ENM TUTORIALS

RRegrs package

RELEASE DATE:	15.9.2015
USE:	A collection of R regression tools to estimate the optimal model via a fully validated procedure.
VERSION:	V 0.0.4
MAIN AUTHOR:	Georgia Tsiliki, Cristian R Munteanu
PARTNER:	NTUA
CONTACT DETAILS:	gtsiliki@central.ntua.gr +30 210 772 3236
AUTHORS:	Georgia Tsiliki, Cristian R. Munteanu, Jose A. Seoane, Carlos Fernandez-Lozano, Haralambos Sarimveis, Egon L. Willighagen

TABLE OF CONTENTS

RRegrs Package Tutorial

Georgia Tsiliki, Cristian R. Munteanu, Jose A. Seoane, Carlos Fernandez-Lozano, Haralambos Sarimveis, Egon L. Willighagen

July 19, 2015

1 Introduction

`RRegrs` is a collection of R regression tools based on the `caret` package. It is used to find the best regression models for any numerical dataset. The initial use of the script is aimed at finding QSAR models for chemoinformatics / nanotoxicology for eNanoMapper European project.

2 Concepts

`RRegrs` represents a simple tool to screen any dataset for the best regression model using ten implemented regression methods:

1. Linear Multi-regression (LM)
2. Generalized Linear Model with Stepwise Feature Selection (GLM)
3. Partial Least Squares Regression (PLS)
4. Lasso regression
5. Elastic Net regression (ENET)
6. Support vector machine using radial functions (SVM radial)
7. Neural Networks regression (NN)
8. Random Forest (RF)
9. Random Forest-Recursive Feature Elimination (RF-RFE)
10. Support Vector Machines Recursive Feature Elimination (SVM-RFE)

`RRegrs` permits you to run all the methods by using only one function call. The main function of the package (*RRegrs*) contains several sections: loading parameters and dataset, remove near zero variance features, scaling dataset, remove correlated features, dataset splitting, run the 10 regression methods, summary of statistics for all methods and splittings, averages for each method and cross-validation type for all splittings, automatic best model statistics, best model Y-randomization. Assessment of Applicability Domain was included in each method.

In order to use RRegrs function, it is needed to specify a minimum set of parameters (the others will get default values). All the parameters will be written into a CSV file (ex: Parameters.csv), in the same working folder where it should be present the input dataset file and the outputs files.The dataset needs to have CSV format, with the first column as the dependent variable. The input and output files will be placed into a working folder. The output files are CSV statistics files, PDF and PNG plots.

The prerequisite libraries needed are `data.table`, `corrplot`, `caret`, `kernlab`, `pls`, `randomForest`, `RSNNS`, `doSNOW`, `foreach`, `doMC`. The minimal call for the `RRegrs()` function could be:

```
> library(RRegrs)

> # Run RRegrs with all default parameters
> # (use default dataset file and working folder,
> # run all regression methods, without wrappers,
> # 10 splitings, 100 times Y-randomization,
> # no parallel calculation = 1 CPU core)
> RRegrsResults = RRegrs()
>
> # Run RRegrs for a specific dataset file and the rest
> # default parameters
> RRegrsResults = RRegrs(DataFileName="MyDataSet.csv")
>
> #Run RRegrs for a specific dataset file,
> # working folder (it should exists and contains dataset file)
> # and the rest default parameters
> RRegrsResults = RRegrs(DataFileName="MyDataSet.csv",
>                        PathDataSet="MyResultsFolder")
```

The default values could be found into the `RRegrs` definition:

```
> RRegrs <- function(DataFileName="ds.House.csv",
>                    PathDataSet="DataResults",
>                    noCores=1,
>                    ResAvgs="RRegsResAvgs.csv",
>                    ResBySplits="RRegrsResAllSplits.csv",
>                    ResBest="RRegrsResBest.csv",
>                    fDet="T",fFilters="F",fScaling="T",
>                    fRemNear0Var="T",fRemCorr="T",
>                    fLM="T",fGLM="T",fPLS="T",fLASSO="T",
>                    fENET="T",fSVRM="T",fNN="T",
>                    fRF="T",fRFREF="T",fSVMRFE="T",
>                    RFE_SVM_C="1;5;15;50",
>                    RFE_SVM_epsilon="0.01;0.1;0.3",
>                    cutoff=0.9,iScaling=1,iScalCol=1,
>                    trainFrac=0.75,iSplitTimes=10,noYrand=100,
>                    CVtypes="repeatedcv;LOOCV",
>                    No0NearVarFile="ds.No0Var.csv",
>                    ScaledFile="ds.scaled.csv",
>                    NoCorrFile="ds.scaled.NoCorrs.csv",
>                    lmFile="LM.details.csv",
>                    glmFile="GLM.details.csv",
>                    plsFile="PLS.details.csv",
>                    lassoFile="Lasso.details.csv",
>                    svrmFile="SVMRadial.details.csv",
>                    nnFile="NN.details.csv",
>                    rfFile="RF.details.csv",
>                    rfrefFile="RFREF.details.csv",
>                    svmrfeFile="SVMRFE.details.csv",
>                    enetFile="ENET.details.csv",
>                    fR2rule="T")
```

The calculations need to be done using a specific folder where all the input, output files can be found. `RRegrs` main function is using an extended set of parameters:

- *DataFileName*: Input dataset file name (default)
- *PathDataSet*: Working folder for all input and output files
- *noCores*: number of CPU cores to be used for calculation - 0=all available, 1=no parallel, n = specific number of cores; depending on operating system, different R package will be needed: `doMC` for Linux or Mac, `doSNOW` and `foreach` for Windows; on Windows, using RStudio, several processes will be created and if all the available cores will be used, the computer will become very slow (it is indicated the use of *available cores-1* and the restart of RStudio to free the RAM between calculations)
- *ResAvgs*: Output file name for averaged statistics (by splittings) for each regression method
- *ResBySplits*: Output file name statistics for each splitting and each regression method (main statistics for all calculations)
- *ResBest*: Output file name statistics for the best model
- *fDet*: If print details for all the functions (default = TRUE)
- *fScaling*: If Scalling dataset
- *fFilters*: if run custom filter (not implemented yet!)
- *fRemNear0Var*: If run Removal of near zero variance columns
- *fRemCorr*: If run Removal of correlated columns
- *fLM*: If run LM
- *fGLM*: If run GLM
- *fPLS*: If run PLS
- *fLASSO*: If run Lasso
- *fENET*: If run ENET
- *fSVRM*: If run SVM radial
- *fNN*: If run NN
- *fRF*: If run RF
- *fSVMRFE*: If run SVM-RFE
- *RFE_SVM_C*: Values of C for SVM-RFE
- *RFE_SVM_epsilon*: Values of epsilon for SVM-RFE
- *cutoff*: Cutoff for correlated features (default = 0.9)
- *iScalCol*: Type of scaling: 1 = normalization; 2 = standardization; 3 = other; any other: no scaling
- *iScalCol*: Scaling columns -¿ 1 = including dependent variable; 2: only all the features
- *trainFrac*: Fraction of training set from the entire dataset (default = 0.75); the rest of dataset is the test set
- *iSplitTimes*: Number of splittings the dataset into train and test (default = 10)
- *noYrand*: Number of Y-Randomization (default = 100)

- *CVtypes*: Cross-validation types: 10-CV (repeatedcv), LOOCV, etc. form caret package (see note below)
- *No0NearVarFile*: Dataset file name without zero near features (if details is chosen)
- *ScaledFile*: Scaled dataset file name (if details is chosen)
- *NoCorrFile*: Dataset file name after correlation removal (if details is chosen)
- *lmFile*: LM output file name with details
- *glmFile*: GLM output file name with details
- *plsFile*: PLS output file name with details
- *lassoFile*: Lasso output file name with details
- *svrmFile*: SVM Radial output file name with details
- *nnFile*: NN output file name with details
- *rfFile*: RF output file name with details
- *rfrefFile*: RFREF output file name with details
- *svmrfeFile*: SVMRFE output file name with details
- *enetFile*: ENET output file name with details
- *fR2rule*: If true, R2 rule will be used to select the best model (else adjR2)

Even if `RRegrs` will create outputs files with all statistics and plots, the function return a list with model's statistics and the the full regression model (resulted from caret training).

Each regression function:

- Uses `caret` package functions such as train function to train the model and trainControl to set the training conditions (10 repetitions, RMSE used as metrics to choose the model)
- Generates the same list of statistics with 17 values: regression name, split number, cross-validation type, number of model features, names of model features, training adjusted R-squared, training root mean squared error (RMSE), training R-squared, training standardized RMSE, test adjusted R-square, test RMSE, test R-squared, test correlation, both (training and test) adjusted R-squared, both RMSE, and both R-squared.
- If details are needed, several output files are generated:
  - a CSV file with detailed statistics about the regression model:
    * Regression method, splitting number, cross-validation type
    * Training set summary
    * Test set summary
    * Fitting summary
    * List of predictors
    * Training predictors
    * Test predictors
    * Full statistics = the list of 17 values defined above
    * Feature importance
    * Residuals of the fitted model

    - Assessment of applicability domain / leverage analysis (if the determinant is not zero): mean of hat values, hat values with warnings (X3 and X2 for values 3 and 2 times than hat mean), leverage threshold, list of points with leverage greater than threshold, Cook's distance cutoff, Cook's distances, points influence
  - 5 − 12 plots for fitting statistics as a PDF file for each splitting and cross-validation method
    - Training Yobs-Ypred
    - Test Yobs-Ypred
    - Feature Importance
    - Fitted vs. Residuals for Fitted Model
    - Leverage for Fitted Model
    - Cook's Distance for Fitted Model
    - 6 standard fitting plots using plot function with cutoff.Cook

The outputs can differ depending on the regression method used.

After filtering the dataset for correlated variables, near-zero variance features and splitting the dataset into training and test sets, the user's selected regression methods will be executed for each splitting and cross-validation type. Some of the complex regression methods (RF, SVM-RFE, RF-RFE) are using only 10-fold cross-validation (other validation methods could be very slow for these complex functions). The parallel support for calculations is presented only for the complex functions.

In the next step, a CSV output file will be created with all the basic statistics (17 values) for each method type, splitting and cross-validation type. These summary statistics are used to generate another CSV file with the averaged statistics by all splittings, for each regression method and cross-validation type. The best regression model is chosen based on the following criteria: from the best test R-squared (+/- 0.05), the model with minimum RMSE is the final one. If fR2rule is False, adjR2 will be used to select the best model. For the best model, an additional CSV file is generated containing detailed statistics as well as PDF plots for important statistics.

Please note that the Cross-validation types available are the resampling methods available from `trainControl caret` and `rfeControl caret`:

- when no feature selection is requested: boot, boot632, cv, repeatedcv, LOOCV, LGOCV, none, oob, adaptive_cv, adaptive_boot or adaptive_LGOCV,
- when feature selection is requested: boot, cv, LOOCV or LGOCV.
- when a mixture of methods is requested (feature and non feature selection), options from the second list should be selected.

If the wrapper function are selected, *SVMRFEreg* and *RFRFEreg* functions could be used.

The following section is presenting the code flow with details about the data objects, input and output files, and functions used into the main *RRegrs* function.

3 RRegrs function code flow

This section presents datails about variable names, input and output files, order of the sections:

1. Load parameters and dataset
   - Load Parameters from the function call as data frame: *Params.df*
   - Write all parameters into a CSV file (default: Parameters.csv)
2. Remove the NA values
3. Remove near zero variance columns using RemNear0VarCols = ds - No0Var CSV

4. Scaling dataset (normalization - default, standardization, etc.) using ScalingDS = ds - Scaled CSV

5. Remove correlated features using RemCorrs = ds

   - Dataset without correlated features: Scaled NoCorrs CSV
   - Correlation matrix: Scaled NoCorrs CorrMAT CSV
   - Correlation plot before removal of features: Scaled NoCorrs Corrs PNG

6. Dataset splitting: Training and Test sets using DsSplit = ds.train, ds.test - CSVs for train and test; for each dataset splitting (default = 10) repeat steps $7 - 9$ = *dfMod*

7. Regression Methods

   - Executed for each cross-validation type (non-wrapper or wrapper)
   - Resulted PDF plots for each method, split and cross-validation type
   - Resulted CSV files for each method with detailed statistics
   - Each method return a list containing: statistics values and the fitter model obtained with `caret` package
   - All models are memorized into variable *dfMod*

   (a) Basic LM using *LMreg* → *lm.model* appended to *dfMod*
   (b) GLM based on AIC regression using *GLMreg* → *glm.model* appended to *dfMod*
   (c) PLS using *PLSreg* → *pls.model* appended to *dfMod*
   (d) Lasso using *LASSOreg* → *lasso.model* appended to *dfMod*
   (e) Elastic Nets *ENETreg* → *enet.model* appended to *dfMod*
   (f) SVM radial regression using *SVLMreg* → *SVRM.model* appended to *dfMod*
   (g) Neural Networks Regression using *NNreg* → *nn.model* appended to *dfMod*
   (h) Random Forest Regression *RFreg* → *rf.model* appended to *dfMod*
   (i) Random Forest-Recursive Feature Elimination (RF-RFE)
   (j) Support Vector Machines Recursive Feature Elimination *SVMRFEreg* → *svmrfe.model* appended to *dfMod*

8. Fitted models comparison plots.
   Models are compared in terms of their resampling results. The `resamples()` `caret` function is used to collect, summarize and contrast the resampling results. Plots are produced per training set (i.e. per data split), and for different estimated R^2 and RMSE values, e.g. 'DifModels.RMSE.iSplits.1.pdf', 'DifModels.R2.iSplits.1.pdf'. Note that only models with the same resampling scheme are compared.

9. Results

   - All statistics results (not ordered) = df.res - RRegrsResBySplit.csv
   - Averaged statistics of the results by each Regression Method and CV type
   - All results as data table - dt.res
   - Averaged results = dt.mean
   - Ordered averaged results = dt.mean.ord - RRegsResAvgs.csv

10. Best model selection – max R2.ts ($+/-0.005$), min RMSE

    - Max R2.ts = Best model statistics - best.dt

- R2.ts for the best model = best.R2 .ts
- Add new conditions (max adjR2.ts (+/ − 0.005), min RMSE) = best.dt
- Regression method for the best model = best.reg

11. Best model detailed statistics
    - Detailed statistics for the best model = RRegrsResBest.csv
    - Run the caret function with the method from the best method = my.stats.reg
    - Plots for best model with 10-fold CV and last split: RRegrsResBest.csv.repeatedcv.split2.pdf
    - Regression method for the best model = best.reg
12. Y-randomization for best model – default = 100 times
    - Using Yrandom = R2Diff.Yrand - RRegrsResBest.csv.Yrand.Hist.pdf

4 Resampling methods

Model performance is estimated using resampling techniques, namely k-fold cross-validation (CV), leave-one-out (LOO) CV, bootstrapping. Particularly, repeated data splitting is performed (default value 10), whereas during the procedure of building the model a set of modified data sets are created from the training samples based on the options offered by `train()` and `rfe()` functions in `caret` package- NAMES in RRgres. Both functions consider a grid of candidate tunning parameters, and the final tunning parameter set is chosen based on aggegating resampling performance estimates for each of the hold-out sample set. These performance estimates are used to evaluate which combination(s) of the tuning parameters are appropriate. Once the final tuning values are assigned, the final model is refit using the entire training set. Finally the performance of the model is evaluated on the test set.

By default the root mean square error (RMSE) is used to calculate performance but R^2 and adjusted-R^2 options are also available.

5 RRegrs package functions

5.1 RRegrs function

RRegrs is the main function of the current package and it permits to execute all regression methods for any dataset in only one call.

RRegrs function is based on 11 regression methods that use *caret* package. The following subsections will present the regression methods which can also be used individually. In order to do that we need to specify all parameters and the data set used, by either defining the parameters file as we have done before:

```
> # flag to calculate and print details for all the functions
> fDet <- as.logical(
>   Param.df[which(Param.df$RRegrs.Parameters=="fDet"),2]
> )
>
> # to reapeat the ds splitting, different values of seed will be used
> iSeed <- i
>
>
> # the fraction of training set from the entire dataset;
> # trainFrac = the rest of dataset, the test set
> trainFrac <- as.numeric(
```

```
>    as.character(
>      Param.df[which(Param.df$RRegrs.Parameters=="trainFrac"),2]
>    )
> )
>
> # dataset folder for input and output files
> PathDataSet <- as.character(
>   Param.df[which(Param.df$RRegrs.Parameters=="PathDataSet"),2]
> )
>
> # input step 1 = ds original file name
> DataFileName<- as.character(
>   Param.df[which(Param.df$RRegrs.Parameters=="DataFileName"),2]
> )
>
> # Generate path + file name = original dataset
> inFile <- file.path(PathDataSet, DataFileName)
>
> #scaled ds file name (in the same folder)
> ScaledFile = as.character(
>   Param.df[which(Param.df$RRegrs.Parameters=="ScaledFile"),2]
> )
>
> ds <- read.csv(inFile,header=T)
>
> # return a list with 2 datasets = dsList$train, dsList$test
> dsList  <- DsSplit(ds,trainFrac,fDet,PathDataSet,iSeed)
>
> # get train and test from the resulted list
> ds.train<- dsList$train
> ds.test <- dsList$test
>
> # types of cross-validation methods
> CVtypes <- strsplit(as.character(
>   Param.df[which(Param.df$RRegrs.Parameters=="CVtypes"),2]),";"
> )[[1]]
```

Or by individually defining all parameters:

```
> # flag to calculate and print details for all the functions
> fDet <- FALSE
>
> # to reapeat the ds splitting, different values of seed will be used
> iSeed <- i
>
> # the fraction of training set from the entire dataset;
> trainFrac <- 0.75
>
>
> # dataset folder for input and output files
> PathDataSet <- 'DataResults'
>
> # upload data set
> ds <- read.csv(ds.Housing,header=T)# !!!!
>
```

```
> # return a list with 2 datasets = dsList$train, dsList$test
> dsList  <- DsSplit(ds,trainFrac,fDet,PathDataSet,iSeed)
>
> # get train and test from the resulted list
> ds.train<- dsList$train
> ds.test <- dsList$test
>
> # types of cross-validation methods
> CVtypes <- c('repeatedcv','LOOCV')
```

For this particular example we are looking at the Boston Housing data [1].

5.2 Basic Linear regression (LM) function

LM is called via `train() caret` function having no extra tuning parameters. RMSE was chosen as the summary metric used to select the optimal model. `trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outLM <- 'LMoutput.csv'
> LM.fit <- LMreg(
>    ds.train,ds.test,CVtypes[1],iSplit=1,fDet=F,outFile=outLM
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.3 Generalized Linear Model with Stepwise Feature Selection (GLM) function

GLM is called via `train() caret` function using the `glmStepAIC` function from the `MASS` package. No tuning parameters are set. RMSE was chosen as the summary metric used to select the optimal model. `trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outGLM<- 'GLMoutput.csv'
> GLM.fit <- GLMreg(
>    ds.train,ds.test,CVtype[1],iSplit=1,fDet=F,outFile=outGLM
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.4 Partial Least Squares Regression (PLS) function

PLS is called via `train() caret` function using the `mvr` function of the `pls` package. RMSE was chosen as the summary metric used to select the optimal model. The number of components is the tuning parameter of the model, which we set to a sequence of integers from 1 to one fifth of the number of features in the training data set. (If the later is smaller than 1, tuning parameter is set to 1.)

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outPLS<- 'PLSoutput.csv'
> PLS.fit <- PLSreg(
>   ds.train,ds.test,CVtype[1],iSplit=1,fDet=F,outFile=outPLS
> )
```

If the details are used, both functions are creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.5 Lasso regression function

Lasso is called via `train() caret` function using the `enet` function of the `elasticnet` package. RMSE was chosen as the summary metric used to select the optimal model. Fraction is the tuning parameter of the model which is the ratio of the L1 norm of the coefficient vector, relative to the norm at the full LS solution. We have set fraction to vary in a sequence of 10 values between zero and one.

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outLASSO<- 'LASSOoutput.csv'
> LASSO.fit <- LASSOreg(
>  ds.train,ds.test,CVtype[1],iSplit=1,fDet=F,outFile=outLASSO
> )
```

Following the guidelines by the `elasticnet` package, LASSOreg only runs for k-fold CV schemes.

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.6 Elastic Net regression (ENET) function

Elastic net from `glmnet` package have mainly two parameters, alpha and lambda. Instead of using the standard `caret` package `sCV` parameterization, the proper alpha value is chosen by sCV (alpha =1 lasso, alpha =0 ridge), and labda is chosen using the glmnet package. RMSE was chosen as the summary metric used to select the optimal model.

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

`sCV` can take the following values: boot, boot632, cv, repeatedcv, LOOCV, LGOCV (for repeated training/test splits), none (only fits one model to the entire training set), oob (only for random forest, bagged trees, bagged earth, bagged flexible discriminant analysis, or conditional tree forest models), adaptive_cv, adaptive_boot or adaptive_LGOCV.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outENET<- 'ENEToutput.csv'
> ENET.fit <- ENETreg(my.datf.train,my.datf.test,sCV,iSplit1,
        fDet=F,outFile=outENET)
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.7 Support vector machine using radial functions (SVM radial) regression function

SVM radial is called via `train() caret` function using the `ksvm` function of the `kernlab` package (the kernel function used is 'rbfdot' Radial Basis Gaussian kernel). RMSE was chosen as the summary metric used to select the optimal model. Tuning parameters in this case are sigma (inverse kernel width) and a regularization parameter C cost (controls how much the regression line can adapt to the data smaller values result in more linear, i.e. flat surfaces.). We have set sigma to be estimated by sigest from kernlab package, and C to vary in c(1,5,10,15,20).

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outSVRM<- 'SVRMoutput.csv'
> SVRM.fit <- SVRMreg(
>   ds.train,ds.test,CVtype[1],iSplit=1,fDet=F,outFile=outSVRM
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.8 Neural Networks regression (NN) function

NN is called via `train() caret` function using the `nnet` function of the `nnet` package. RMSE was chosen as the summary metric used to select the optimal model. Tuning parameters in this case are size and decay, where size refers to the number of units in the hidden layer and decay to the weight decay. Size is set to vary in $c(1, 5, 10, 15)$ and decay within a sequence of values in $[0, 0.001]$.

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outNN<- 'NNoutput.csv'
> NN.fit <- NNreg(
>   ds.train,ds.test,CVtype[1],iSplit=1,fDet=F,outFile=outNN
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.9 Random Forest (RF) regression function

RF (random forest) is called via `train() caret` function using the `randomForest` function of the `randomForest` package. RMSE was chosen as the summary metric used to select the optimal model. Tuning parameters in this case are the number of features selected randomly for each tree in the forest. The recommended value is the square root of the total number of features, however, we recommend a set of values between this value and the total number of features. Of course, the bigger this number, the lower the algorithm. The adjustment values are TO COMPLETE

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times. Each random forest grows 1500 trees.

During the model selection process, the sCV method try to find the best number of features automaticaly chosen in each tree of the RF. The possible values are: numberFeatures/3 (default in randomForest Package), numberFeatures and numberFeatures/2.

`sCV` can take the following values: boot, boot632, cv, repeatedcv, LOOCV, LGOCV (for repeated training/test splits), none (only fits one model to the entire training set), oob (only for random forest, bagged trees, bagged earth, bagged flexible discriminant analysis, or conditional tree forest models), adaptive_cv, adaptive_boot or adaptive_LGOCV.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outRF <- 'RFoutput.csv'
> RF.fit <- RFreg(
>   my.datf.train,my.datf.test,sCV,iSplit1,fDet=F,outFile=outRF
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.10 Support Vector Machines Recursive Feature Elimination (SVM-RFE) regression function

SVM is called via `eps-svr` function of the `kernlab` package and uses the `RFE` function of the `caret` package to obtaing the best SVM model with the best feature set. RMSE was chosen as the summary metric used to select the optimal model.

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times.

`sCV` can take the following values: boot, boot632, cv, repeatedcv, LOOCV, LGOCV (for repeated training/test splits), none (only fits one model to the entire training set), oob (only for random forest, bagged trees, bagged earth, bagged flexible discriminant analysis, or conditional tree forest models), adaptive_cv, adaptive_boot or adaptive_LGOCV.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outSVMRFE<- 'SVMRFEoutput.csv'
> SVMRFE.fit <- SVMRFEreg(
>   my.datf.train,my.datf.test,sCV,iSplit1,
>   fDet=F,outFile=outSVMRFE,cs=c(1,5,15,50),eps=c(0.01,0.1,0.3)
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

5.11 Random Forest-Recursive Feature Elimination (RF-RFE) regression function

RF-RFE represents a wrapper version of RF and, therefore, it will be executed only of feature selection flag was choose.

RF (random forest) is called via `train() caret` function using the `randomForest` function of the `randomForest` package and uses the `RFE` function of the `caret` package to obtaing the best SVM model withe the best feature set. RMSE was chosen as the summary metric used to select the optimal model. Tuning parameters in this case are the number of features selected randomly for each tree in the forest. The recommended value is the square root of the total number of

features, however, we recommend a set of values between this value and the total number of features. Of course, the bigger this number, the lower the algorithm. The adjustment values are TO COMPLETE

`trainControl() caret` function is used to set the resampling method used and its parameters, namely for the k-fold CV we set k=10, and the default value is to repeat the resampling procedure 10 times. Each random forest grows 1500 trees.

During the model selection process, the sCV method try to find the best number of features automaticaly chosen in each tree of the RF. The possible values are: numberFeatures/3 (default in randomForest Package), numberFeatures and numberFeatures/2.

`sCV` can take the following values: boot, boot632, cv, repeatedcv, LOOCV, LGOCV (for repeated training/test splits), none (only fits one model to the entire training set), oob (only for random forest, bagged trees, bagged earth, bagged flexible discriminant analysis, or conditional tree forest models), adaptive_cv, adaptive_boot or adaptive_LGOCV.

```
> # define the output file where all results (CSV, PDF files)
> # will be stored
> outRFRFE<- 'RFRFEoutput.csv'
> RFRFE.fit <- RFRFEreg(
>   my.datf.train,my.datf.test,sCV,iSplit1,
>   fDet=F,outFile=outRFRFE
> )
```

If the details are used, the function is creating several output files such as a CSV file with all calculation details and PDF files for each cross-validation type and split.

Several function have been created in order to split the dataset, remove near-zero variance features, remove correlated features, etc. (the the code flow section).

5.12 Removal of near zero variance columns

This function is based on nearZeroVar function from caret and it has several parameters as input: ds = dataset features without predicted variable (as data frame), fDet = if details (default is FALSE), outFile = output file with modified dataset (default is "ds3.No0Var.csv"):

```
> # Removal of near zero variance columns and add the predicted variable
> # => ds.new = new dataset (as data frame )
> ds.new <- cbind(
>  "net.c" = ds[,1],RemNear0VarCols(ds[,2:dim(ds)[2]],fDet,outFile)
> )
```

5.13 Scaling dataset

This function is based on scale function from caret and it has several parameters as input: ds = dataset features (as data frame), s = 1,2,3 - type of scaling: 1 = normalization, 2 = standardization, 3 = other (default = 1 = Normalization), c = the number of column into the dataset to start scaling (default = 1; if c = 1, included the dependent variable; if c = 2, only the features will be scaled), fDet = if details (default is FALSE), outFile = output file with modified dataset (default is "ds4.scaled.csv"). If s different of 1,2,3 is used, there is no scaling.

```
> # Scaling dataset => ds.new = new dataset (as data frame)
> ds.new <- ScalingDS(ds,iScaling,iScalCol,fDet,outFile)
```

5.14 Remove correlated features

This function is based on scale functions from caret and corrplot packages. It has several parameters as input: ds = dataset frame, fDet = flag for details (TRUE/FALSE), cutoff = correlation cut off (ex: 0.9), outFileName = file name with the corrected dataset (it could include the path).

```
> # Remove the correlated columns => ds.new = new dataset
> # (as data frame) and rebuild the new data frame with the
> # predicted variable
> ds.new <- cbind(
>   "net.c" = ds[,1], RemCorrs(ds[,2:dim(ds)[2]], fDet, cutoff, outFile)
> )
```

Additional files are generated if fDet is TRUE such as correlation matrix (CSV) and plot the correlation plot before correlation removal (PDF).

5.15 Dataset splitting in Training and Test

This function is based on createDataPartition function from caret and it has several parameters as input: ds = frame dataset object, trainFrac = training set ratio from the entire dataset (default = 3/4 = 75%), fDet = flag for detais (default = FALSE), PathDataSet = pathway for results, iSeed = number to be used as seed.

```
> # Dataset spliting in Training and Test => dsList = list with
> # two data frames for training and test
> dsList  <- DsSplit(ds, trainFrac, fDet, PathDataSet, iSeed)
```

5.16 Y-randomization for the best model

It uses only one splitting, 10-cross validation (repeatedcv) and the best method. Best R2 for test (best.R2.ts) will be compared with R2 in test with Y-randomization (Yrand.R2.ts) and it returns ratios between the difference of R2 and best R2 (DiffsR2/bestR2). DsSplit is using for splitting the dataset.

The function is appending outputs to the general CSV statistics file and a histogram as PDF plot.

It is based on createDataPartition function from caret and it has several parameters as input: ds = frame dataset object, trainFrac = training set ratio from the entire dataset, best.reg = label of the method, best.r2.ts = value of R2 for the best model in test set, noYrand = number of randomization (optimal = 100), ResBetF = file name for the best model output CSV file.

```
> # Y-randomization test
> # parallel support using 2 CPU cores
> R2Diff.Yrand <- Yrandom(
>   ds, trainFrac, best.reg, best.R2.ts,
>   noYrand, ResBestF
> )
```

This function calls the best model regression method and, if it is a complex method, there is a need of parallel calculation.

5.17 Auxiliary functions

Several functions have been developed in order to print specific data types to text or CSV files or to calculate several statistics:

- r2.adj.funct = calculates adjusted R2
- r2.adj.lm.funct = calculates adjusted R2 for LM
- rmse.funct = calculates RMSE
- r2.funct = calculates R2

- AppendList2CSv = writes a list to CSV file
- AppendList2txt = writes a list to TXT file
- findResamps.funct = find the number of re-samples for caret, rfe or sbf objects from caret package
- svmFuncsW$fit = calculates the best model with the best feature set (`c` and `epsilon`)
- svmFuncsW$rank = calculates the $\| w^2 \|$ as ranking criterion for measuring the importance of a particular featur in the RFE process.
- svmFuncsW$pred = [to be completed]
- svmFuncsGradW$rank = calculates gradient w, as proposed by Rakotomamonjy et. al based on the gradient of SVM coefficients.

6 Final model

When all models are build based on the resampling scheme discussed above, the best model is selected given RMSE values on the test set, averaged over the 10 data splits. In fact only one winning model is reported at the end of the process with all model parameters and performance statistics. Nevertheless, the user can ask for full details in the working folder.

7 Output

RRegrs returns a list with three items: the name of the best method, the statistics for the best model, the list with all the fitted models based on `caret` functions (including the best model).

8 Example: Regression model for Boston House dataset

The following examples show simple calls of the `RRegrs()` function using a specific dataset file entitled "MyDataSet.csv" that it should be provided by the user:

```
> library(RRegrs)
>
> # Run RRegrs with all default parameters
> #  default data set file ("ds.House.csv") and
> #  working directory ("DataResults")
> #  run all regression methods
> #  10 splittings, 100 times Y-randomization,
> #  no parallel support for CPU cores
> RRegrsResults = RRegrs()
>
> # Run RRegrs for a specific data set file with default parameters
> #  including the default directory ("DataResults")
> RRegrsResults = RRegrs(DataFileName="MyDataSet.csv")
>
> # Run RRegrs for a specific data set file ("MyDataSet.csv") and
> # working folder ("MyResultsFolder"); both should exist
> # the rest of RRegrs parameters have default values
> RRegrsResults = RRegrs(DataFileName="MyDataSet.csv",
>                        PathDataSet="MyResultsFolder")
```

Method	repeatedcv	LOOCV
LM	18.50	1.65
GLM	3.02	8.86
PLS	1.14	1.58
Lasso	1.30	-
ENET	13.74	49.45
SVM radial	5.55	14.61
NN	12.70	49.97
RF	92.44	-
RF-RFE	3.65	-
SVM-RFE	60.75	-

Table 1: RRegrs execution time for a split of Boston House dataset.

The output variable RRegrsResults is a complex object which contains the object of the fitted models and the main statistics for each regression model. Details about each function are presented into the tutorial of the RRegrs package.

The following example could be used to test the RRegrs package using a the Boston housing dataset [1] from RRegrs GitHub URL. It has 13 features and 506 cases:

```
> library(RRegrs)
>
> # Create default working directory "DataResults"
> dir.create("DataResults")
>
> # Get Housing dataset "ds.House.csv" from RRegrs GitHub
> # in the default RRegrs directory "DataResults"
>  download.file("https://raw.githubusercontent.com/enanomapper/RRegrs/master/TEST/ds
>              "DataResults/ds.House.csv",method="auto",quiet=FALSE)
>
> # RRegrs call with default parameters
> RRegrsResults = RRegrs()
```

If you already have this dataset locally in the default working directory of RRegrs ("DataResults"), the following call could be used:

```
> # Search for the best regression model using parameter file
> ComplexOutput <- RRegrs(DataFileName="ds.House.csv",
>                                 noCores=0,iSplitTimes=2,noYrand=2)
```

The *RRegrs* call uses all available CPU cores for the complex methods, 2 splittings, 2 Y-randomizations, and all the regression methods.Table 1 presents the execution times on an Windows 8.1 64bit with i7-4790 CPU (3.60GHz, 4 cores, 8 logical cores), 16G RAM. *repeatedcv* represents the 10-fold cross-validation. The total execution time was 11.63 minutes.

The most important files into the working folder are:

- Data set file = ds.House.csv (Fig. 1). First column represents the predicted variable (dependent variable) and the other 13 columns are the original features.
- Parameter file = Parameters.csv (Fig. 2). It contains all the parameters used to run *RRegrs* function.
- RRegrsResAllSplits.csv = Statistics for all data set splitting, method and CV type (Fig. 3)
- RRegsResAvgs.csv = Averages statistics by method/CV type (Fig. 4)
- RRegrsResBest.csv = Detailed statistics for the automatically best model (best R2 in test dataset with minimum RMSE)

- Comparison plots = DifModels.R2.iSplits.1.pdf (Fig. 5), DifModels.RMSE.iSplits.1.pdf (Fig. 6), ModelsComp.iSplits.1.pdf (Fig. 7), DifModels.R2.iSplits.2.pdf, DifModels.RMSE.iSplits.2.pdf, ModelsComp.iSplits.2.pdf.
- ResBestF,".repeatedcv.split",i,".pdf" = Best model plots only for *repeatedcv*
- ResBestF,".Yrand.Hist.pdf" = Best model Y-randomization plot

Additional files are presented for each regression method such as CSV detailed statistics and PDF plots. One PNG is created (ds.scaled.NoCorrs.csv.corrs.png) in order to show the correlated features in the original dataset (Fig. 8). The initial names of the features have been replaced with V2-V14.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	MEDV	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
2	24	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98
3	21.6	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14
4	34.7	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
5	33.4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
6	36.2	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33
7	28.7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21
8	22.9	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43
9	27.1	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15
10	16.5	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93
11	18.9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1
12	15	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45
13	18.9	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27
14	21.7	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71
15	20.4	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26
16	18.2	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26
17	19.9	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47
18	23.1	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58
19	17.5	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67
20	20.2	0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21	288.99	11.69
21	18.2	0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95	11.28
22	13.6	1.25179	0	8.14	0	0.538	5.57	98.1	3.7979	4	307	21	376.57	21.02

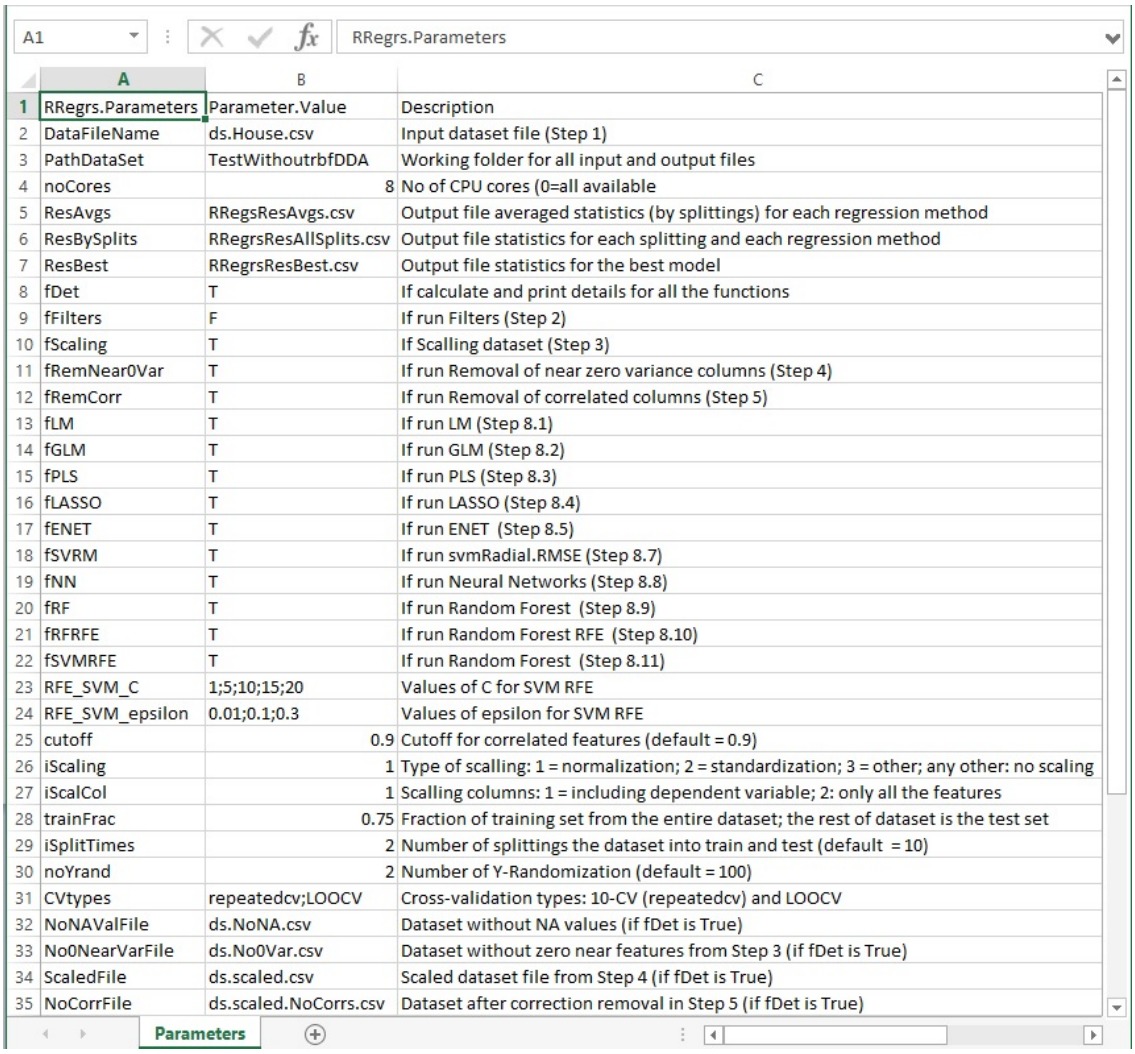
Figure 1: Boston House data set header.

9 Acknowledgment

The eNanoMapper project is funded by the European Union's Seventh Framework Programme for research, technological development and demonstration (FP7-NMP-2013-SMALL-7) under grant agreement no 604134.

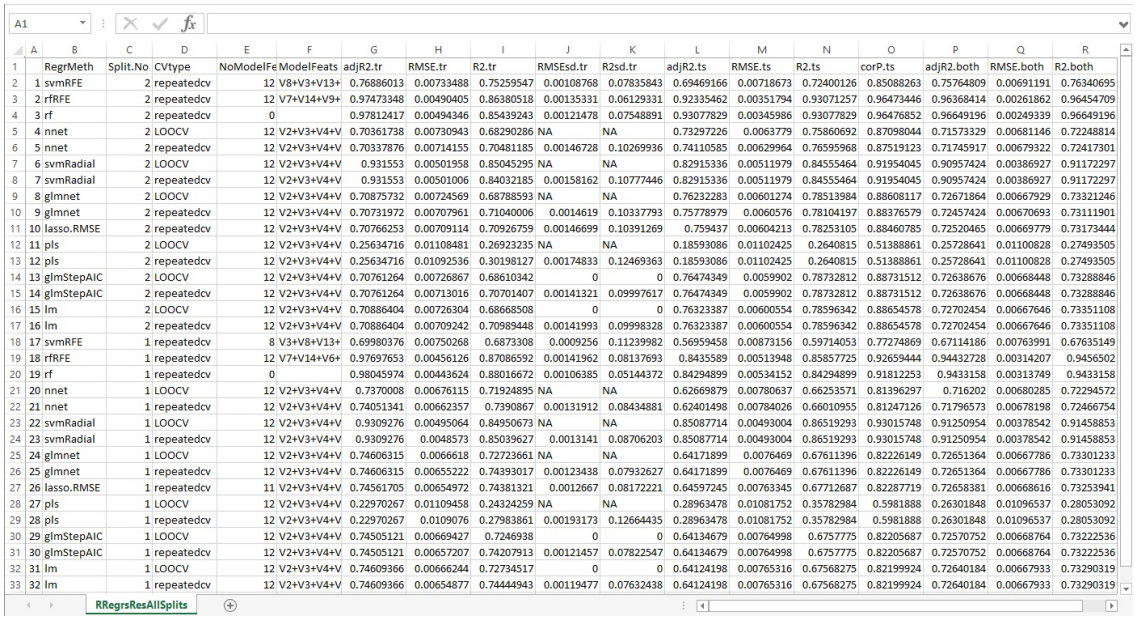
References

[1] Harrison, D., Rubinfeld, D.L.: Hedonic housing prices and the demand for clean air. Journal of Environmental Economics and Management **5**(1), 81–102 (1978)



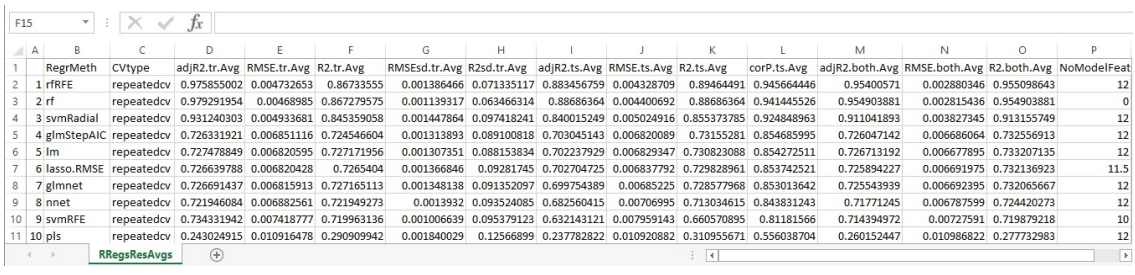
	A	B	C
1	RRegrs.Parameters	Parameter.Value	Description
2	DataFileName	ds.House.csv	Input dataset file (Step 1)
3	PathDataSet	TestWithoutrbfDDA	Working folder for all input and output files
4	noCores	8	No of CPU cores (0=all available
5	ResAvgs	RRegsResAvgs.csv	Output file averaged statistics (by splittings) for each regression method
6	ResBySplits	RRegrsResAllSplits.csv	Output file statistics for each splitting and each regression method
7	ResBest	RRegrsResBest.csv	Output file statistics for the best model
8	fDet	T	If calculate and print details for all the functions
9	fFilters	F	If run Filters (Step 2)
10	fScaling	T	If Scalling dataset (Step 3)
11	fRemNear0Var	T	If run Removal of near zero variance columns (Step 4)
12	fRemCorr	T	If run Removal of correlated columns (Step 5)
13	fLM	T	If run LM (Step 8.1)
14	fGLM	T	If run GLM (Step 8.2)
15	fPLS	T	If run PLS (Step 8.3)
16	fLASSO	T	If run LASSO (Step 8.4)
17	fENET	T	If run ENET (Step 8.5)
18	fSVRM	T	If run svmRadial.RMSE (Step 8.7)
19	fNN	T	If run Neural Networks (Step 8.8)
20	fRF	T	If run Random Forest (Step 8.9)
21	fRFRFE	T	If run Random Forest RFE (Step 8.10)
22	fSVMRFE	T	If run Random Forest (Step 8.11)
23	RFE_SVM_C	1;5;10;15;20	Values of C for SVM RFE
24	RFE_SVM_epsilon	0.01;0.1;0.3	Values of epsilon for SVM RFE
25	cutoff	0.9	Cutoff for correlated features (default = 0.9)
26	iScaling	1	Type of scalling: 1 = normalization; 2 = standardization; 3 = other; any other: no scaling
27	iScalCol	1	Scalling columns: 1 = including dependent variable; 2: only all the features
28	trainFrac	0.75	Fraction of training set from the entire dataset; the rest of dataset is the test set
29	iSplitTimes	2	Number of splittings the dataset into train and test (default = 10)
30	noYrand	2	Number of Y-Randomization (default = 100)
31	CVtypes	repeatedcv;LOOCV	Cross-validation types: 10-CV (repeatedcv) and LOOCV
32	NoNAValFile	ds.NoNA.csv	Dataset without NA values (if fDet is True)
33	No0NearVarFile	ds.No0Var.csv	Dataset without zero near features from Step 3 (if fDet is True)
34	ScaledFile	ds.scaled.csv	Scaled dataset file from Step 4 (if fDet is True)
35	NoCorrFile	ds.scaled.NoCorrs.csv	Dataset after correction removal in Step 5 (if fDet is True)

Figure 2: Parameters used for RRegrs with Boston House dataset.



	RegrMeth	Split.No	CVtype	NoModelFe	ModelFeats	adjR2.tr	RMSE.tr	R2.tr	RMSEsd.tr	R2sd.tr	adjR2.ts	RMSE.ts	R2.ts	corP.ts	adjR2.both	RMSE.both	R2.both
1	svmRFE	2	repeatedcv	12	V8+V3+V13+	0.76886013	0.00733488	0.75259547	0.00108768	0.07835843	0.69469166	0.00718673	0.72400126	0.85088263	0.75764809	0.00691191	0.76340695
2	rfRFE	2	repeatedcv	12	V7+V14+V9+	0.97473348	0.00490405	0.86380518	0.00135331	0.06129331	0.92335462	0.00351794	0.93071257	0.96473446	0.96368414	0.00261862	0.96454709
3	rf	2	repeatedcv	0		0.97812417	0.00494346	0.85439243	0.00121478	0.07548891	0.93077829	0.00345986	0.93077829	0.96476852	0.96649196	0.00249339	0.96649196
4	nnet	2	LOOCV	12	V2+V3+V4+V	0.70361738	0.00730943	0.68290286	NA	NA	0.73297226	0.0063779	0.75860692	0.87098044	0.71573329	0.00681146	0.72248814
5	nnet	2	repeatedcv	12	V2+V3+V4+V	0.70337876	0.00714155	0.70481185	0.00146728	0.10269936	0.74110585	0.00629964	0.76595968	0.87519123	0.71745917	0.00679322	0.72417301
6	svmRadial	2	LOOCV	12	V2+V3+V4+V	0.931553	0.00501958	0.85045295	NA	NA	0.82915336	0.00511979	0.84555464	0.91954045	0.90957424	0.00386927	0.91172297
7	svmRadial	2	repeatedcv	12	V2+V3+V4+V	0.931553	0.00501006	0.84032185	0.00158162	0.10777446	0.82915336	0.00511979	0.84555464	0.91954045	0.90957424	0.00386927	0.91172297
8	glmnet	2	LOOCV	12	V2+V3+V4+V	0.70875732	0.00724569	0.68788593	NA	NA	0.76232283	0.00601274	0.78513984	0.88608117	0.72671864	0.00667929	0.73321246
9	glmnet	2	repeatedcv	12	V2+V3+V4+V	0.70731972	0.00707961	0.71040006	0.0014619	0.10337793	0.75778979	0.0060576	0.78104197	0.88376579	0.72457424	0.00670693	0.73111901
10	lasso.RMSE	2	repeatedcv	12	V2+V3+V4+V	0.70766253	0.00709114	0.70926759	0.00146699	0.10391269	0.759437	0.00604213	0.78253105	0.88460785	0.72520465	0.00669779	0.73173444
11	pls	2	LOOCV	12	V2+V3+V4+V	0.25634716	0.01108481	0.26923235	NA	NA	0.18593086	0.01102425	0.2640815	0.51388861	0.25728641	0.01100828	0.27493505
12	pls	2	repeatedcv	12	V2+V3+V4+V	0.25634716	0.01092536	0.30198127	0.00174833	0.12469363	0.18593086	0.01102425	0.2640815	0.51388861	0.25728641	0.01100828	0.27493505
13	glmStepAIC	2	LOOCV	12	V2+V3+V4+V	0.70761264	0.00726867	0.68610342	0	0	0.76474349	0.0059902	0.78732812	0.88731512	0.72638676	0.00668448	0.73288846
14	glmStepAIC	2	repeatedcv	12	V2+V3+V4+V	0.70761264	0.00713016	0.70701407	0.00141321	0.09997617	0.76474349	0.0059902	0.78732812	0.88731512	0.72638676	0.00668448	0.73288846
15	lm	2	LOOCV	12	V2+V3+V4+V	0.70886404	0.00726304	0.68668508	0	0	0.76323387	0.00600554	0.78596342	0.88654578	0.72702454	0.00667646	0.73351108
16	lm	2	repeatedcv	12	V2+V3+V4+V	0.70886404	0.00709242	0.70989448	0.00141993	0.09998328	0.76323387	0.00600554	0.78596342	0.88654578	0.72702454	0.00667646	0.73351108
17	svmRFE	1	repeatedcv	8	V3+V8+V13+	0.69980376	0.00750268	0.6873308	0.0009256	0.11239982	0.56959458	0.00873156	0.59714053	0.77274869	0.67114186	0.00763991	0.67635149
18	rfRFE	1	repeatedcv	12	V7+V14+V6+	0.97697653	0.00456126	0.87086592	0.00141962	0.08137693	0.8435589	0.00513948	0.85857725	0.92659444	0.94432728	0.00314207	0.9456502
19	rf	1	repeatedcv	0		0.98045974	0.00443624	0.88016672	0.00106385	0.05144372	0.84294899	0.00534152	0.84294899	0.91812253	0.9433158	0.00313749	0.9433158
20	nnet	1	LOOCV	12	V2+V3+V4+V	0.7370008	0.00676115	0.71924895	NA	NA	0.62669879	0.00780637	0.66253571	0.81396297	0.716202	0.00680285	0.72294572
21	nnet	1	repeatedcv	12	V2+V3+V4+V	0.74051341	0.00662357	0.7390867	0.00131912	0.08434881	0.62401498	0.00784026	0.66010955	0.81247126	0.71796573	0.00678198	0.72466754
22	svmRadial	1	LOOCV	12	V2+V3+V4+V	0.9309276	0.00495064	0.84950673	NA	NA	0.85087714	0.00493004	0.86519293	0.93015748	0.91250954	0.00378542	0.91458853
23	svmRadial	1	repeatedcv	12	V2+V3+V4+V	0.9309276	0.0048573	0.85039627	0.0013141	0.08706203	0.85087714	0.00493004	0.86519293	0.93015748	0.91250954	0.00378542	0.91458853
24	glmnet	1	LOOCV	12	V2+V3+V4+V	0.74606315	0.0066618	0.72723661	NA	NA	0.64171899	0.0076469	0.67611396	0.82226149	0.72651364	0.00667786	0.73301233
25	glmnet	1	repeatedcv	12	V2+V3+V4+V	0.74606315	0.00655222	0.74393017	0.00123438	0.07932627	0.64171899	0.0076469	0.67611396	0.82226149	0.72651364	0.00667786	0.73301233
26	lasso.RMSE	1	repeatedcv	11	V2+V3+V4+V	0.74561705	0.00654972	0.74381321	0.0012667	0.08172221	0.64597245	0.00763345	0.67712687	0.82287719	0.72658381	0.00668616	0.73253941
27	pls	1	LOOCV	12	V2+V3+V4+V	0.22970267	0.01109458	0.24324259	NA	NA	0.28963478	0.01081752	0.35782984	0.5981888	0.26301848	0.01096537	0.28053092
28	pls	1	repeatedcv	12	V2+V3+V4+V	0.22970267	0.0109076	0.27983861	0.00193173	0.12664435	0.28963478	0.01081752	0.35782984	0.5981888	0.26301848	0.01096537	0.28053092
29	glmStepAIC	1	LOOCV	12	V2+V3+V4+V	0.74505121	0.00669427	0.7246938	0	0	0.64134679	0.00764998	0.6757775	0.82205687	0.72570752	0.00668764	0.73222536
30	glmStepAIC	1	repeatedcv	12	V2+V3+V4+V	0.74505121	0.00657207	0.74207913	0.00121457	0.07822547	0.64134679	0.00764998	0.6757775	0.82205687	0.72570752	0.00668764	0.73222536
31	lm	1	LOOCV	12	V2+V3+V4+V	0.74609366	0.00666244	0.72734517	0	0	0.64124198	0.00765316	0.67568275	0.82199924	0.72640184	0.00667933	0.73290319
32	lm	1	repeatedcv	12	V2+V3+V4+V	0.74609366	0.00654877	0.74444943	0.00119477	0.07632438	0.64124198	0.00765316	0.67568275	0.82199924	0.72640184	0.00667933	0.73290319

Figure 3: All main statistics for all the splittings, cross-validation methods and regression types.



	RegrMeth	CVtype	adjR2.tr.Avg	RMSE.tr.Avg	R2.tr.Avg	RMSEsd.tr.Avg	R2sd.tr.Avg	adjR2.ts.Avg	RMSE.ts.Avg	R2.ts.Avg	corP.ts.Avg	adjR2.both.Avg	RMSE.both.Avg	R2.both.Avg	NoModelFeat
1	rfRFE	repeatedcv	0.975855002	0.004732653	0.86733555	0.001386466	0.071335117	0.883456759	0.004328709	0.89464491	0.945664446	0.95400571	0.002880346	0.955098643	12
2	rf	repeatedcv	0.979291954	0.00468985	0.867279575	0.001139317	0.063466314	0.88686364	0.004400692	0.88686364	0.941445526	0.954903881	0.002815436	0.954903881	0
3	svmRadial	repeatedcv	0.931240303	0.004933681	0.845359058	0.001447864	0.097418241	0.840015249	0.005024916	0.855373785	0.924848963	0.911041893	0.003827345	0.913155749	12
4	glmStepAIC	repeatedcv	0.726331921	0.006851116	0.724546604	0.001313893	0.089100818	0.703045143	0.006820089	0.73155281	0.854685995	0.726047142	0.006686064	0.732556913	12
5	lm	repeatedcv	0.727478849	0.006820595	0.727171956	0.001307351	0.088153834	0.702237929	0.006829347	0.730823088	0.854272511	0.726713192	0.006677895	0.733207135	12
6	lasso.RMSE	repeatedcv	0.726639788	0.006820428	0.7265404	0.001366846	0.09281745	0.702704725	0.006837792	0.729828961	0.853742521	0.725894227	0.006691975	0.732136923	11.5
7	glmnet	repeatedcv	0.726691437	0.006815913	0.727165113	0.001348138	0.091352097	0.699754389	0.00685225	0.728577968	0.853013642	0.725543939	0.006692395	0.732065667	12
8	nnet	repeatedcv	0.721946084	0.006882561	0.721949273	0.0013932	0.093524085	0.682560415	0.00706995	0.713034615	0.843831243	0.71771245	0.006787599	0.724420273	12
9	svmRFE	repeatedcv	0.734331942	0.007418777	0.719963136	0.001006639	0.095379123	0.632143121	0.007959143	0.660570895	0.81181566	0.714394972	0.00727591	0.719879218	10
10	pls	repeatedcv	0.243024915	0.010916478	0.290909942	0.001840029	0.12566899	0.237782822	0.010920882	0.310955671	0.556038704	0.260152447	0.010986822	0.277732983	12

Figure 4: Averages statistics by method and cross-validation type.

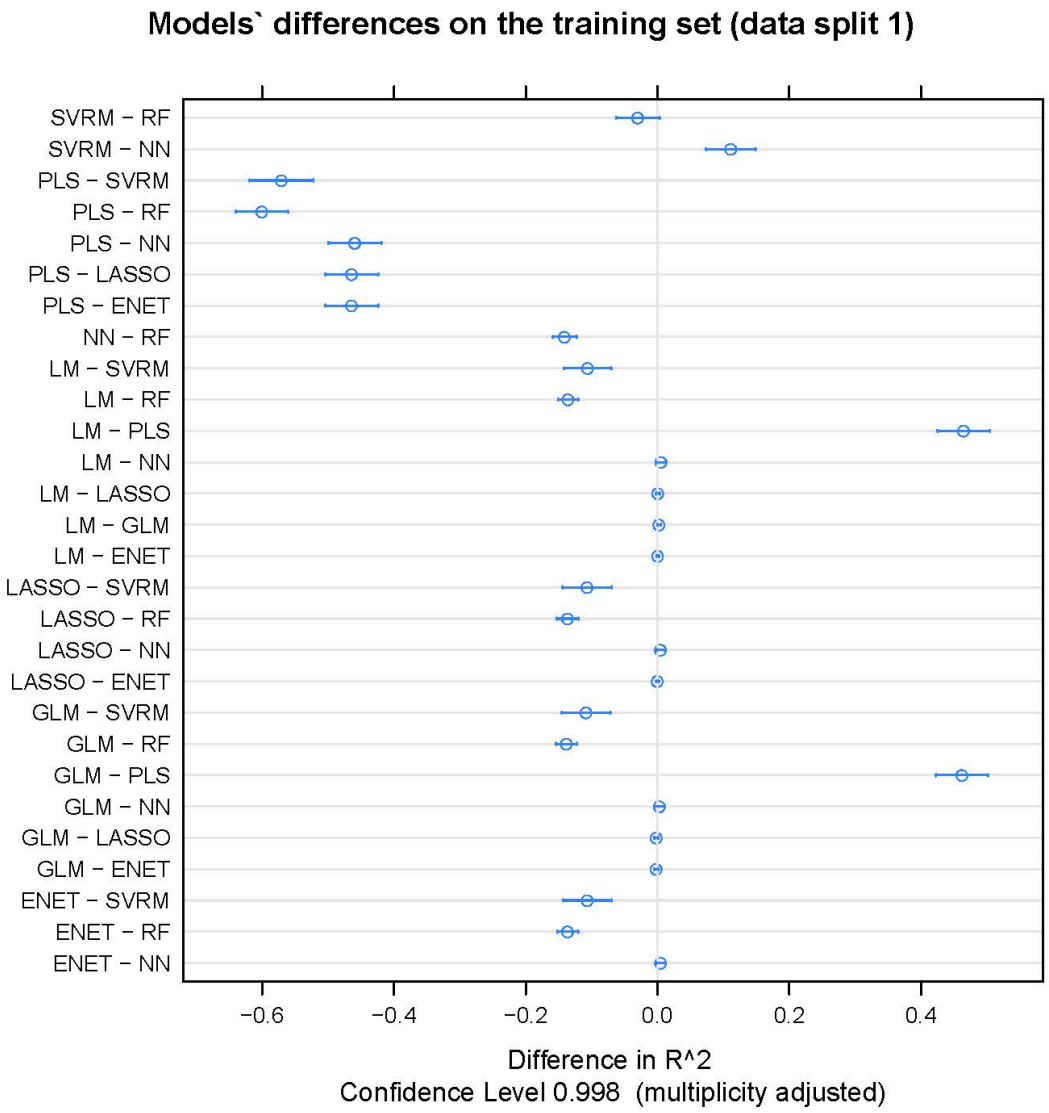


Figure 5: Model's differences in R2 on the training set for split 1.

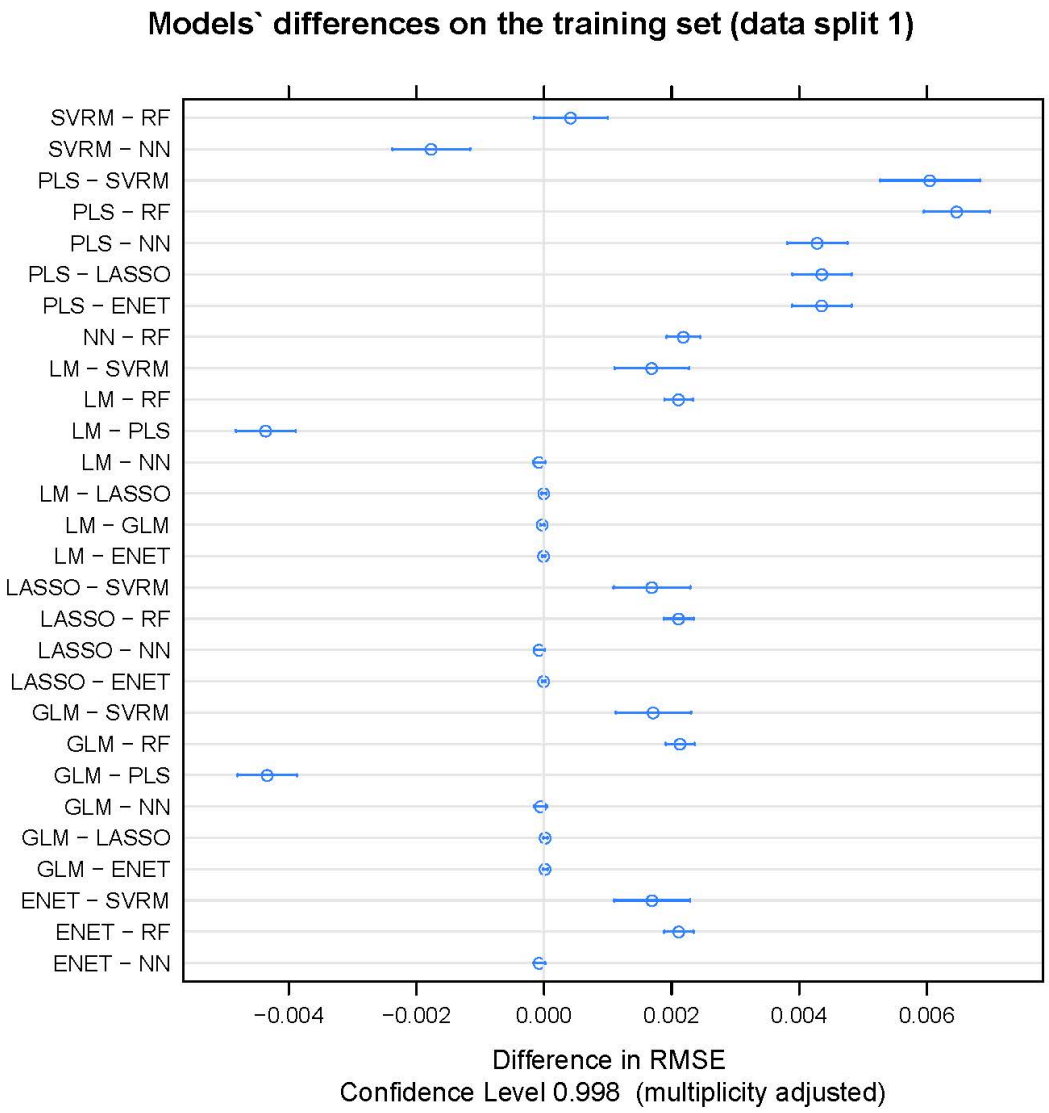


Figure 6: Model's differences in RMSE on the training set for split 1.

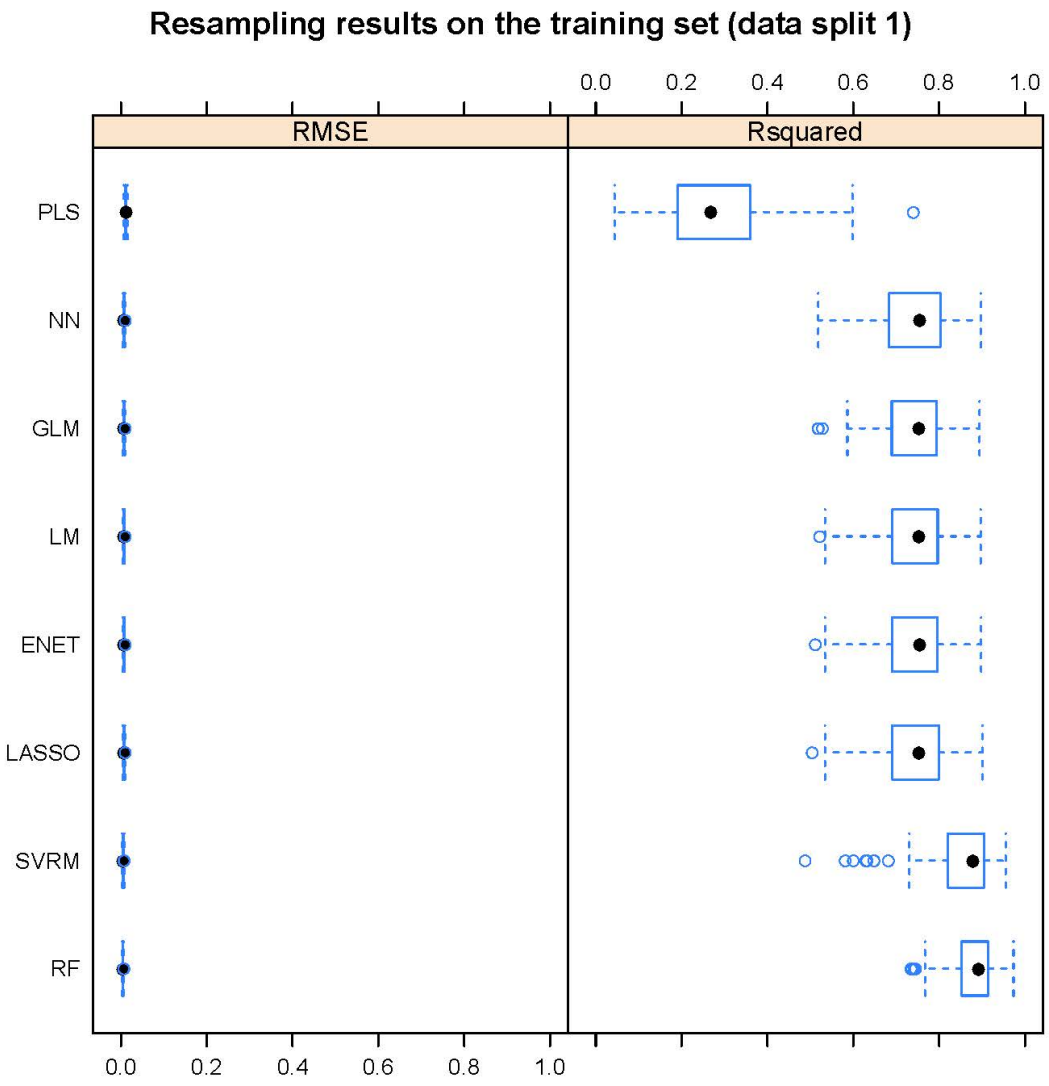


Figure 7: Resampling results on the training set for split 1.

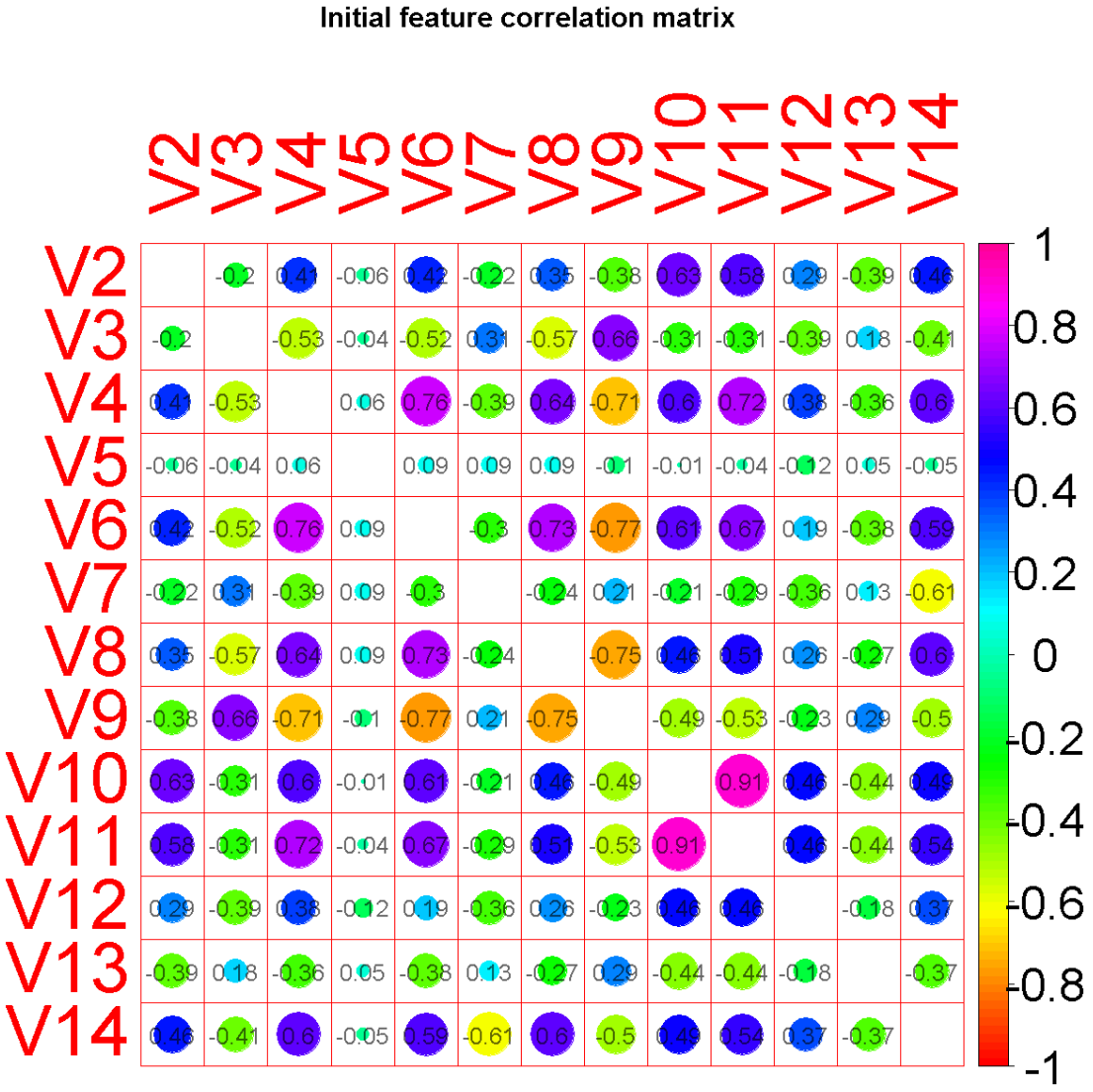


Figure 8: Feature correlation matrix for the original scaled dataset.