NBS-Predict

NBS-PREDICT USER MANUAL

Version 1.0.0-beta.13

Table of Contents

1 OVERVIEW 1

1.1 LITERATURE AND AIM 1
1.2 DEVELOPMENT AND TEST ENVIRONMENT 2
1.3 AUTHORS 2
1.4 REFERENCE 3
1.5 COPYRIGHT 3

2 THE NBS-PREDICT METHODOLOGY 4

3 GETTING STARTED WITH THE TOOLBOX 7

3.1 PREREQUISITES 7
3.2 INSTALLATION 7
3.3 SAMPLE DATA 7

4 WORKSPACE 8

4.1 GENERATING A NEW WORKSPACE 9
4.2 LOADING, REMOVING AND STARTING WORKSPACE 9
4.3 NON-EXISTENT WORKSPACES 9

5 ANALYSIS SETUP INTERFACE 10

5.1 THE DATA PANEL 12
5.1.1 CORRELATION MATRICES 12
5.1.2 BRAIN REGIONS (NODES) 12
5.2 THE MODELS PANEL 13
5.2.1 DESIGN MATRIX 13
5.2.2 CONTRAST: 14
5.2.3 ML MODELS 15
5.3 THE ADVANCED OPTIONS PANEL 16
5.3.1 K-FOLD 16

5.3.2 REPEATED CV 16
5.3.3 P-VALUE 16
5.3.4 SEED 17
5.3.5 PERMUTATION 17
5.3.6 METRIC 17
5.3.7 SCALING 18
5.3.8 CPU CORES 18
5.3.9 HYPERPARAMETER OPTIMIZATION 19

6 RESULTS VIEWER 21

6.1 DISPLAY OPTIONS PANEL 21
6.2 WEIGHT THRESHOLD 22
6.3 EVALUATE SUBNETWORK 23
6.4 DISPLAY PANEL 25
6.5 NODAL DEGREE TABLE 27
6.6 SAVE FIGURE 27
6.7 SAVE TABLE 28

7 ADVANCED TOPICS 29

7.1 HISTORY FILE 29
7.2 OUTPUT FILE 29
7.3 THE STRUCTURE OF THE OUTPUT FILE 29
7.3.1 NBSPREDICT.PARAMETER 30
7.3.2 NBSPREDICT.DATA 30
7.3.3 NBSPREDICT.FEATSELHANDLE 30
7.3.4 NBSPREDICT.INFO 30
7.3.5 NBSPREDICT.RESULTS 30
7.4 COMMAND LINE 31
7.5 MODEL EXTRACTION 32
7.6 TEST FUNCTION 33
7.7 SIMULATION 34
7.8 PLOTTING SIMULATION RESULTS 35
7.9 BUGS AND FEATURE REQUESTS 36

REFERENCES 37

1 Overview

1.1 Literature and Aim

The human brain's structural and functional connectivity has gained vast interest in neuroscience (Bullmore & Sporns, 2009). Graph models are among the most well-known models to investigate effect-associated connections (i.e., possible neuroimaging-based biomarkers) in structural or functional connectivity. To identify these connections, mass-univariate testing of hypotheses has been widely used by researchers. However, this common approach is associated with a major challenge of multiple comparisons (Benjamini, 2010). One method that overcomes this challenge is the Network-based Statistic (NBS) (Zalesky et al., 2010). NBS is a powerful method that controls the family-wise error rate in a weak sense, combining cluster-based permutation technique and the graph-theoretical concept of connected components. However, NBS comes with two drawbacks that originate in the traditional statistical method (i.e., GLM) used in its algorithm: it does not allow for (1) evaluation of generalizability of the results and (2) making predictions (i.e., decoding) about the outcome variable (e.g., diagnosis). Those two points are crucially important in the development of generalizable neuroimaging-based biomarkers.

The development of generalizable neuroimaging-based biomarkers is critical. Generalizable biomarkers can help to define diseases and aid diagnosis. However, erroneous biomarkers could lead to misdiagnosis and put innocent people's lives at risk. Cross-validation is a method to estimate the generalizability of biomarkers (Tabe-Bordbar et al., 2018). Studies that do not use any out-of-sample estimation technique (i.e. cross-validation) can fail to evaluate the generalizability of their results (Kriegeskorte et al., 2009; Shen et al., 2017).

Machine learning models are great tools to identify neuroimaging-based biomarkers to make predictions about many things, such as diagnosis and prognosis of a disease. However, the lack of interpretability is a common issue in machine learning models. The coefficients derived from machine learning models (even from the linear ones) are not straightforward to interpret (Haufe et al., 2014; Hebart & Baker, 2018).

To alleviate the lack of interpretability and lack of generalizability, we developed a new method called NBS-Predict. This new approach combines the powerful features of machine learning and the network-based statistic. NBS-Predict aims to provide a fast way to identify neuroimaging-based biomarkers with high generalizability by combining machine learning with graph theory in a cross-validation structure.

1.2 Development and Test Environment

NBS-Predict was developed on MATLAB, under a Mac OS environment. NBS-Predict has been successfully tested on MATLAB R2017b and R2018b under Windows 10 x64, Linux (Ubuntu x64), and Mac OS environments.

1.3 Authors

NBS-Predict was designed by Emin Serin[1,2], Andrew Zalesky[3], Johann D. Kruschwitz[4,] and Henrik Walter[4] and developed by Emin Serin.

[1] Berlin School of Mind and Brain, Humboldt Universität zu Berlin, Germany

[2] Einstein Center for Neurosciences Berlin, Charité – Universitätsmedizin Berlin, Germany

[3] Melbourne Neuropsychiatry Centre and Department of Biomedical Engineering, University of Melbourne, Australia

[4] Charité – Universitätsmedizin Berlin, Division of Mind and Brain Research, Department of Psychiatry and Psychotherapy, Berlin, Germany.

Mail to Authors:

Emin Serin – emin.serin@charite.de

Johann D. Kruschwitz - johann.kruschwitz@charite.de

1.4 Reference

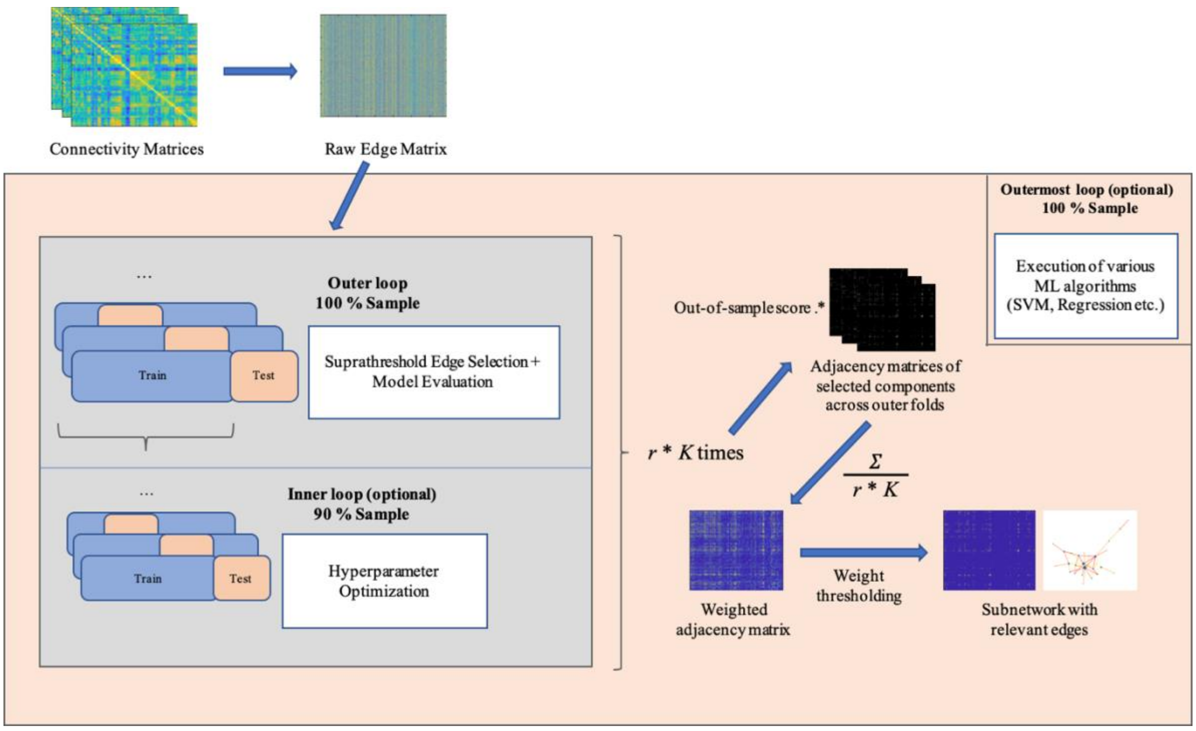
If you use the toolbox, please cite the following paper:

Serin, E., Zalesky, A., Matory, A., Walter, H., & Kruschwitz, J. D. (2021). NBS-Predict: A Prediction-based Extension of the Network-based Statistic. *NeuroImage*, 118625.

1.5 Copyright

2 The NBS-Predict Methodology

NBS-Predict operates in a cross-validation structure (nested if hyperparameter optimization is desired). The general algorithm consists of model evaluation, feature selection like suprathreshold edge selection, hyperparameter optimization (optionally), and machine learning algorithm optimization (optionally). The workflow of the NBS-Predict algorithm is depicted in Fig 1.



Connectivity Matrices
Raw Edge Matrix
Outermost loop (optional)
100 % Sample
Execution of various ML algorithms (SVM, Regression etc.)
Outer loop
100 % Sample
Train
Test
Suprathreshold Edge Selection + Model Evaluation
Out-of-sample score .*
Adjacency matrices of selected components across outer folds
r * K times
Σ / r * K
Inner loop (optional)
90 % Sample
Train
Test
Hyperparameter Optimization
Weighted adjacency matrix
Weight thresholding
Subnetwork with relevant edges

Figure 1. Workflow of NBS-Predict

Suprathreshold Edge Selection: The feature selection like suprathreshold edge selection is performed in the outer loop (also in the inner loop if hyperparameter optimization is desired) to detect a connected component comprising selected edges. The suprathreshold edge selection is the expansion of filter-based feature selection with the graph-theoretical concept of connected components, and it is the part of the algorithm that originated from the original NBS method (Zalesky et al., 2010). Briefly, a linear model fits each edge (i.e., feature), and the corresponding p-value is computed. The largest connected component that presents in the set of edges with p-values below a predefined p-value threshold (e.g., 0.05) is then selected as features in the machine learning model. The workflow of the suprathreshold edge selection algorithm is given in Fig. 2.

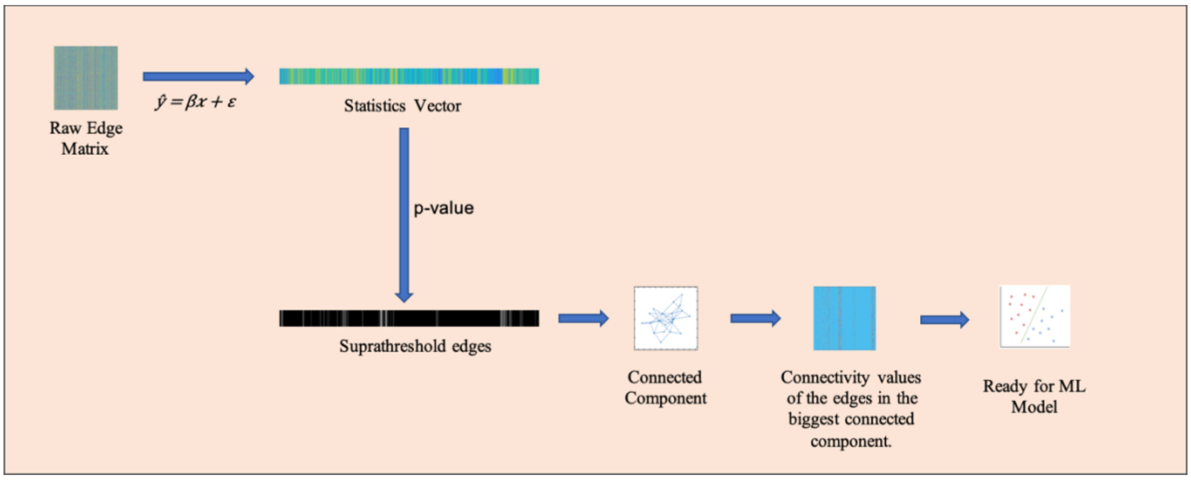

Raw Edge Matrix
$\hat{y} = \beta x + \varepsilon$
Statistics Vector
p-value
Suprathreshold edges
Connected Component
Connectivity values of the edges in the biggest connected component.
Ready for ML Model

Figure 2. Suprathreshold edge selection workflow.

Model Evaluation: Following the suprathreshold edge selection, model evaluation (training and testing) is done using these edges. The model's performance (e.g., classification accuracy) using the set of selected edges is assigned to the selected edges (0 is set to unselected edges). In this way, the presence of an edge in the evaluated model and the model's performance in the corresponding fold is represented as a single weight value for each edge. That allows us to evaluate the contribution of each edge to the overall model straightforwardly.

The suprathreshold edge selection and model evaluation steps are repeated *r* x *K* times, where *r* represents the number of repetitions of the CV structure, and *K* is the number of folds. The CV scheme is repeated *r* times to provide a more generalizable and realistic estimation of a model on a given dataset (Braga-Neto & Dougherty, 2004; Kim, 2009; Krstajic et al., 2014).

If desired, hyperparameter optimization for corresponding machine learning algorithms is performed in an additional inner loop. Specifically, in each fold of the inner loop, the suprathreshold feature selection and hyperparameter optimization are performed to find the parameters with the best prediction performance.

Optionally, users may run multiple machine learning algorithms to find the best performing algorithm on a specific dataset. If selected, the CV structure explained above runs in an additional non-CV loop, in which NBS-Predict runs with different machine learning algorithms suitable for the given problem (regression vs. classification) and returns the best performing algorithm.

3 Getting Started with the Toolbox

3.1 Prerequisites

NBS-Predict requires the following software and toolboxes to run properly:

MATLAB (2016b or newer)

Statistics and Machine Learning Toolbox

Parallel Computing Toolbox (optional)

3.2 Installation

The NBS-Predict toolbox is very straightforward to install. Download the newest release of the toolbox from https://github.com/eminSerin/NBS-Predict. After downloading, unzip the toolbox and put it in your preferred directory. Then, open the MATLAB window. Click **Set Path**, then click **Add with Subfolders** and locate the directory of the toolbox. Click **Save**.

Sometimes, having multiple custom toolboxes on your path might cause some problems if they overlap with each other. If you worry about that, navigate to the directory wherein NBS-Predict locates, using either the command window or **Current Folder** window. Then, either click **Add to Path - Selected Folders and Subfolders** or type the following command to add NBS-Predict to your MATLAB path:

```
addpath(genpath(pwd));
```

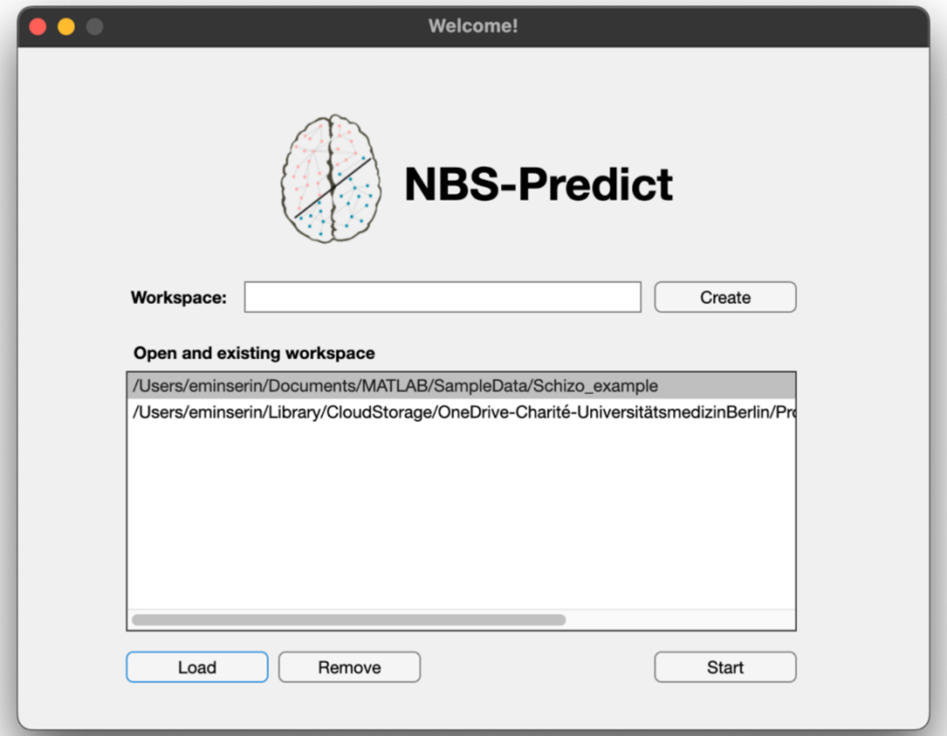
This way, NBS-Predict will be added to your path for this current session. Please keep in mind that if you use the second method, you need to add it to your path every time you open MATLAB.

3.3 Sample Data

You can download sample data using the following link: https://www.nitrc.org/docman/view.php/1517/179438/. This dataset comprises synthetic network data simulating regression and classification problems. Since the sample data is synthetic, please keep in mind that, results obtained following the analysis of this data do not imply any significant information. This sample dataset should only be used to check whether the toolbox works properly and to serve as an example input structure so that you can organize your input data accordingly.

4 Workspace

After downloading the toolbox and adding it to your MATLAB path, simply type "start_NBSPredict" into the MATLAB command line. It will launch NBS-Predict and open the welcome window using which you can create and load and start workspaces, as shown in Fig. 3.



/Users/eminserin/Documents/MATLAB/SampleData/Schizo_example
/Users/eminserin/Library/CloudStorage/OneDrive-Charité-UniversitätsmedizinBerlin/Pr

Figure 3. Welcome window

This welcome window is divided into two panels: 1) to create a new workspace, 2) to open an existing workspace.

4.1 Generating a New Workspace

In this panel, you see an editable text bar and the "Create" button. To create a new workspace, you simply write the workspace's name to create, and press enter. A window will pop up automatically to select the directory into which you want to create the workspace. Following choosing the directory, please hit the "Create" button. If the workspace is successfully generated, the "Create" button will turn into green color, and the workspace's directory will appear on the list box below. Suppose the workspace already exists in the directory you selected or cannot be created due to file permission issues. In that case, you will get an error dialog window informing you about the problems.

4.2 Loading, Removing and Starting Workspace

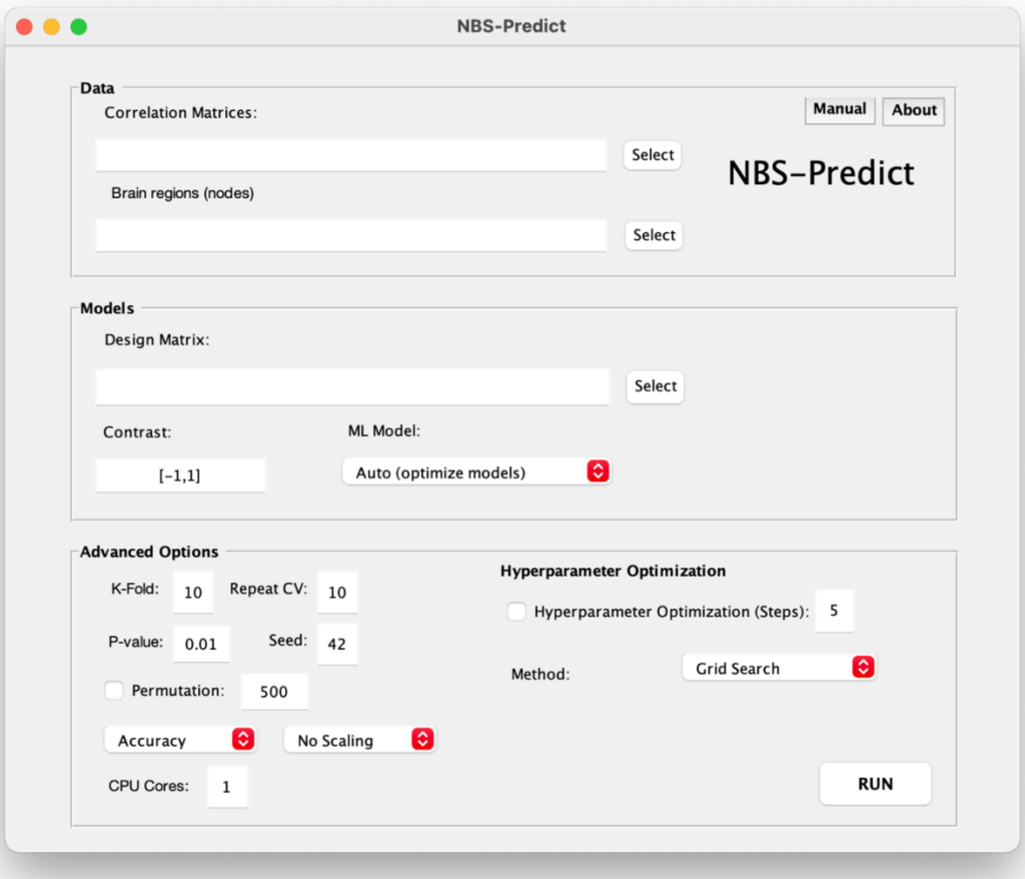
In this panel, there are a list box showing the workspaces in the workspace database, “Load”, “Remove” and “Start” buttons. To load an existing workspace, simply hit the “Load” button and select the workspace in the window appeared. The loaded workspace will automatically appear on the list box. To remove workspace from the workspace database, select the workspace you want to remove and hit the “Remove” button. To start workspace, please choose the workspace from the list box, and hit the “Start” button. Then, the welcome window will automatically close, the MATLAB’s current directory will be set to the workspace directory, and the NBS-Predict analysis setup window will open.

4.3 Non-existent Workspaces

All the generated or loaded workspaces will be saved in a workspace database located in the main NBS-Predict directory (“workspaces.mat”). When you start the NBS-Predict welcome window by simply typing “start_NBSPredict” into the MATLAB command line, NBS-Predict checks whether the workspaces in the database still exist in given directories. If not, the non-existent workspaces will be automatically removed from the workspace database and the list box on the workspace window. In this case, please check MATLAB’s command window for warnings about the non-existent workspaces. If you experience NBS-Predict welcome window crashing or messed up workspace, please simply delete the “workspaces.mat” file in the main directory and let us know via our emails or GitHub.

5 Analysis Setup Interface

After launching the workspace, you will encounter one of two versions of the NBS-Predict analysis window, depending on your MATLAB version. If you are using a version older than R2021b, the analysis window will appear as shown in Figure 4. However, if you are using a version R2021b or newer, the analysis window will have a different interface (see Fig 5).



Data
Correlation Matrices:
Brain regions (nodes)

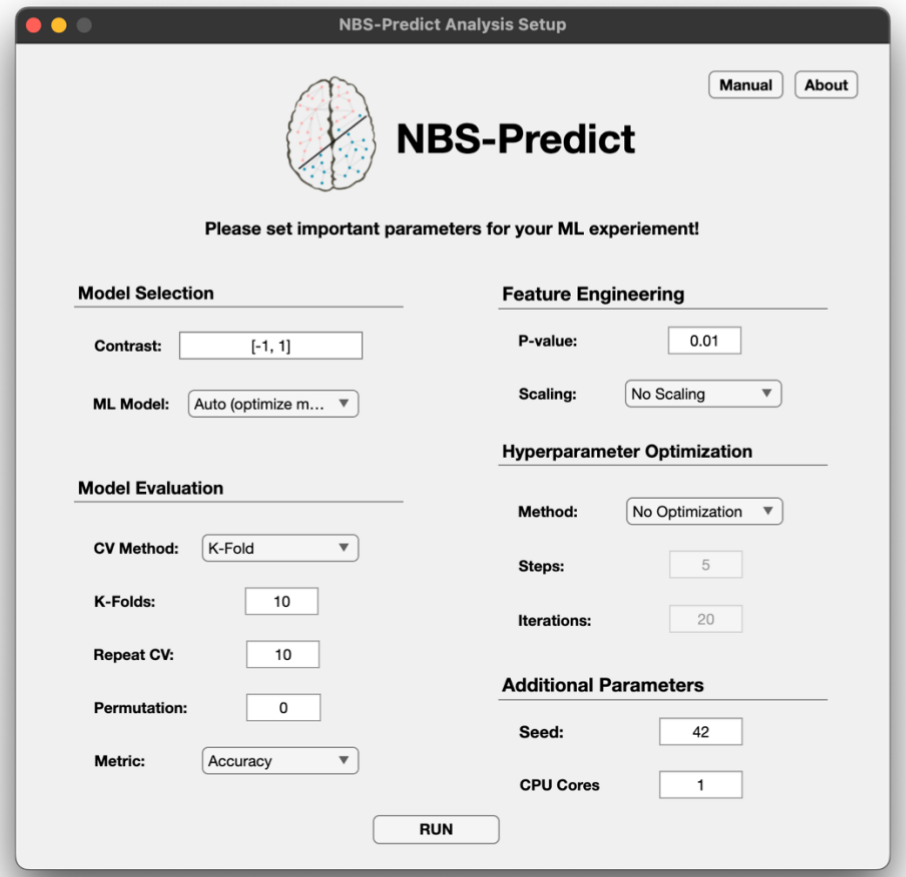
Models
Design Matrix:
Contrast: [-1,1]
ML Model: Auto (optimize models)

Advanced Options
K-Fold: 10
Repeat CV: 10
P-value: 0.01
Seed: 42
Permutation: 500
Accuracy
No Scaling
CPU Cores: 1

Hyperparameter Optimization
Hyperparameter Optimization (Steps): 5
Method: Grid Search

Figure 4. Analysis setup window

The analysis window is divided into three panels: Data, Models and Advanced Options.


NBS-Predict Analysis Setup

Manual | About

NBS-Predict

Please set important parameters for your ML experiement!

Model Selection

Contrast: [-1, 1]

ML Model: Auto (optimize m...

Model Evaluation

CV Method: K-Fold

K-Folds: 10

Repeat CV: 10

Permutation: 0

Metric: Accuracy

Feature Engineering

P-value: 0.01

Scaling: No Scaling

Hyperparameter Optimization

Method: No Optimization

Steps: 5

Iterations: 20

Additional Parameters

Seed: 42

CPU Cores 1

RUN

Figure 5. Updated analysis setup window

The key distinction between these two versions is that the updated analysis setup window eliminates the need for users to manually load correlation matrices, brain regions, and the design matrix. Instead, it assumes that the required data has already been placed in the working directory according to a specific structure outlined below:

```
sample_project
        matrices
                subject_001.mat
                subject_002.mat
                …
        design.mat
        BrainRegions.csv
```

For detailed instructions on the structure and format of each file, please refer to sections 5.1.1, 5.1.2, and 5.2.1.

5.1 The Data Panel

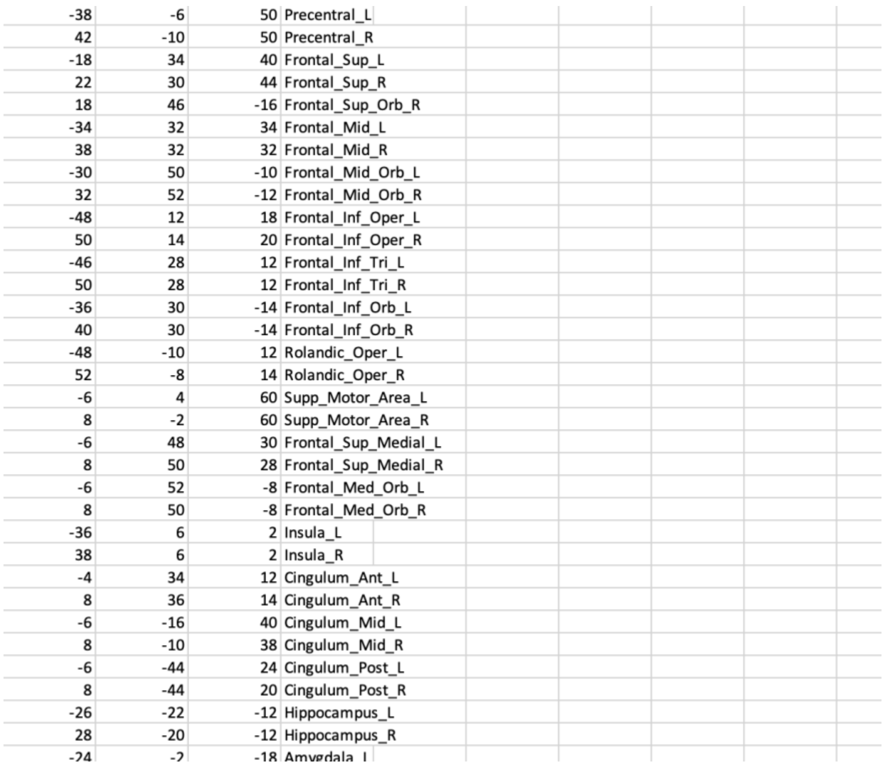
5.1.1 Correlation Matrices

Here, you can browse and select the directory containing connectivity matrices. Subjects' correlation matrices should be stored in. mat or .csv file format. If you do not have connectivity matrices yet, you can use the GraphVar Toolbox (Kruschwitz et al., 2015) to create them quickly.

Here, you can browse and select the directory containing connectivity matrices. Subjects' correlation matrices should be stored in. mat or .csv file format. If you do not have connectivity matrices yet, you can use the GraphVar Toolbox (Kruschwitz et al., 2015) to create them quickly.

5.1.2 Brain Regions (nodes)

Here, you can select the brain parcellation file (.mat or .csv format) containing node coordinates and labels. In the brain parcellation file, there should be 4 columns. The first 3 columns should be x, y, and z coordinates of nodes, and the 4th column should include the node labels (see Fig. 6). Please keep in mind that, parcellation .csv file **must not** contain column names. Please check the sample data for the example parcellation file structure.



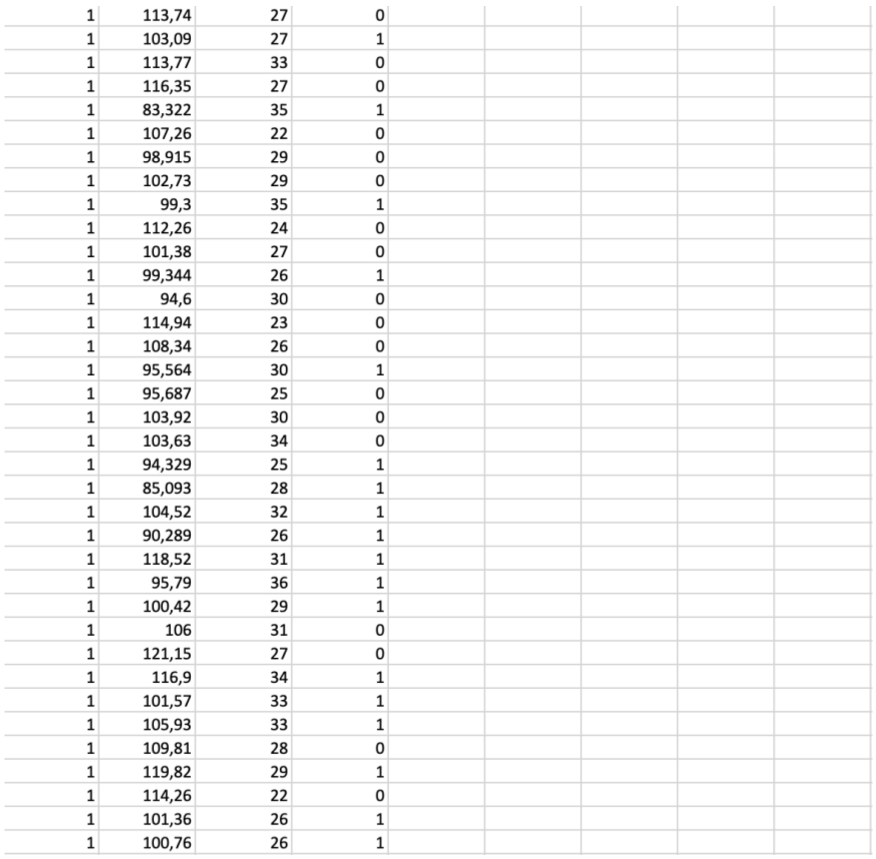
-38	-6	50	Precentral_L
42	-10	50	Precentral_R
-18	34	40	Frontal_Sup_L
22	30	44	Frontal_Sup_R
18	46	-16	Frontal_Sup_Orb_R
-34	32	34	Frontal_Mid_L
38	32	32	Frontal_Mid_R
-30	50	-10	Frontal_Mid_Orb_L
32	52	-12	Frontal_Mid_Orb_R
-48	12	18	Frontal_Inf_Oper_L
50	14	20	Frontal_Inf_Oper_R
-46	28	12	Frontal_Inf_Tri_L
50	28	12	Frontal_Inf_Tri_R
-36	30	-14	Frontal_Inf_Orb_L
40	30	-14	Frontal_Inf_Orb_R
-48	-10	12	Rolandic_Oper_L
52	-8	14	Rolandic_Oper_R
-6	4	60	Supp_Motor_Area_L
8	-2	60	Supp_Motor_Area_R
-6	48	30	Frontal_Sup_Medial_L
8	50	28	Frontal_Sup_Medial_R
-6	52	-8	Frontal_Med_Orb_L
8	50	-8	Frontal_Med_Orb_R
-36	6	2	Insula_L
38	6	2	Insula_R
-4	34	12	Cingulum_Ant_L
8	36	14	Cingulum_Ant_R
-6	-16	40	Cingulum_Mid_L
8	-10	38	Cingulum_Mid_R
-6	-44	24	Cingulum_Post_L
8	-44	20	Cingulum_Post_R
-26	-22	-12	Hippocampus_L
28	-20	-12	Hippocampus_R
-24	-2	-18	Amygdala_L

Figure 6. Brain parcellation file

5.2 The Models Panel

5.2.1 Design Matrix

Here, you can select a design matrix (.mat or .csv format) that specifies the statistical test, which will be used to specify the statistical model required for the suprathreshold feature selection algorithm. The design matrix used in NBS-Predict is the same as in general linear models and in NBS (Zalesky et al., 2010). The first 2 columns should contain one-hot encoded data labels (e.g., 0 for healthy controls, 1 for patients with disease) for classification problems. For regression problems (see Fig. 7), the first column should be the column of ones (i.e., intercept term), and the second column should represent the target values (e.g., intelligence scores). Please keep in mind that NBS-Predict does not support multilabel classifications (e.g., classifying more than 2 diseases). As shown in Fig. 7, the additional columns represent covariances. If covariances are provided in the design matrix and correctly specified in the contrast panel (see **5.2.2**), NBS-Predict will perform cross-validated confound regression prior to suprathreshold feature selection using the method shown in Snoek et al., 2019. Design matrix **must not** contain any column name.



1	113,74	27	0
1	103,09	27	1
1	113,77	33	0
1	116,35	27	0
1	83,322	35	1
1	107,26	22	0
1	98,915	29	0
1	102,73	29	0
1	99,3	35	1
1	112,26	24	0
1	101,38	27	0
1	99,344	26	1
1	94,6	30	0
1	114,94	23	0
1	108,34	26	0
1	95,564	30	1
1	95,687	25	0
1	103,92	30	0
1	103,63	34	0
1	94,329	25	1
1	85,093	28	1
1	104,52	32	1
1	90,289	26	1
1	118,52	31	1
1	95,79	36	1
1	100,42	29	1
1	106	31	0
1	121,15	27	0
1	116,9	34	1
1	101,57	33	1
1	105,93	33	1
1	109,81	28	0
1	119,82	29	1
1	114,26	22	0
1	101,36	26	1
1	100,76	26	1

Figure 7. Design matrix

5.2.2 Contrast:

Here, you need to specify a contrast vector for the statistical model used in the suprathreshold feature selection algorithm (e.g. [-1, 1] for group 1 < group 2; review FEAT User Guide for further examples). The contrast vector should be the same length as the number of columns in the design matrix. Keep in mind that the contrast values should be within square brackets. Unlike the common GLM method, confound control is done before the statistical model; thereby t-test could still be done if you enter nuisance covariates (i.e., no need to model ANCOVA design). To do this, simply specify contrast values for groups and nuisance variables (e.g. [1, -1, 0, 0]).

5.2.3 ML Models

Here, you select a machine learning algorithm to perform your predictions. NBS-Predict will allow you to select suitable machine learning algorithms for a given design matrix[1]. You can either select a specific machine learning algorithm or **Auto (optimize models)**, which runs all suitable machine learning algorithms one by one**.** The available machine learning algorithms in the current version of NBS-Predict are given in Table 1. Importantly, as of NBS-Predict v1.0.0-beta.9, the decision tree classifier and regressor were dropped due to possible compatibility issues with confound regression. Specifically, the decision tree algorithms are simple yet non-linear ML algorithms that can map between target and feature set non-linearly. However, the cross-validated confound correction only removes the linear association between confounding variables and feature set, possibly leaving the non-linear ones intact (Snoek et al., 2019). Therefore, although they are simply algorithms, the decision tree algorithms can still detect the non-linear patterns in the feature set associated with confounding variables.

Table 1*. List of available machine learning algorithms in NBS-Predict and their hyperparameters and their corresponding spacing scales and possible ranges.*

	ML Algorithm	**Hyperparameter**	**Spacing Scale**	**Range**
Regression	Linear Regression	Lambda (L_2)	logarithmic	10^{-2} to 10^{3}
	Linear Support Vector Regression	Lambda (L_2)	logarithmic	10^{-2} to 10^{3}
Classification	Logistic Regression	Lambda (L_2)	logarithmic	10^{-2} to 10^{3}
	Linear Support Vector Classification	Lambda (L_2)	logarithmic	10^{-2} to 10^{3}
	Linear Discriminant Analysis	Gamma	linear	0 to 1

[1] To do this, NBS-Predict checks the second column in the design matrix. NBS-Predict allows you to select classification algorithms if the second column contains categorical values (e.g., 0 and 1) and regression algorithms if it contains continuous values. Please keep in mind that NBS-Predict only allows for binary classification problems. More than two unique values found in the second column in the design matrix (i.e., target variable) would be considered as regression problems.

5.3 The Advanced Options Panel

5.3.1 K-Fold

Here, you can set the number of folds in the k-fold cross-validation procedure. The number of folds should be at least 2.

5.3.1.1 Leave-One-Out Cross Validation (LOOCV)

Please keep in mind that <u>you can only run LOOCV for classification problems</u>.
To do that, you can toggle between K-Fold and LOOCV on the updated analyses setup window, while you need to set K-Fold to -1 on the old setup window.

5.3.2 Repeated CV

This number defines how many times the CV procedure is repeated. As k-fold CV is a stochastic model evaluation technique, it gives slightly different estimates of a model in each execution (Braga-Neto & Dougherty, 2004). Repeating K-fold CV *r* times reduces this variability in the estimation of model performance by generating an average estimate of K-fold CV.

This number defines how many times the CV procedure is repeated. As k-fold CV is a stochastic model evaluation technique, it gives slightly different estimates of a model in each execution (Braga-Neto & Dougherty, 2004). Repeating K-fold CV *r* times reduces this variability in the estimation of model performance by generating an average estimate of K-fold CV.

5.3.3 P-value

P-value is a thresholding parameter used in the suprathreshold edge selection algorithm. Features (i.e., edges) with a p-value below the given p-value threshold are selected for further process and training. This parameter has a direct impact not only on the number of features selected in each fold but also on the weight distribution of the weighted network provided after analysis. You should take the size of an input network into account when you define a p-value threshold. A p-value threshold of 0.01 is a good start for most cases. However, if you want to analyze data with very small or immense network size, you should consider using a higher or lower p-value threshold than the default value, respectively.

5.3.4 Seed

This number specifies the seed for MATLAB random number generator. This way, you could store the state for the random number generator, thus could get the same results in further executions of NBS-Predict using the same parameters. If you run your NBS-Predict in parallel, NBS-Predict automatically sets different seed numbers for each worker using your defined seed number.[2] If you want to randomize the random number generator, simply enter **-1**.

5.3.5 Permutation

This allows evaluating the significance of the model's CV score using permutation testing. Permutation testing is done by permuting the outcome variables (i.e., target), then computing the p-value against null-hypothesis, which proposes features and outcome variables are independent. The p-value represents the percentage of cases where the prediction performance of randomized data is equal to or better than the tested model's performance (Read more in the scikit-learn's user guide).

5.3.6 Metric

Here, you can select the performance metric used to evaluate the model's performance. NBS-Predict automatically provides you the suitable metrics for your data (see Footnote 1 for details). The list of available performance metrics is given in Table 2.

Table 2. *List of available performance metrics in NBS-Predict*

Regression	R^2	Coefficient of determination. Ranges from 1 (perfect) to 0.
	Correlation	Pearson's correlation coefficient.

[2] It uses consecutives numbers starting from the defined seed number for the random number generator in each worker. Let's say you define 42 as seed, and you utilize 4 CPU cores during your analysis, NBS-Predict automatically uses 42, 43, 44, and 45 as seed numbers in random number generators in the workers.

	Explained variance	How much variance of an observed data is described by a model? Ranges from 1 (perfect) to 0.
	MSE	Mean squared error. Lower values represent a better fit.
	RMSE	Root means squared error. Lower values represent a better fit.
	MAD	Median absolute deviation. Lower values represent a better fit.
Classification	Accuracy	The ratio of true positives. Ranges from 1 (perfect) to 0.
	Balanced Accuracy	The average of sensitivity obtained on each class.
	Sensitivity (i.e., Recall)	The relative number of actual positives identified correctly. Ranges from 1 (perfect) to 0.
	Specificity	The fraction of true negatives identified correctly. Ranges from 1 (perfect) to 0.
	Precision	The fraction of true positives. Ranges from 1 (perfect) to 0.
	F_1	Harmonic average of precision and recall.
	Matthew's CC	Matthews Correlation Coefficient. Ranges from 1 (perfect agreement) to -1 (perfect disagreement) and 0 indicates random prediction.
	Cohen's Kappa	The measure of inter-rater reliability. Ranges from 1 (perfect) to 0.
	AUC	The area under the ROC Curve. Ranges from 1 (perfect) to 0.

5.3.7 Scaling

This option allows you to scale your feature datasets prior to analysis in each fold. For further details, please refer to the user guide on the scaling methods written by scikit-learn (Pedregosa et al., 2011).

5.3.8 CPU Cores

This option allows you to run NBS-Predict in parallel. You can set the number of CPU cores to use for your analyses. To utilize all the CPU cores, please set it to -1. MATLAB uses **only**

physical cores due to hyperthreading. Note that you need the **Parallel Computing Toolbox** installed on your computer to use this option.

5.3.9 Hyperparameter Optimization

This option allows you to optimize the hyperparameters for corresponding machine learning algorithms. Click the checkbox to activate the hyperparameter optimization.

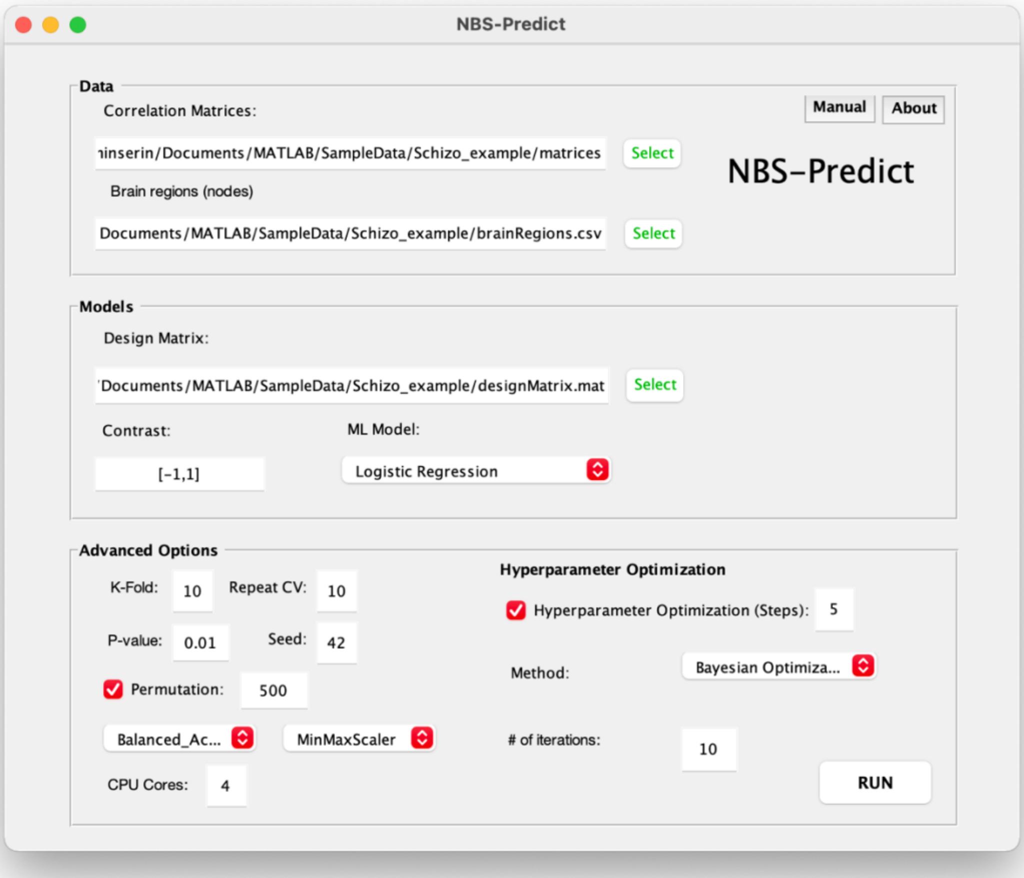
5.3.9.1 Hyperparameter Optimization (Steps)

This number represents the number of optimization steps for each hyperparameter for machine learning algorithms. It directly determines the size of the parameter space on which a searching algorithm runs. Higher numbers allow for finer grained hyperparameter optimization with the cost of computational power and time, while lower numbers allow for coarse-grained but fast optimization. The list of hyperparameters and their corresponding ranges is given in Table 1. The current version of NBS-Predict does not allow users to set a custom range for hyperparameters, and this option will be added in future updates.

5.3.9.2 Method

Here, you can select the searching method used in hyperparameter optimization. The current version of NBS-Predict allows users to select **Grid Search**, **Random Search,** and **Bayesian Optimization**[3] algorithms to select the best performing hyperparameters (refer Tune parameters, Bayesian optimization, Hyperparameter optimization for details). If **Random search** or **Bayesian optimization** is selected, NBS-Predict allows you to define the total number of iterations that the algorithm will run to search parameters (see Fig. 8). This option will be hidden if **Grid Search** is selected as grid search exhaustively considers every single combination of parameters.

[3] Expected Improvement is used as an acquisition function in Bayesian optimization.


NBS-Predict

Data

Correlation Matrices:

ıinserin/Documents/MATLAB/SampleData/Schizo_example/matrices

Brain regions (nodes)

Documents/MATLAB/SampleData/Schizo_example/brainRegions.csv

Models

Design Matrix:

'Documents/MATLAB/SampleData/Schizo_example/designMatrix.mat

Contrast: [-1,1]

ML Model: Logistic Regression

Advanced Options

K-Fold: 10 Repeat CV: 10

P-value: 0.01 Seed: 42

✓ Permutation: 500

Balanced_Ac... MinMaxScaler

CPU Cores: 4

Hyperparameter Optimization

✓ Hyperparameter Optimization (Steps): 5

Method: Bayesian Optimiza...

# of iterations: 10

Figure 8. Analysis setup window after loading data and setting parameters

6 Results Viewer

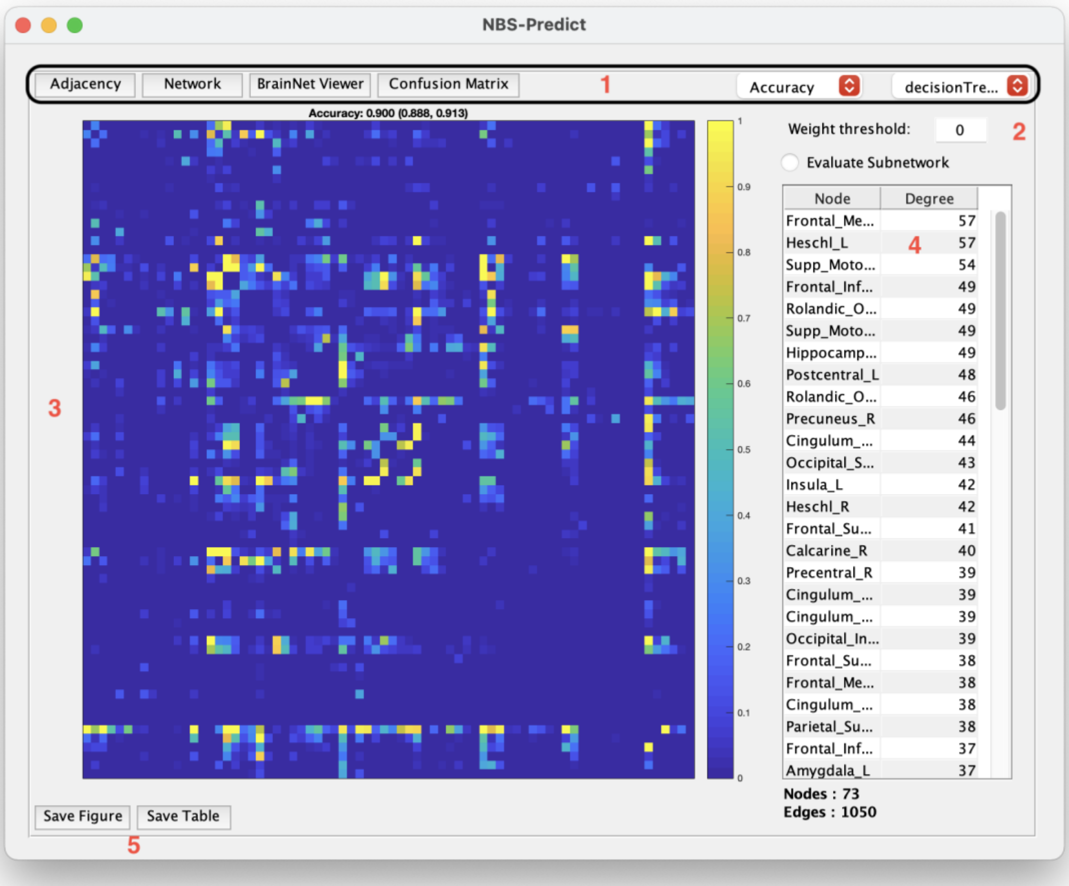
As shown in Fig. 8, the **Results Viewer** of NBS-Predict is roughly divided into 5 sections: (1) allows for selection of different figures, performance metrics, and machine learning algorithms (if **Auto (optimize models)** option is selected in the **Analysis Setup Window**) to visualize results; (2) allows to set weight threshold to visualize subnetwork and evaluate its out-of-sample performance; (3) display view for the figures; (4) table indicating the nodal degree of brain regions presented in the network; (5) and allows to save figures and tables. The number of edges and nodes present in the network is shown below the nodal degree table. If you run NBS-Predict using multiple machine learning algorithms, the **Results Viewer** will show results from the best-performing algorithm by default.

6.1 Display Options Panel

Here, you can select to display weighted adjacency matrix (Adjacency, see Fig. 9), weighted network on a circular graph[4] (Network, see Fig. 10), weighted network on a 3D brain surface[5] (Xia et al., 2013) (BrainNet Viewer, see Fig. 11) and confusion matrix (Confusion Matrix, see Fig. 12). In this panel, you can also find pop-up menus to select various suitable performance metrics and machine learning algorithms (if the Auto (optimize models) option is selected in the Analysis Setup Window). After selecting a performance metric or machine learning algorithm, the result viewer window will be automatically updated.

[4] NBS-Predict uses the circularGraph function developed by Paul Kassebaum to visualize the network on a circular graph.

[5] BrainNet Viewer was adapted into NBS-Predict to visualize the network on a 3D brain surface.



Weight threshold: 0

Evaluate Subnetwork

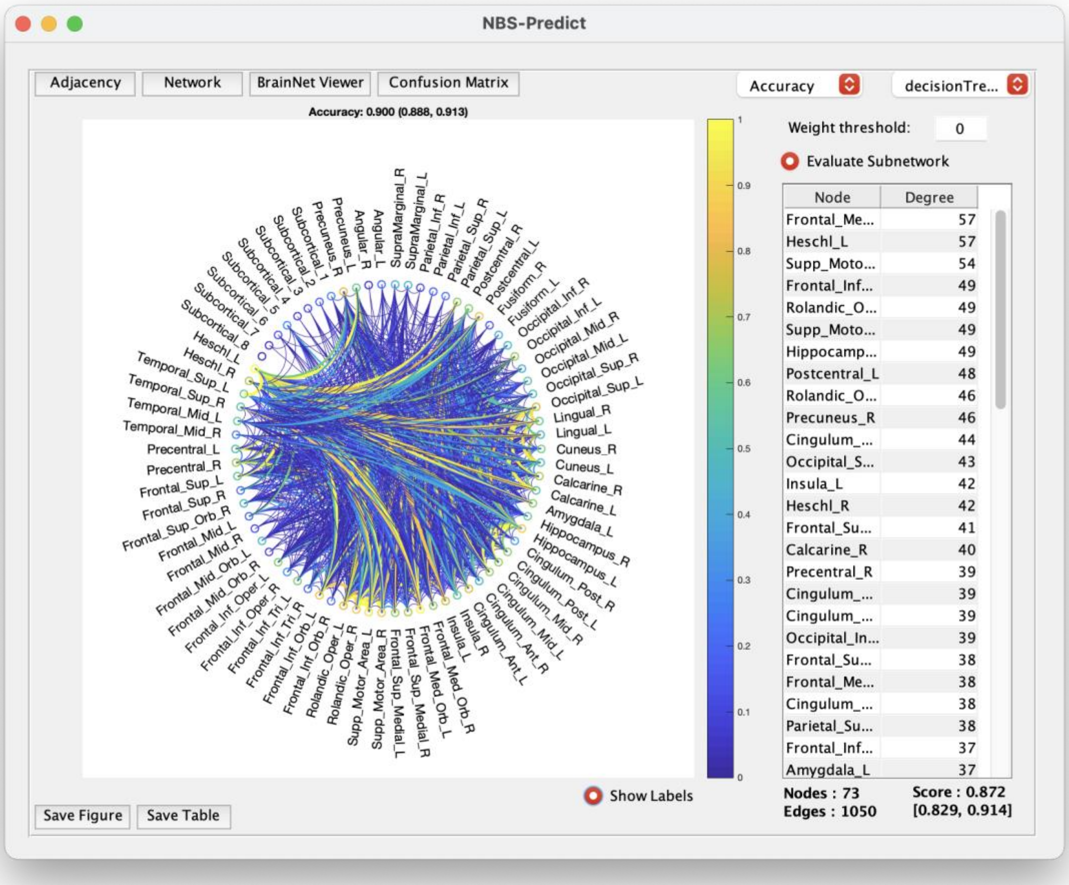
Node	Degree
Frontal_Me...	57
Heschl_L	57
Supp_Moto...	54
Frontal_Inf...	49
Rolandic_O...	49
Supp_Moto...	49
Hippocamp...	49
Postcentral_L	48
Rolandic_O...	46
Precuneus_R	46
Cingulum_...	44
Occipital_S...	43
Insula_L	42
Heschl_R	42
Frontal_Su...	41
Calcarine_R	40
Precentral_R	39
Cingulum_...	39
Cingulum_...	39
Occipital_In...	39
Frontal_Su...	38
Frontal_Me...	38
Cingulum_...	38
Parietal_Su...	38
Frontal_Inf...	37
Amygdala_L	37

Nodes : 73
Edges : 1050

Figure 9. Results viewer window. A weighted adjacency matrix is presented on the display panel.

6.2 Weight threshold

Here, you can set a weight threshold to visualize a subnetwork comprising important features. This weight threshold is a cut-off for the contribution of edges to the overall model performance. That allows for a straightforward interpretation of the results and the contribution of the edges to the overall model. Hence, you could evaluate the robustness and importance of a biomarker in a neuroimaging-based prediction. Notably, this weight threshold is arbitrary, but there are several strategies to pick a good one. One strategy would be to pick the weight threshold of 1 to obtain the subnetwork comprising the most relevant edges. If this is very conservative for your specific data, you could pick more lenient thresholds (e.g., 0.9 - 0.8). However, the best strategy would be to report subnetworks derived after applying conservative and more lenient thresholds along with the full weighted network to provide more information on the distribution of edge weights and the structure of the whole brain network.



Accuracy | decisionTre...

Weight threshold: 0

Evaluate Subnetwork

Node	Degree
Frontal_Me...	57
Heschl_L	57
Supp_Moto...	54
Frontal_Inf...	49
Rolandic_O...	49
Supp_Moto...	49
Hippocamp...	49
Postcentral_L	48
Rolandic_O...	46
Precuneus_R	46
Cingulum_...	44
Occipital_S...	43
Insula_L	42
Heschl_R	42
Frontal_Su...	41
Calcarine_R	40
Precentral_R	39
Cingulum_...	39
Cingulum_...	39
Occipital_In...	39
Frontal_Su...	38
Frontal_Me...	38
Cingulum_...	38
Parietal_Su...	38
Frontal_Inf...	37
Amygdala_L	37

Nodes : 73
Edges : 1050

Score : 0.872
[0.829, 0.914]

Figure 10. A plot of a weighted network on a circular graph inside NBS-Predict Results Viewer.

6.3 Evaluate Subnetwork

By clicking this option, you can simply evaluate the out-of-sample performance of the suprathreshold subnetwork.[6] It is important that you should not use the out-of-sample performance of the suprathreshold subnetwork to choose an "optimal" weight threshold since that could lead to a severe overfitting problem. The results will be shown on the right-hand side below the table (see Fig. 10).

[6] The out-of-sample performance of the suprathreshold subnetwork is evaluated using 10-repeated 10-fold cross-validation procedure. The current version of the toolbox does not allow for hyperparameter optimization. Scaling and confound regression will be done if selected.

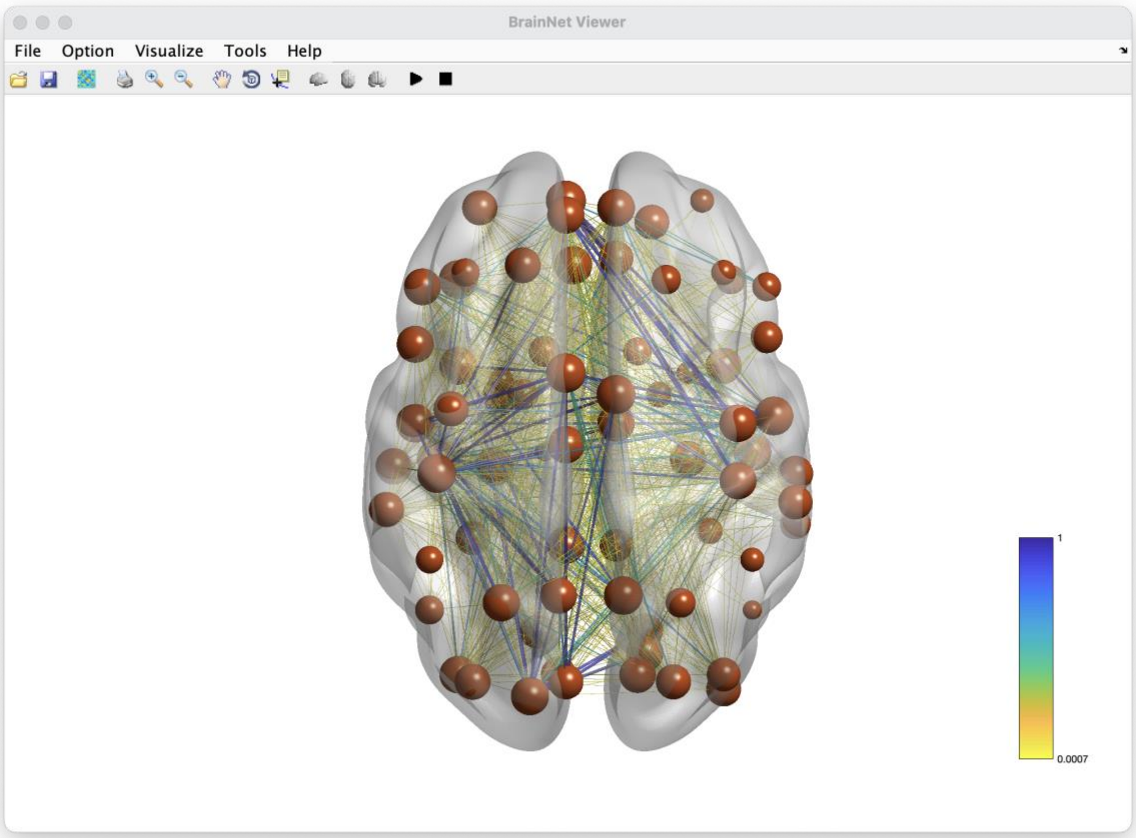
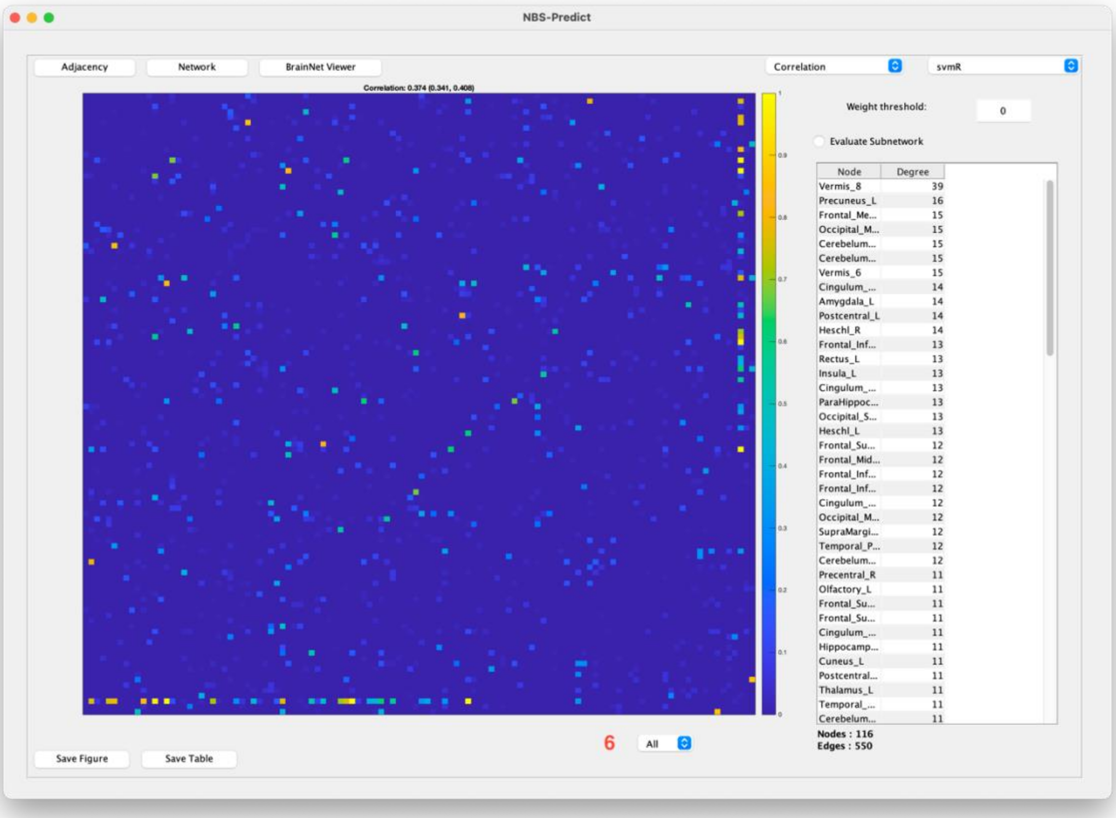


Figure 11. Weighted network on a 3D brain surface.



Node	Degree
Vermis_8	39
Precuneus_L	16
Frontal_Me...	15
Occipital_M...	15
Cerebelum...	15
Cerebelum...	15
Vermis_6	15
Cingulum_...	14
Amygdala_L	14
Postcentral_L	14
Heschl_R	14
Frontal_Inf...	13
Rectus_L	13
Insula_L	13
Cingulum_...	13
ParaHippoc...	13
Occipital_S...	13
Heschl_L	13
Frontal_Su...	12
Frontal_Mid...	12
Frontal_Inf...	12
Frontal_Inf...	12
Cingulum_...	12
Occipital_M...	12
SupraMargi...	12
Temporal_P...	12
Cerebelum...	12
Precentral_R	11
Olfactory_L	11
Frontal_Su...	11
Frontal_Su...	11
Cingulum_...	11
Hippocamp...	11
Cuneus_L	11
Postcentral...	11
Thalamus_L	11
Temporal_...	11
Cerebelum...	11

Nodes : 116
Edges : 550

Figure 12. Results viewer window in a regression problem. A weighted adjacency matrix is presented on the display panel.

6.4 Display Panel

Here, you can display the selected plots. Specifically, the weighted adjacency matrix and the weighted networks on a circular graph plot offer interactive features. By clicking on edges and nodes in these plots, you can access more detailed information such as nodal degree and edge weight. If you choose to view the plots using the Network or BrainNet Viewer option, you will notice the "Show Labels" option located in the lower left corner below the display panel. Enabling this option allows you to include node labels in these plots.

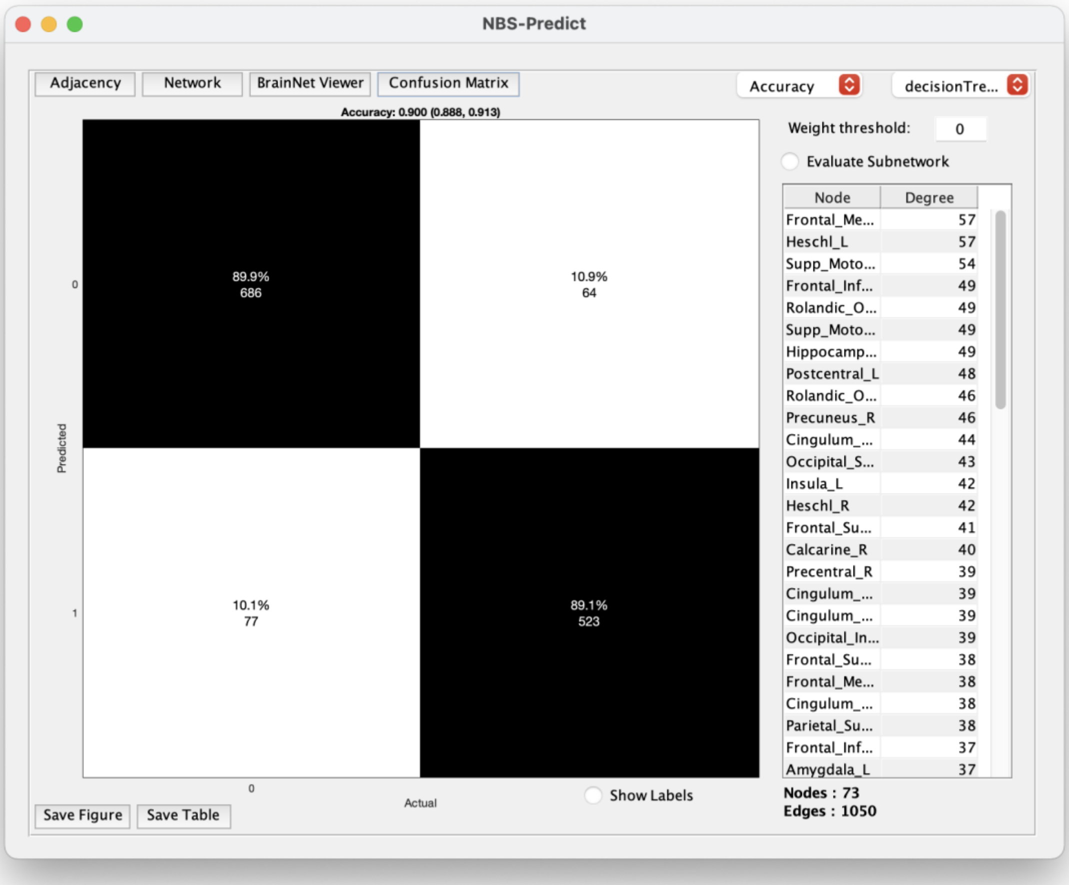
In cases where regression analysis has been performed, users have the option to visualize combined, positive, or negative networks (Fig. 12) by utilizing the drop-down list available below the display panel. This feature enables you to explore the relationship between relevant edges and the target variable. The direction of the relationship is determined as follows:

- In each outer cross-validation (CV) fold, the pairwise correlation between edges and the target variable is computed and stored in a matrix.
- After the completion of nested-CV, these correlation coefficients are summed to yield a single coefficient for each edge.
- The direction of the summed coefficient (positive or negative) is then assigned to the corresponding edge in the outcome weighted network.

The confusion matrix (Fig. 13) is displayed as:

True Negative, False Negative
False Positive, True Positive

The numbers displayed in each block represent the total observations across all outer folds. For example, if you observe a value of 686 in the true positives block, it signifies the total number of true positives across all outer folds.



Weight threshold: 0

Evaluate Subnetwork

Node	Degree
Frontal_Me...	57
Heschl_L	57
Supp_Moto...	54
Frontal_Inf...	49
Rolandic_O...	49
Supp_Moto...	49
Hippocamp...	49
Postcentral_L	48
Rolandic_O...	46
Precuneus_R	46
Cingulum_...	44
Occipital_S...	43
Insula_L	42
Heschl_R	42
Frontal_Su...	41
Calcarine_R	40
Precentral_R	39
Cingulum_...	39
Cingulum_...	39
Occipital_In...	39
Frontal_Su...	38
Frontal_Me...	38
Cingulum_...	38
Parietal_Su...	38
Frontal_Inf...	37
Amygdala_L	37

Nodes : 73
Edges : 1050

Figure 13. Confusion matrix plot inside NBS-Predict Results Viewer.

6.5 Nodal degree table

Here, you can review the nodal degree of nodes presented in the weighted network (or suprathreshold network) (see Fig 9-13).

6.6 Save Figure

This button allows you to save the figure presented in the display panel[7].

[7] Available formats are: .png, .fig, .jpeg, .tiff

6.7 Save Table

This button allows you to save the nodal degree table[8].

[8] Available formats are: .txt, .mat, .xlsx, .csv

7 Advanced Topics

7.1 History File

After you start the analysis by clicking the "Run" button on the **Analysis Setup Window** (see Fig. 8), NBS-Predict will automatically save a history file in the application's main directory. This history file stores all parameters and directories entered for the current run. Additionally, it will be automatically loaded and used in further runs using the parameters stored so that users do not need to reenter the parameters for the same data. If you would like to analyze a different dataset and use another parameter, simply navigate the main directory of NBS-Predict and remove the "history.mat" file.

7.2 Output File

Following the analysis, NBS-Predict automatically saves a "NBSPredict.mat" file under the "/Results/{current date}" directory in the input's directory[9]. This file stores parameters, directories, and results. If a user runs analysis multiple times, the output file will not be overwritten on the existing file; instead, it will be saved as "NBSPredict(number).mat" (e.g. "NBSPredict2.mat"). This file could also be used to run the same analysis with the same parameters or open the NBS-Predict **Results Viewer** window (see Fig. 9) using the command line (see Section **7.4.**).

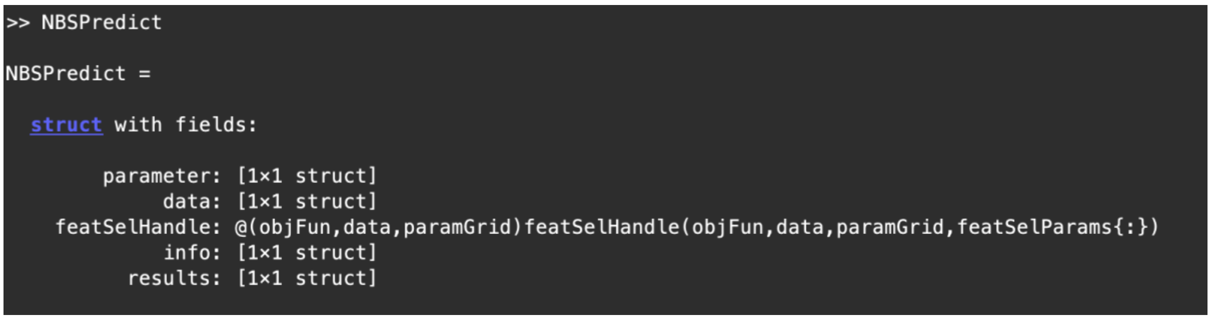
7.3 The Structure of the Output File

If you may further want to access specific parameters, directories, or results, you can easily load the output file saved in the "/Results/{current date}" directory under the input's directory[9]. Once you locate the output file using the command line or **Current Folder** window, simply load the file. To load the file, you can either use the right mouse click and load or write the following command on the command window:

```
load("FileName.mat");
```

[9] NBS-Predict uses parent directory of the directory containing correlation matrices as input directory.

After you load the output file, you will see the MATLAB structure called "NBSPredict" on the MATLAB **Workspace**. This "NBSPredict" structure comprises 5 substructures (see Fig. 14): *NBSPredict.parameter*, *NBSPredict.data*, *NBSPredict.featSelHandle*, *NBSPredict.info*, *NBSPredict.results*. Simply write "NBSPredict" at the command window to check the substructures.



```
>> NBSPredict

NBSPredict =

  struct with fields:

        parameter: [1×1 struct]
             data: [1×1 struct]
    featSelHandle: @(objFun,data,paramGrid)featSelHandle(objFun,data,paramGrid,featSelParams{:})
             info: [1×1 struct]
          results: [1×1 struct]
```

Figure 14. Output "NBSPredict" Structure

7.3.1 NBSPredict.parameter

It stores the parameters entered in the NBS-Predict **Analysis Setup** window (see Fig. 8).

7.3.2 NBSPredict.data

It stores vectorized adjacency matrices (features), target, brain regions, confounds, and data directories.

7.3.3 NBSPredict.featSelHandle

It stores the MATLAB function handle (please refer to Function Handles for details) for the feature selection algorithm.

7.3.4 NBSPredict.info

It stores the start and end date of the analysis and time elapsed.

7.3.5 NBSPredict.results

It stores results derived from the machine learning algorithms used during the analysis, and the "NBSPredict.results.bestEstimator" substructure, indicating the machine learning algorithm with the best performance. In each machine learning result substructure (e.g. “NBSPredict.results.svmR”), you can access scaled and unscaled versions of the mean connected components across outer loops (i.e. weighted adjacency matrix) and their vectorized versions, true and predicted labels, mean CV score and its confidence intervals. Additionally, if hyperparameter tuning has been performed, the hyperparameters selected in each CV fold can be viewed.

7.4 Command Line

If the output “NBSPredict.mat” file is present and a user wants to perform the same analysis again, it is possible to easily run the analysis using the command line. To do that, the user can either directly run the following command:

```
run_NBSPredict("fileName.mat");
```

where “fileName.mat” is the full directory of the output “NBSPredict.mat” file, or navigate to the directory of the output file, load it and run the following command:

```
run_NBSPredict(NBSPredict);
```

Additionally, a user may check the results on the NBSPredict **Results Viewer** window later. To do that, the user can either write the following command or select the output file on the file manager window:

```
view_NBSPredict;
```

or load the output file and write the following command:

```
view_NBSPredict(NBSPredict);
```

or simply write the following command:

```
view_NBSPredict("fileName.mat");
```

where “fileName.mat” is the full directory of the output file.

7.5 Model Extraction

NBS-Predict automatically fits the trained model using the whole dataset following the evaluation of the model within the CV structure and saves it inside the “model” substructure under the corresponding ML model results substructure (e.g., “NBSPredict.results.lda.model”). Using this model substructure, users can make predictions about the novel data. The substructure contains the trained model (with the selected hyperparameters if hyperparameter optimization performed) and function handles for performed preprocessing steps (e.g., scaling, confound regression, etc.). To predict novel data, use the “NBSPredict_predict” function. This function requires you to provide the trained model and the directory for the connectivity matrices of unseen subjects and, optionally, confound matrix (if deconfounding performed). The example input structure for the function is as follows:

```
NBSPredict_predict(NBSPredict.results.lda.model, 'connectome',...
                  '~/Documents/holdout/connMats/subject-001.mat',...
                  'confMat', '~/Documents/holdout/confoundMat.mat');
```

The function returns a label for the novel subject (e.g., 0 or 1 if binary classification). Using this functionality, the holdout performance of the model can be estimated. The ML community always recommends using a holdout dataset to maximize the confidence in assessing the model performance. The holdout dataset performance can represent the degree to which the trained model performs on a “real-world” scenario if the holdout dataset can mostly represent the population. Additionally, by estimating the holdout performance of the trained model, users can assess whether any overfitting or “information leaking” occurred during training. For example, if there is a significant performance discrepancy between the holdout set and repeated-CV, one might suspect overfitting. The sample script for evaluating the holdout performance is given below.

```
% Set main directory for the holdout prediction.
holdoutDir = '~Documents/holdoutPrediction/';

fprintf('Started predicting holdout subjects... \n');
```

```
% Model to predict holdout subjects.
model = NBSPredict.results.lda.model;

% Predict holdout subjects.
yPred = NBSPredict_predict(model,…
                            'connectome', [holdoutDir, 'connMats/'],...
                            'confMat', [holdoutDir, 'confoundMat.csv']);

% Load true labels
yTrue = csvread([holdoutDir, 'trueLabels.csv']);

% Compute explained variance between predicted and true labels, and print.
score = compute_modelMetrics(yTrue, yPred, 'explained_variance');
fprintf('Explained variance: %.3f \n', score);
```

Of note, it is critical to ensure that the holdout and training sets are entirely independent of each other. Otherwise, the performance estimation might be optimistically biased. Please ensure that the subjects in both sets do not have any codependencies (e.g., no members of the same family). Further, using the same method above, users can also evaluate the model's performance across various recording sites. In such cases, we recommend users preprocess data using the same preprocessing pipeline to minimize the dataset variability.

7.6 Test Function

Users may test the prediction performance of NBS-Predict on several synthetic networks (small-world, scale-free or random) data by typing the following command:

```
test_NBSPredict(parameters);
```

where the parameters are extensively documented in the test_NBSPredict.m function. To review the parameters suitable for the function, please type the following command:

```
help test_NBSPredict
```

7.7 Simulation

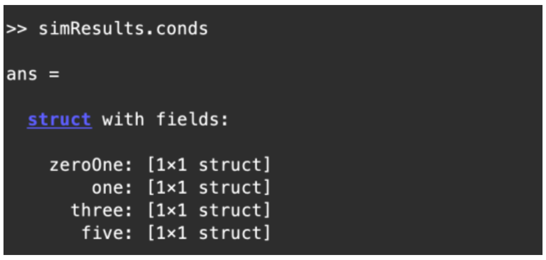
Users may evaluate the prediction performance of NBS-Predict and alternative algorithms (elastic net, lasso, p-value thresholding, top 5% of features, and connectome-based predictive modeling (Shen et al., 2017)) on simulated network data. Regarding simulated data, different network topologies are available: scale-free, small-world and random. To run the simulation, please type the following command:

```
sim_testNBSPredict(parameters);
```

where the parameters are extensively documented in the "sim_testNBSPredict.m" function. To review the parameters suitable for the function, please type the following command:

```
help sim_testNBSPredict
```

After you run the simulation, a structure called "simResults" will be generated and saved into the "simRes_[algorithmName]_[task]_[date].mat" file under the "~/NBSPredict/simulationResults/" directory. This "simResults" structure comprises two substructures: *simResults.info* and *simResults.conds*. *simResults.conds* substructure consists of simulation results that are located under substructures named with ground truth conditions (see Fig. 15).



```
>> simResults.conds

ans =

  struct with fields:

    zeroOne: [1×1 struct]
        one: [1×1 struct]
      three: [1×1 struct]
       five: [1×1 struct]
```

Figure 15. "simResults.conds" Structure

In this substructure, you will find the following data that are derived from each iteration of the simulation: CPU time, target prediction performance, vectorized raw weighted network, vectorized scaled weighted network.

7.8 Plotting simulation results

Users may further plot the simulation results. To do that, you need to prepare data for plotting using the "prep_plotData" function. This function computes the performance of the algorithms in identifying the set of edges with ground truth derived using a set of various weight thresholds (e.g., 100 evenly spaced weight thresholds between 0 and 1) using desired metrics (e.g., ROC, AUC, precision) and organize "simResults" structure generated by the "sim_testNBSPredict" function for plotting. To run the preparation function, please enter the following command:

```
prep_plotData(parameters);
```

where the parameters are extensively documented in the "prep_plotData" function. To review the parameters suitable for the function, please type the following command:

```
help prep_plotData
```

After you run the preparation function, it will ask you to select the simulation results files. You can select multiple results files. The prepared files are then saved into the "~/PlotData/" directory under the directory of the selected simulation results files. Each file named as the name of the input file with the suffix of "plotData" (e.g., "simRes_[algorithmName]_[task]_[date]_plotData.mat").

Now, you can plot the results by entering the following command:

```
plot_plotData(parameters);
```

where the parameters are extensively documented in the "plot_plotData" function. To review the parameters suitable for the function, please type the following command:

```
help plot_plotData
```

The figures will be saved into the "~/figures/" directory under the directory of selected "_plotData" files.

7.9 Bugs and Feature Requests

In case you encounter a bug or want to request a feature, there are two options to contact us. First, you can write one of the corresponding authors: Emin Serin[10] (emin.serin@charite.de) or Johann D. Kruschwitz (johann.kruschwitz@charite.de). A better way is to create an issue on NBS-Predict's GitHub site using the following link: https://github.com/eminSerin/NBS-Predict/issues

[10] Main author.

References

Benjamini, Y. (2010). Simultaneous and selective inference: Current successes and future challenges. *Biometrical J*. https://doi.org/10.1002/bimj.200900299

Braga-Neto, U. M., & Dougherty, E. R. (2004). Is cross-validation valid for small-sample microarray classification? *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btg419

Bullmore, E., & Sporns, O. (2009). Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci*. https://doi.org/10.1038/nrn2575

Haufe, S., Meinecke, F., Görgen, K., Dähne, S., Haynes, J.-D., Blankertz, B., & Bießmann, F. (2014). On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, *87*, 96–110. https://doi.org/10.1016/j.neuroimage.2013.10.067

Hebart, M. N., & Baker, C. I. (2018). Deconstructing multivariate decoding for the study of brain function. *NeuroImage*, *180*, 4–18. https://doi.org/10.1016/j.neuroimage.2017.08.005

Kim, J. H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Comput. Stat. Data Anal*. https://doi.org/10.1016/j.csda.2009.04.009

Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S. F., & Baker, C. I. (2009). Circular analysis in systems neuroscience: The dangers of double dipping. *Nature Neuroscience*, *12*(5), 535–540. https://doi.org/10.1038/nn.2303

Krstajic, D., Buturovic, L. J., Leahy, D. E., & Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. *J. Cheminform*. https://doi.org/10.1186/1758-2946-6-10

Kruschwitz, J. D., List, D., Waller, L., Rubinov, M., & Walter, H. (2015). GraphVar: A user-friendly toolbox for comprehensive graph analyses of functional brain connectivity. *Journal of Neuroscience Methods*, *245*, 107–115. https://doi.org/10.1016/j.jneumeth.2015.02.021

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (n.d.). Scikit-learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON*, 6.

Shen, X., Finn, E. S., Scheinost, D., Rosenberg, M. D., Chun, M. M., Papademetris, X., & Constable, R. T. (2017). Using connectome-based predictive modeling to predict individual behavior from brain connectivity. *Nature Protocols*, *12*(3), 506–518. https://doi.org/10.1038/nprot.2016.178

Snoek, L., Miletić, S., & Scholte, H. S. (2019). How to control for confounds in decoding analyses of neuroimaging data. *NeuroImage*, *184*, 741–760. https://doi.org/10.1016/j.neuroimage.2018.09.074

Tabe-Bordbar, S., Emad, A., Zhao, S. D., & Sinha, S. (2018). A closer look at cross-validation for assessing the accuracy of gene regulatory networks and models. *Scientific Reports*, *8*(1), 6620. https://doi.org/10.1038/s41598-018-24937-4

Xia, M., Wang, J., & He, Y. (2013). *BrainNet Viewer: A Network Visualization Tool for Human Brain Connectomics*. https://doi.org/10.1371/journal.pone.0068910

Zalesky, A., Fornito, A., & Bullmore, E. T. (2010). Network-based statistic: Identifying differences in brain networks. *Neuroimage*, *53*, 1197–1207. https://doi.org/10.1016/j.neuroimage.2010.06.041