A short fscaret package introduction with examples

Jakub Szlek (j.szlek@uj.edu.pl)

June 30, 2013

1 Installation

As it is in case of caret, the fscaret uses large number of R packages but it loads them when needed. To fully take advantage of the package it is recommended to install it both with dependent and suggested packages. Install fscaret with the command

```
> install.packages("fscaret", dependencies = c("Depends", "Suggests"))
```

from R console.

2 Overview

In general fscaret is a wrapper module. It uses the engine of caret to build models and to get the variable ranking from them. When models are build package tries to draw variable importance from them also getting the generalization error (RMSE and MSE) which is used during the scaling process. Finally the output is produced. It contains the data frame of variable importance, generalization error for all build models and preprocessed data set if the preprocessData = TRUE when calling the main fscaret() function.

In summary the whole feature ranking process can be divided into:

1. User provides input data sets and a few settings
2. Models are build
3. Variable rankings are draw out of the models
4. Generalization error is calculated for each model

5. Variable rankings are scaled according to generalization error
6. The results are gathered in tables

3 Input

3.1 Data set format

Be advised that fscaret assumes that data sets are in MISO format (multiple input single output). The example of such (with header) is:

Input_no1	Input_no2	Input_no3	...	Output
2	5.1	32.06	...	1.02
5	1.21	2.06	...	7.2

For more information on reading files in R, please write ?read.csv in R console.

3.2 An example

There are planty of methods to introduce data sets into R. The best way is to read file (presumably csv with `tab` as column separator) as follows:

1. Select file name

```
> basename_file <- "My_database"
> file_name <- paste(basename_file,".csv",sep="")
```

2. Read data file into matrix

```
> matrixTrain <- read.csv(file_name,header=TRUE,sep="\t",
+                  strip.white = TRUE, na.strings = c("NA",""))
```

3. Put loaded matrix into data.frame

```
> matrixTrain <- as.data.frame(matrixTrain)
```

Be advised to use header=TRUE when you have data set with column names as first row and header=FALSE when there are no column names. The last step is obligatory to introduce data into fscaret functions as it checks if the data presented is in data.frame format.

4 Function fscaret()

4.1 Settings

All the settings are documented in `Reference manual` of fscaret `http://cran.r-project.org/web/packages/fscaret/fscaret.pdf`. Here we will concentrate only on a few valuable ones.

- installReqPckg The default setting is FALSE, but if set to TRUE prior to calculations it installs all packages from the sections 'Depends' and 'Suggests' of DESCRIPTION. Please be advised to be logged as root (admin) if you want to install packages for all users.
- preprocessData The default setting is FALSE, but if set to TRUE prior to calculations it performs the data preprocessing, which in short is realized in two steps:
  1. Check for near zero variance predictors and flag as near zero if:
     - the percentage of unique values is less than 20
     - the ratio of the most frequent to the second most frequent value is greater than 20,
  2. Check for susceptibility to multicollinearity
     - Calculate correlation matrix
     - Find variables with correlation 0.9 or more and delete them
- regPred Default option is TRUE and so the regression models are applied
- myTimeLimit Time limit in seconds for single model development, be advised that some models need as time to be build, if the option is omitted, the standard 24-hours time limit is applied. This function is off on non-Unix like systems.
- Used.funcRegPred Vector of regression models to be used, for all available models please enter Used.funcRegPred="all", the listing of functions is:

```
> library(fscaret)
> data(funcRegPred)
> funcRegPred
```

```
 [1] "glm"            "glmStepAIC"     "gam"            "gamLoess"
 [5] "gamSpline"      "rpart"          "rpart2"         "ctree"
 [9] "ctree2"         "evtree"         "obliqueTree"    "gbm"
[13] "blackboost"     "bstTree"        "glmboost"       "gamboost"
[17] "bstLs"          "bstSm"          "rf"             "parRF"
[21] "cforest"        "Boruta"         "RRFglobal"      "RRF"
[25] "treebag"        "bag"            "logicBag"       "bagEarth"
[29] "nodeHarvest"    "partDSA"        "earth"          "gcvEarth"
[33] "logreg"         "glmnet"         "nnet"           "mlp"
[37] "mlpWeightDecay" "pcaNNet"        "avNNet"         "rbf"
[41] "pls"            "kernelpls"      "simpls"         "widekernelpls"
```

```
[45] "spls"          "svmLinear"     "svmRadial"     "svmRadialCost"
[49] "svmPoly"       "gaussprLinear" "gaussprRadial" "gaussprPoly"
[53] "knn"           "xyf"           "bdk"           "lm"
[57] "lmStepAIC"     "leapForward"   "leapBackward"  "leapSeq"
[61] "pcr"           "icr"           "rlm"           "neuralnet"
[65] "qrf"           "qrnn"          "M5Rules"       "M5"
[69] "cubist"        "ppr"           "penalized"     "ridge"
[73] "lars"          "lars2"         "enet"          "lasso"
[77] "relaxo"        "foba"          "krlsRadial"    "krlsPoly"
[81] "rvmLinear"     "rvmRadial"     "rvmPoly"       "superpc"
```

- no.cores The default setting is NULL as to maximize the CPU utilization and to use all available cores. This option is off for Windows OS.
- missData This option handles the missing data. Possible values are:
  - missData=”delRow” - for deletion of observations (rows) with missing values,
  - missData=”delCol” - for deletion of attributes (columns) with missing values,
  - missData=”meanCol” - for imputing mean to missing values,
  - missData=NULL - no action is taken.
- supress.output Default option is FALSE, but it is sometimes justifed to supress the output of intermediate functions and focus on ranking predictions.

4.2 An example

A simple example utilizing the data provided in the fscaret:

```
> library(fscaret)
> data(dataset.train)
> data(dataset.test)
> trainDF <- dataset.train
> testDF <- dataset.test
> myFS<-fscaret(trainDF, testDF, myTimeLimit = 15, preprocessData=TRUE,
+               Used.funcRegPred=c("pcr","pls"), with.labels=TRUE,
+               supress.output=TRUE)
> myRES_tab <- myFS$VarImp$matrixVarImp.MSE[1:10,]
> myRES_tab <- subset(myRES_tab, select=c("pcr","pls","SUM%","ImpGrad","Input_no"))
> myRES_rawMSE <- myFS$VarImp$rawMSE
> myRES_PPlabels <- myFS$PPlabels
```

5 Output

As it was stated previously there are three lists of outputs.

1. Feature ranking and generalization errors for models:

```
> # Print out the Variable importance results for MSE scaling
> print(myRES_tab)

            pcr          pls         SUM%    ImpGrad Input_no
4  6.254152e+01 5.060028e+01 1.000000e+02  0.0000000        4
5  1.672441e+01 2.639949e+01 3.811492e+01 61.8850846        5
22 2.044427e+01 2.248590e+01 3.794368e+01  0.4492507       22
23 2.688175e-01 4.425572e-01 6.287461e-01 98.3429491       23
2  1.997007e-02 4.251718e-02 5.522915e-02 91.2159860        2
13 3.073112e-04 1.650892e-03 1.730751e-03 96.8662355       13
9  6.272900e-04 3.838266e-04 8.936720e-04 48.3650873        9
1  7.670071e-05 9.631531e-05 1.529196e-04 82.8886182        1
17 1.675776e-06 3.498273e-06 4.573066e-06 97.0094969       17
21 1.154120e-06 1.459194e-06 2.309769e-06 49.4918861       21
```

2. Raw generalization errors for each model

```
> # Print out the generalization error for models
> print(myRES_rawMSE)

       pcr     pls
1 716.6597 716.854
```

```
> # Print out the reduced number of inputs after preprocessing
> print(myRES_PPlabels)

   Orig Input No                    Labels
1              1             Balaban.index
2              2           Dreiding.energy
3              3 Fused.aromatic.ring.count
4              4         Hyper.wiener.index
5              5              Szeged.index
6              6         Ring.count.of.atom
7              7                        pI
8              8      Quaternary_structure
9              9                   PLGA_Mw
10            10                 La_to_Gly
11            11      PVA_conc_inner_phase
12            12      PVA_conc_outer_phase
13            13                    PVA_Mw
14            14         Inner_phase_volume
```

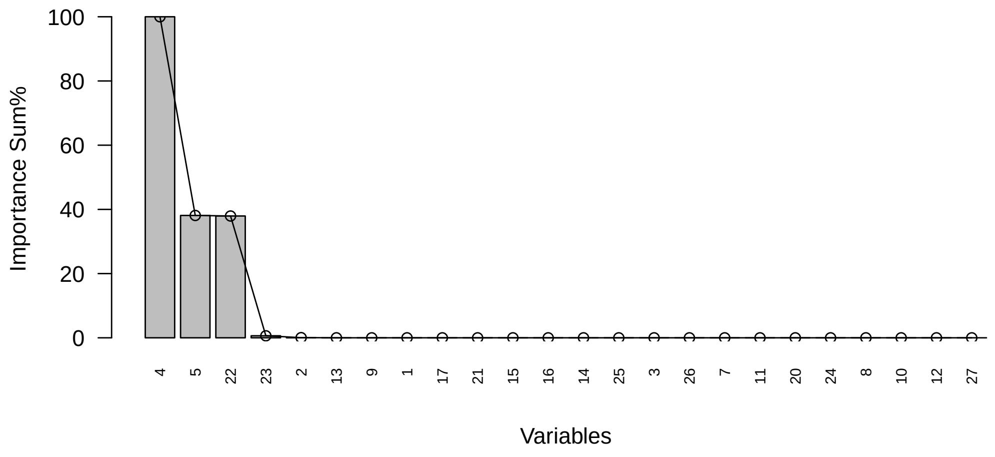

Figure 1: A sum of feature ranking of models trained and tested on dataset.train, two models were used "pcr","pls".

```
15            15              Encaps_rate
16            16                PLGA_conc
17            17       PLGA_to_Placticizer
18            18                  diss_pH
19            19                 diss_add
20            20              Prod_method
21            22   Asymmetric.atom.count.1
22            23      Hyper.wiener.index.1
23            24            Szeged.index.1
24            25                    count
25            26              pH_14_logd
26            27                    bpKa2
27            29       Cyclomatic.number.2
28            30            Mean_part_size
```

As one can see in the example there were only two models used "pcr","pls", to use all available models please set option Used.funcRegPred="all". The results can be presented on a bar plot (see Figure 1). Then the arbitrary feature reduction can be applied.