# QSAR biodegradation

*June, 2019*

## 1 INTRODUCTION

The QSAR biodegradation dataset was built in the Milano Chemometrics and QSAR Research Group.It is available in the UC Irvine Machine Learning Repository. The objective of this work is to obtain a model to classify the chemical compounds of said dataset into ready (RB) or not ready (NRB) biodegradable molecules. To this end we have 41 molecular descriptors and 1 experimental class:

1) SpMax_L: Leading eigenvalue from Laplace matrix
2) J_Dz: Balaban-like index from Barysz matrix weighted by Sanderson electronegativity
3) nHM: Number of heavy atoms
4) F01_N_N: Frequency of N-N at topological distance 1
5) F04_C_N: Frequency of C-N at topological distance 4
6) NssssC: Number of atoms of type ssssC
7) nCb_: Number of substituted benzene C(sp2)
8) C_percent: Percentage of C atoms
9) nCp: Number of terminal primary C(sp3)
10) nO: Number of oxygen atoms
11) F03_C_N: Frequency of C-N at topological distance 3
12) SdssC: Sum of dssC E-states
13) HyWi_B: Hyper-Wiener-like index (log function) from Burden matrix weighted by mass
14) LOC: Lopping centric index
15) SM6_L: Spectral moment of order 6 from Laplace matrix
16) F03_C_O: Frequency of C - O at topological distance 3
17) Me: Mean atomic Sanderson electronegativity (scaled on Carbon atom)
18) Mi: Mean first ionization potential (scaled on Carbon atom)
19) nN_N: Number of N hydrazines
20) nArNO2: Number of nitro groups (aromatic)
21) nCRX3: Number of CRX3
22) SpPosA_B: Normalized spectral positive sum from Burden matrix weighted by polarizability
23) nCIR: Number of circuits
24) B01_C_Br: Presence/absence of C - Br at topological distance 1
25) B03_C_Cl: Presence/absence of C - Cl at topological distance 3
26) N_073: Ar2NH / Ar3N / Ar2N-Al / R..N..R
27) SpMax_A: Leading eigenvalue from adjacency matrix (Lovasz-Pelikan index)
28) Psi_i_1d: Intrinsic state pseudoconnectivity index - type 1d
29) B04_C_Br: Presence/absence of C - Br at topological distance 4
30) SdO: Sum of dO E-states
31) TI2_L: Second Mohar index from Laplace matrix
32) nCrt: Number of ring tertiary C(sp3)
33) C_026: R–CX–R
34) F02_C_N: Frequency of C - N at topological distance 2
35) nHDon: Number of donor atoms for H-bonds (N and O)
36) SpMax_B: Leading eigenvalue from Burden matrix weighted by mass
37) Psi_i_A: Intrinsic state pseudoconnectivity index - type S average
38) nN: Number of Nitrogen atoms
39) SM6_B: Spectral moment of order 6 from Burden matrix weighted by mass
40) nArCOOR: Number of esters (aromatic)
41) nX: Number of halogen atoms
42) experimental class: ready biodegradable (RB) and not ready biodegradable (NRB)

This is a standard supervised classification task: the labels are included in the training data, what we have to do is to train a model to learn to predict the labels from the features. The label is binary: RB or NRB.

After a preliminary analysis of the dataset, we will try different classification models, also using different techniques (cross-validationk, normalization, PCA, tuning, staking...) in order to obtain the best performance from the algorithms and get the model that best suits us.

As for the metric to evaluate the models, we choose Accuracy.

# 2 DATA REVIEW

## 2.1 Dimensions

```
## [1] 1055   42
```

The file has 1055 instances and 42 variables.

## 2.2 Structure

```
## 'data.frame':    1055 obs. of  42 variables:
##  $ SpMax_L  : num  3.92 4.17 3.93 3 4.24 ...
##  $ J_Dz     : num  2.69 2.11 3.25 2.71 3.39 ...
##  $ nHM      : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ F01_N_N  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ F04_C_N  : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ NssssC   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nCb_     : int  0 0 0 0 0 0 0 0 2 2 ...
##  $ C_percent: num  31.4 30.8 26.7 20 29.4 28.6 11.1 31.6 44.4 41.2 ...
##  $ nCp      : int  2 1 2 0 2 2 0 3 2 0 ...
##  $ nO       : int  0 1 4 2 4 4 3 2 0 4 ...
##  $ F03_C_N  : int  0 0 0 0 0 0 0 0 0 3 ...
##  $ SdssC    : num  0 0 0 0 -0.271 -0.275 0 -0.039 0 -1.29 ...
##  $ HyWi_B   : num  3.11 2.46 3.28 2.1 3.45 ...
##  $ LOC      : num  2.55 1.393 2.585 0.918 2.753 ...
##  $ SM6_L    : num  9 8.72 9.11 6.59 9.53 ...
##  $ F03_C_O  : int  0 1 0 0 2 1 0 5 0 8 ...
##  $ Me       : num  0.96 0.989 1.009 1.108 1.004 ...
##  $ Mi       : num  1.14 1.14 1.15 1.17 1.15 ...
##  $ nN_N     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nArNO2   : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ nCRX3    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpPosA_B : num  1.2 1.1 1.09 1.02 1.14 ...
##  $ nCIR     : int  0 1 0 0 0 0 0 0 1 1 ...
##  $ B01_C_Br : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ B03_C_Cl : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ N_073    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpMax_A  : num  1.93 2.21 1.94 1.41 1.99 ...
##  $ Psi_i_1d : num  0.011 -0.204 -0.008 1.073 -0.002 ...
##  $ B04_C_Br : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SdO      : num  0 0 0 8.36 10.35 ...
##  $ TI2_L    : num  4.49 1.54 4.89 1.33 5.59 ...
##  $ nCrt     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_026    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ F02_C_N  : int  0 0 0 0 0 0 0 0 0 2 ...
##  $ nHDon    : int  0 0 1 1 0 0 1 0 0 1 ...
```

```
##  $ SpMax_B  : num  2.95 3.31 3.08 3.05 3.35 ...
##  $ Psi_i_A  : num  1.59 1.97 2.42 5 2.4 ...
##  $ nN       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ SM6_B    : num  7.25 7.26 7.6 6.69 8 ...
##  $ nArCOOR  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nX       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Eclass   : Factor w/ 2 levels "NRB","RB": 2 2 2 2 2 2 2 2 2 2 ...
```

The dependent variable is a factor with 2 levels. The rest of the variables ara integer or numeric.

## 2.3 Dependent variable distribution

```
##     freq percentage
## NRB  699   66.25592
## RB   356   33.74408
```

There are 66% instances in the NRB class and 33% in the RB class. That is, the file is imbalance, but not so much that we have to rebalance the dataset.

## 2.4 Summarize Data

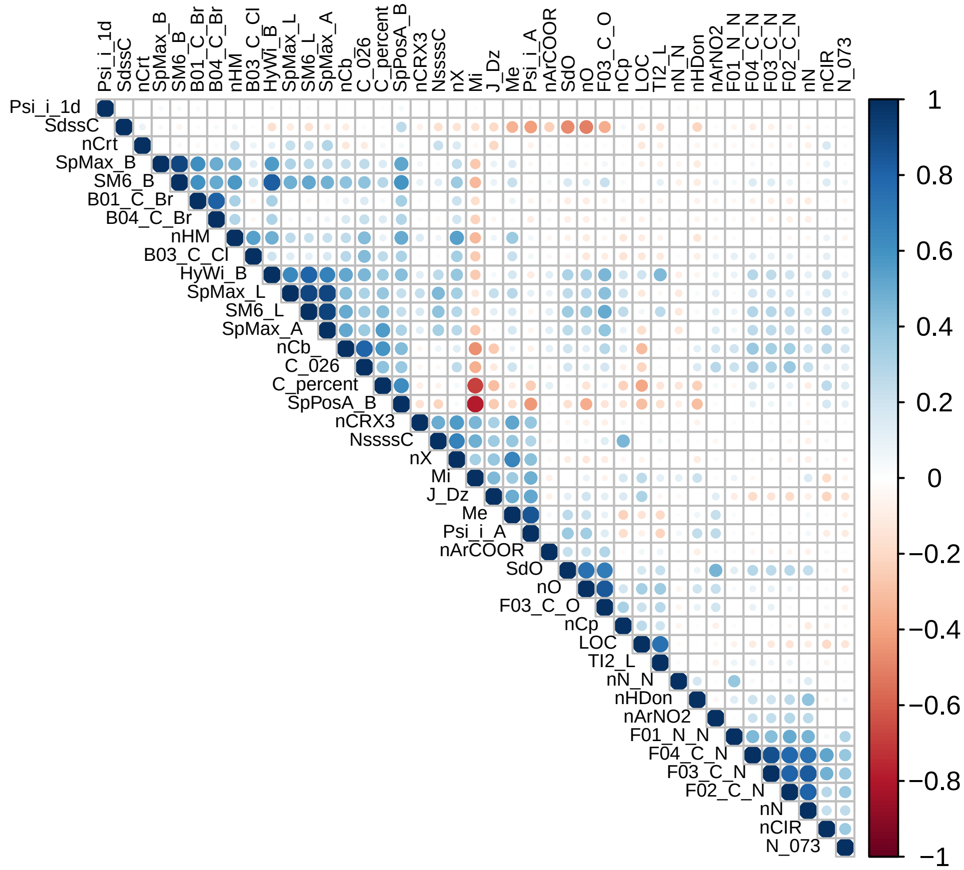
```
##     SpMax_L          J_Dz             nHM             F01_N_N
##  Min.   :2.000   Min.   :0.8039   Min.   : 0.0000   Min.   :0.00000
##  1st Qu.:4.481   1st Qu.:2.5027   1st Qu.: 0.0000   1st Qu.:0.00000
##  Median :4.828   Median :3.0463   Median : 0.0000   Median :0.00000
##  Mean   :4.783   Mean   :3.0695   Mean   : 0.7166   Mean   :0.04265
##  3rd Qu.:5.125   3rd Qu.:3.4377   3rd Qu.: 1.0000   3rd Qu.:0.00000
##  Max.   :6.496   Max.   :9.1775   Max.   :12.0000   Max.   :3.00000
##     F04_C_N           NssssC           nCb_          C_percent
##  Min.   : 0.0000   Min.   : 0.00   Min.   : 0.000   Min.   : 0.00
##  1st Qu.: 0.0000   1st Qu.: 0.00   1st Qu.: 0.000   1st Qu.:30.45
##  Median : 0.0000   Median : 0.00   Median : 1.000   Median :37.50
##  Mean   : 0.9801   Mean   : 0.29   Mean   : 1.646   Mean   :37.06
##  3rd Qu.: 1.0000   3rd Qu.: 0.00   3rd Qu.: 3.000   3rd Qu.:43.40
##  Max.   :36.0000   Max.   :13.00   Max.   :18.000   Max.   :60.70
##       nCp              nO             F03_C_N          SdssC
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :-5.2560
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:-0.1910
##  Median : 1.000   Median : 2.000   Median : 0.000   Median : 0.0000
##  Mean   : 1.376   Mean   : 1.804   Mean   : 1.437   Mean   :-0.1971
##  3rd Qu.: 2.000   3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.: 0.0000
##  Max.   :24.000   Max.   :12.000   Max.   :44.000   Max.   : 4.7220
##      HyWi_B          LOC            SM6_L           F03_C_O
##  Min.   :1.544   Min.   :0.000   Min.   : 4.174   Min.   : 0.00
##  1st Qu.:3.105   1st Qu.:0.875   1st Qu.: 9.533   1st Qu.: 0.00
##  Median :3.442   Median :1.187   Median :10.039   Median : 2.00
##  Mean   :3.477   Mean   :1.351   Mean   : 9.937   Mean   : 3.63
##  3rd Qu.:3.825   3rd Qu.:1.705   3rd Qu.:10.514   3rd Qu.: 6.00
##  Max.   :5.701   Max.   :4.491   Max.   :12.609   Max.   :40.00
##        Me              Mi             nN_N              nArNO2
##  Min.   :0.957   Min.   :1.022   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.983   1st Qu.:1.116   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :1.003   Median :1.130   Median :0.000000   Median :0.00000
##  Mean   :1.013   Mean   :1.131   Mean   :0.008531   Mean   :0.07393
##  3rd Qu.:1.029   3rd Qu.:1.143   3rd Qu.:0.000000   3rd Qu.:0.00000
```

```
##  Max.   :1.311   Max.   :1.377   Max.   :2.000000   Max.   :3.00000  
##      nCRX3           SpPosA_B          nCIR            B01_C_Br      
##  Min.   :0.00000   Min.   :0.863   Min.   :  0.000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:1.182   1st Qu.:  0.000   1st Qu.:0.00000  
##  Median :0.00000   Median :1.243   Median :  1.000   Median :0.00000  
##  Mean   :0.02938   Mean   :1.239   Mean   :  1.406   Mean   :0.03981  
##  3rd Qu.:0.00000   3rd Qu.:1.296   3rd Qu.:  2.000   3rd Qu.:0.00000  
##  Max.   :3.00000   Max.   :1.641   Max.   :147.000   Max.   :1.00000  
##     B03_C_Cl          N_073             SpMax_A         Psi_i_1d        
##  Min.   :0.0000   Min.   :0.00000   Min.   :1.000   Min.   :-1.099000  
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:2.101   1st Qu.:-0.008000  
##  Median :0.0000   Median :0.00000   Median :2.247   Median : 0.000000  
##  Mean   :0.1479   Mean   :0.03128   Mean   :2.216   Mean   :-0.001206  
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:2.358   3rd Qu.: 0.005000  
##  Max.   :1.0000   Max.   :3.00000   Max.   :2.859   Max.   : 1.073000  
##     B04_C_Br            SdO              TI2_L             nCrt       
##  Min.   :0.00000   Min.   : 0.000   Min.   : 0.444   Min.   :0.0000  
##  1st Qu.:0.00000   1st Qu.: 0.000   1st Qu.: 1.446   1st Qu.:0.0000  
##  Median :0.00000   Median : 0.000   Median : 2.052   Median :0.0000  
##  Mean   :0.02654   Mean   : 8.781   Mean   : 2.668   Mean   :0.1299  
##  3rd Qu.:0.00000   3rd Qu.:12.465   3rd Qu.: 3.146   3rd Qu.:0.0000  
##  Max.   :1.00000   Max.   :71.167   Max.   :17.537   Max.   :8.0000  
##      C_026            F02_C_N           nHDon           SpMax_B      
##  Min.   : 0.0000   Min.   : 0.000   Min.   :0.0000   Min.   : 2.267  
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 3.487  
##  Median : 0.0000   Median : 0.000   Median :1.0000   Median : 3.726  
##  Mean   : 0.8834   Mean   : 1.275   Mean   :0.9611   Mean   : 3.918  
##  3rd Qu.: 1.0000   3rd Qu.: 2.000   3rd Qu.:2.0000   3rd Qu.: 3.987  
##  Max.   :12.0000   Max.   :18.000   Max.   :7.0000   Max.   :10.695  
##     Psi_i_A            nN             SM6_B           nArCOOR       
##  Min.   :1.467   Min.   :0.0000   Min.   : 4.917   Min.   :0.00000  
##  1st Qu.:2.103   1st Qu.:0.0000   1st Qu.: 7.991   1st Qu.:0.00000  
##  Median :2.458   Median :0.0000   Median : 8.499   Median :0.00000  
##  Mean   :2.558   Mean   :0.6863   Mean   : 8.629   Mean   :0.05119  
##  3rd Qu.:2.870   3rd Qu.:1.0000   3rd Qu.: 9.021   3rd Qu.:0.00000  
##  Max.   :5.825   Max.   :8.0000   Max.   :14.700   Max.   :4.00000  
##        nX          Eclass   
##  Min.   : 0.0000   NRB:699  
##  1st Qu.: 0.0000   RB :356  
##  Median : 0.0000            
##  Mean   : 0.7232            
##  3rd Qu.: 0.0000            
##  Max.   :27.0000            
```

We can observe that some of the variables take few values diferent from 0; some take only positive values, but others take both positive and negative values.

# 3 DATA VISUALIZATION

In first place, we are going to calculate the correlations of the features.




We have some strong correlations: * SpMax_B with SM6B, HyWi_B, SM6_L and SpMax_A. * B01_C_Cr with B04 * SpMax_L with SM6_L and SpMax_A. * nCb_ with C_026 and C_percent * F04_C_N with F03_C_N , F02_C_N and nN * Me with Psi_i_A * nO with F03_C_O and SdO

Some of them have negative correlations. Such is the case of SpPosA_B and Mi and C_percent.

Therefore, we could remove some of these variables, since the information they provide is redundant and some algorithms work better with not highly correlated features.

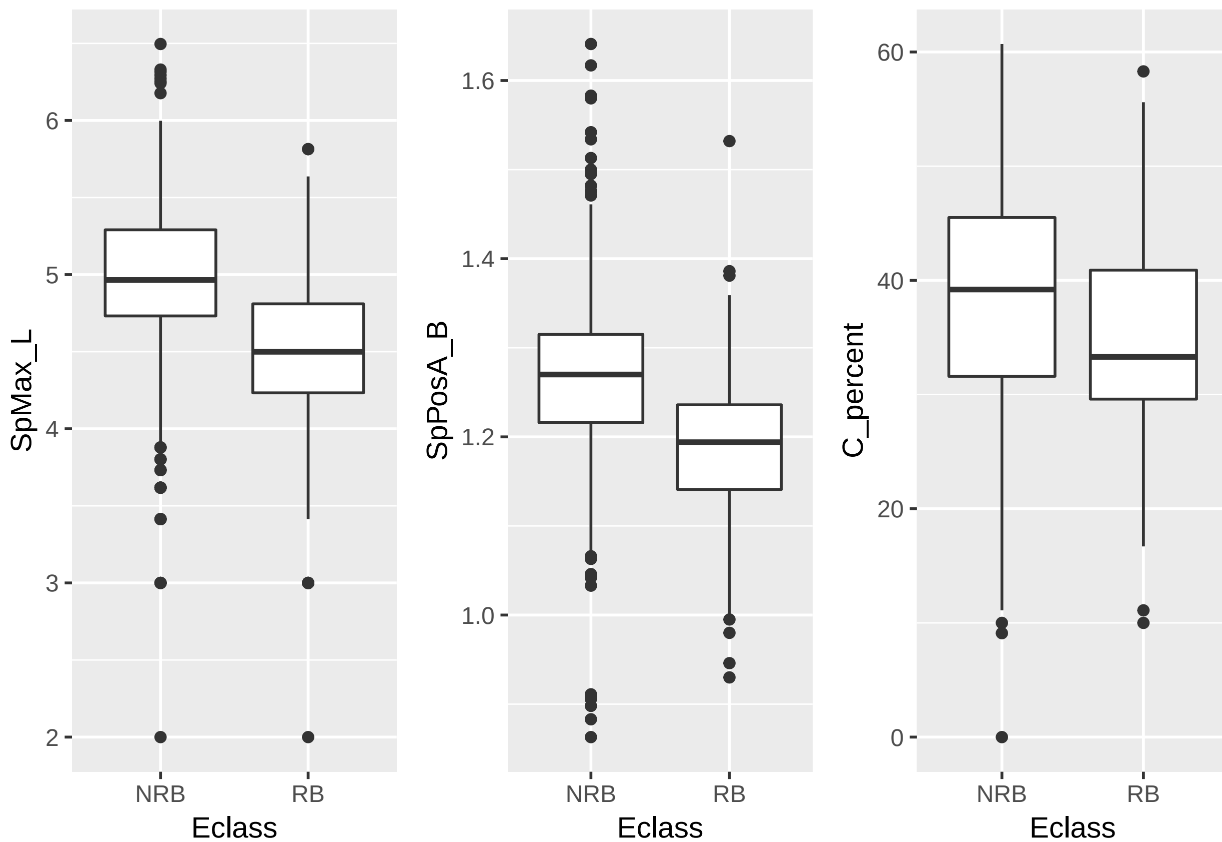
As Eclass is a qualitative variable, in order to obtain the Pearson's correlation, we transfort it into a quantitative varible. Its correlation with each of the features is:

```
##      SpMax_L         J_Dz          nHM      F01_N_N      F04_C_N
## -0.396138020 -0.001900062 -0.299107095 -0.103290258 -0.234618065
##       NssssC         nCb_    C_percent          nCp           nO
## -0.170449688 -0.337267836 -0.201603321 -0.056141620  0.177183328
##      F03_C_N        SdssC       HyWi_B          LOC        SM6_L
## -0.242325352 -0.112425177 -0.343778868  0.275320658 -0.343376690
##      F03_C_O           Me           Mi         nN_N       nArNO2
## -0.002878905 -0.091519764  0.131555361 -0.059831142 -0.153639506
##        nCRX3     SpPosA_B         nCIR     B01_C_Br     B03_C_Cl
## -0.096238814 -0.372253904 -0.116612921 -0.114554019 -0.252103161
##        N_073      SpMax_A     Psi_i_1d     B04_C_Br          SdO
## -0.091820393 -0.389950708 -0.025021552 -0.092893259  0.053636307
##        TI2_L         nCrt        C_026      F02_C_N       nHDon
##  0.173571596 -0.106590117 -0.318546591 -0.268874987 -0.027387003
##      SpMax_B      Psi_i_A           nN        SM6_B      nArCOOR
## -0.289618975  0.114895695 -0.261750540 -0.366793219  0.149510351
```

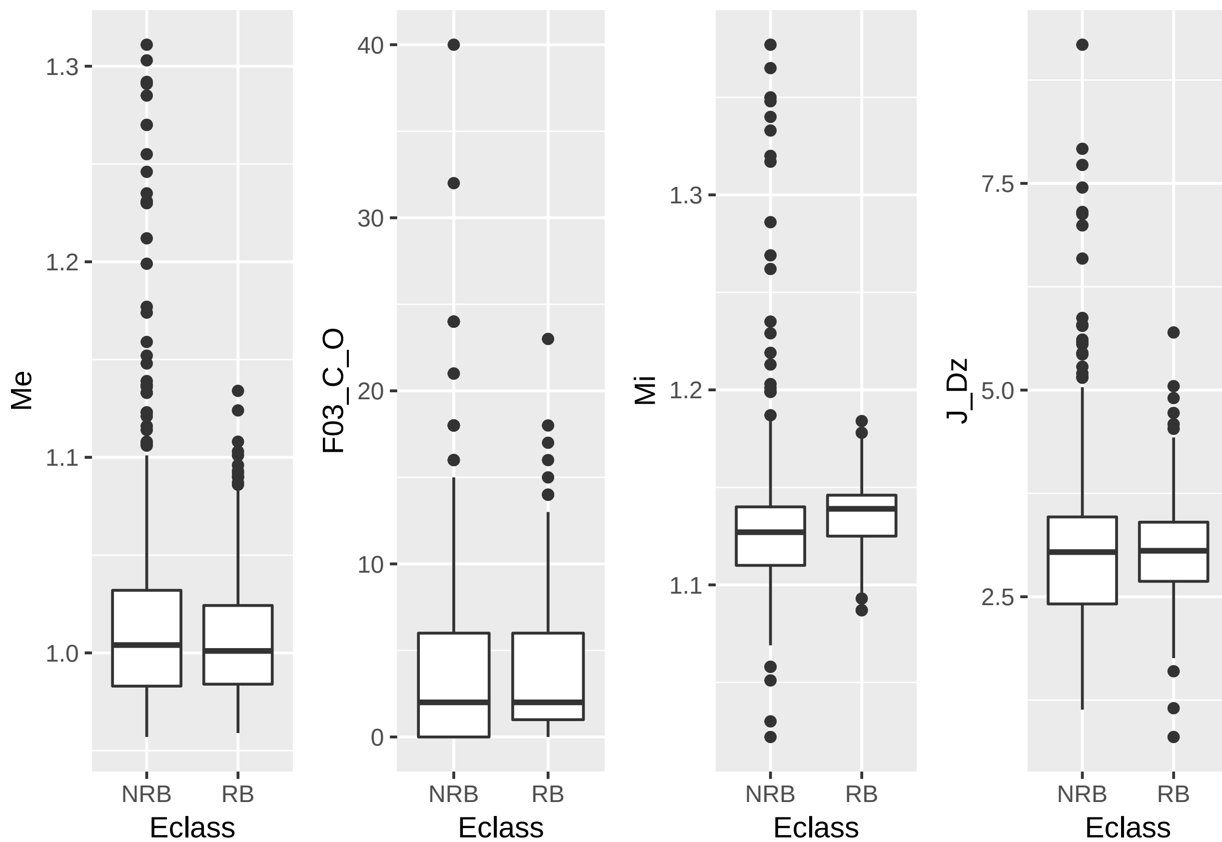
```
##           nX
## -0.214476110
```

None of the variables is strongly correlated with the variable to be predicted, although some of them have a slight correlation, with Pearson coefficients between -0.39 and 0.17.

As we can see in following the examples, the values that some of features take vary according to Eclass:




In contrast, in less highly correlated features we see that the boxplots are very similar for both values of Eclass:




# 4 DATA CLEAN

```
##    SpMax_L       J_Dz        nHM    F01_N_N    F04_C_N     NssssC       nCb_
##          0          0          0          0          0          0          0
## C_percent        nCp         nO    F03_C_N      SdssC    HyWi_B        LOC
##          0          0          0          0          0          0          0
##      SM6_L    F03_C_O         Me         Mi      nN_N    nArNO2      nCRX3
##          0          0          0          0          0          0          0
##   SpPosA_B       nCIR   B01_C_Br   B03_C_Cl      N_073   SpMax_A   Psi_i_1d
##          0          0          0          0          0          0          0
##   B04_C_Br        SdO      TI2_L       nCrt      C_026    F02_C_N      nHDon
##          0          0          0          0          0          0          0
##    SpMax_B    Psi_i_A         nN      SM6_B    nArCOOR         nX     Eclass
##          0          0          0          0          0          0          0
```

There are not NA values

# 5 RESULTS

We will try several linear and non-linear algorithms of the Caret package, using 10-fold cross-validation with 3 repeats. To evaluate them We will use the Accuracy and Kappa metrics.

## 5.1 Data split

After dividing the original dataset, we verify that the RB / NRB proportion in the training set is similar to the original:

```
##     freq percentage
## NRB  699   66.25592
## RB   356   33.74408
```

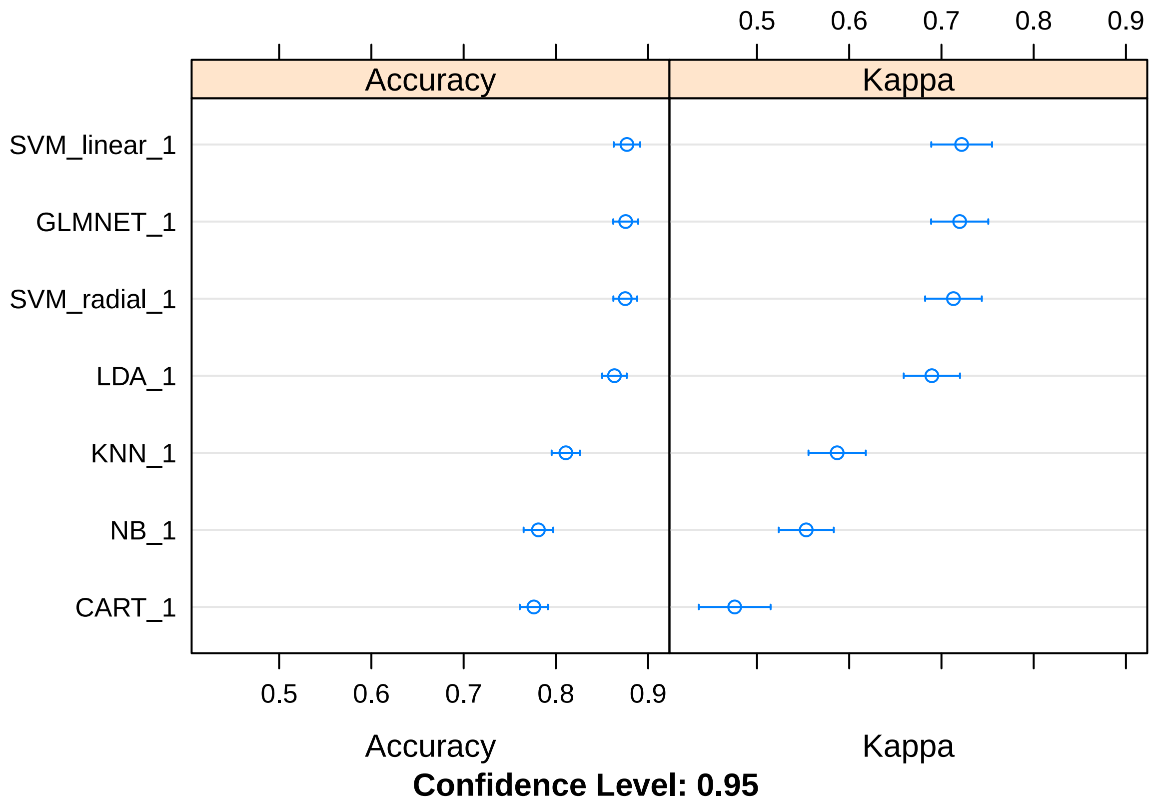
## 5.2 Basic models

As a first step, we will evaluate our chosen algorithms. We will use 10-fold cross-validation with 3 repeats, without any tranformation or tuning. The first algorithms that we are going to try are:

- k-Nearest Neighbors (KNN)
- Linear Discriminant Analysis (LDA)
- Penalized Linear Regression (GLMNET)
- Classification and Regression Trees (CART)
- Naive Bayes (NB)
- Support Vector Machines with Radial Basis Functions (SVM Radial)
- Support Vector Machines with Linear Basis Functions (SVM Linear)

```
##
## Call:
## summary.resamples(object = results_1)
##
## Models: LDA_1, GLMNET_1, KNN_1, CART_1, NB_1, SVM_linear_1, SVM_radial_1
## Number of resamples: 30
##
## Accuracy
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_1        0.7702703 0.8513514 0.8648649 0.8635135 0.8918919 0.9189189
## GLMNET_1     0.7702703 0.8513514 0.8783784 0.8756757 0.9054054 0.9324324
## KNN_1        0.7432432 0.7837838 0.8040541 0.8108108 0.8479730 0.8783784
## CART_1       0.6891892 0.7567568 0.7702703 0.7761261 0.7972973 0.8648649
## NB_1         0.6621622 0.7702703 0.7837838 0.7810811 0.7972973 0.8513514
## SVM_linear_1 0.7567568 0.8547297 0.8783784 0.8770270 0.9054054 0.9459459
## SVM_radial_1 0.7972973 0.8547297 0.8783784 0.8752252 0.9020270 0.9459459
##              NA's
## LDA_1           0
## GLMNET_1        0
## KNN_1           0
## CART_1          0
## NB_1            0
## SVM_linear_1    0
## SVM_radial_1    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_1        0.4709840 0.6576955 0.6823697 0.6895723 0.7522823 0.8151540
## GLMNET_1     0.4709840 0.6725437 0.7281372 0.7197723 0.7854023 0.8504446
## KNN_1        0.4316896 0.5282548 0.5766808 0.5869155 0.6577390 0.7308003
## CART_1       0.2891125 0.4090845 0.4762175 0.4758431 0.5405817 0.6919234
## NB_1         0.3526942 0.5200842 0.5675676 0.5533853 0.5910096 0.6890756
## SVM_linear_1 0.4341546 0.6735173 0.7308003 0.7218842 0.7810480 0.8791837
## SVM_radial_1 0.5135846 0.6793406 0.7227034 0.7130183 0.7716465 0.8791837
```

```
##               NA's
## LDA_1            0
## GLMNET_1         0
## KNN_1            0
## CART_1           0
## NB_1             0
## SVM_linear_1     0
## SVM_radial_1     0
```



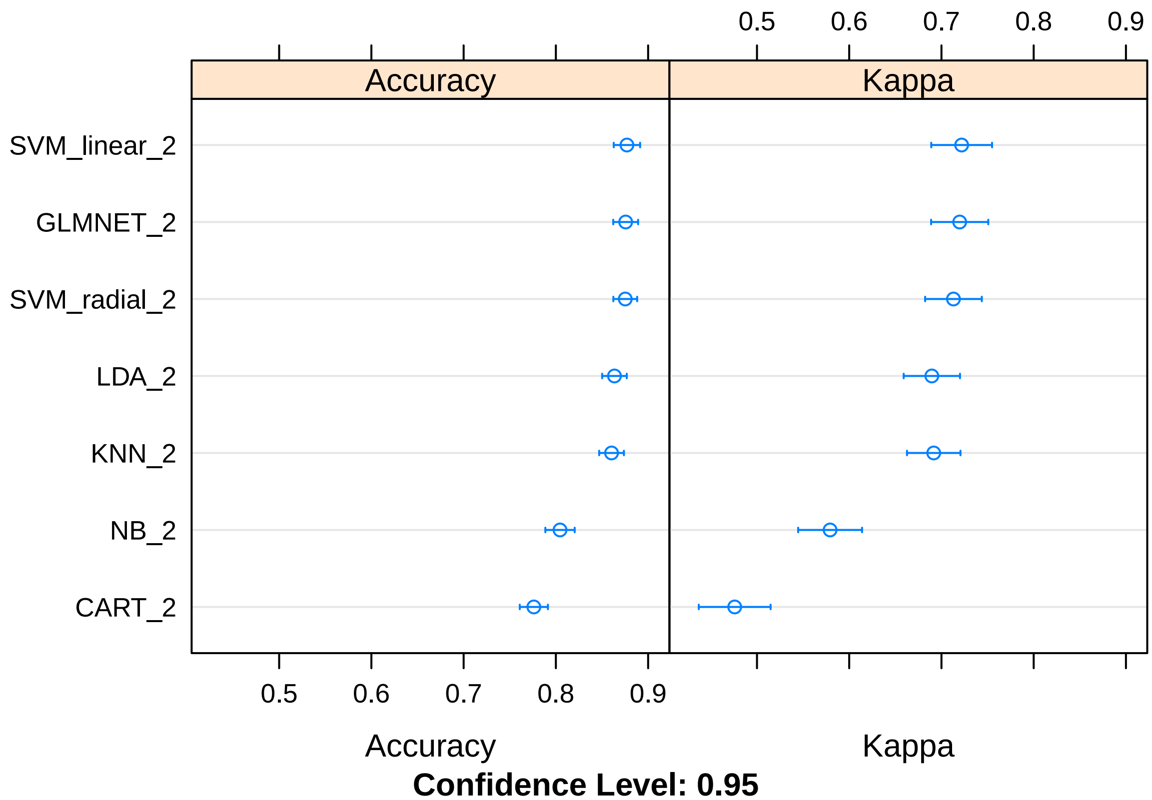

Except CART and NB, all the other algorithms have a mean Accuracy above 80%. SVM linear (87.70%), GLMNET (87.56%) and SVM Radial (87.52%) have the highest Accuracy. The same four also head the rank in terms of kappa values.

### 5.3 Applying transformations

We know than some algorithms work better if the data is regularized. We are going to try again the previous algorithms, but applying regularizations. In this case, we are going to center and use the same scale for all the features:

```
##
## Call:
## summary.resamples(object = results_2)
##
## Models: LDA_2, GLMNET_2, KNN_2, CART_2, NB_2, SVM_linear_2, SVM_radial_2
## Number of resamples: 30
##
## Accuracy
```

```
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_2        0.7702703 0.8513514 0.8648649 0.8635135 0.8918919 0.9189189
## GLMNET_2     0.7702703 0.8513514 0.8783784 0.8756757 0.9054054 0.9324324
## KNN_2        0.8108108 0.8378378 0.8648649 0.8603604 0.8783784 0.9594595
## CART_2       0.6891892 0.7567568 0.7702703 0.7761261 0.7972973 0.8648649
## NB_2         0.7027027 0.7736486 0.8108108 0.8045045 0.8378378 0.8783784
## SVM_linear_2 0.7567568 0.8547297 0.8783784 0.8770270 0.9054054 0.9459459
## SVM_radial_2 0.7972973 0.8547297 0.8783784 0.8752252 0.9020270 0.9459459
##              NA's
## LDA_2           0
## GLMNET_2        0
## KNN_2           0
## CART_2          0
## NB_2            0
## SVM_linear_2    0
## SVM_radial_2    0
##
## Kappa
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_2        0.4709840 0.6576955 0.6823697 0.6895723 0.7522823 0.8151540
## GLMNET_2     0.4709840 0.6725437 0.7281372 0.7197723 0.7854023 0.8504446
## KNN_2        0.5686928 0.6352318 0.6855107 0.6916197 0.7280834 0.9102668
## CART_2       0.2891125 0.4090845 0.4762175 0.4758431 0.5405817 0.6919234
## NB_2         0.3947955 0.5264431 0.6004800 0.5792778 0.6375510 0.7408560
## SVM_linear_2 0.4341546 0.6735173 0.7308003 0.7218842 0.7810480 0.8791837
## SVM_radial_2 0.5135846 0.6793406 0.7227034 0.7130183 0.7716465 0.8791837
##              NA's
## LDA_2           0
## GLMNET_2        0
## KNN_2           0
## CART_2          0
## NB_2            0
## SVM_linear_2    0
## SVM_radial_2    0
```

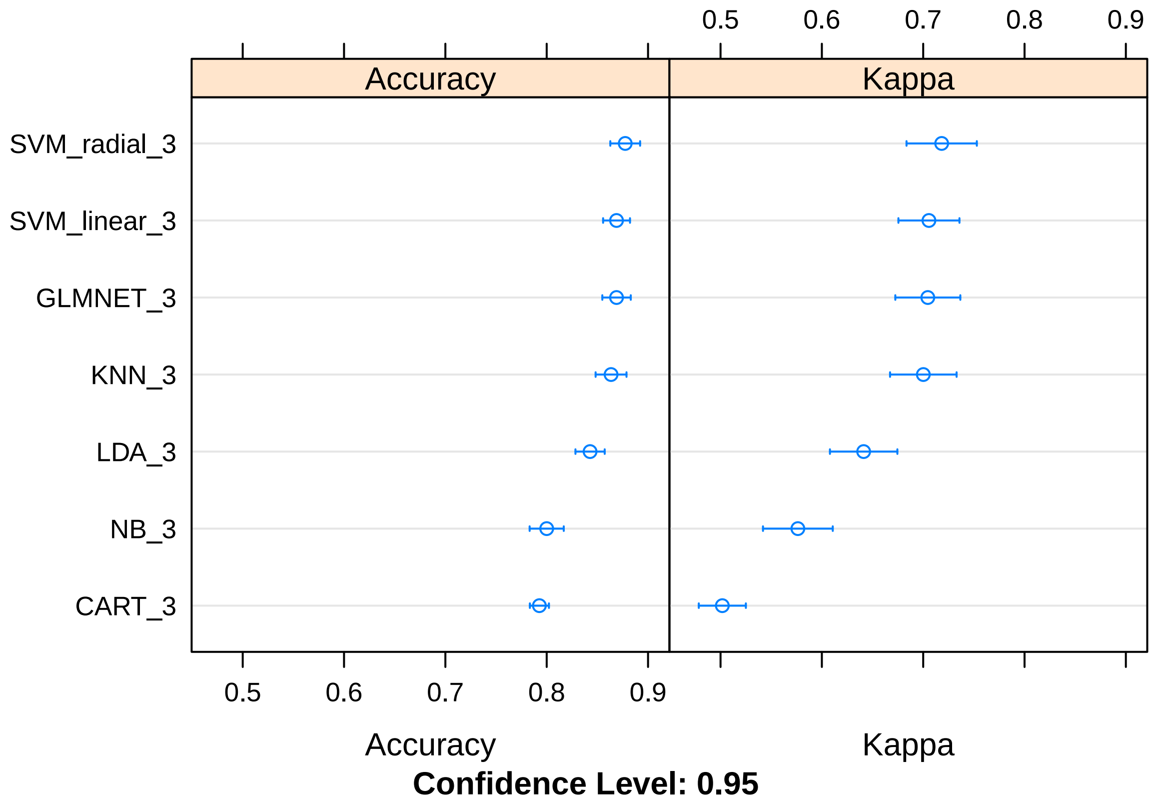



With this transformation, KNN has improved from 81.08% to 86.03%, and NB from 78.10% to 80.45%. The Accuracy of the other algorithms is the same than before the transformation.

We could try a PCA transformation, too, to avoid correlated attributes:

```
##
## Call:
## summary.resamples(object = results_3)
##
## Models: LDA_3, GLMNET_3, KNN_3, CART_3, NB_3, SVM_linear_3, SVM_radial_3
## Number of resamples: 30
##
## Accuracy
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_3        0.7567568 0.8243243 0.8378378 0.8427928 0.8648649 0.9054054
## GLMNET_3     0.7972973 0.8378378 0.8716216 0.8689189 0.8918919 0.9459459
## KNN_3        0.7972973 0.8412162 0.8513514 0.8635135 0.8885135 0.9459459
## CART_3       0.7432432 0.7837838 0.7972973 0.7927928 0.8108108 0.8513514
## NB_3         0.7027027 0.7601351 0.8108108 0.8000000 0.8243243 0.9054054
## SVM_linear_3 0.7972973 0.8412162 0.8648649 0.8689189 0.8918919 0.9459459
## SVM_radial_3 0.7837838 0.8513514 0.8851351 0.8774775 0.9054054 0.9459459
##              NA's
## LDA_3           0
## GLMNET_3        0
## KNN_3           0
## CART_3          0
## NB_3            0
```

```
## SVM_linear_3    0
## SVM_radial_3    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LDA_3        0.4223764 0.5982767 0.6445156 0.6412351 0.6964502 0.7864798
## GLMNET_3     0.5598980 0.6375510 0.7200742 0.7045600 0.7583673 0.8791837
## KNN_3        0.5598980 0.6389871 0.6890756 0.7001221 0.7539895 0.8815052
## CART_3       0.3838738 0.4553325 0.5000799 0.5017615 0.5391955 0.6506438
## NB_3         0.3838002 0.5133532 0.5852682 0.5763180 0.6291513 0.7864798
## SVM_linear_3 0.5411869 0.6593707 0.7037630 0.7056621 0.7618496 0.8791837
## SVM_radial_3 0.4970263 0.6576955 0.7370394 0.7182402 0.7854023 0.8791837
##              NA's
## LDA_3           0
## GLMNET_3        0
## KNN_3           0
## CART_3          0
## NB_3            0
## SVM_linear_3    0
## SVM_radial_3    0
```




In this case we could see improvements in the values of KNN from 86.03% to 86.35%, CART from 77.61% to 79.27%, and SVM Radial from 87.52% to 87.74%. All the other cases show worse values.

We could conclude that some transformations work better with some algorithms than with others.

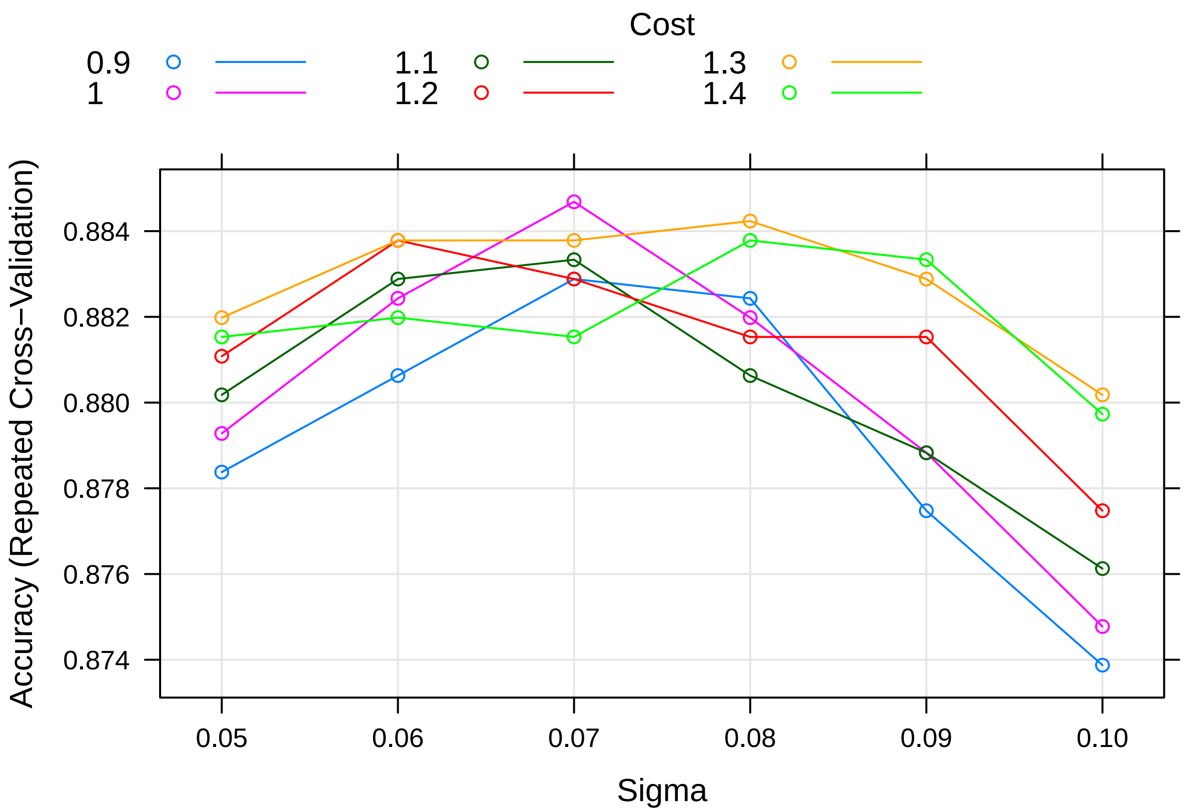
### 5.4 Tuning algorithms

Taking into account the previous results, we will take the two best algorithms and modify their parameters in order to get better predictions.

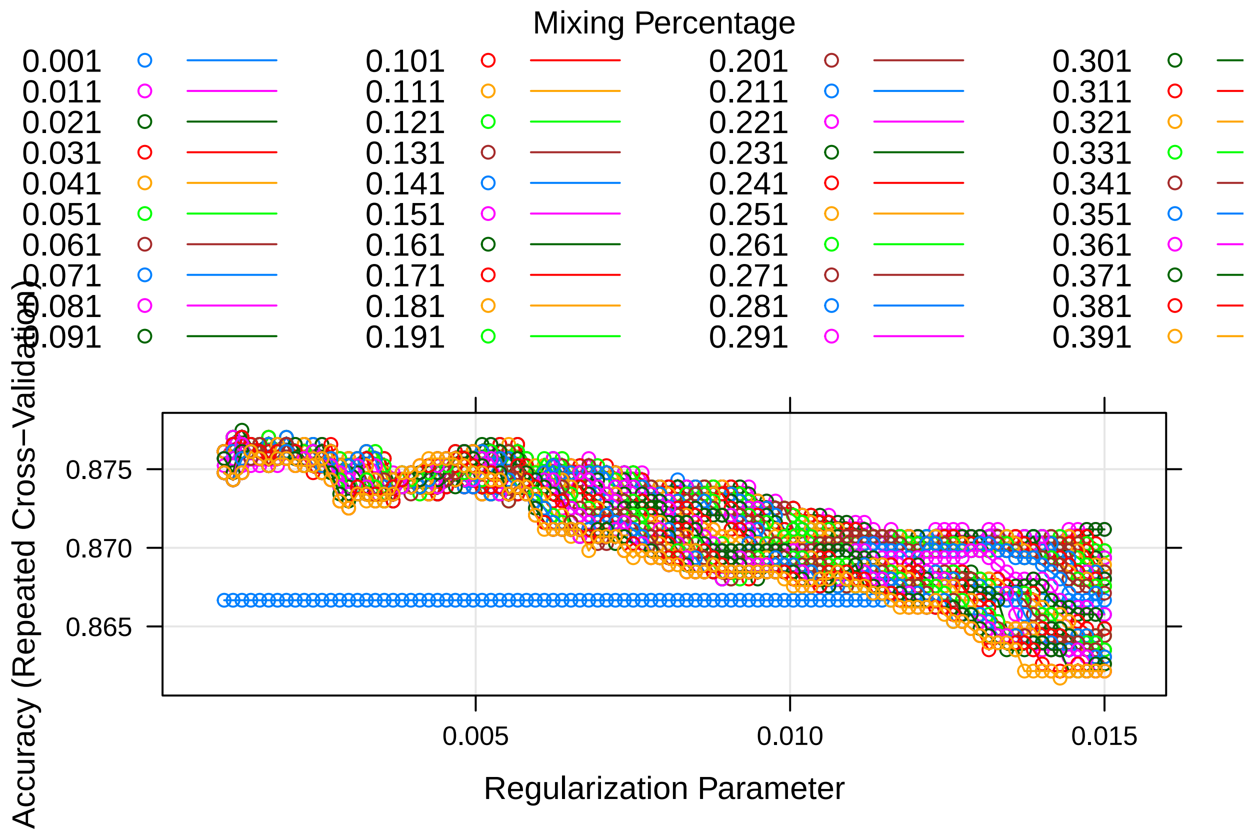
The SVM implementation has two parameters that we can tune: C and sigma. We will try values around the ones that got us the previous best result for this algorithm.

We do the same with the two parameters that can be tuned in the implementation of GMNLT.

```
## [1] "SVM RADIAL"
```

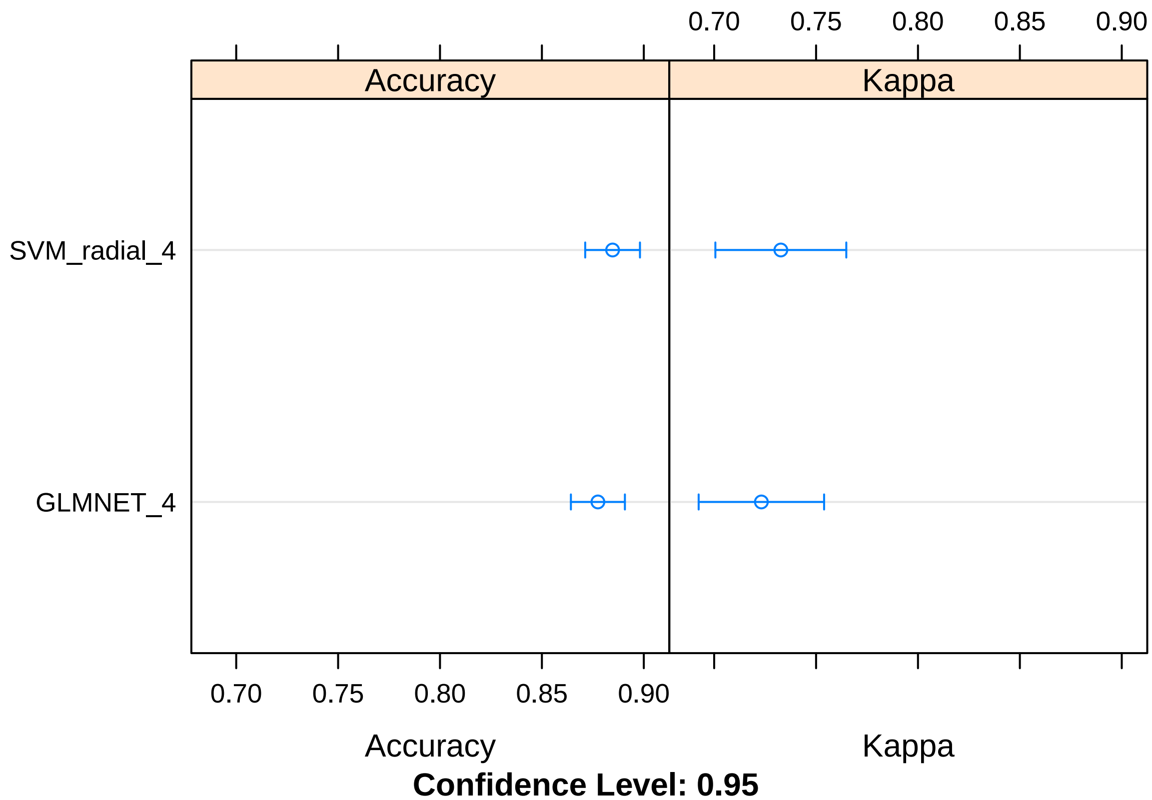



```
## [1] "GLMNET"
```




After repeating the process several times, adjusting the parameters, we choose the optimal ones.

```
##
## Call:
## summary.resamples(object = results_4)
##
## Models: GLMNET_4, SVM_radial_4
## Number of resamples: 30
##
## Accuracy
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## GLMNET_4     0.7702703 0.8513514 0.8783784 0.8774775 0.9054054 0.9324324
## SVM_radial_4 0.8108108 0.8648649 0.8918919 0.8846847 0.9054054 0.9594595
##              NA's
## GLMNET_4        0
## SVM_radial_4    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## GLMNET_4     0.4709840 0.6777047 0.7308003 0.7231663 0.7821699 0.8504446
## SVM_radial_4 0.5507372 0.6948833 0.7432784 0.7327172 0.7854023 0.9084913
##              NA's
## GLMNET_4        0
## SVM_radial_4    0
```




In both cases, the tuning makes a small difference. With GLMNET we go from 87.56% to 87.74%. In the SVM Radial case, we go from 87.52% to 88.46%.

We could try too some ensemble methods:

- Random Forest (RF)
- Bagged CART (Treebag)

```
##
## Call:
## summary.resamples(object = results_5)
##
## Models: RF, TREEBAG
## Number of resamples: 30
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF      0.7837838 0.8513514 0.8648649 0.8702703 0.9054054 0.9459459    0
## TREEBAG 0.7567568 0.8513514 0.8783784 0.8671171 0.8918919 0.9189189    0
##
## Kappa
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF      0.4865568 0.6506438 0.6885286 0.7006851 0.7776824 0.8791837    0
## TREEBAG 0.4223764 0.6664203 0.7141631 0.6967783 0.7535387 0.8187755    0
```

Comparing these results with those of the previous algorithms, we see that the Accuracy values are better now than what we get with some of the previous algorithms.

## 5.5 Validation

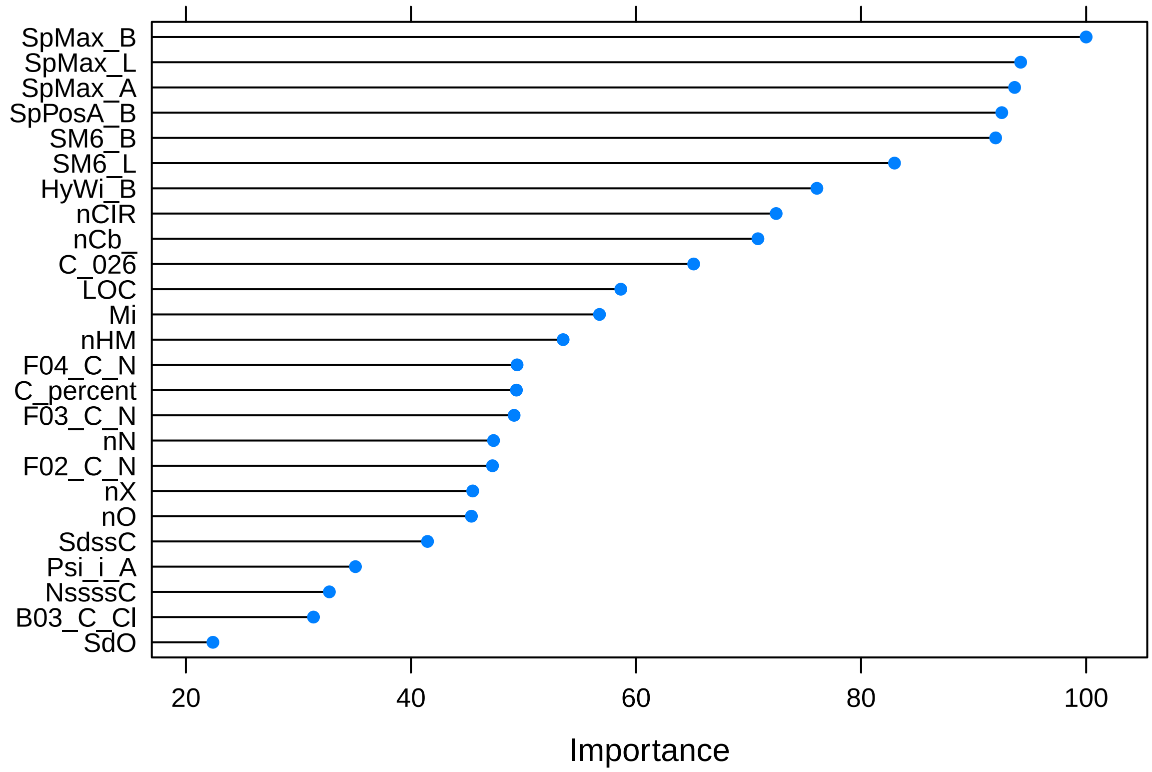
Our best model until now is the tuned SVM Radial. In the next steps we will calculate the confusion matrix and some more metrics using de test dataset.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 187  23
##        RB   22  83
##
##                Accuracy : 0.8571
##                  95% CI : (0.8136, 0.8939)
##     No Information Rate : 0.6635
##     P-Value [Acc > NIR] : 4.922e-15
##
##                   Kappa : 0.6793
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8947
##             Specificity : 0.7830
##          Pos Pred Value : 0.8905
##          Neg Pred Value : 0.7905
##              Prevalence : 0.6635
##          Detection Rate : 0.5937
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8389
##
##        'Positive' Class : NRB
##
```

Of the 209 cases of NRB, they are correctly predicted as NRB 187, and incorrectly, 23. Of the 106 cases of RB, they are correctly predicted as RB 83 and incorrectly 22.

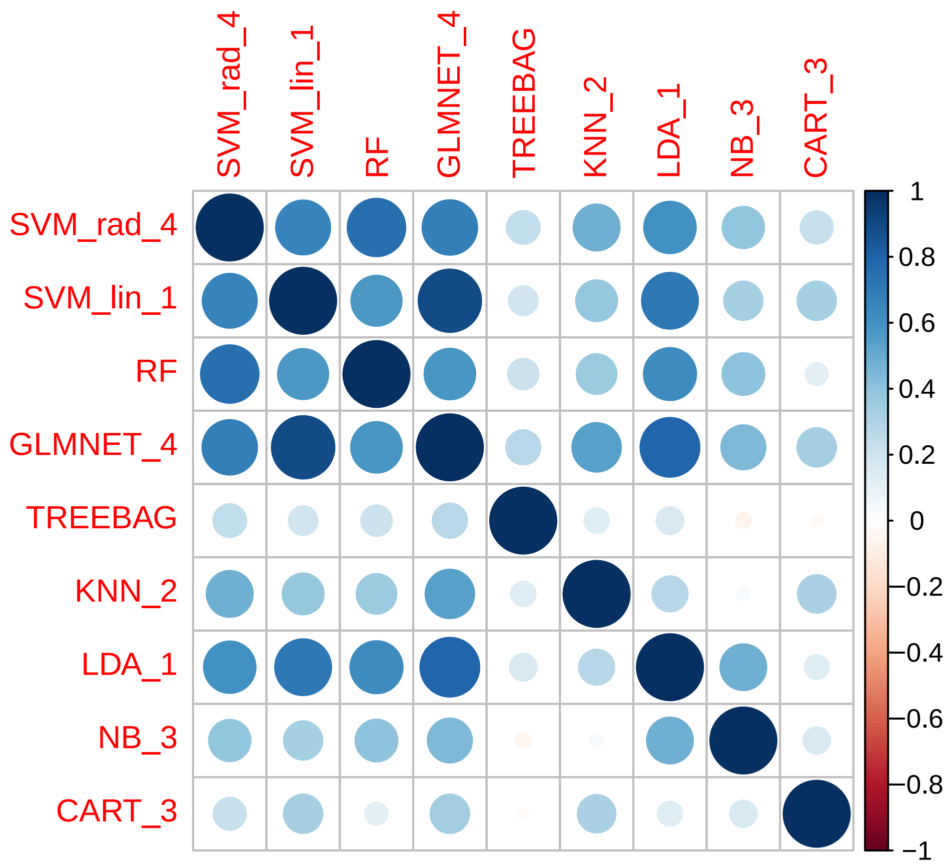
The Accuracy in the validation dataset is 85.71%, quite similar to the one obteined in the train dataset. Sensitivity (in this case the probability of predicting NRB when NRB), is quite high (89.47%). Specificity (in this case, the probability of not predicting RB when it is not RB) is slightly lower, but it remains an more than acceptable value (78.30%). Balanced Accuracy is 83.89%.

We can see also the importance of each feature in this model:




Could the result of the tuned SVM Radial model be improved? We can use another strategy: Stacking Algorithms. It consists in combining the predictions of several sub-models. It is better that the results of these sub-models have low correlation:

```
##           SVM_rad_4 SVM_lin_1        RF  GLMNET_4     TREEBAG      KNN_2
## SVM_rad_4 1.0000000 0.6683558 0.7592942 0.6812448  0.24930518 0.48594298
## SVM_lin_1 0.6683558 1.0000000 0.5774961 0.8940023  0.19246725 0.38490853
## RF        0.7592942 0.5774961 1.0000000 0.5882477  0.21499554 0.36331795
## GLMNET_4  0.6812448 0.8940023 0.5882477 1.0000000  0.27081468 0.54057860
## TREEBAG   0.2493052 0.1924672 0.2149955 0.2708147  1.00000000 0.13832053
## KNN_2     0.4859430 0.3849085 0.3633179 0.5405786  0.13832053 1.00000000
## LDA_1     0.6089222 0.7169082 0.6234378 0.7939254  0.16542823 0.28475907
## NB_3      0.3939494 0.3378672 0.4027902 0.4464534 -0.05387262 0.03890538
## CART_3    0.2365975 0.3397522 0.1129073 0.3407483 -0.02786910 0.32511752
##               LDA_1        NB_3     CART_3
## SVM_rad_4 0.6089222  0.39394938  0.2365975
## SVM_lin_1 0.7169082  0.33786720  0.3397522
## RF        0.6234378  0.40279018  0.1129073
## GLMNET_4  0.7939254  0.44645337  0.3407483
## TREEBAG   0.1654282 -0.05387262 -0.0278691
## KNN_2     0.2847591  0.03890538  0.3251175
## LDA_1     1.0000000  0.48319767  0.1330099
## NB_3      0.4831977  1.00000000  0.1608749
## CART_3    0.1330099  0.16087494  1.0000000
```




We could take as submodels SVM with a radial function (the best one so far) and some others with low correlation with it. For example, Treebag and Cart.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 187  22
##        RB   22  84
##
##                Accuracy : 0.8603
##                  95% CI : (0.8171, 0.8966)
##     No Information Rate : 0.6635
##     P-Value [Acc > NIR] : 1.592e-15
##
##                   Kappa : 0.6872
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8947
##             Specificity : 0.7925
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.7925
##              Prevalence : 0.6635
##          Detection Rate : 0.5937
##    Detection Prevalence : 0.6635
##       Balanced Accuracy : 0.8436
##
```

```
##        'Positive' Class : NRB
##
```

The Accuracy is lightly better (86.03%), as well as Balanced Accuray (84.36%)

We could try 3 other models, with no so good accuracy in the trainset as SVM RADIAL, but with a very low correlationship between them: RF, Treebag and GLMNET:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NRB  RB
##        NRB 189  19
##        RB   20  87
##
##                Accuracy : 0.8762
##                  95% CI : (0.8347, 0.9105)
##     No Information Rate : 0.6635
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7234
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9043
##             Specificity : 0.8208
##          Pos Pred Value : 0.9087
##          Neg Pred Value : 0.8131
##              Prevalence : 0.6635
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6603
##       Balanced Accuracy : 0.8625
##
##        'Positive' Class : NRB
##
```

The Accuracy in this case is 87.62%, that is, an improvement of 1%. Balanced Accuracy has improved, too,and now is 86.25%.

When we combine the predictions of these models with low correlation using staking, we obtain a better Accuracy than using our previous best model: combining models that are skillful in different ways we could improve our prediction.

Sensitivity is slightly lower than in the previous case, but Specifity is higher.

# 6 CONCLUSION

We have built and tested several models. From the initial models, applying various techniques, the values of the evaluation variable have been improved, although in no case has the improvement been extremely substantial. The best result has been achieved by combining three submodels: Random Forest (RF), Bagged CART (Treebag) and Penalized Linear Regression (GLMNET). Although individually each of these models offered results not as good as others (for example, SVM with Radial Basis Functions), the majority vote ensemble produces a model with a considerable improvement over the sub-models.

As we have seen, the dataset is very slightly unbalanced (there are more NRB than RB records), so a priori there could be other metrics that offer better results int the models evaluation than Accuracy. But after reviewing the confusion matrix of our last model, we could see that the values of Accuracy (87.62%) and

Balance Accuracy (86.25%) are quite good. Even so, other possibilities could be explored using other metrics to evaluate the models, such as ROAC.

# 7 REFERENCES


- Irizzary, Rafael.(2019). Introduction to Data Science. https://rafalab.github.io/dsbook/
- Brownlee, Jason. (2017).Machine Learning Mastery With R. https://machinelearningmastery.com/machine-learning-with-r/