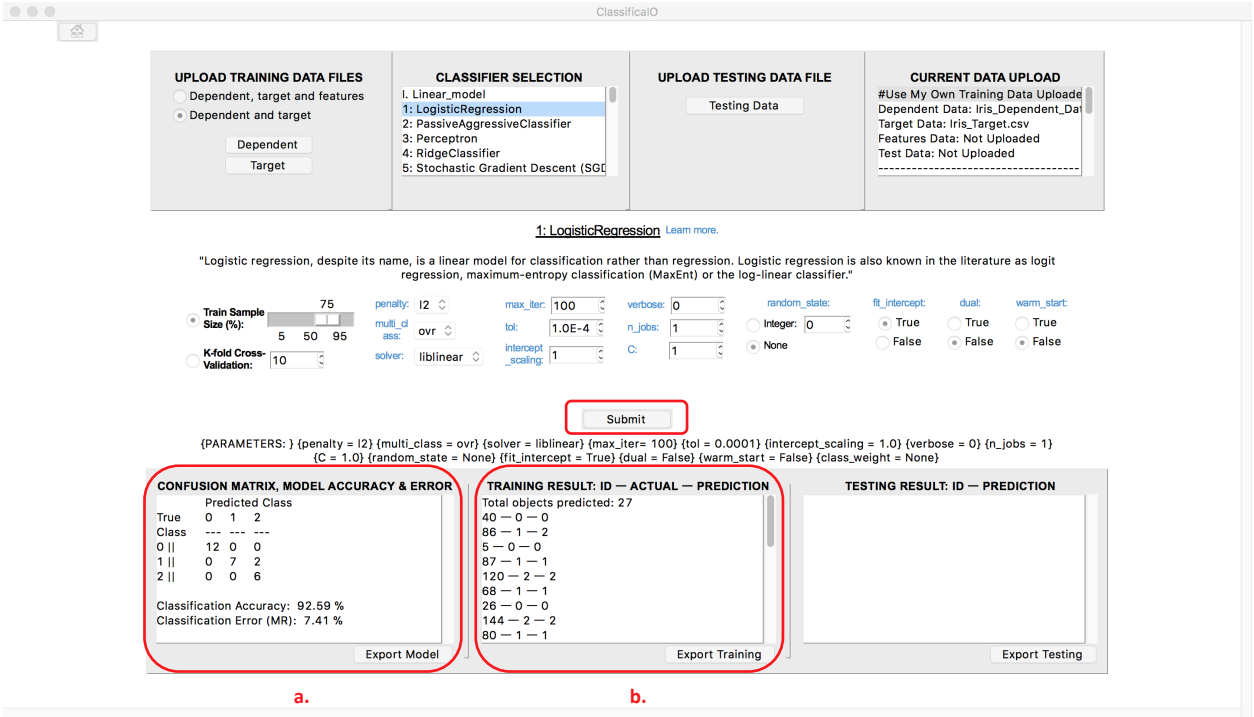
Supplementary Figure 4. Selected Logistic Regression Classifier. a. classifier definition, **b**. training methods with Train Sample Size(%) method selected, **c**. classifier default parameters.

You can now click 'submit' to train your selected classifier using already uploaded training data (dependent and target), evaluate classifier accuracy and error. Or, upload testing data and then click 'submit' to both train classifier and test your data at the same time!

For this example, **first**: we will train selected '1: LogisticRegression' classifier using its default parameters and default train-validate split method, evaluate its accuracy and error, **second**, we will upload testing data to test the trained model.

Model training, evaluation, validation and result output:

After clicking on 'submit' selected '1: LogisticRegression' classifier is trained using the loaded training data (dependent and target). Internally, ClassificaIO uses the scikit-learn train_test_split function when "Train Sample Size (%)" method is selected, this to allow for fast training data split into training and validation subsets. Also, with this method the parameters: train_size = takes the train sample size set by you, shuffle = always set to True (default) to shuffle training data before splitting (e.g. Train Sample Size (%): set to 75% means train_size = 0.75 and test_size= 0.25), random_state = always set to 0 (can be other integer or other value, but preferred 0) to guarantee same training data split using train_size and test_size, and for result consistency, rather than random.
After training is completed, the confusion matrix, classifier accuracy and error are displayed in the "CONFUSION MATRIX, MODEL ACCURACY & ERROR" panel (**Supplementary Figure 5.a**). Model validation data results are displayed in the "TRAINING RESULT: ID – ACTUAL – PREDICTION" panel (**Supplementary Figure 5.b**) with object ID = 1st, actual target value = 2st, predicted target value = 3rd.

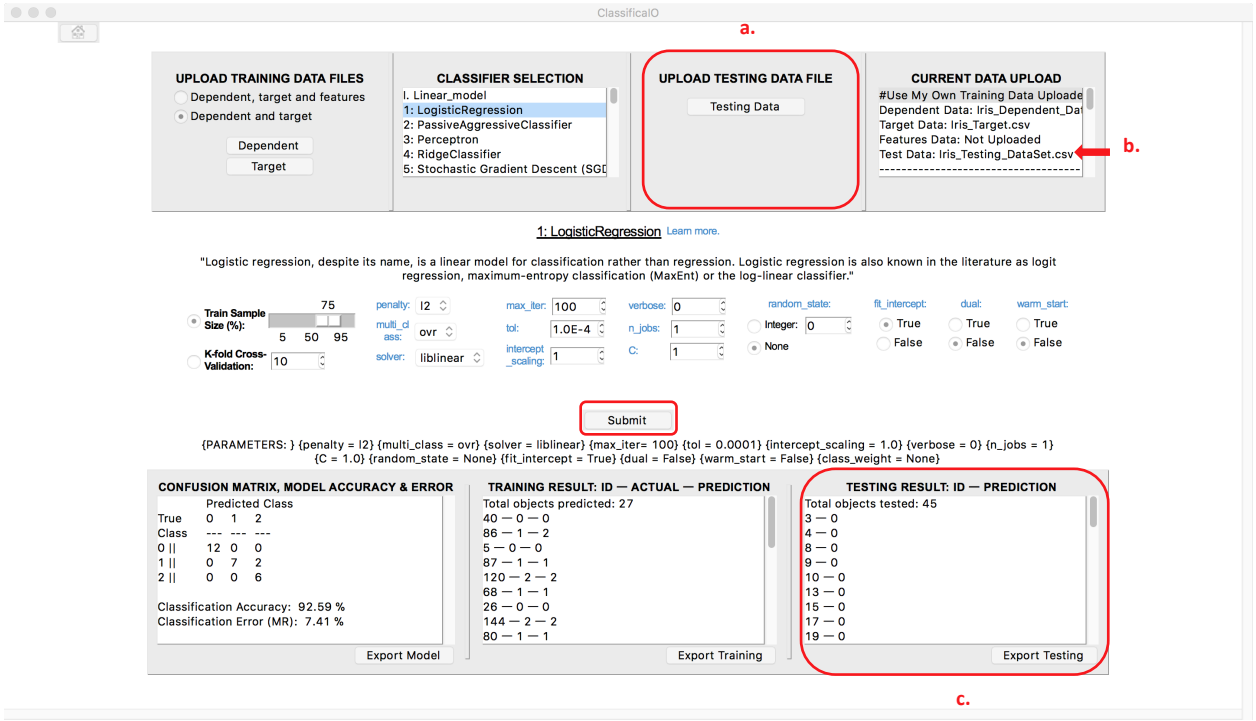


Supplementary Figure 5. Trained Logistic Regression Classifier Result. a. trained model using 78 data points (75% of 105), classifier evaluation (confusion matrix, model accuracy and error), **b**. model validated using 27 data points (25% of 105 objects).

Testing data input and result output:

Following same steps in training data input, you will need to upload testing data file by clicking the 'Testing Data' button in the "UPLOAD TESTING DATA FILE" panel (**Supplementary Figure 6.a**). Once clicked, a file selector directs you to upload testing data file, see **Supplementary Figure 2.a & b**. Once testing data file is uploaded file name is automatically saved in the 'CURRENT DATA UPLOAD' panel (**Supplementary Figure 6.b**). Testing data file format is same as of dependent data file, see **Supplementary Figure 3.a.**

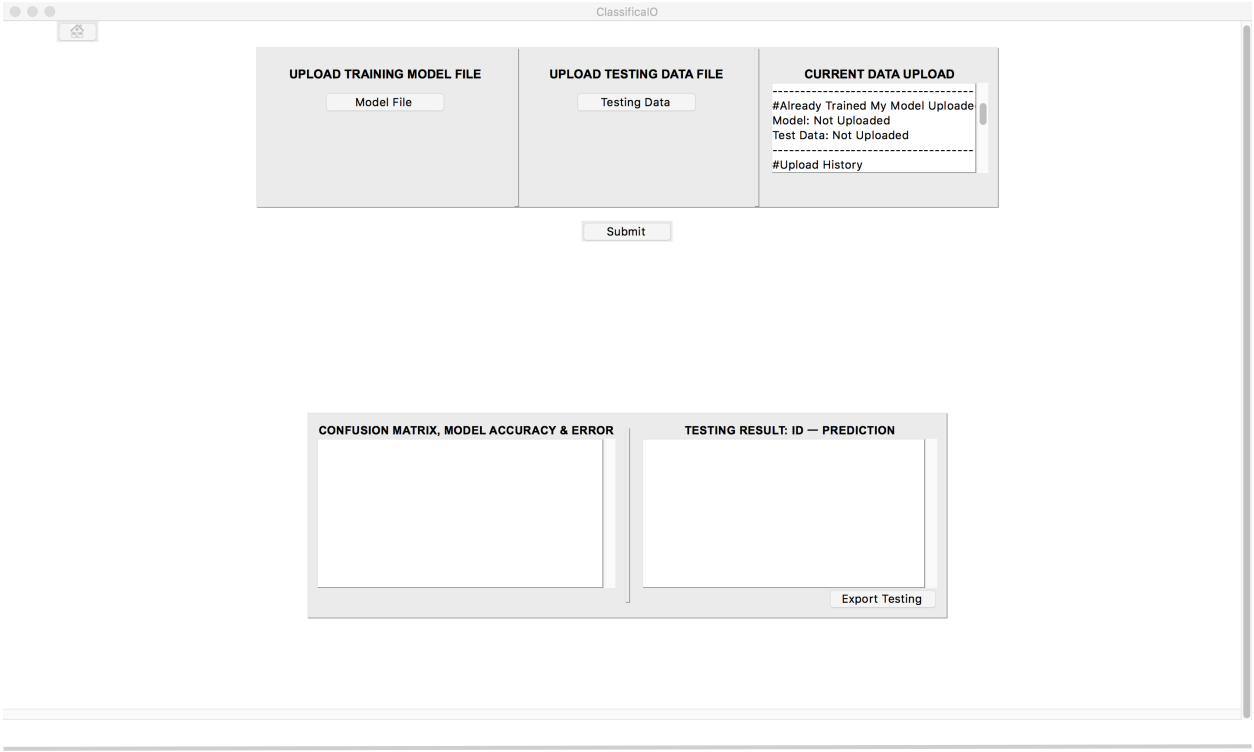
After clicking on 'submit' and after testing is completed, tesing data results are displayed in the "Testing RESULT: ID – PREDICTION" panel (**Supplementary Figure 6.c**) with object ID = 1st and predicted target value = 2st.



Supplementary Figure 6. Tested Logistic Regression Classifier Result. a. upload testing data panel, **b**. Uploaded testing data file (Red arrows point to files names), **c**. model tested using 45 data points (30% of 150 total data points).

Result export:

Now you are ready to export your trained model to preserve for future use without having to retrain. Simply, click the “Export Model” button (**Supplementary Figure 5.a**) and save your model. Now, your exported ClassificaIO model can then be used for future testing of new data in the ‘Already Trained My Model’ window.

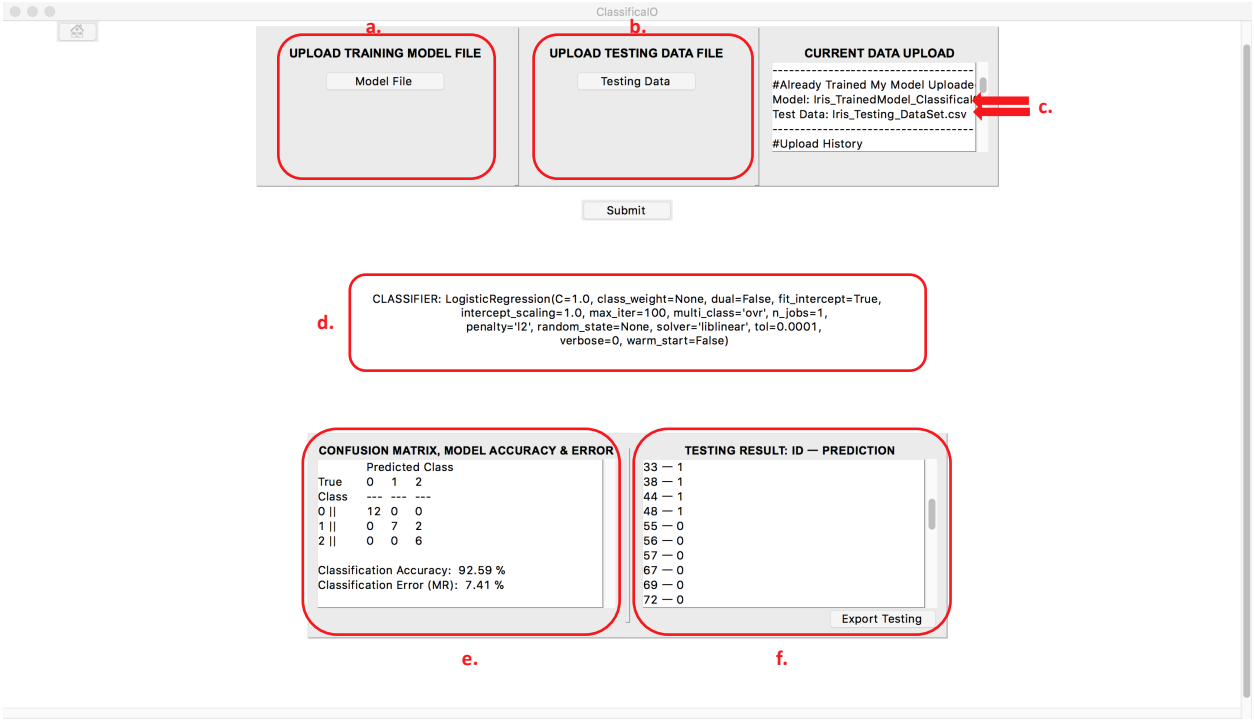


ClassificaIO model and testing data input:

You will need to upload ClassificaIO model by clicking the 'Model File' button in the 'UPLOAD TRAINING MODEL FILE' panel (**Supplementary Figure 7.a**). Once clicked, a file selector directs you to upload ClassificaIO already trained model file, see **Supplementary Figure 3.a & b**. You will also need to upload testing data file by clicking the 'Testing Data' button in the "UPLOAD TESTING DATA FILE" panel (**Supplementary Figure 7.b**). Once ClassificaIO model and testing data files are uploaded, files names are automatically saved in the 'CURRENT DATA UPLOAD' panel (**Supplementary Figure 7.c**). Testing data file format is the same as previously explained.

After clicking on 'submit' and after testing is completed, uploaded model parameters will populate (**Supplementary Figure 7.d**) to show the classifier used to train the uploaded model. The confusion matrix, classifier accuracy and error of trained model are then displayed in the "CONFUSION MATRIX, MODEL ACCURACY & ERROR" panel (**Supplementary Figure 7.e**). Testing data results are then displayed in the "Testing RESULT: ID – PREDICTION" panel (**Supplementary Figure 7.f**) with object ID = 1st and predicted target value = 2st.

Results Export:



Supplementary Figure 7. 'Already Trained My Model' window

Full results (trained models, models parameters, and both validated, and tested data) for both windows (**'Use My Own Training Data' and 'Already Trained My Model'**) can be exported as CSV files for further analysis for publication, sharing, or later use (for more details on the export data file formats, see Supplementary data).

References

Anderson, E. The Irises of the Gaspe peninsula. *Bulletin of American Iris Society* 1935;59:2-5.
Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann Eugenic* 1936;7:179-188.