The rminer package for regression

Gabriele Venturato

Abstract

The aim of this work is to have an insight into the *rminer* package for regression analysis. Starting from a brief theoretical introduction, towards the description of the main functions of the package, and concluding with a simple case study to show how the package can be used.

Introduction

Regression

Regression is the problem of learning a *functional relationship* between variables using a dataset where the specific functional form learned depends on the choice of the model (it can be linear or not). The parameters of the function are learned using the *explanatory variables (features)* into the training set, and then performance are evaluates testing the model on the test set. The aim of a regression model — as opposed to a classification model — is to perform a *numeric prediction* based on the features in input.

Linear Regression

Random Forest

The rminer package

The goal of this package is to facilitate the use of data mining algorithms for classification and regression. It offers a short and coherent set of functions in order to easily develop a project, letting the user to follow in particular three CRISP-DM stages: *data preparation*, *modeling* and *evaluation*.

The package can be installed and loaded with:

```
install.packages("rminer")
```

And loaded with:

```
library(rminer)
```

As usual, a complete list of all functions available can be found in the documentation of the package:

```
help(package=rminer)
```

For the purpose of this work instead of reporting what can be found easily — and with more details — inside the help, I preferred to report a brief list of the function organized by their purpose, in order to quickly move toward the practical example that is more useful to show the package capabilites.

First of all, for the data preparation phase, after having loaded the dataset, the first function that can be used are mainly:

- `delevels(x, levels, label = NULL)` – reduce or replace factor x with *levels*, with an optional new label;
- `imputation(imethod = "value", D, Attribute = NULL, Missing = NA, Value = 1)` – perform imputation to remove missing values from dataset D and from a specific attribute, with the value specified.

- `CaseSeries` – create a data.frame from a time series (vector) using a sliding window. This function is not used in this work and its behavior can be further analized in official documentation.

Case Study: Life Expectancy

In this section it will be given a tour through the main functionalities of rminer by mean of a real life case study.

The dataset

The dataset is about Life Expectancy and can be found in Kaggle ("Life Expectancy (Who)," n.d.). This dataset is available thanks to the World Health Organization who keeps track of the health status for all countries. It contains data about 193 countries from the year 2000 to 2015. All data column have a pretty self-explanatory name. For more details one can have a look into the official website from which the dataset has been taken.

For the purpose of this work a quick idea about the data can be achieved with the summary function in R, after lodaing it.

```
lifeexp.df = read.csv("Life Expectancy Data.csv")
str(lifeexp.df)
summary(lifeexp.df) # here we can see NAs
```

```
## 'data.frame':    2938 obs. of  22 variables:
##  $ Country                        : Factor w/ 193 levels "Afghanistan",..: 1 1 1 1 1 ..
##  $ Year                           : int  2015 2014 2013 2012 2011 2010 2009 2008 2007..
##  $ Status                         : Factor w/ 2 levels "Developed","Developing": 2 2 ..
##  $ Life.expectancy                : num  65 59.9 59.9 59.5 59.2 58.8 58.6 58.1 57.5 5..
##  $ Adult.Mortality                : int  263 271 268 272 275 279 281 287 295 295 ...
##  $ infant.deaths                  : int  62 64 66 69 71 74 77 80 82 84 ...
##  $ Alcohol                        : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.03 0.02..
##  $ percentage.expenditure         : num  71.3 73.5 73.2 78.2 7.1 ...
##  $ Hepatitis.B                    : int  65 62 64 67 68 66 63 64 63 64 ...
##  $ Measles                        : int  1154 492 430 2787 3013 1989 2861 1599 1141 1..
##  $ BMI                            : num  19.1 18.6 18.1 17.6 17.2 16.7 16.2 15.7 15.2..
##  $ under.five.deaths              : int  83 86 89 93 97 102 106 110 113 116 ...
##  $ Polio                          : int  6 58 62 67 68 66 63 64 63 58 ...
##  $ Total.expenditure              : num  8.16 8.18 8.13 8.52 7.87 9.2 9.42 8.33 6.73 ..
##  $ Diphtheria                     : int  65 62 64 67 68 66 63 64 63 58 ...
##  $ HIV.AIDS                       : num  0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
##  $ GDP                            : num  584.3 612.7 631.7 670 63.5 ...
##  $ Population                     : num  33736494 327582 31731688 3696958 2978599 ...
##  $ thinness..1.19.years           : num  17.2 17.5 17.7 17.9 18.2 18.4 18.6 18.8 19 1..
##  $ thinness.5.9.years             : num  17.3 17.5 17.7 18 18.2 18.4 18.7 18.9 19.1 1..
##  $ Income.composition.of.resources: num  0.479 0.476 0.47 0.463 0.454 0.448 0.434 0.4..
##  $ Schooling                      : num  10.1 10 9.9 9.8 9.5 9.2 8.9 8.7 8.4 8.1 ...
```

```
##                 Country          Year            Status
##  Afghanistan        :  16   Min.   :2000   Developed : 512
##  Albania            :  16   1st Qu.:2004   Developing:2426
##  Algeria            :  16   Median :2008
##  Angola             :  16   Mean   :2008
##  Antigua and Barbuda:  16   3rd Qu.:2012
##  Argentina          :  16   Max.   :2015
##  (Other)            :2842
##  Life.expectancy Adult.Mortality infant.deaths       Alcohol
##  Min.   :36.30   Min.   :  1.0   Min.   :   0.0   Min.   : 0.0100
##  1st Qu.:63.10   1st Qu.: 74.0   1st Qu.:   0.0   1st Qu.: 0.8775
##  Median :72.10   Median :144.0   Median :   3.0   Median : 3.7550
##  Mean   :69.22   Mean   :164.8   Mean   :  30.3   Mean   : 4.6029
##  3rd Qu.:75.70   3rd Qu.:228.0   3rd Qu.:  22.0   3rd Qu.: 7.7025
##  Max.   :89.00   Max.   :723.0   Max.   :1800.0   Max.   :17.8700
##  NA's   :10      NA's   :10                       NA's   :194
##  percentage.expenditure  Hepatitis.B       Measles              BMI
##  Min.   :    0.000      Min.   : 1.00   Min.   :     0.0   Min.   : 1.00
##  1st Qu.:    4.685      1st Qu.:77.00   1st Qu.:     0.0   1st Qu.:19.30
##  Median :   64.913      Median :92.00   Median :    17.0   Median :43.50
##  Mean   :  738.251      Mean   :80.94   Mean   :  2419.6   Mean   :38.32
##  3rd Qu.:  441.534      3rd Qu.:97.00   3rd Qu.:   360.2   3rd Qu.:56.20
##  Max.   :19479.912      Max.   :99.00   Max.   :212183.0   Max.   :87.30
##                         NA's   :553                        NA's   :34
##  under.five.deaths     Polio       Total.expenditure   Diphtheria
##  Min.   :   0.00   Min.   : 3.00   Min.   : 0.370    Min.   : 2.00
##  1st Qu.:   0.00   1st Qu.:78.00   1st Qu.: 4.260    1st Qu.:78.00
##  Median :   4.00   Median :93.00   Median : 5.755    Median :93.00
##  Mean   :  42.04   Mean   :82.55   Mean   : 5.938    Mean   :82.32
##  3rd Qu.:  28.00   3rd Qu.:97.00   3rd Qu.: 7.492    3rd Qu.:97.00
##  Max.   :2500.00   Max.   :99.00   Max.   :17.600    Max.   :99.00
##                    NA's   :19      NA's   :226       NA's   :19
##     HIV.AIDS           GDP               Population
##  Min.   : 0.100   Min.   :     1.68   Min.   :3.400e+01
##  1st Qu.: 0.100   1st Qu.:   463.94   1st Qu.:1.958e+05
##  Median : 0.100   Median :  1766.95   Median :1.387e+06
##  Mean   : 1.742   Mean   :  7483.16   Mean   :1.275e+07
##  3rd Qu.: 0.800   3rd Qu.:  5910.81   3rd Qu.:7.420e+06
##  Max.   :50.600   Max.   :119172.74   Max.   :1.294e+09
##                   NA's   :448         NA's   :652
##  thinness..1.19.years thinness.5.9.years Income.composition.of.resources
##  Min.   : 0.10        Min.   : 0.10      Min.   :0.0000
##  1st Qu.: 1.60        1st Qu.: 1.50      1st Qu.:0.4930
##  Median : 3.30        Median : 3.30      Median :0.6770
##  Mean   : 4.84        Mean   : 4.87      Mean   :0.6276
##  3rd Qu.: 7.20        3rd Qu.: 7.20      3rd Qu.:0.7790
##  Max.   :27.70        Max.   :28.60      Max.   :0.9480
##  NA's   :34           NA's   :34         NA's   :167
##    Schooling
##  Min.   : 0.00
##  1st Qu.:10.10
##  Median :12.30
##  Mean   :11.99
##  3rd Qu.:14.30
##  Max.   :20.70
##  NA's   :163
```

From here can be seen that there are 22 columns and that some of them have missing values that will need to be taken care of. The purpose is to use the *Life.expectancy* variable as dependent, and all the others as predictors.

An important note here about the package is that since the country variable is stores as a factor, using this dataset I've find out that rminer can't handle factors with more than 53 levels, so I transformed the country factor as a numerical.

```
lifeexp.df$Country = as.numeric(lifeexp.df$Country)
```

Imputation

Here I manage the missing value taking advantage of the `imputation()` function of the package.

```
## IMPUTATION
# save column with missing values indexes
nacol = NULL
for (i in 1:ncol(lifeexp.df)) {
  if ( any(is.na(lifeexp.df[,i])) ) {
    nacol = c(nacol,i)
  }
}

# 1st method: case deletion
lifeexp.na.del = na.omit(lifeexp.df)

# 2nd method: imputation by mode
lifeexp.imp.mode = lifeexp.df
for (i in nacol) {
  lifeexp.imp.mode = imputation("value", lifeexp.imp.mode, i,
                                Value=which.max(table(na.omit(lifeexp.df[,i]))))
}

# 3rd mode: imputation by hotdeck
lifeexp.imp.hotdeck = lifeexp.df
for (i in nacol) {
  lifeexp.imp.hotdeck = imputation("hotdeck", lifeexp.imp.hotdeck, i)
}
```

The first part is for convenience: I extract the column indexes that correspod to variables in which there are missing values. Then, just to check out for different methods, I tried to trivially remove missing values, and then I used the imputation function: firstly substituting NAs with the mode, and secondly then with the hotdeck method implemented inside the rminer package.

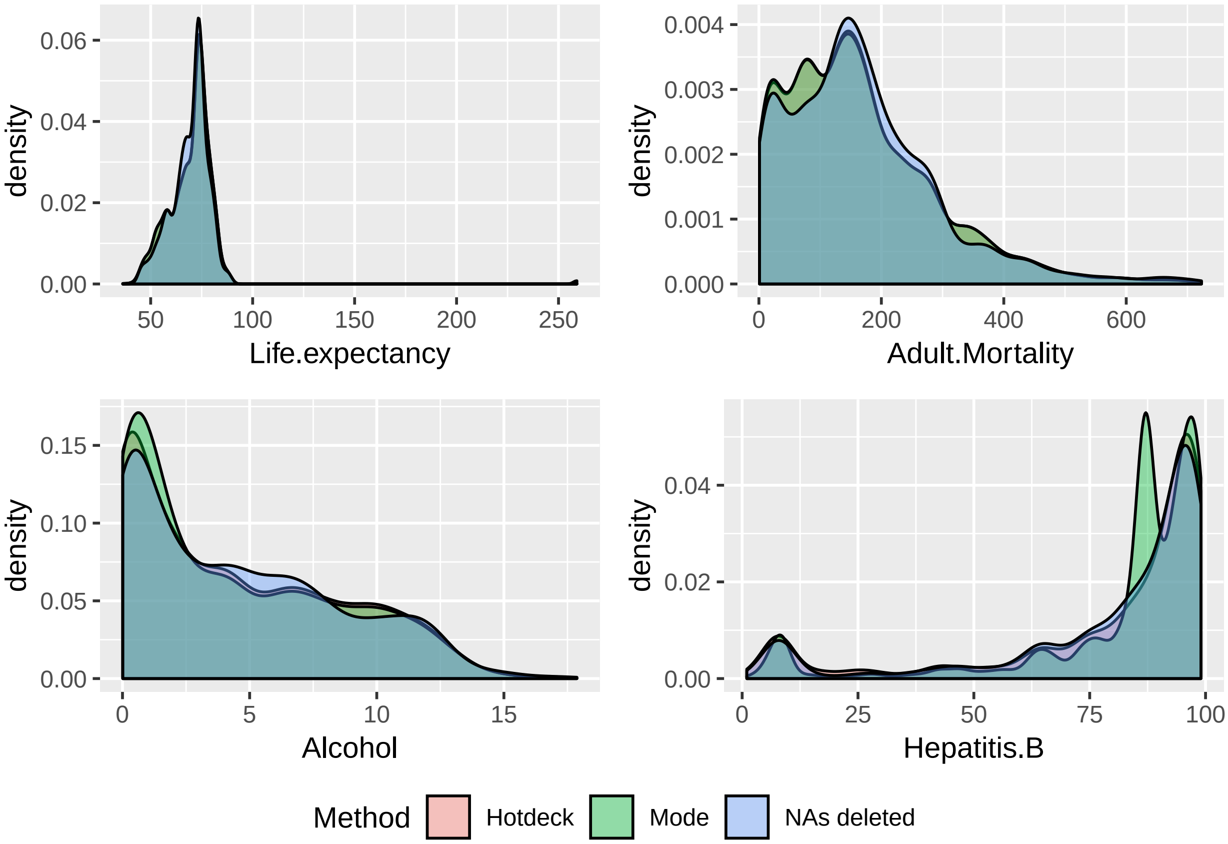
After this manupulation its possible to check the summary of the dataframe again to check the results (for example about the hotdeck method):

```
summary(lifeexp.imp.hotdeck)
```

```
##     Country           Year              Status     Life.expectancy
##  Min.   :  1.0   Min.   :2000   Developed : 512   Min.   :36.30
##  1st Qu.: 47.0   1st Qu.:2004   Developing:2426   1st Qu.:63.20
##  Median : 94.0   Median :2008                     Median :72.10
##  Mean   : 96.1   Mean   :2008                     Mean   :69.24
##  3rd Qu.:146.0   3rd Qu.:2012                     3rd Qu.:75.67
##  Max.   :193.0   Max.   :2015                     Max.   :89.00
##  Adult.Mortality infant.deaths       Alcohol        percentage.expenditure
##  Min.   :  1.0   Min.   :   0.0   Min.   : 0.0100   Min.   :    0.000
##  1st Qu.: 74.0   1st Qu.:   0.0   1st Qu.: 0.6425   1st Qu.:    4.685
##  Median :144.0   Median :   3.0   Median : 3.5650   Median :   64.913
##  Mean   :164.7   Mean   :  30.3   Mean   : 4.4763   Mean   :  738.251
##  3rd Qu.:227.0   3rd Qu.:  22.0   3rd Qu.: 7.5600   3rd Qu.:  441.534
##  Max.   :723.0   Max.   :1800.0   Max.   :17.8700   Max.   :19479.912
##   Hepatitis.B       Measles              BMI        under.five.deaths
##  Min.   : 1.00   Min.   :     0.0   Min.   : 1.00   Min.   :   0.00
##  1st Qu.:73.00   1st Qu.:     0.0   1st Qu.:19.20   1st Qu.:   0.00
##  Median :91.00   Median :    17.0   Median :43.00   Median :   4.00
##  Mean   :78.24   Mean   :  2419.6   Mean   :38.14   Mean   :  42.04
##  3rd Qu.:96.00   3rd Qu.:   360.2   3rd Qu.:56.10   3rd Qu.:  28.00
##  Max.   :99.00   Max.   :212183.0   Max.   :87.30   Max.   :2500.00
##      Polio       Total.expenditure   Diphtheria       HIV.AIDS
##  Min.   : 3.00   Min.   : 0.370    Min.   : 2.00   Min.   : 0.100
##  1st Qu.:78.00   1st Qu.: 4.290    1st Qu.:78.00   1st Qu.: 0.100
##  Median :93.00   Median : 5.750    Median :93.00   Median : 0.100
##  Mean   :82.43   Mean   : 5.951    Mean   :82.29   Mean   : 1.742
##  3rd Qu.:97.00   3rd Qu.: 7.470    3rd Qu.:97.00   3rd Qu.: 0.800
##  Max.   :99.00   Max.   :17.600    Max.   :99.00   Max.   :50.600
##       GDP              Population        thinness..1.19.years
##  Min.   :     1.68   Min.   :3.400e+01   Min.   : 0.100
##  1st Qu.:   462.23   1st Qu.:1.816e+05   1st Qu.: 1.600
##  Median :  1723.17   Median :1.363e+06   Median : 3.400
##  Mean   :  6924.41   Mean   :1.228e+07   Mean   : 4.881
##  3rd Qu.:  5468.43   3rd Qu.:7.538e+06   3rd Qu.: 7.200
##  Max.   :119172.74   Max.   :1.294e+09   Max.   :27.700
##  thinness.5.9.years Income.composition.of.resources   Schooling
##  Min.   : 0.100     Min.   :0.0000                  Min.   : 0.0
##  1st Qu.: 1.600     1st Qu.:0.4920                  1st Qu.:10.1
##  Median : 3.400     Median :0.6770                  Median :12.3
##  Mean   : 4.911     Mean   :0.6277                  Mean   :12.0
##  3rd Qu.: 7.300     3rd Qu.:0.7790                  3rd Qu.:14.3
##  Max.   :28.600     Max.   :0.9480                  Max.   :20.7
```

At the end, a brief comparison between the first four columns in which missing values have been managed (similar analysis can be checked for the others but requires more space) suggests that the hotdock method is a better — and less naif — compromise and tends to be more aligned with original data.

```r
plots = list()
j = 1
for (i in nacol[1:4]) {
    meth1=data.frame(v=lifeexp.na.del[[i]])
    meth2=data.frame(v=lifeexp.imp.mode[[i]])
    meth3=data.frame(v=lifeexp.imp.hotdeck[[i]])
    meth1$Method="NAs deleted"
    meth2$Method="Mode"
    meth3$Method="Hotdeck"
    all = rbind(meth1,meth2,meth3)
    plots[[j]] = ggplot(all,aes(v,fill=Method))+
      geom_density(alpha = 0.4)+
      xlab(colnames(lifeexp.df)[i])
    j = j+1
}
ggarrange(plotlist = plots, ncol=2, nrow=2, common.legend = TRUE, legend="bottom")
```

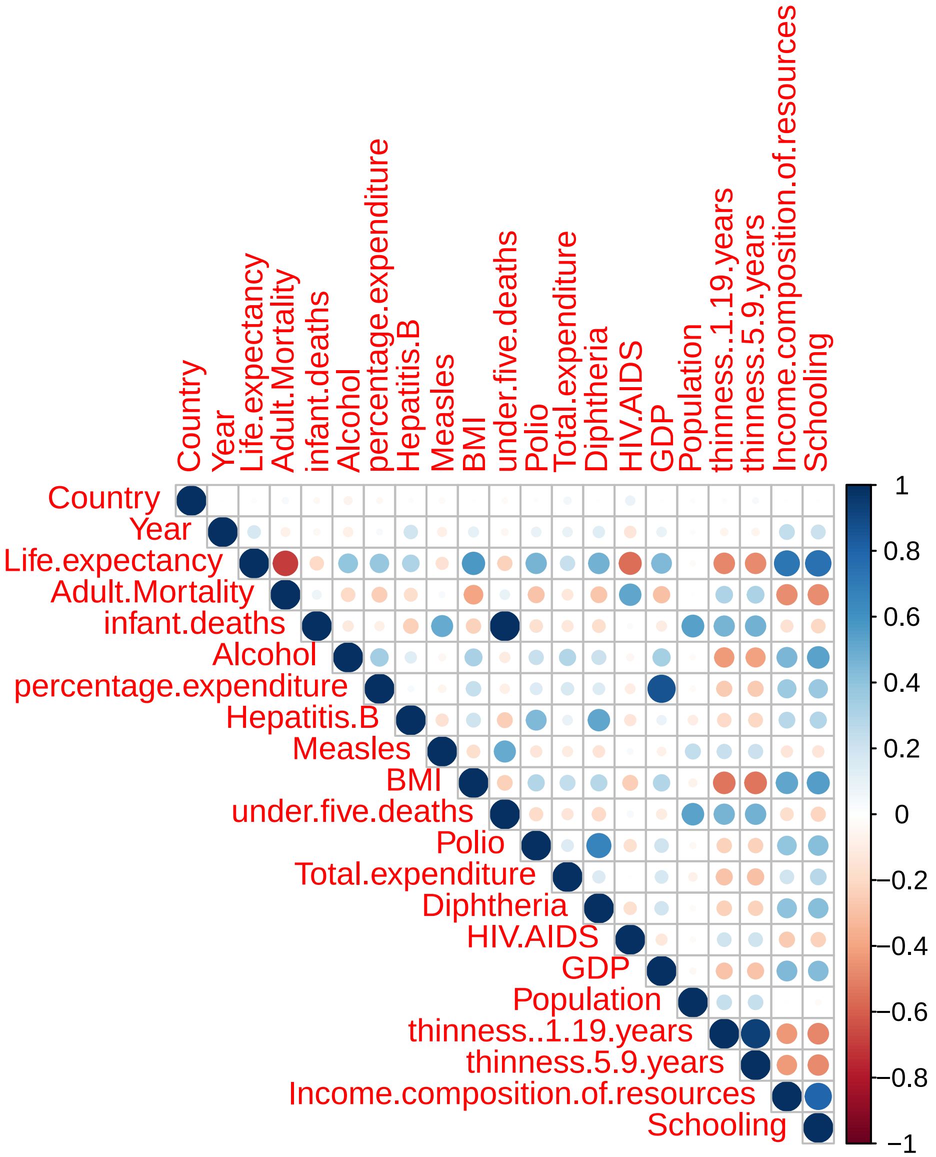


```
# we keep hotdeck version
lifeexp = lifeexp.imp.hotdeck
```

The final dataset is stored in a new variable with a more concise name for further model definition and evaluation.

Another quick insight that can be explored is to check the correlation matrix. Here we can see that there are not serius problem: some correlations are abvious considering the variables meaning, and anyway those with high correlation are the first to check out later in case of poor model.

```
correlation = cor(within(lifeexp, rm("Status")))
corrplot(correlation, type="upper", method="circle")
```



The model

As described above, the rminer package contains different models that can be used for regression analysis. *Random Forest* is only one of them. I've taken it as example of the package capabilities, but with small changes any other model can be used as same as this one.

In order to perform an analysis with a model it's important to have a train set to train the model, but it's necessary to have also a test set to evaluate the performance. Evaluating the model in the train set would lead to over-optimistic results.

For this purpose the package rminer lets the user to easily split the dataset into train and test sets, taking care of selecting random units in the right proportions. To this aim, I've trained the model in two different ways: one with the holdout method and one with 10-fold cross-validation.

Here's the code for model training:

```
# Holdout - Random Forest
H = holdout(lifeexp$Life.expectancy, ratio=2/3, seed=42)
```

```
summary(H)
```

```
##      Length Class  Mode
## tr   1958   -none- numeric
## itr     0   -none- NULL
## val     0   -none- NULL
## ts    980   -none- numeric
```

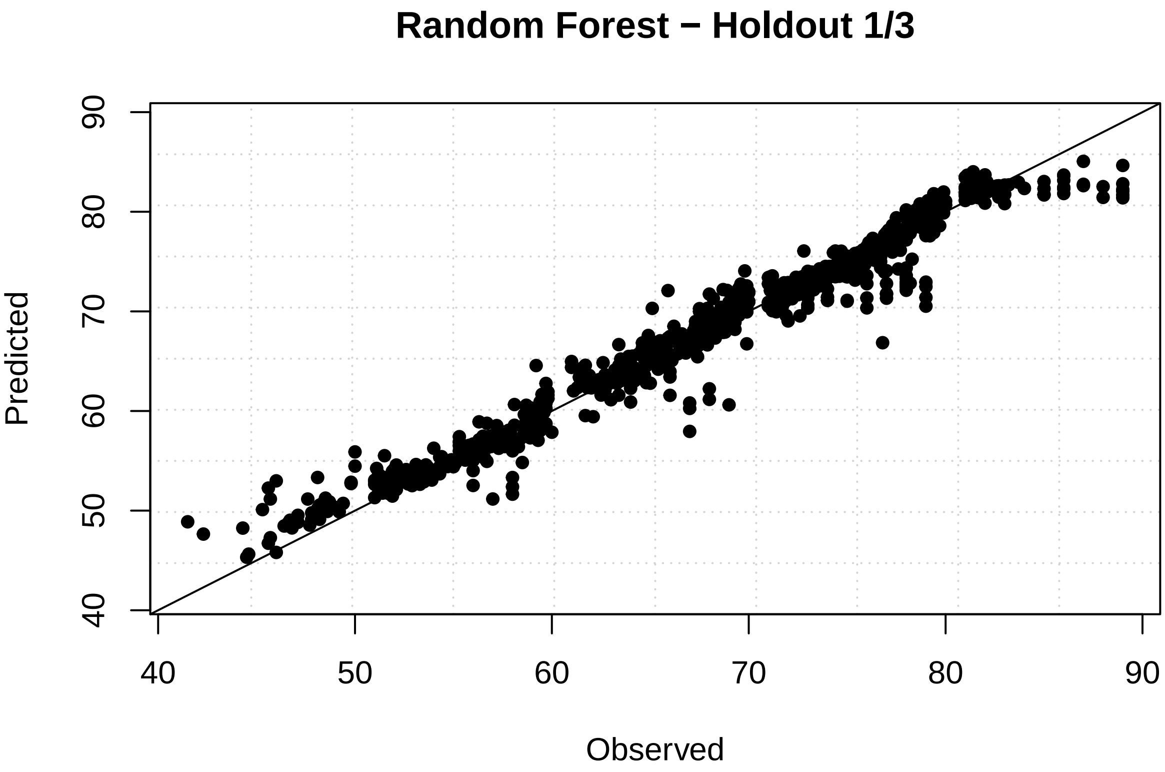
```
model1 = fit( Life.expectancy~., lifeexp[H$tr,], model="randomForest")

# 10-fold Cross-validation - Random Forest
model2 = crossvaldata(Life.expectancy~., lifeexp, fit, predict, ngroup=10, seed=42,
                      model="randomForest", task="reg")
```

As can be seen, thanks to the rminer package, it's a very easy task to accomplish. After this, one can proceed with model evaluation.

Evaluation

The evaluation of the model is easy as the training. Using functions `mgraph` and `mmetric` can be printed the *Regression Scatter Plot* and all the metrics.

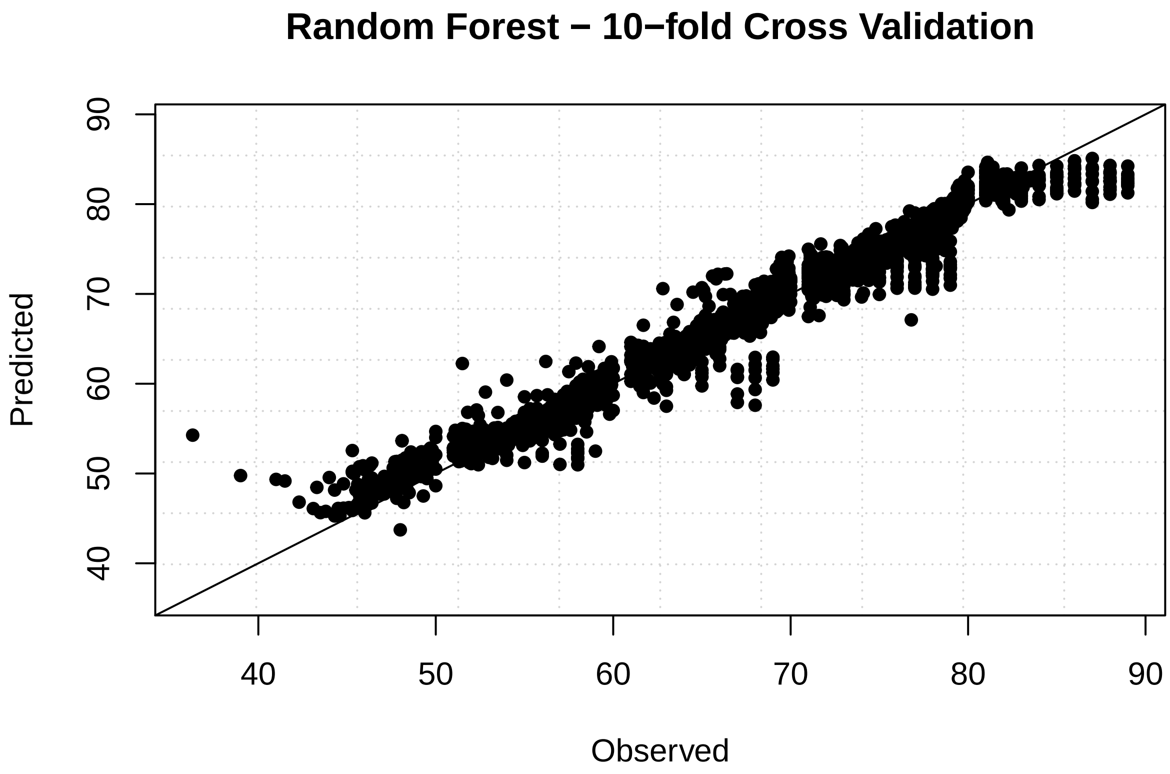
```
# Holdout
pred1 = predict(model1, lifeexp[H$ts,]) # get predictions on test set (new data)
target1 = lifeexp[H$ts,]$Life.expectancy
mgraph(target1, pred1, graph="RSC", Grid=10, main="Random Forest - Holdout 1/3")
```



```
mmetric(target1, pred1, metric="ALL")
```

```
##             SAE            MAE          MdAE           GMAE          MaxAE
##   1.149393e+03  1.172850e+00  6.897750e-01  6.047759e-01  9.945883e+00
##           NMAE           RAE           SSE           MSE          MdSE
##   2.469158e+00  1.498127e+01  3.344539e+03  3.412795e+00  4.757896e-01
##           RMSE          GMSE         HRMSE           RSE          RRSE
##   1.847375e+00  0.000000e+00  2.835374e-02  3.756377e+00  1.938137e+01
##             ME           COR            q2            R2            Q2
##  -2.278791e-02  9.815993e-01  3.646290e-02  9.635371e-01  3.756377e-02
##          NAREC     TOLERANCE          MAPE         MdAPE         RMSPE
##   3.647441e-01  6.321942e-01  1.766593e+00  9.992816e-01  2.835374e-01
##         RMdSPE         SMAPE        SMdAPE   SMinkowski3   MMinkowski3
##   9.992818e-02  1.766986e+00  9.949653e-01  1.582929e+04  1.582929e+04
##   MdMinkowski3
##   1.582929e+04
```

```
# 10-fold cross-validation
pred2 = model2$cv.fit # k-fold predictions on full dataset
mgraph(lifeexp$Life.expectancy, pred2, graph="RSC", Grid=10,
       main="Random Forest - 10-fold Cross Validation")
```



```
mmetric(lifeexp$Life.expectancy, pred2, metric="ALL")
```

```
##            SAE           MAE          MdAE          GMAE         MaxAE
##   3.201716e+03  1.089760e+00  6.322617e-01  0.000000e+00  1.797014e+01
##           NMAE           RAE           SSE           MSE          MdSE
##   2.067856e+00  1.401288e+01  9.234248e+03  3.143039e+00  3.997553e-01
##           RMSE          GMSE         HRMSE           RSE          RRSE
##   1.772862e+00  0.000000e+00  2.870430e-02  3.474923e+00  1.864115e+01
##             ME           COR            q2            R2            Q2
##  -4.877063e-02  9.827976e-01  3.410884e-02  9.658912e-01  3.474923e-02
##          NAREC     TOLERANCE          MAPE         MdAPE         RMSPE
##   3.919582e-01  6.639671e-01  1.654958e+00  9.179158e-01  2.870430e-01
##        RMdSPE         SMAPE        SMdAPE   SMinkowski3   MMinkowski3
##  9.179158e-02  1.649981e+00  9.159594e-01  4.749332e+04  4.749332e+04
##  MdMinkowski3
##  4.749332e+04
```

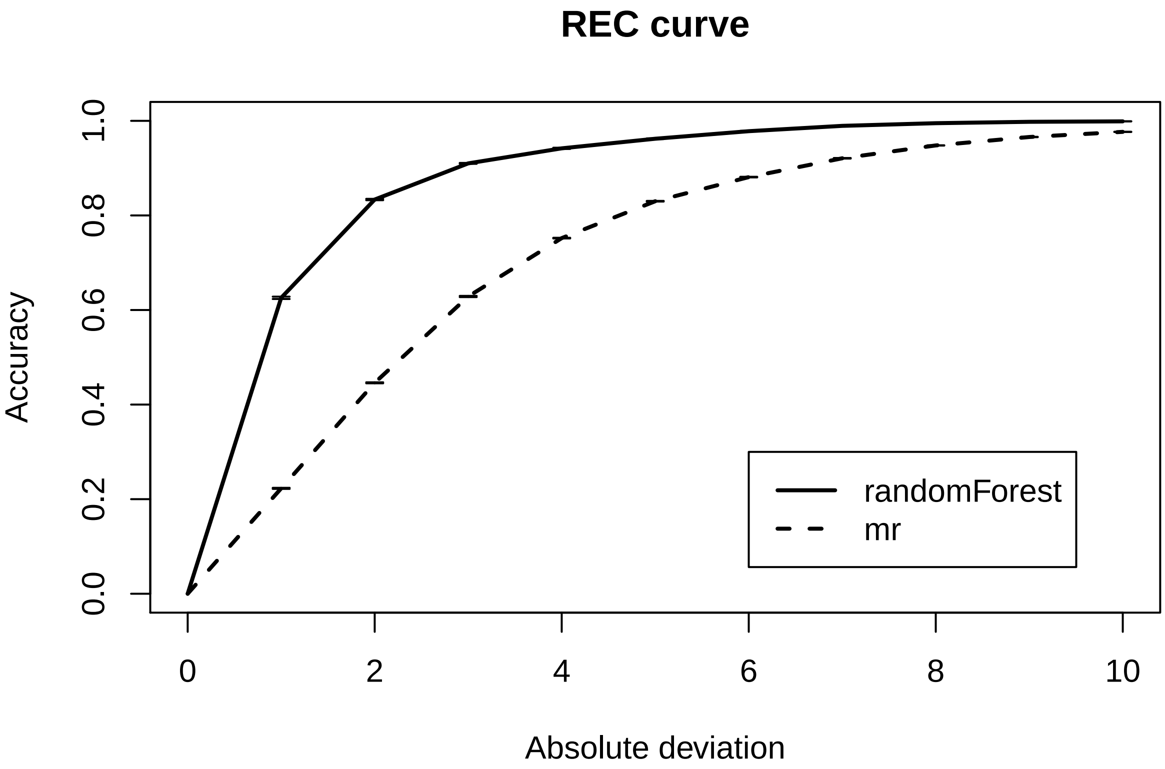
A further very useful function that can be used is the `mining` function. It lets the user to execute several fit and predict runs with a single line of code. And after mining, all the metrics are available for examination.

```
# mining for randomForest, external 3-fold, 20 Runs (=60 fitted models)
model3.mining = mining(Life.expectancy~., lifeexp,
                       model="randomForest", method=c("kfold",3,42), Runs=20)
m=mmetric(model3.mining, metric=c("MAE","RMSE","R2"))
print(m) # show metrics for each run
```

```
##         MAE     RMSE        R2
## 1  1.206532 1.939575 0.9592773
## 2  1.186874 1.859235 0.9626054
## 3  1.203920 1.902113 0.9608968
## 4  1.195467 1.885594 0.9613499
## 5  1.209864 1.925625 0.9597574
## 6  1.184972 1.876700 0.9618660
## 7  1.186807 1.883079 0.9616018
## 8  1.202549 1.907914 0.9606777
## 9  1.201508 1.900423 0.9607804
## 10 1.200257 1.903958 0.9607660
## 11 1.200390 1.907964 0.9605318
## 12 1.201168 1.931154 0.9598137
## 13 1.209126 1.904068 0.9608273
## 14 1.200502 1.909922 0.9604048
## 15 1.193869 1.876577 0.9617693
## 16 1.184531 1.879316 0.9617824
## 17 1.183369 1.875478 0.9618425
## 18 1.192731 1.881884 0.9615822
## 19 1.209022 1.917107 0.9600985
## 20 1.183677 1.870401 0.9620338
```

Finally one can be interest in comparing the mining of a model with the mining of another model, and this can be achieved with this commands:

```
# mining for standard multiple linear regression
model4.mining = mining(Life.expectancy~., lifeexp,
                       model="mr", method=c("kfold",3,42), Runs=20)

L=vector("list",2)
L[[1]]=model3.mining
L[[2]]=model4.mining
mgraph(L, graph="REC", leg=c("randomForest","mr"), main="REC curve", xval=10)
```



In this case the Random Forest model is compared with a standard multiple linear regression model. They are compared with REC curves. The Regression Error Characteristic (REC) curve is the corresponding of the ROC curve for regression. It plots the error tolerance on the x-axis versus the percentage of points predicted within the tolerance on the y-axis. More information about the REC curve can be found in (Bi and Bennett 2003).

Conclusion

Eventually, from this work it's evident that the package rminer is a good tool to perform regression analysis. With its small set of functions — but with a wide spread of options and parameters — can be useful to someone who want to do an overall analysis, but also to someone that want a finer granularity for personalization in model hyperparameters. In this brief tour of the package I didn't analyze the details about hyperparameters tuning, but with a quick look into the documentation one can affort this task too as easily as what done here. Must be said also that for an advanced user with very specific requirements, this package can be a bit limiting, but anyway, it's a very good starting point for a regression analysis.

References

Bi, Jinbo, and Kristin P Bennett. 2003. "Regression Error Characteristic Curves." In *Proceedings of the 20th International Conference on Machine Learning (Icml-03)*, 43–50.

Cortez, P. 2010. "Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool." In *Advances in Data Mining – Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*, edited by P. Perner, 572–83. Berlin, Germany: LNAI 6171, Springer.

"Life Expectancy (Who)." n.d. https://www.kaggle.com/kumarajarshi/life-expectancy-who/.

Trevor, Hastie, Tibshirani Robert, and Friedman JH. 2009. "The Elements of Statistical Learning: Data

Mining, Inference, and Prediction." New York, NY: Springer.

Witten, Ian H, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.