PRESULT user guide

Written by

Shankaracharya

And

Chad Huff

Huff Lab
Department of Epidemiology
The University of Texas MD Anderson Cancer Center
Houston, Texas

Table of Contents

1. Introduction

The prediction by Supervised Learning Toolkit (PRESULT) is designed to simplify the development, validation and optimization of machine learning (ML) risk prediction models. PRESULT also applies ML models to predict the absolute risk of developing disease over one or more years. PRESULT supports four advanced ML methods: Random Forest (RF), Mixture of Experts (ME), Support Vector Machine (SVM) and Regression Tree (RT).

PRESULT is designed to mitigate many of the major challenges of risk prediction modeling with ML methods. The tool produces the portable model that can be used anytime to validate the new model, provides program objects in the form of executable for easy distribution, allows parameter tuning through simple, well defined command line arguments, and clearly documents tuning parameters to ensure reproducibility and provide support for external validation. PRESULT also generates receiver operating characteristics (ROC) curves and calculate the area under the curve (AUC) for training and validation datasets. The difference between ROC curves for training and testing datasets as well as k-fold cross validation helps detect model overtraining. The optimization of ML parameters values is one way to minimize overtraining.

Software utilities: PRESULT is designed to classify data into binary classes (for example diseased/non-diseased) to predict the present status of any new test dataset to one of the two classes based on the training network. It accepts raw data for every field with “missing values” marked as “?” for all covariates in one file.

The performance of PRESULT will often depend on the choice of tuning parameter values. The default parameter values function well in our internal tests, but we recommend parameter optimization whenever possible.

2. Getting ready to run PRESULT

2.1 System requirements

PRESULT runs on any Linux or Mac OS system with X-windows (XQuartz for mac), ghostscript and xpdf-tools installed. The ghostscript program is necessary for producing high quality pdf files and installer is available at, http://pages.uoregon.edu/koch/. The xpdf-tools are available at http://www.foolabs.com/xpdf/download.html. X-windows must be installed on Linux to display the ROC curve, but high quality pdf versions of the ROC curve will be saved after successful execution of the program even if X-windows is not installed.

2.2 Installation

2.2.1 Windows (7 and above):

1. MCR (matlab compiler runtime environment) Installation -
   Click on windows installer for MCR supplied in the folder "Windows_MCR" and follow the instructions on the screen.

2. Install strawberry perl (A free version of windows perl) by following the instructions here https://learn.perl.org/installing/windows.html. Setup the path of perl library by navigating through environmental variable setting option on your windows PC. Restart the system and verify perl installation with command "perl -v". This command should show the installed version of perl and confirm the installation.

3. Install ghostscript program by following the steps documented in "Installing-Ghostscript-on-Windows-7.pdf", "Installing-Ghostscript-on-Windows-8.1.pdf" and "Installing-Ghostscript-on-Windows-10.pdf" distributed with the binaries in the folder "ghostscript_install_instructions".

4. Important: restart the computer.

5. Download xpdf for windows version and install it by following the steps mentioned in the "INSTALL" file inside the xpdf folder.

6. Important: restart the computer.

7. Open the "Command Prompt", navigate through the "PRESULT_WINDOWS" folder on the command line (using command cd) and run the program (for eg. PRESULT.exe train_test1 RF --input_data pid_raw_data.txt)

2.2.1 Macs:

1. MCR (matlab compiler runtime environment) Installation -
   Click on mac installer for MCR supplied and follow the instructions on the screen.

2. Setting up environment
   Add the following two lines to the ~/.bash_profile file in your home directory, then restart the terminal (Note: incase matlab compiler is installed at some other location, change the path):

   export
   DYLD_LIBRARY_PATH=/Applications/MATLAB/MATLAB_Runtime/v91/runtime/maci64:/Applications/MATLAB/MATLAB_Runtime/v91/bin/maci64:/Applications/MATLAB/MATLAB_Runtime/v91/sys/os/maci64

   export
   XAPPLRESDIR=/Applications/MATLAB/MATLAB_Runtime/v91/X11/app-default

2.2.2 Linux:

1. Find the MCR for linux in the folder "MCR_linx_installer" or download the MCR from the following link:
   https://www.mathworks.com/supportfiles/downloads/R2014b/deployment_files/R2014b/installers/glnxa64/MCR_R2014b_glnxa64_installer.zip

2. Install the MCR –

   #### Install as superuser-

   < sudo ./install -mode automated -argreeToLicense yes> in the MCR_installer directory and follow the instructions on the screen.

   #### Install locally-

   Step 1- create installation directory
   (example: /path/directory/MATLAB/MATLAB_Compiler_Runtime).

   Step 2- run the command
   <./install -mode automated -argreeToLicense yes >

3. Setting up environment
   Run the command < sh run_PRESULT.sh <mcr_directory> [argument_list] >

Example: sh run_PRESULT.sh /path/to/mcr directory/v84 train_test1 RF --input_data pid_raw_data.txt --out test

<mcr_directory> = the complete path where your matlab compiler runtime is installed
Example: /path to mcr/MATLAB_Compiler_Runtime/v84

[argument_list] = list of arguments to launch the program
Example: train_test1 RF --input_data pid_raw_data.txt --out test

Copy and paste the below mentioned LD_LIBRARY_PATH in your .bash_profile and restart the terminal

export LD_LIBRARY_PATH=<mcr_directory>/runtime/glnxa64:/<mcr_directory>/bin/glnxa64:<mcr_directory>/sys/os/glnxa64:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/native_threads:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/server:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/client:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/

export XAPPLRESDIR=<mcr_directory>/X11/app-defaults

2.3 Need help?

If your question is not answered in this user's guide, feel free to send your questions and comments at, FShankaracharya@mdanderson.org

3. Five Modules of PRESULT

3.1 Train module

Description: This module train the network for the training input data and saves the portable model for four different ML methods along with standard logistic regression model.

Command:
PRESULT train (RF|ME|RT|SVM) --train_data <input_dataset> [PARAMETERS]

Input:

1. One training dataset (a compulsory input parameter)
2. Optional parameters: "--out" for output prefixes, "randm" for choice of randomization of samples in the input data with the choice of 'yes' or 'no', the desired p value input to select variables in train data for logistic regression "p_val" whose default value is set to 0.01 and "--title" write user's title on ROC curve.
3. Other method related parameter options for different method (More details in 4).

Output:

The train module create the following files:

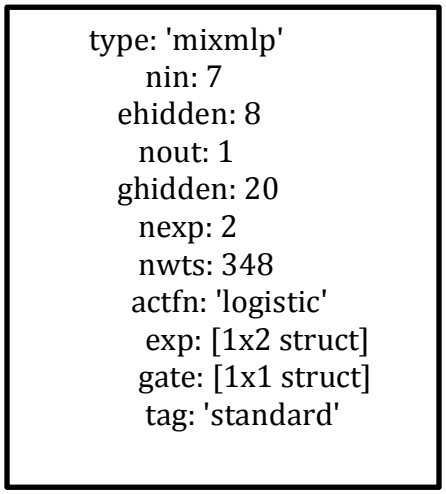
- The trained model file named as [method name]_trained_net.mat (Example: ME_trained_net.mat).
- ROC curve in two file formats, pdf and eps.
- The training score of the trained model.
- Logistic regression model and the p-value for each variable.
- Normalized training data matrix

Trained model description:

The training model is saved as trained network for each ML method (for example ME trained model as ME_trained_net.mat and RF trained model as RF_trained_net.mat etc.). Training model with each method contain their own parameter values and architecture. Here is the example architecture of mixture of experts model. A typical ME_trained_net.mat file incorporates the original parameter values used to train the model which include:

- Type of network = mixture of expert multi layer perceptron,
- Numbers of input (nin) = 7
- Numbers of hidden layers in expert network (ehidden)=8,
- Number of output (nout) = 1,
- Numbers of hidden layers in gate network (ghidden)=20,
- Numbers of expert (nexp)=2,
- Network weight (nwts)=348,
- Activation function used (actfn)=logistic,
- Expert architecture (exp)=[1x2 struct],)

- Gate architecture (gate)=(1x1 struct) and
- tag (distribution=standard).



3.2 Test module

Description: This module tests a trained network file from the training module on a testing dataset. The number and order of columns in the training dataset should match the testing dataset.

Command:
PRESULT test (RF|ME|RT|SVM) --test_data <input_dataset> --trained_net <input_network_file> [PARAMETERS]

Input:

1. One testing dataset (a compulsory input parameter)
2. One trained network file (a compulsory input parameter)
3. Optional parameter: The test module will validate the test dataset with selected ML method in a default setting. Logistic regression can also be incorporated for comparison with ML method with the inclusion of following parameters: "--log_train_model" for logistic regression model, "--log_p_value" for p-value of logistic regression for each variable in the trained data and "--logistic_regression" as "on" (don't include the inverted commas in the command line). The first two parameters are the standard output from the training module. "p_val" is the cutoff parameter used to select the variables for logistic regression (default value is set to 0.01). The "--out" parameter is for output file name prefixes and the "--title" gives freedom to the users to choose their own title to be printed on ROC curve.
4. Other method related parameter options, which vary according to the ML method used (More details can be found in the ML methods parameter optimization section on page 16).

Output:

The train module creates the following files:

1. The trained model file named as [method name]_trained_net.mat (Example: ME_trained_net.mat).
2. ROC curve in two file formats, pdf and eps.
3. The training score of the trained model.
4. Logistic regression model file and text file with the p-value for each variable.
5. Normalized test data matrix
6. Other standard output on the command line:
   - Confusion matrix in the form of

True Positive (TP)	*False Positive(FP)*
False Negative (FN)	*True Negative (TN)*

- Sensitivity = $\frac{Numbers\ of\ true\ positive\ outputs}{Numbers\ of\ actually\ positive\ cases} = \frac{TP}{(TP+FN)}$
- Specificity = $\frac{Numbers\ of\ true\ negative\ outputs}{Numbers\ of\ actually\ negative\ cases} = \frac{TN}{(TN+FP)}$
- Accuracy = $\frac{Numbers\ of\ correct\ decisions}{Total\ numbers\ of\ cases} = \frac{(TP+TN)}{(TN+TP+FN+FP)}$

3.3 Training and testing module

Description: This module conducts training and testing in a single command line. When invoked with the "train_test1" option, PRESULT will divide a single input dataset into separate training and testing datasets. By default, 80% of individuals will be assigned to the training dataset and 20% of individuals will be assigned to the testing dataset; these percentages can be modified with the test_pc parameter. In addition to the output produced from the training and testing modules, option "train_test1" also produces two additional output files: the divided training and testing datasets, named [ML METHOD]_train_data.txt and [ML METHOD]_test_data.txt. When invoked with the "train_test2" option, the training and testing datasets are provided as separate input files.

Command:
PRESULT train_test1 (RF|ME|RT|SVM) --input_data <input_dataset> [--test_pc <x>] [PARAMETERS]

PRESULT train_test2 (RF|ME|RT|SVM) --train_data <input_training_dataset> --test_data <input_testing_dataset> [PARAMETERS]

Input:

1. Single input data (a compulsory input parameter)
2. Optional parameter: "--test_pc" as any integer, "--logistic_regression" as "on" or "off", The desired p value input to select variables in train data for logistic regression "p_val" whose default value is set to 0.01. The "--out" parameter is for output file name prefixes and the "--title" gives freedom to the users to choose their own title to be printed on ROC curve.
3. Other method related parameter options, which vary according to the ML method used (More details can be found in the ML methods parameter optimization section on page 16).

Output:

The training and testing module create the following files:

1. The trained model file named as [method name]_trained_net.mat (Example: ME_trained_net.mat).
2. ROC curve in two file formats, pdf and eps.
3. The training score of the trained model.
4. The logistic regression model file and text file with the p-value file for each variable.
5. Normalized training and testing data matrix files.
6. The standard command line output will also report the confusion matrix, specificity, sensitivity, accuracy, AUC's of the ML model as well as the logistic model (if logistic_regression was specified), name of the ML model file, name of the logistic regression model file, and name of the logistic regression p-value file.

3.4 K-fold cross validation module

Description: This module conducts k-fold cross-validation on an input dataset. The dataset is first subdivided into k files, which are written as [method name]_kfold_train_data_kfold_1 to 10 and [method name]_kfold_test_data_kfold_1 to 10 (example ME_kfold_train_data_kfold_1 and ME_kfold_test_data_kfold_1). Then for each partition, PRESULT conducts training on the other k-1 partitions and applies the trained model to the remaining partition. The output includes the accuracy and AUC for each partition as well as a plot with the average ROC curve and ROC curves for all partition.

Command:
PRESULT kfold (RF|ME|RT|SVM|Logistic) --input_data <input_file> --kfold <k> --title <ROC curve title> [PARAMETERS]

Input:

1. Single input data (s compulsory input parameter)
2. Optional parameters: The parameter "--randm" as "no" will divide the data evenly into k partitions without shuffling the data, otherwise the data are randomly sorted prior to partitioning. The "--out" parameter is for output file name prefixes and the "--title" gives freedom to the users to choose their own title to be printed on ROC curve.

Output:

1. The trained model file named as [method name]_trained_net_kfold_[k].mat (Example: ME_trained_net_kfold_1.mat). For 10-fold cross validation, ten models are created.
2. Ten sets of training and testing data are generated with the name ending with kfold_1 to kfold_k. The typical dataset named as [method name]_test_data_kfold_[k] (example ME_test_data_kfold_1).
3. ROC curve in two file formats, pdf and eps.
4. Normalized training and testing data matrix files
5. The standard command line output reports specificity, sensitivity, accuracy, and AUC for each partition.
6. Summary of k-fold cross validation is presented as the standard output along with average sensitivity, specificity, accuracy and AUC of the test model. Example:

############ Summary for 10-fold cross validation ###############

Kfold	Sensitivity	Specificity	Accuracy	AUC_test
1	0.82353	0.64	76.316	0.84706
2	0.77358	0.73913	76.316	0.84153
3	0.72917	0.71429	72.368	0.78224
4	0.91837	0.7037	84.211	0.89171
5	0.66038	0.78261	69.737	0.80764
6	0.85185	0.63636	78.947	0.88047
7	0.79365	0.61538	76.316	0.80346
8	0.91489	0.68966	82.895	0.8726
9	0.84	0.65385	77.632	0.85098
10	0.81818	0.71429	78.947	0.88863

3.5 Absolute risk prediction module

Description: This module calculates absolute disease risk in 5, 10 or x-years in two steps: 1) training with "train_abs_risk" and 2) absolute yearly risk prediction with "yearly_risk_prediction".

3.5.1 Training:

Description: To train the data input training dataset must contain sample id in the first column with other variables to match with another input file, id_age. The id_age file also must have sample id, their corresponding age and labels (outcome) in the first, second and third consecutive columns.

Command:
PRESULT train_abs_risk (RF|ME|RT|SVM)-train_data <input_file> --randm no --id_age LC_id_age_label.txt [PARAMETERS]

Input:

1. The Training dataset with sample id (only in numeric form) as tag (--train_data).
2. The sample id, age and label dataset (as described above) as tag (--id_age)
3. Optional parameters: The parameter "--randm" as "no" help reproduce the result by preventing shuffling of data. The "--out" parameter is for output file name prefixes and the "--title" gives freedom to the users to choose their own title to be printed on ROC curve.

Output:

1. Trained network model file for the selected ML method.
2. The β_1 value from logistic regression.
3. Attributable risk (AR) values.
4. Training score file.
5. Logistic regression model file.
6. Baseline risk file with the median values of variables from the training dataset.

3.5.2 Absolute yearly risk prediction:

Description: The second step in the absolute risk prediction workflow uses the training output as input. This module accept user's entered baseline risk file as input. Incidence and mortality rate table should be provided which contain age specific incidence rate and mortality rate excluding the disease in the study. In the table first, second and third column must be arranged as age, incidence rate and mortality rate respectively either block-wise or yearly format. It also requires prediction data (first column must contain sample id), which contains variables for the samples whose risk needs to predict. The sample id in different datasets must be numeric. The output of the "yearly_risk" sub-module includes relative-risk with corresponding ID of the sample as "id_and_ri.txt", absolute year-wise 10 years risk of developing the disease as "yearwise_absolute_risk".

Command:
PRESULT yearly_risk (RF|ME|RT|SVM) --trained_net <risk_trained_net.mat> --baseline_risk <baseline_risk.txt> --AR <float> --predictive_data <prediction_data.txt> --id_age_data <id_age_data.txt> --beta_one <float> --incidance_rate <inc_rate_blocks.txt> [PARAMETERS]

Input:

1. Trained network model file.
2. Sample id, age and label data (same file as in first step).
3. Baseline risk file containing baseline risk for each variable as a text file.
4. Attributable risk and β_1 values.
5. Prediction dataset contain the sample data whose yearly risk are to be predicted. The file format expects the sample id in the first column and other variables in the successive columns. Column order in this file must match the training dataset.
6. The incidence and mortality rate table (as described above).
7. The "--out" tag for the output file name prefixes.

Output:

1. The relative risk file containing relative risk for every individual in prediction dataset with his or her id.
2. The ten years (or any number of years) risk for individuals in the prediction dataset.

4. ML methods parameter optimization

4.1 Mixture of Experts

Introduction:
Mixture of Experts (ME) is made up of different number of experts and one gate network. The ME algorithm first divides the whole learning task into multiple subtasks to process them separately by simple expert network and then combines output from each of them to produce the final output. The overall performance of the final output is better than the individual network output. The ME uses Multilayer perceptron (MLP) as a unit for training in the expert and gate network. The MLP uses back propagation (BP) algorithm for learning weight in order to maximize the log likelihood of training.
The gating network accepts vector input, operates on a generalized linear function and produces scalar output which indicates the probability of which input x is attributed to expert j. The output from each expert network for an input vector x is based on the generalized linear equation $O_i(x) = f\ (w_i x)$, where, w_i is a weight matrix. The final output of ME is the sum of multiplications of the outputs from gating and expert networks.

The gating networks input vector and the conditional densities of target vector t for expert j are used to calculate the probabilistic interpretation of ME. The Expectation and Maximization (EM) optimization process updates the weights with conjugate gradient descent method. Overall, ME architecture posses the mechanism of soft competitions between expert networks (for learning on the basis of supervised error) and gate network (compete for the right to select an appropriate expert network). The implementation of ME in PRESULT is based on matlab libraries mixlab (Moerland, 1997) and netlab.

Parameter optimization:
We use following optimization parameter for ME model:

1. Numbers of iterations in EM algorithm (--emiters, default is 10).
2. Numbers of experts (can be any integer, default is 2).
3. Numbers of hidden layers in each experts (--hexp, default is 8).
4. Numbers of hidden layers in gate network (--hgate, default is 20).
5. Numbers of iterations in maximization step of EM algorithm (--mstep_iterations, default is 7).
6. Activation function (--actfunc, ‘logistic’ or ‘linear’, default is logistic).
7. Randomized shuffling before the selection of train dataset to train the ME model (--randm, default is yes).

4.2 Random Forest

Introduction:
Random Forest (RF) method first creates ensembles (forests) of many regression trees then uses an average of all ensemble predictions to predict the disease risk. Each regression tree is grown on an independent bootstrap sample from the training data. At each node the method select random numbers of randomly selected training input variables then find the best split on the selected random variables. Afterwards the method grows trees to maximum depth and makes the predictions. Mathematically, $\{r_n(x, \theta_m, D_n), m \geq 1\}$ indicates tree, where $\theta_1, \theta_2 \, to \, \theta_m$ are identical independent outputs of a randomizing variable θ. These trees combine to form an ensemble of aggregated regression estimate of $r_n(X, \mathcal{D}_n) = \mathbb{E}_\theta[r_n(X, \theta, \mathcal{D}_n)]$,where $\mathbb{E}_\theta$ denotes expectation conditional on X and the data set $\mathcal{D}_n$. This process is achieved by using the forked code of RF method from Stochastic Bosque code (https://www.mathworks.com/matlabcentral/fileexchange/31036-random-forest) provided by Eren Golge (https://github.com/erogol/Random_Forests).

Parameter optimization:
We use following optimization parameter for RF method:

1. Minimum numbers of samples in an impure node for it to be considered for splitting with tag --min_parents (default is 100).
2. Minimum numbers of samples in a leaf with tag --min_leaf (default is 70).
3. Minimum numbers of trees for training with tag --n_trees (default is 300).

4.3 Regression Tree

Introduction:
Regression tree (RT) is an intuitive method in which tree grows in upside down direction having the root on the top. The information passes down the tree through a series of splits, or nodes at which the decision of tree growth is determined on the basis of value of one or more explanatory variables. The terminal leaf gives the predicted response. Splitting of node in RT is analogous to variable selection in regression. Trees are typically fit via binary (parents nodes will always spit into two nodes), recursive (each child node will become the parent nodes, unless it is terminal node), or partitioning method.

Parameter optimization:
We use following optimization parameter for RT method:

1. Minimum numbers of parents with tag –min_parents (default is 50).
2. Minimum numbers of leaves with tag –min_leaf (default is 12).

4.4 Support Vector Machine

Introduction:
Support Vector Machine (SVM) divides the data into two binary classes with the largest margins by drawing the best hyper plane between them. We used SVM libraries fitcsvm of matlab to train the model. Fitcsvm library calculates the score as a function of input data $f(x) = \sum_{j=1}^{n} \propto_j y_j G(x_j, x) + b$ where, $(\propto_j, \ldots\ldots, \propto_n, b)$ are the estimated SVM parameters, $G(x_j, x)$ is the dot product in the predictor space between x and the support vectors, and the sum includes the training set observations. We used matlab function fitSVMPosterior to convert the score into posterior probability by passing the model with predictor data and class labels. The resulting training model contains the parameters and their values used originally to train the model. We use the matlab function 'predict' to predict the probability values for the class from validation data.

Parameter optimization:

1. Limiting numbers of iterations with tag --iterations_limit (default is 300).
2. Kernel function with tag --kernel_function (can be "rbf" or "linear", default is "linear").

.

5. Frequently Asked Questions

Q: Where can I get PRESULT?
A: PRESULT is available at www.hufflab.org/PRESULT.

Q: How can I cite PRESULT?
A: Please cite the following paper:

Q: How can I report a bug in PRESULT?
A: Please feel free to send your questions, comments and suggestions at FShankaracharya@mdanderson.org

Q: what if I am getting this error message
"./PRESULT: error while loading shared libraries: libmwlaunchermain.so: cannot open shared object file: No such file or directory"?
A: Check your matlab compiler runtime, either they are not installed or library path has not been set.

Additionally, if your program is not working even after library path setup and you got this error message, do the steps:

Run the below mentioned LD_LIBRARY_PATH on the terminal (please replace the "<mcr_directory>" with actual path of mcr installed directory)

export
LD_LIBRARY_PATH=<mcr_directory>/runtime/glnxa64:/<mcr_directory>/bin/glnxa64:<mcr_directory>/sys/os/glnxa64:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/native_threads:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/server:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/client:<mcr_directory>/sys/java/jre/glnxa64/jre/lib/amd64/

export XAPPLRESDIR=<mcr_directory>/X11/app-defaults

Q. What if I am getting the Error Message:
Reason: no suitable image found. Did find:
/System/Library/Frameworks/JavaVM.framework/JavaVM/libmwlaunchermain.dylib: stat() failed with errno=20
Trace/BPT trap: 5

A. Please try to set the environment correctly by pasting the following commands:
export
DYLD_LIBRARY_PATH=/Applications/MATLAB/MATLAB_Runtime/v91/runtime/maci64:/Applications/MATLAB/MATLAB_Runtime/v91/bin/maci64:/Applications/MATLAB/MATLAB_Runtime/v91/sys/os/maci64:/System/Library/Frameworks/JavaVM.framework/JavaVM:/System/Library/Frameworks/JavaVM.framework/Lib

raries

export XAPPLRESDIR=/Applications/MATLAB/MATLAB_Runtime/v91/X11/app-default