

ClassificalO: machine learning for classification graphical user interface

Raeuf Roushangar^{1,2}, George Mias^{1,2,*}

¹Department of Biochemistry and Molecular Biology,

²Institute for Quantitative Health Science and Engineering,

Michigan State University, East Lansing MI 48824

*To whom correspondence should be addressed.

Abstract

Summary: ClassificalO is an open-source Python graphical user interface (GUI) for machine learning classification for the scikit-learn module. ClassificalO 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.

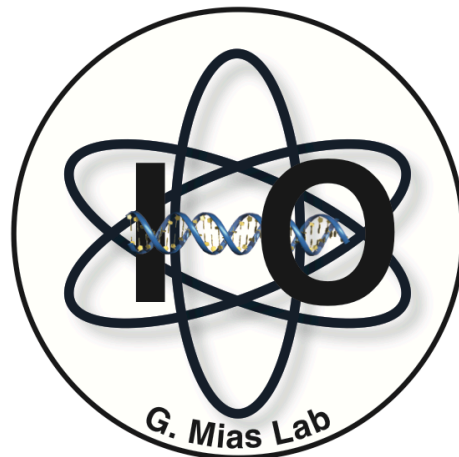
Availability: ClassificalO is implemented as a Python application and is available for download and installation through the Python Package Index (PyPI) (<http://pypi.python.org/pypi/ClassificalO>) and it can be deployed using the “import” function once installed. The application is distributed under an MIT license and source code is available for download (for Mac OS X, Unix and Microsoft Windows) through PyPI and GitHub (<http://github.com/gmiaslab/ClassificalO>), and additionally released at Zenodo, <https://doi.org/10.5281/zenodo.1133266>.

Contact: gmias@msu.edu

SUPPLEMENTARY INFORMATION: ClassificalO User Manual 2

ATTACHMENTS: SUPPLEMENTARY FILES

	<i>File Name</i>	<i>Description</i>
1.	<i>S1_Iris_Dependent_DataSet</i>	Iris data set (150 data points)
2.	<i>S2_Iris_Target</i>	Iris Target data set (150 labels)
3.	<i>S3_Iris_Testing_DataSet</i>	Iris Testing data set (150 data points)
4.	<i>S4_Iris_FeatureNames</i>	Example Iris features (2 features: sepal length and petal width)
5.	<i>S5_LogisticRegression_IrisTrainedModel.pkl</i>	Example ClassificalO trained model using logistic regression
6.	<i>S6_TestingResult.CSV</i>	Example ClassificalO testing result using logistic regression
7.	<i>S7_TrainValidationResult.csv</i>	Example ClassificalO validation result using logistic regression



ClassificalO

Machine Learning for Classification Graphical
User Interface User Manual 1.0.5 (12/2017)

Summary:

ClassificalO is an open-source Python graphical user interface (GUI) for supervised machine learning classification for the scikit-learn module (Pedregosa, et al., 2011). ClassificalO 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:

ClassificalO is a Python library with the following external dependencies: nltk \geq 3.2.5, Tcl/Tk \geq 8.6.7, Pillow \geq 4.3, pandas \geq 0.21, numpy \geq 1.13, scikit-learn \geq 0.19.1. ClassificalO 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:

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

```
$ pip install ClassificalO
```

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

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

In case you do not have pip installed, you must install it first. Or you can obtain and install ClassificalO by downloading or cloning ClassificalO source code from ClassificalO GitHub repository (<https://github.com/gmiaslab/ClassificalO>)

Getting started:

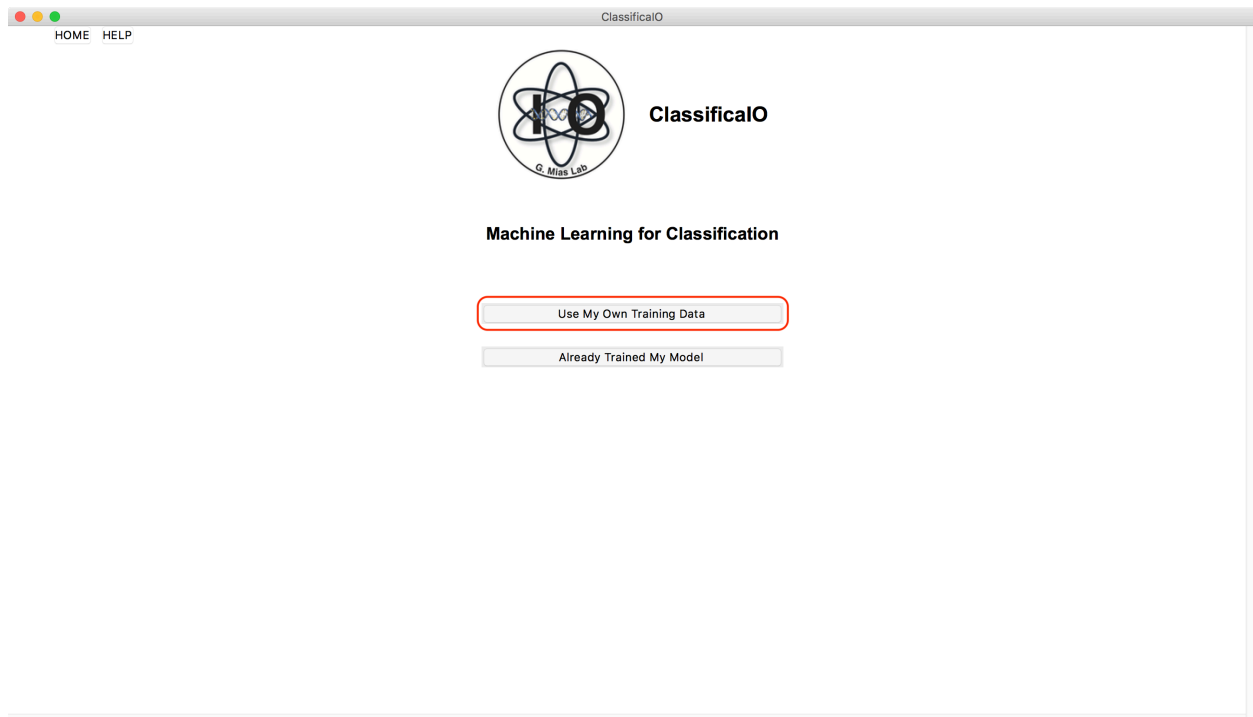
Please Note:

- ClassificalO supports comma-separated values (CSV) input files only.
- In this document we use the machine learning Iris dataset (Anderson, 1935; Fisher, 1936) (150 data points) as an example, to demonstrate model training, validation, and testing, as well as the data formats that ClassificalO relies on.
- In the classification example below, we use 70% of the Iris dataset (105 data points) for model training and 30% (45 data points) for model testing.

After installing ClassificalO, please run it using the “import” function in Python:

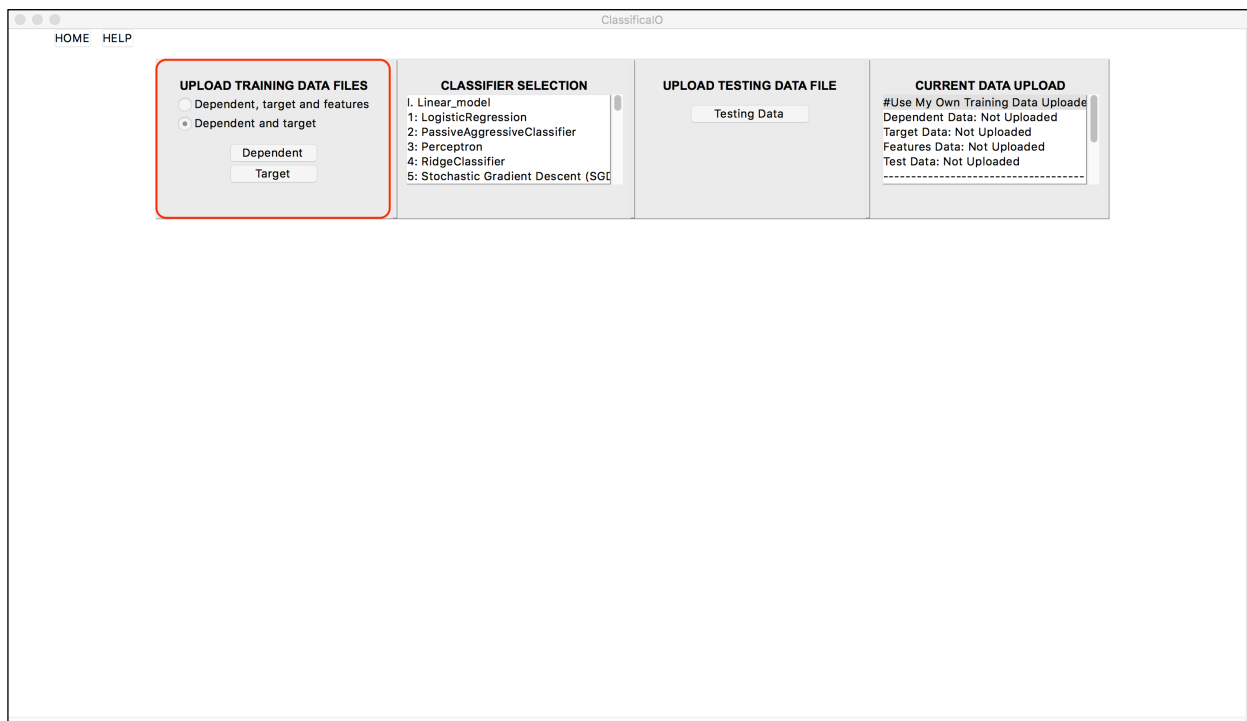
```
>>> from ClassificalO import ClassificalO  
>>> ClassificalO.gui()
```

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

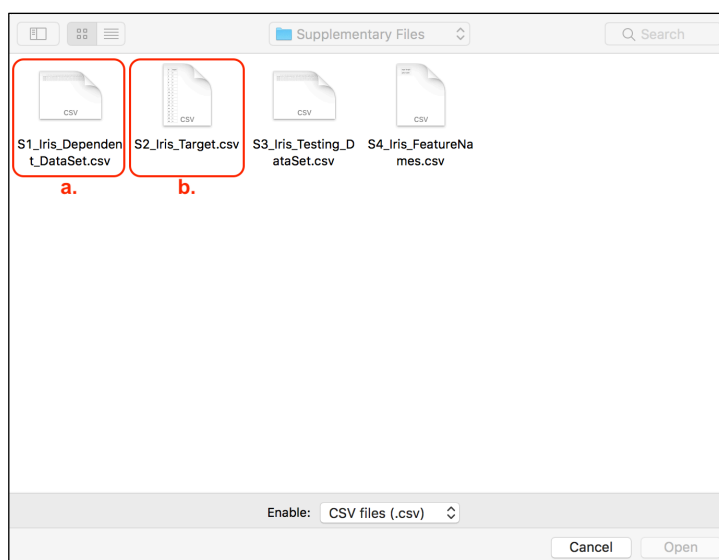


Training data input:

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



To begin uploading files, click the corresponding buttons in the 'UPLOAD TRAINING DATA FILES' panel: a file selector directs you to upload both, dependent data file (**Supplementary Figure 1.a**) and target data file (**Supplementary Figure 1.b**). Once a file is uploaded to ClassificalO, the file name and directory are automatically saved in the 'CURRENT DATA UPLOAD' panel (**Supplementary Figure 2**). This updatable log allows for tracking current data files in use, and maintains a history of all files uploaded to the software.



Supplementary Figure 1. Graphical Control Element Dialog Box. a. Dependent data file selected for upload. **b.** selected target data file to upload. N.B. each file selection has to be done one at a time.

Supplementary Figure 2. Current Data Upload Panel. Both dependent and target data file names shown (red boxes). Scroll down for uploaded data files directories.

CURRENT DATA UPLOAD

#Use My Own Training Data Upload

Dependent Data: S1_Iris_Dependent

Target Data: S2_Iris_Target.csv

Features Data: Not Uploaded

Test Data: Not Uploaded

Data format:

Data formats are shown in **Supplementary Figure 3.a** for dependent data and **Supplementary Figure 3.b** for target data.

		Objects															
Attributes		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 Dependent Data. Example of partial dependent data file format. Testing data (not shown) uses the same format.

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

		Attributes	
		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

Classifier selection:

Once you have uploaded all required training data files, you can select between 25 different machine learning classification algorithms in the 'CLASSIFIER SELECTION' panel.

The screenshot shows the ClassificalO web interface. At the top, there are navigation links for HOME and HELP. The main content area is divided into four panels: UPLOAD TRAINING DATA FILES, CLASSIFIER SELECTION, UPLOAD TESTING DATA FILE, and CURRENT DATA UPLOAD. The CLASSIFIER SELECTION panel is highlighted with a red box and contains a list of 5 classifiers: 1: Linear_model, 2: LogisticRegression, 3: PassiveAggressiveClassifier, 4: Perceptron, and 5: RidgeClassifier. Below this list, there is a description of Logistic Regression and a set of parameters for training and testing. The parameters include Train Sample Size (%), K-fold Cross-Validation, random_state, penalty, multi_class, solver, max_iter, tol, verbose, fit_intercept, dual, warm_start, n_jobs, and C. A Submit button is located below the parameters. At the bottom, there are three panels: CONFUSION MATRIX, MODEL ACCURACY & ERROR, TRAINING RESULT: ID — ACTUAL — PREDICTION, and TESTING RESULT: ID — PREDICTION. Each panel has an Export button (Export Model, Export Training, Export Testing).

Here are all classification algorithms in order of appearance in the 'CLASSIFIER SELECTION' panel. Also, immutable (unchangeable) parameters with their default values are also listed for each classifier:

I. Linear_model

- 1: LogisticRegression. (class_weight = None)
- 2: PassiveAggressiveClassifier. (class_weight = None, n_iter= None)
- 3: Perceptron. (class_weight = None)
- 4: RidgeClassifier. (class_weight = None)
- 5: Stochastic Gradient Descent (SGDClassifier).

II. Discriminant_analysis

- 6: LinearDiscriminantAnalysis. (shrinkage= None, priors = None)
- 7: QuadraticDiscriminantAnalysis. (store_covariances = None, priors = None)

III. Support vector machines (SVMs)

- 8: LinearSVC. (class_weight = None)
- 9: NuSVC. (class_weight = None)
- 10: SVC. (class_weight = None)

IV. Neighbors

- 11: KNeighborsClassifier. (metric_params = None)
- 12: NearestCentroid.
- 13: RadiusNeighborsClassifier. (metric_params = None)

V. Gaussian_process

- 14: GaussianProcessClassifier. (kernel = None)

VI. Naive_bayes

- 15: BernoulliNB. (class_prior = None)
- 16: GaussianNB. (class_prior = None)
- 17: MultinomialNB. (class_prior = None)

VII. Trees

- 18: DecisionTreeClassifier. (class_weight = None)
- 19: ExtraTreeClassifier. (min_impurity_split = None, class_weight = None)

VIII. Ensemble

- 20: AdaBoostClassifier. (base_estimator = None)
- 21: BaggingClassifier. (base_estimator = None)
- 22: ExtraTreesClassifier. (class_weight = None)
- 23: RandomForestClassifier. (class_weight = None)

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 scikit-learn documentation for the selected classifier.
- **Supplementary Figure 4.b:** Easy interactive way to select between train-validate split and cross-validation methods (radio buttons), 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

The screenshot displays the ClassificalIO web application interface. At the top, there are navigation links for HOME and HELP. The main content area is divided into several sections:

- UPLOAD TRAINING DATA FILES:** Includes radio buttons for "Dependent, target and features" and "Dependent and target", with corresponding "Dependent" and "Target" buttons below.
- CLASSIFIER SELECTION:** A list of classifiers: 1: Linear_model, 2: LogisticRegression (highlighted), 3: PassiveAggressiveClassifier, 4: Perceptron, and 5: Stochastic Gradient Descent (SGD).
- UPLOAD TESTING DATA FILE:** Includes a "Testing Data" button.
- CURRENT DATA UPLOAD:** A text area showing upload status for training, target, features, and test data.

Below these sections, a red box labeled **a.** contains the definition of Logistic Regression: "Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier." A "Learn more" link is provided.

Below this, a red box labeled **b.** shows training methods: "Train Sample Size (%)" with a slider set to 75, and "K-fold Cross-Validation" with a dropdown set to 10. To the right, a red box labeled **c.** contains various classifier parameters such as "random_state", "penalty", "max_iter", "verbose", "fit_intercept", "dual", "warm_start", "multi_class", "tol", "n_jobs", "C", "solver", and "intercept_scaling", each with a corresponding input field or dropdown menu.

A "Submit" button is located below the parameter section. At the bottom, there are three panels for results: "CONFUSION MATRIX, MODEL ACCURACY & ERROR", "TRAINING RESULT: ID — ACTUAL — PREDICTION", and "TESTING RESULT: ID — PREDICTION". Each panel has an "Export" button (Export Model, Export Training, Export Testing).

Supplementary Figure 4. Selected Logistic Regression Classifier. **a.** Classifier definition is displayed, together with, **b.** the training methods with “Train Sample Size(%)” method selected, and **c.** the classifier parameters set to their default values.

Model training, evaluation, validation and result output:

You can now click 'submit' to train your classifier using the uploaded training, dependent, and target data in this example, and evaluate your result. Or, alternatively you can upload testing data first, and then click 'submit' to train and test a classifier on your data at the same time! For this example, **first**: we train a selected classifier, 'LogisticRegression', using its default parameters, and default train-validate split method 'Train Sample Size(%)', and then, **second**: we upload testing data to test the trained model.

After clicking 'submit', our selected classifier, 'LogisticRegression' for this example, is trained using the loaded training data, 'Dependent and Target' for this example.

Notes

ClassificalO always shuffles your training data before splitting to eliminate mini batch effects.

Internally, when 'Train Sample Size(%)' method is selected, ClassificalO uses the scikit-learn `train_test_split` function, to allow for fast training data split into training and validation subsets. With this method the parameter set is `train_size`, which takes the train sample size set by you (e.g. Train Sample Size (%): set to 75% means `train_size = 0.75` and `test_size = 0.25`).

If the 'K-fold Cross-Validation' method is selected instead, ClassificalO uses the scikit-learn `cross_val_predict` function where the training data is split into k-sets. The model is trained on k-1 of the folds followed by a validation step on the remaining part of the data. This will be repeated for each of the k-folds.

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 each data point ID is the first value, actual target value is displayed 2nd, and predicted target value third.

Testing data input and result output:

To test your trained model, first upload the 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 the testing data file, see **Supplementary Figure 1**. Once testing data is uploaded, the file name is automatically saved in the 'CURRENT DATA UPLOAD' panel to indicate that your file has been uploaded. The Testing Data file format is the same as for the dependent data file, see **Supplementary Figure 3.a**.

After clicking 'Submit', testing results are displayed in the 'TESTING RESULT: ID – PREDICTION' panel (**Supplementary Figure 6.b**) with each data point ID shown first, and the corresponding predicted target value displayed after it, separated by a hyphen.

The screenshot displays the ClassificationIO web interface. At the top, there are four main panels: 'UPLOAD TRAINING DATA FILES', 'CLASSIFIER SELECTION', 'UPLOAD TESTING DATA FILE' (labeled 'a'), and 'CURRENT DATA UPLOAD'. The 'CLASSIFIER SELECTION' panel shows '1: LogisticRegression' selected. The 'UPLOAD TESTING DATA FILE' panel has a 'Testing Data' button. The 'CURRENT DATA UPLOAD' panel shows the uploaded files: 'S1_Iris_Dependent', 'S2_Iris_Target.csv', and 'S3_Iris_Testing_DataSet.csv'.

Below these panels, the '1: LogisticRegression' section is expanded, showing a description and various parameters. The 'Train Sample Size (%)' is set to 75, and 'K-fold Cross-Validation' is set to 10. The 'Submit' button is visible.

Below the 'Submit' button, the 'PARAMETERS' section shows the following configuration: {random_state = None} {shuffle = True} {penalty = l2} {multi_class = ovr} {solver = liblinear} {max_iter= 100} {tol = 0.0001} {intercept_scaling = 1.0} {verbose = 0} {n_jobs = 1} {C = 1.0} {fit_intercept = True} {dual = False} {warm_start = False} {class_weight = None}.

At the bottom, there are three panels: 'CONFUSION MATRIX, MODEL ACCURACY & ERROR', 'TRAINING RESULT: ID — ACTUAL — PREDICTION', and 'TESTING RESULT: ID — PREDICTION' (labeled 'b'). The 'TESTING RESULT' panel shows 45 objects tested, with a list of IDs and predicted values (e.g., 3 — 0, 4 — 0, 8 — 0, 9 — 0, 10 — 0, 13 — 0, 15 — 0, 17 — 0, 19 — 0).

Supplementary Figure 6. Tested Logistic Regression Classifier. a. Upload testing data panel. **b.** Model tested using 45 data points.

Result export:

Now you are ready to export your trained model to preserve it for future use without having to retrain. Simply, click the 'Export Model' button (**Supplementary Figure 5.a**) and save your model. Your exported ClassificalO model can then be used for future testing on new data in the 'Already Trained My Model' window in ClassificalO, shown below.

The screenshot displays the ClassificalO web application interface. At the top, there is a navigation bar with 'HOME' and 'HELP' links. The main content area is divided into three columns for file uploads: 'UPLOAD TRAINING MODEL FILE' with a 'Model File' button, 'UPLOAD TESTING DATA FILE' with a 'Testing Data' button, and 'CURRENT DATA UPLOAD' which contains a text area with pre-filled data: '#Use My Own Training Data Upload', 'Dependent Data: S1_Iris_Dependent', 'Target Data: S2_Iris_Target.csv', 'Features Data: Not Uploaded', and 'Test Data: S3_Iris_Testing_DataSet.c'. Below these columns is a 'Submit' button. At the bottom, there are two side-by-side panels: 'CONFUSION MATRIX, MODEL ACCURACY & ERROR' and 'TESTING RESULT: ID — PREDICTION', both of which are currently empty. An 'Export Testing' button is located at the bottom right of the 'TESTING RESULT' panel.

ClassificalO model input:

You will need to upload ClassificalO 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 a ClassificalO trained model. Also, you will need to upload a testing data file (the testing data file format is the same as explained above), by clicking the 'Testing Data' button in the "UPLOAD TESTING DATA FILE" panel (**Supplementary Figure 7.b**). Once a ClassificalO model and testing data files are uploaded, files names are automatically displayed in the 'CURRENT DATA UPLOAD' panel (**Supplementary Figure 7.c**).

After clicking 'submit', the uploaded model preset parameters will populate (**Supplementary Figure 7.d**) to show the classifier used to originally 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 displayed in the 'Testing RESULT: ID – PREDICTION' panel (**Supplementary Figure 7.f**) with the data point ID shown first, followed by a hyphen and the predicted value displayed right after it.

The screenshot shows the ClassificalO web interface. At the top, there are 'HOME' and 'HELP' links. The main area is divided into several panels:

- a. UPLOAD TRAINING MODEL FILE:** Contains a 'Model File' button.
- b. UPLOAD TESTING DATA FILE:** Contains a 'Testing Data' button.
- c. CURRENT DATA UPLOAD:** Displays the status of the upload. It shows: '#Already Trained My Model Uploaded', 'Model: S5_LogisticRegression_IrisTr', 'Test Data: S3_Iris_Testing_DataSet.c', and '#Upload History'.
- Submit:** A button to submit the model and data.
- d. CLASSIFIER:** Displays the model parameters: 'PARAMETERS: ', 'random_state = None', 'shuffle = True', 'penalty = l2', 'multi_class = ovr', 'solver = liblinear', 'max_iter = 100', 'tol = 0.0001', 'intercept_scaling = 1.0', 'verbose = 0', 'n_jobs = 1', 'C = 1.0', 'fit_intercept = True', 'dual = False', 'warm_start = False', 'class_weight = None'.
- e. CONFUSION MATRIX, MODEL ACCURACY & ERROR:** Displays the confusion matrix and model evaluation metrics.

	Predicted Class		
True \ Class	0	1	2
0	7	0	0
1	0	9	0
2	0	2	9

Classification Accuracy: 92.59 %
Classification Error (MR): 7.41 %
- f. TESTING RESULT: ID – PREDICTION:** Displays the testing results for 45 objects.

ID	Prediction
3	0
4	0
8	0
9	0
10	0
13	0
15	0
17	0
19	0

Export Testing

Supplementary Figure 7. 'Already Trained My Model' window a. Upload ClassificalO trained model panel. b. Upload testing data panel. c. Current data upload panel with both model and testing data files names shown (red boxes). d. Model preset parameters. e. Trained model result and model evaluation (confusion matrix, model accuracy and error). f. Model testing result.

Results Export:

Full results (trained models, 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 later use, e.g. further analysis, publication, sharing, etc. (for more details on the export data file formats, see the Supplementary data files).

References

Anderson, E. The Irises of the Gaspé peninsula. *Bulletin of American Iris Society* 1935;59:2-5.

Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann Eugenics* 1936;7:179-188.

Pedregosa, F., *et al.* Scikit-learn: Machine Learning in Python. *J Mach Learn Res* 2011;12:2825-2830.