Project Linear & Non-Linear Models for Classification and Regression

Table of contents

1. Project Profile

Project Package overview for documentation and codes:

- **Title:** **Non-Linear Models for Classification and Prediction**
- **Project platform:** **Rstudio**
- **Submitted to:** **Dr. Amir Nakib**
- **Developed by:** **Michaël Faivre**

2. Introduction

2.1 Project Summary

The project encompasses both aspects of (I) testing linear models (ii) testing non-linear models and (iii) comparing the performances of all models. This is a stimulating project as it involves the use of various techniques in linear and non-linear modeling. I decided to implement the algorithm of analysis in R due to quickly accessible references and faster in coding compared with Python. However, I will code again the project in Python with sklearn package.

As the time is quite limited, **my aim in the study is to assess and quantify in terms of train/test accuracy how non-linear model(s) improve the classification vs the linear model.**

I am plainly conscious of making use of a tiny use of the proficient course we were , but at the current status, I can not do much better.

NB : All, many online references used are acknowledged in the codes.

2.2 Purpose

This document describes the framework and analytical choices made to solve the problem of classification from multivariate analysis. I am interested in having a predictive tool to in regards of several explanatory variables.

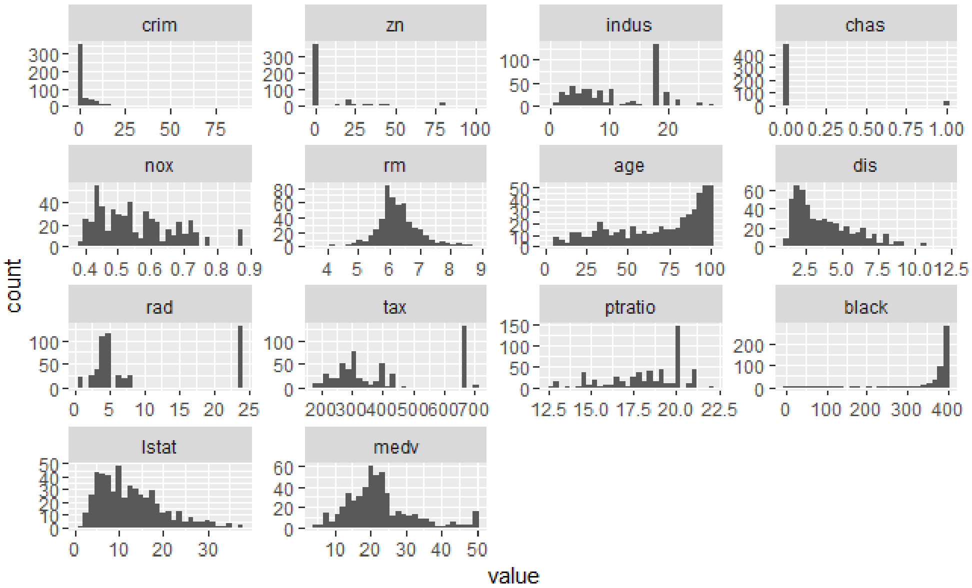
PART.1 : Classification

Dataset : Boston area housing real estate price

DS1.1 Dataset description

```
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
```

```
$ rm     : num  6.58 6.42 7.18 7 7.15 ...
$ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
$ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
$ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
$ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
$ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
$ black  : num  397 397 393 395 397 ...
$ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
$ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```



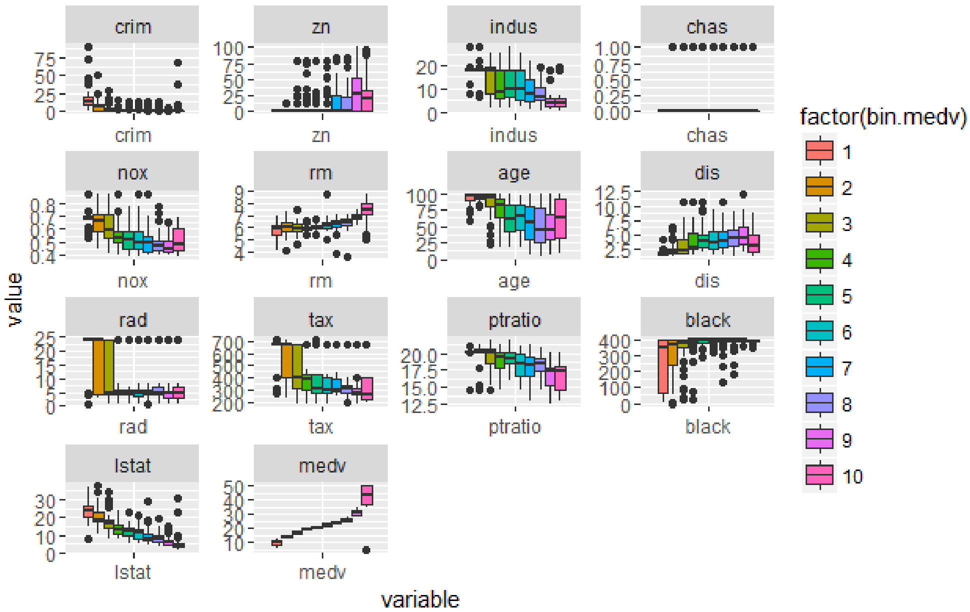
Note on variable distributions : exponential, uniform, normal-like distributions
Kernel Density Estimation is applied in the code : PROJET_exploratory_data_analysis.

DS1.2 Pre-processing operations

Table.1 List of preprocessing and basic tests

Any missing values ?	Checked with is.na filter
Variable selection and Dim reduction	PCA : 9+ variables required to have a cumulated variance greater than 80%. So, no obvious varialbe selection
Homoscedasticity	Not applied
Gaussianity	Not applied
Kernel Density Estimation (Parzen-Rosenblatt)	Tested for a couple of variables in PROJET_exploratory_data_analysis.Rmd
Classes building in uniform distribution from continuous response variable	Yes applied with 6 classes fitting min-max medv range

Box plot dispersion separated for each



One can notice quite significant dependencies of some explanatory variables with respect to response = 'medv'.

Methods explored :

- **Linear Classifiers : OLS, Generalized Linear Model, Random Forest, Naive Bayesian**
- **Non-Linear Classifiers : Knn,**

DS2.2 Random Forest on discretized response

Case 1. 10 uniformly distributed classes : bin.mdev

house.bin.rf.pred

	1	2	3	4	5	6	7	8	9	10
1	**15**	4	4	0	1	0	0	0	0	1
2	3	**9**	2	1	0	0	1	1	0	1
3	0	4	**4**	2	2	0	0	1	0	0
4	0	0	1	**4**	4	4	1	0	0	0
5	0	0	1	6	**3**	**2**	1	1	0	0
6	0	0	1	0	3	**1**	**2**	1	0	0
7	0	0	0	1	1	**5**	**0**	**4**	0	0
8	0	0	0	1	1	1	**8**	**8**	**3**	0
9	0	0	0	0	0	0	1	0	**8**	**3**
10	0	0	0	0	0	1	0	0	**3**	**11**

So, with 10 classes (quasi-uniform distribution out of the Gaussian like distribution of continuous variable 'mdev') corresponding to the bin.mdev variable does not give satisfactory classification. In peculiar, classes n° 5, 6, 7 perform quite poorly.

Following results are fom R code : PROJET_simplified_housing.Rmd

Detailed Confusion matrices for each method are provided in the 16 pages document of the folder. Here is only provided the list of method and accuracy (on test set).

DS2.1 Linear Discriminant Analysis on discretized response

Accuracy (average) : 0.5071

DS2.2 Multinomial Regression on discretized response

Accuracy (average) : 0.5202

DS2.3 Support Vector Machine on discretized response

Accuracy (average) : 0.5476

DS2.4 Naive Bayesian on discretized response

Accuracy (average) : 0.4966

DS2.5 Tree Bagged on discretized response

Accuracy (average) : 0.5678

DS2.6 Random Forest on discretized response

Accuracy (average) : 0.5782

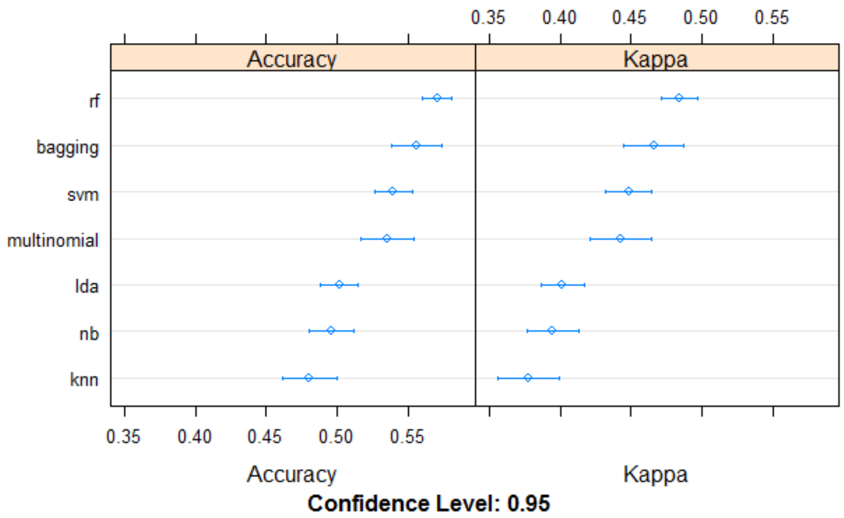
Model Comparison

Models are ranked fom lowest performance (top) to highest one (bottom). **Random Forest is the most performant model in this study.**

```
p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Accuracy
            lda       multinomial svm       knn       nb        bagging   rf
lda                   -0.034063   -0.038168  0.020978  0.005545 -0.054442 -0.069240
multinomial 0.1335807             -0.004105  0.055042  0.039608 -0.020378 -0.035177
svm         0.0214000 1.0000000              0.059147  0.043713 -0.016273 -0.031071
knn         0.8188253 0.0200346   0.0011525           -0.015433 -0.075420 -0.090218
nb          1.0000000 0.0174729   0.0032755 1.0000000           -0.059987 -0.074785
bagging     0.0003694 1.0000000   1.0000000 3.651e-05 0.0015108           -0.014798
rf          5.452e-08 0.1257553   0.0518518 2.243e-07 2.395e-07 1.0000000

Kappa
            lda       multinomial svm       knn       nb        bagging   rf
lda                   -0.041037   -0.046479  0.024300  0.007075 -0.064419 -0.082502
multinomial 0.1105625             -0.005442  0.065337  0.048112 -0.023383 -0.041465
svm         0.0173491 1.0000000              0.070779  0.053554 -0.017940 -0.036023
knn         0.8536219 0.0155059   0.0008664           -0.017225 -0.088720 -0.106802
nb          1.0000000 0.0101977   0.0023335 1.0000000           -0.071494 -0.089577
bagging     0.0005107 1.0000000   1.0000000 3.649e-05 0.0016697           -0.018083
rf          5.231e-08 0.1203189   0.0678302 1.727e-07 2.731e-07 1.0000000
```



Random Forest gives the best result here (6 uniform classes)

PART.2 : Regression

Neural Network

Following results are fom R code : PROJET_NeuralNetwork_Housing.Rmd

Neural Network setup from the R code :

```
#=== 3.a) prepare data
```{r}
maxs <- apply(dataHouseOrig, 2, max)
mins <- apply(dataHouseOrig, 2, min)
scaled <- as.data.frame(scale(dataHouseOrig, center = mins, scale = maxs - mins))

scale.train_ <- scaled[index,]
scale.test_  <- scaled[-index,]
```
```

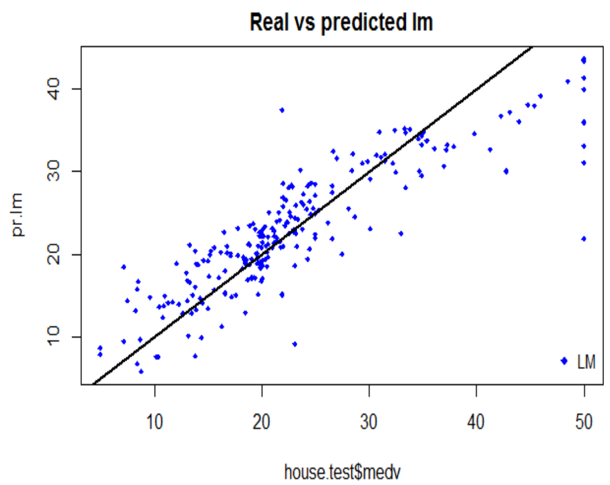
#=== 3.d) build the neural network (NN) for response = medv

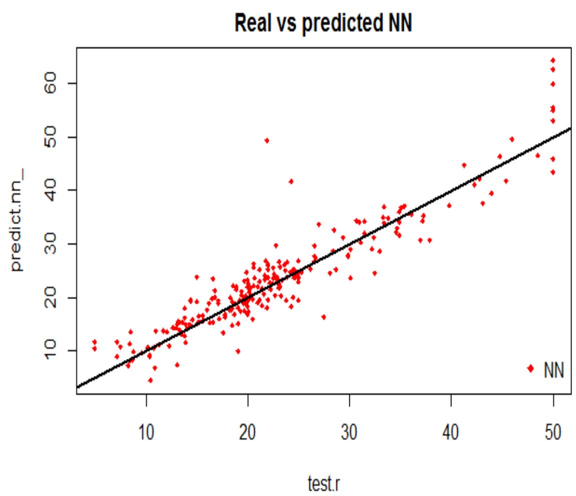
```
house.mdev.net <- neuralnet(formula.nn, data=scale.train_, hidden=c(10,8), learningrate = 0.01,act.fct
= "logistic",linear.output=T,algorithm="rprop+",lifesign = "minimal")
```

Result : training.ratio = 55%

Linear Logistic Regression on the 1 Class for House Median Value

Neural Network setup for 1 target continuous regression!





Neural Network brings here a clear and strong improvement compared to Logistic Regresion !
To quantify the achieved improvement, let use here the RMSE :

Mean Square Error for Logistic Regression = 21.4 vs MSE for NeuralNetwork = 13.6

Cross-validation for both Linear and Neural Network performance

Cross-validation result for the neuralnetwork tunned with 90% training, number of units in 2 hidden layers = (5,2) results in a MSE of 10.3 for neural network vs 13.5 for Logistic Regression.

APPENDIX : Boston correlogram