INSA
ROUEN NORMANDIE

Machine Learning

Random Forests

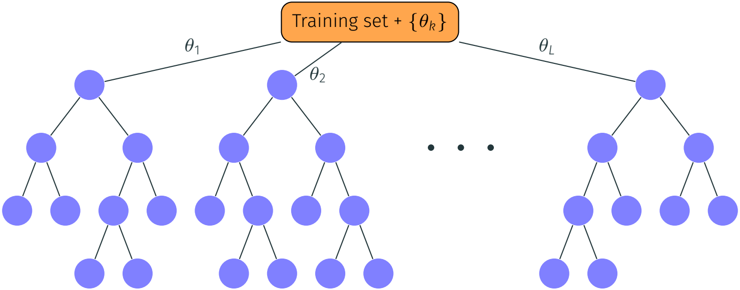
Simon BERNARD
simon.bernard@univ-rouen.fr

Chapters :

Random Forest

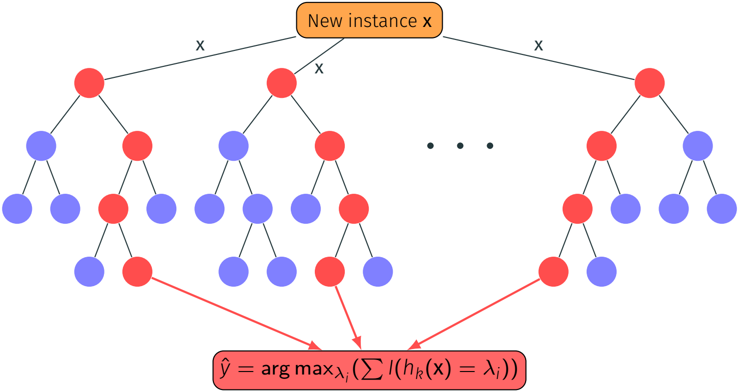
A Random Forest (RF) is an ensemble of DT, each of which is ”randomized”

- Collection of L decision trees $\{h_k = h(\mathbf{x}, \theta_k), k = 1, \ldots, L\}$
- $\{\theta_k\}$ are i.i.d. random vectors



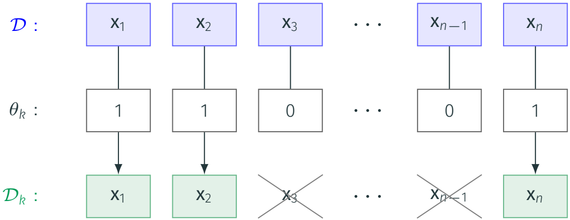
A Random Forest (RF) is an ensemble of DT, each of which is "randomized"

- Each tree casts a unit vote for the most popular class at input **x**



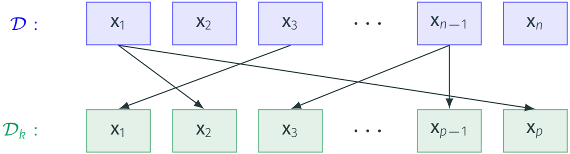
Random Forest

- θ_k : vector of n random values in $\{0, 1\}$ (coin flip)
- For each θ_k, build a replicate $\mathcal{D}_k$ of the training set $\mathcal{D}$ using θ_k as a mask :

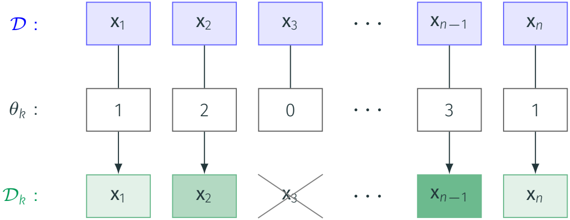


- Train the k^{th} decision tree on $\mathcal{D}_k$
- The L trees are different due to instability (cf. end of previous chapter)

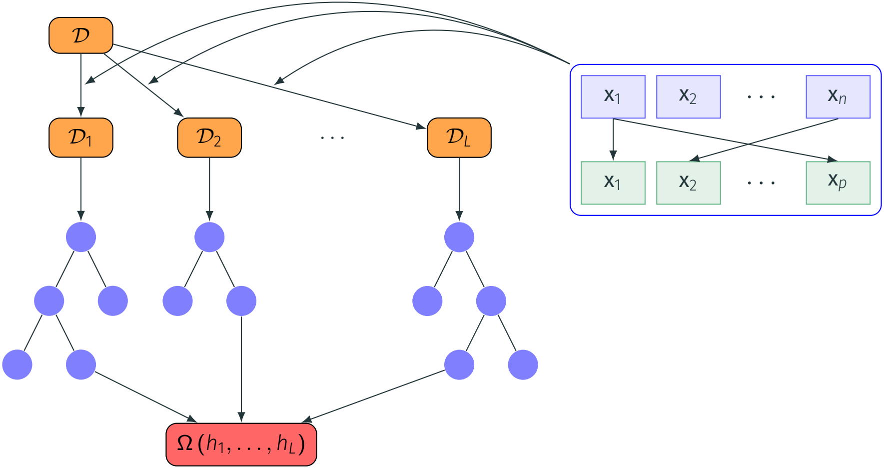
- Problem : we do not control the size of $\mathcal{D}_k$
- Solution : randomly draw p instances from $\mathcal{D}$, with $p < n$
- Problem : diversity is inversely proportional to p
- Solution : randomly draw with replacement p instances from $\mathcal{D}$

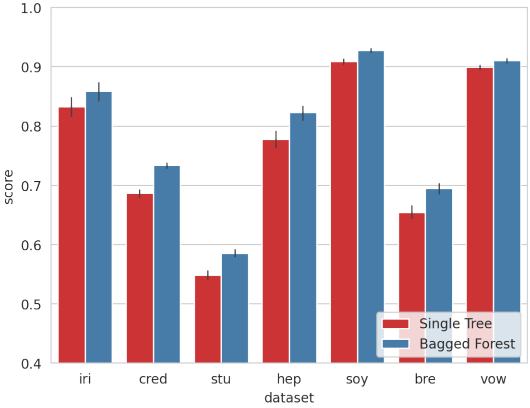


- Problem : we do not control the size of $\mathcal{D}_k$
- Solution : randomly draw p instances from $\mathcal{D}$, with $p < n$
- Problem : diversity is inversely proportional to p
- Solution : randomly draw with replacement p instances from $\mathcal{D}$



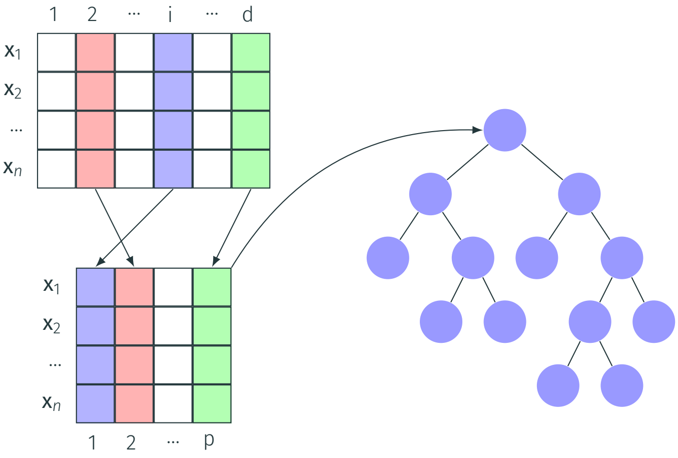
This is called Bagging!

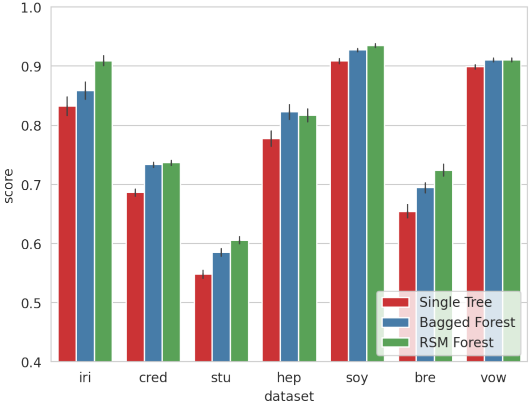

$\mathcal{D}$
$\mathcal{D}_1$
$\mathcal{D}_2$
$\cdots$
$\mathcal{D}_L$
x_1
x_2
$\cdots$
x_n
x_1
x_2
$\cdots$
x_p
$\Omega\,(h_1, \ldots, h_L)$



Note : p set equal to n (good results in general)

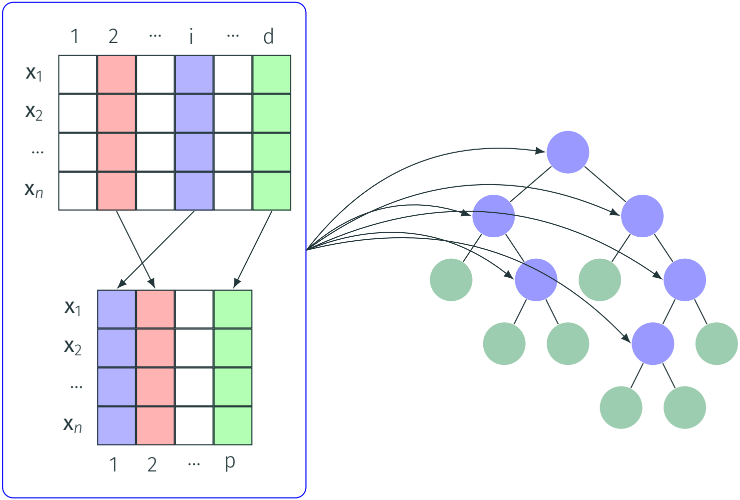
The Random Subspaces Method (RSM)

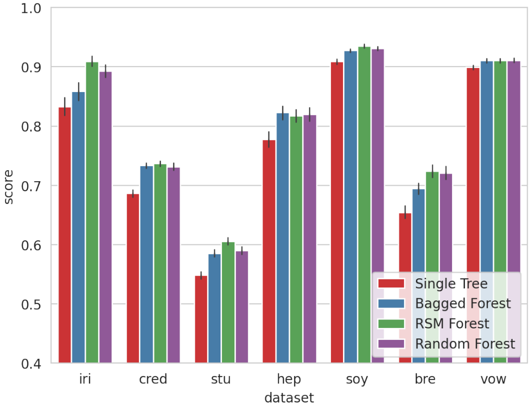




Note : the number p of features to be drawn is more complex to set (here $p = d/2$)

Random Feature Selection



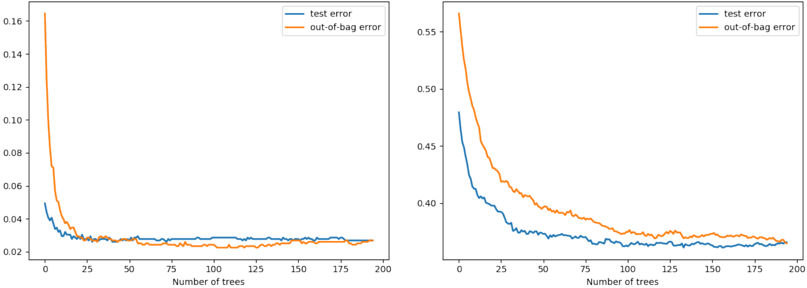


Note : In softwares, Random Forest stands for Bagging + Random Feature Selection

Random forest in practice

2 main hyperparameters to set for optimum performance

L : the number of random trees



2 main hyperparameters to set for optimum performance

L : the number of random trees

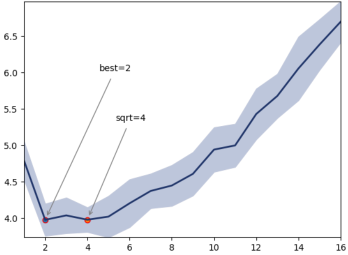
- Analysis : convergence is reached for different amounts of trees from a dataset to another
- Problem : How many trees is enough for a given dataset ?
- Solutions :
  - Empirical : several hundreds
    $\longrightarrow$ no guarantee but computational times being low, the most popular solution
  - Validation (or *out-of-bag*) error to detect convergence
    $\longrightarrow$ Parallelizing is not possible anymore, and sometimes over-optimistic

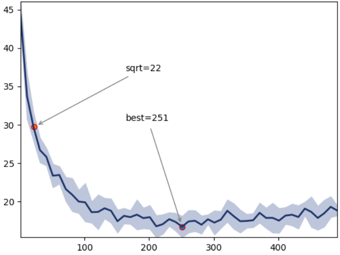
2 main hyperparameters to set for optimum performance

p : the number of random features at each node

$p =$	1	2	...	$d-1$	d
Randomness	max		$\leftarrow \oplus \ldots \ominus \rightarrow$		$\emptyset$

Popular values : 1, $\sqrt{d}$, $\lceil log_2(d) \rceil$



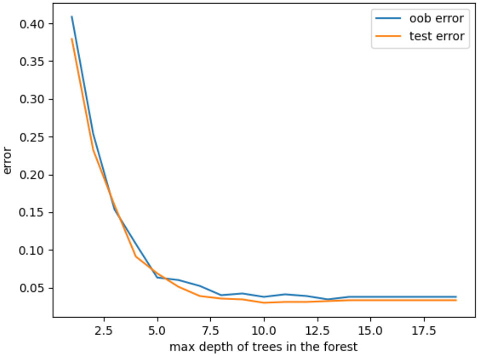


2 main hyperparameters to set for optimum performance

p : the number of random features at each node

- Analysis : best value depends on irrelevant features
  - few irrelevant features $\Rightarrow p$ low ($\approx \sqrt{d}$)
  - many irrelevant features $\Rightarrow p$ high, but difficult to set *a priori*
- Problem : How to set this value without *a priori* knowledge on the features
- Solutions : Know your dataset, feature analysis, feature selection, cross-validation

In most cases, it is best to use unpruned trees

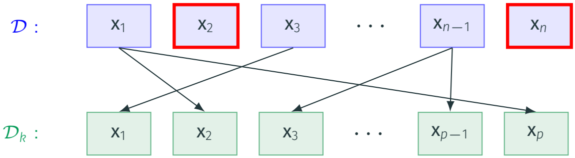


Note : digits dataset, $p = \sqrt{d}$, $L = 100$

The Swiss knife of machine learning

One of the main reasons to use Bagging is the out-of-bag mechanism

- When $p = n$, 36.8% of $\mathcal{D}$ are NOT present in $\mathcal{D}_k$ in average (provable)[1]
- These instances are called *out-of-bag* (oob)
- The oob instances are different from one $\mathcal{D}_k$ to another



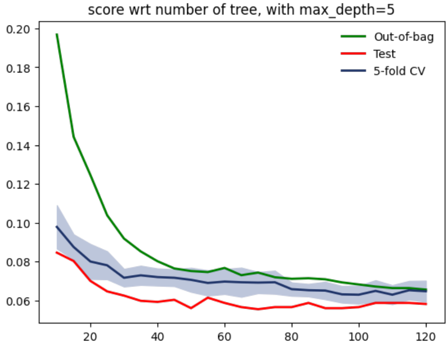
1. Note : when $p > n$, there is less oob instances and the diversity is limited

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

The *out-of-bag* instances can be used to estimate generalization capabilities

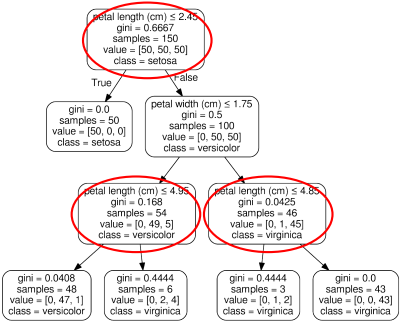
- Validation dataset required for :
  - tuning hyperparameters
  - estimating generalization performance
  - estimating diversity and individual accuracies
  - learning/optimizing combination operators
  - etc.
- Alternative : using oob instances instead of an independant dataset
- Oob estimates are reliable estimates for generalization capabilities, although they tend to underestimate some of them.

Example : generalization error



Forests embed 2 feature importance measures

Mean Decrease Impurity (MDI) :



For a given feature $x^{(i)}$:

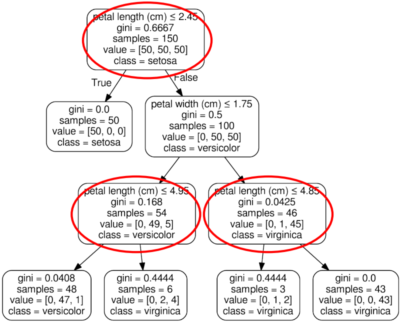
- Consider each node N_k for which $x^{(i)}$ is used in S_{N_k}
- Cumulate their impurity gain values

$$I_{MDI}(x^{(i)}) = \frac{1}{|\mathcal{N}_i|} \sum_{N_k \in \mathcal{N}_i} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \Delta(\mathcal{D}_k, S_{N_k})$$

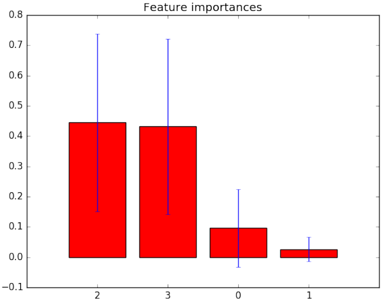
where $\mathcal{N}_i$ is the set of nodes, all trees considered, that uses $x^{(i)}$ for splitting

Forests embed 2 feature importance measures

Mean Decrease Impurity (MDI) :



For example, for *Iris* :



INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Forests embed 2 feature importance measures

Mean Decrease Accuracy (MDA) : based on out-of-bag votes

1. For each h_k, record the correct *out-of-bag* votes, noted V_k
2. For each feature $x^{(i)}$:
   (a) Randomly permute all the values of $x^{(i)}$ in $\mathcal{D}$
   (b) For all trees h_k
      (i) Counts the new *out-of-bag* correct votes from h_k, noted $V_k^{(i)}$
      (ii) Compute $S_k^{(i)} = V_k - V_k^{(i)}$
   (c) The importance measure for $x^{(i)}$ is

$$I_{MDA}(x^{(i)}) = \frac{1}{L} \sum_{k=1}^{L} S_k^{(i)}$$

Random Forests embed a similarity measure on pairs of instances

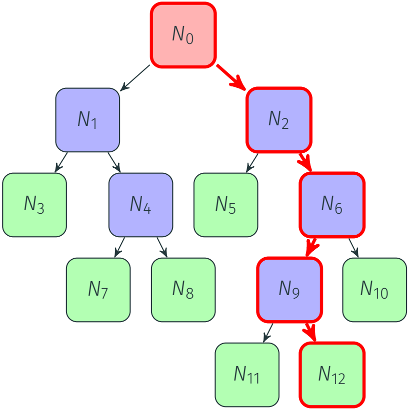
- Similarity = measure the resemblance between 2 instances $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Takes the class membership into account, contrary to distance measure

  Example : Euclidean distance (no y_i, y_j in the equation)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

- $\mathbf{x}_i$, $\mathbf{x}_j$ similar if "close" to each other but also if they belong to the same class
- **Key idea** : $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar if they follow the same path down the trees

Random Forests embed a similarity measure on pairs of instances

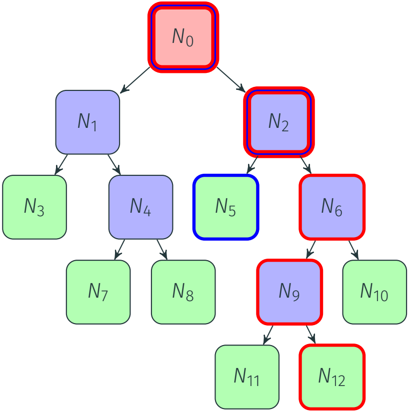


- Let $\mathcal{L}_k$ bet the set of leaves in the k^{th} tree
- Let

$$l_k : \mathcal{X} \to \mathcal{L}_k$$

  be a function that maps all $\mathbf{x}$ to the leaf from $\mathcal{L}_k$ in which it lands
- Here, $l_k(\mathbf{x}_i) = N_{12}$

Random Forests embed a similarity measure on pairs of instances



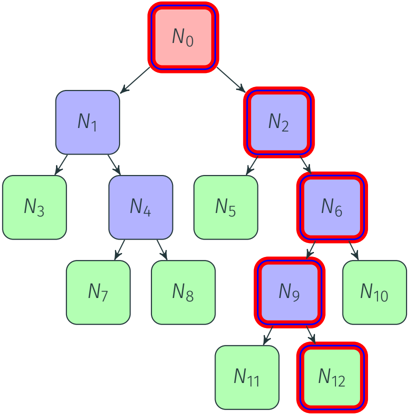
- The similarity $d_k(\mathbf{x}_i, \mathbf{x}_j)$ between $\mathbf{x}_i$ and $\mathbf{x}_j$, given by the k^{th} tree, is

$$d_k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \textit{if } \ell_k(\mathbf{x}_i) = \ell_k(\mathbf{x}_j) \\ 0 & \textit{otherwise} \end{cases}$$

- Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ don't land in the same leaf :

$$d_k(\mathbf{x}_i, \mathbf{x}_j) = 0$$

Random Forests embed a similarity measure on pairs of instances



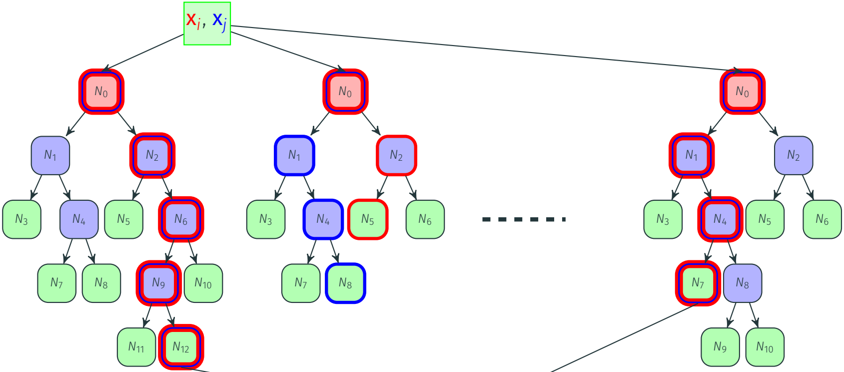
- The similarity $d_k(\mathbf{x}_i, \mathbf{x}_j)$ between $\mathbf{x}_i$ and $\mathbf{x}_j$, given by the k^{th} tree, is

$$d_k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \textit{if } \ell_k(\mathbf{x}_i) = \ell_k(\mathbf{x}_j) \\ 0 & \textit{otherwise} \end{cases}$$

- Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ land in the same leaf :

$$d_k(\mathbf{x}_i, \mathbf{x}_j) = 1$$

Random Forests embed a similarity measure on pairs of instances



$$d(x_i, x_j) = \frac{1}{L}\sum_{k=1}^{L} d_k(x_i, x_j)$$

Some other tools (not detailed here)

- Unsupervised learning
    - Generation of artificial of negative samples to simulate a second class
    - Use the tree structure to perform **clustering** tasks
- Outliers detection
- Novelty detection
- Missing values and labels
- Prototypes selection

Random Forest methods...

- are easy to understand and easy to use
- are among the most accurate methods for ”tabular” data
- are robust to many machine learning settings (e.g. high dimension, imbalanced classes, etc.)
- are very versatile with many embedded tools for interpretability
- have been successfully used for many applications, to name a few :
  - Giga-pixel image segmentation (biomedical imaging)
  - Real-time tracking in videos
  - Real-time body part recognition (Kinect)
  - Intelligent/autonomous vehicle
  - Medical diagnosis/prognosis

Fernandez-Delgado et al., "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems ?", Journal of Machine Learning Research, 2014

- Huge comparison of many classifiers : 179 different classifiers and 121 public datasets

Rank	Acc.	κ	Classifier
32.9	82.0	63.5	parRF_t (RF)
33.1	**82.3**	**63.6**	rf_t (RF)
36.8	81.8	62.2	svm_C (SVM)
38.0	81.2	60.1	svmPoly_t (SVM)
39.4	81.9	62.5	rforest_R (RF)
39.6	82.0	62.0	elm_kernel_m (NNET)
40.3	81.4	61.1	svmRadialCost_t (SVM)
42.5	81.0	60.0	svmRadial_t (SVM)
42.9	80.6	61.0	C5.0_t (BST)
44.1	79.4	60.5	avNNet_t (NNET)
45.5	79.5	61.0	nnet_t (NNET)
47.0	78.7	59.4	pcaNNet_t (NNET)
47.1	80.8	53.0	BG_LibSVM_w (BAG)

"The classifiers most likely to be the bests are the random forest (RF) versions [...]. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel [...]"