Automatic prediction of village-wise soil fertility for several nutrients in India using a wide range of regression methods

M.S. Sirsat, E. Cernadas, M. Fernández-Delgado*, S. Barro

Centro Singular de Investigación en Tecnoloxías da Información da USC (CiTIUS)
Universidade de Santiago de Compostela, Rúa de Jenaro de la Fuente Domínguez
Campus Vida 15782, Santiago de Compostela, Spain

Abstract

In low quality soils, as in the Indian state of Maharashtra, a sustainable land management practice is very important to enhance the soil quality and to maintain proper values for several nutrients that are relevant for an optimal crop yield. The evaluation of a soil fertility index for these nutrients and for each geographical place allows to create maps of village-wise fertility indices which are very useful for fertility management. An automatic prediction of such fertility indices would be very important to reduce the amount of chemical measurements of nutrients to develop in different cultivation lands. The current study develops such prediction for five important soil nutrients (organic carbon, phosphorus pentoxide, iron, manganese and zinc) using 76 regression methods which belong to a collection of 20 regressor families, including neural networks, deep learning, support vector regression, random forests, bagging and boosting, lasso and ridge regression, Bayesian models and more. The best results are achieved by the extremely randomized regression trees (extraTrees), which achieves an acceptable prediction accuracy (average squared correlations between 0.57 and 0.70), being also relatively fast. Other regressors with high performance are random forests and regularized random forest, generalized boosting regression model (gbm) and epsilon-support vector regression.

*Corresponding author
Email address: manuel.fernandez.delgado@usc.es (M. Fernández-Delgado)

Keywords: Indian agriculture, soil fertility index, machine learning, regression, extremely randomized regression trees

1. Introduction

Agriculture is one of the most important economic fields in India, but urbanization and industrialization reduces the cultivation land. There is a need of increase agricultural production without harm the environment and sustainability, which requires to plan the soil fertility by supplying essential nutrients to the crop in sufficient amount and at right time for its best growth. The soil quality is a highly significant factor for achieving high crop production, and imbalances in soil quality reduce the crop health and lead to lower crop yield (Research Council, 1989). Declining status of soil fertility and mismanagement of soil nutrients may be factors for food crises for the world's population (Gruhn et al., 2000). Generally, Indian soil fertility data are summarized for block and district level. These data are useful for decision making about application of suitable amounts of fertilizers, policies of fertilizer distribution and consumption in the view of changes in fertility levels. The creation of maps for village-wise fertility indices and for several relevant nutrients would be very useful to compare levels of soil fertility among villages, and to make fertilizer recommendations specific for each village. For the development of such maps, much effort in terms of chemical analysis and time of specialized staff might be avoided if the direct measurement of the soil fertility through nutrient levels, for each village, might be replaced by an automatic, accurate enough, prediction. Most of the literature about prediction of soil parameters is based on the concept of pedotransfer function (PTF), which allows to describe mathematical relations among soil properties, using measurements to predict or estimate certain soil parameters which are missing or whose measurement is time-consuming or expensive (Bouma, 1989; Pachepsky et al., 2015). The PTF can be formulated using data mining, exploration and machine learning regression methods. After reviewing the literature about PTFs, our objective is to use regression techniques as PTFs

that predict the village-wise soil fertility indices for several relevant nutrients as organic carbon (OC), phosphorus pentoxide (P_2O_5), iron (Fe), manganese (Mn), and zinc (Zn), using data from the Marathwada region in the Maharashtra state of India. The paper is organized as follows: section 2 analyzes previous works which use machine learning methods to predict soil parameters; section 3 describes the calculation of village-wise soil fertility indices which are predicted in the current study; section 4 describes the datasets and regression methods used for the prediction of fertility indices, and section 5 discusses the results of the experimental work. Finally, section 7 compiles the conclusions of this work.

2. Related work

Several studies applied machine learning techniques to solve soil problems in agriculture. Mucherino et al. (2009) provides a review of the methods used, among other objectives, to predict the soil fertility, defined as its ability to supply the required nutrient levels and water for high quality crop yield. Soil fertility was predicted using neural networks with Levenberg-Marquadt based back-propagation (Sheela and Sivaranjani, 2015) and partial least squares regression (Obade and Lal, 2016), whose inputs included soil bulk density, electrical conductivity (EC), available water capacity, soil OC, pewamo silty clay loam, glynwood silt loam, kibbie fine sandy loam, crosby silt loam and crosby celina silt loams soil. Terhoeven-Urselmans et al. (2010) predicted acidity (pH), alongside with OC and cation exchange capacity from mid-infrared spectra for several soils using partial least-squares regression and the prediction root mean square error as quality measure. Jia et al. (2010) applied a Bayesian network for soil fertility grading using the soil pH and nutrients as copper (Cu), Fe, potassium (K), Mn, nitrogen (N), phosphorus (P), OC and Zn. Lamorski et al. (2008) found that SVM outperforms neural networks to provide a PTF which predicts the soil total nitrogen using bulk density and soil contents of water, silt, sand and clay. Jain et al. (2004) focused on PTFs for the prediction of water retention and saturated/unsaturated hydraulic conductivity, properties which

are expensive to measure. Minasny et al. (1999) used multiple-linear regression, extended nonlinear regression and neural networks to estimate water-retention PTFs for soils in Australia. In a previous study (Sirsat et al., 2017), we used a collection composed by 20 classifiers, including random forests, neural networks, adaboost, SVM and bagging, among others, to classify several nutrient levels and village-wise soil fertility indices. The class labels were quantified values (low, medium and high) of their numeric values. We also classified the soil type and pH, and the recommended crop for the next cycle. In the current paper, we use an even larger and more diverse collection of regression methods in order to create PTFs which directly predict, without discretization, the numeric values of fertility indices for several important soil nutrients which will be described below.

	Major nutrients		Micro nutrients (parts per million)		
	OC (%)	P_2O_5 (Kg/ha)	Fe	Mn	Zn
Low <	0.5	10	1	2.5	108
Medium	0.5-0.75	10-24.6	1-2	2.5-4.5	108-280
High >	0.75	24.6	2	4.5	280

Table 1: Intervals defined by the Department of Agriculture & Cooperation (2011) of the Indian Government for the major and micro nutrients respectively (Muhr et al., 1965; Katyal and Rattan, 2003).

3. Prediction of village-wise soil fertility indices

The soil of Marathwada is intensively cultivated with novel agricultural practices in order to achieve a high crop production. A major factor for soil productivity is fertility, which primarily deals with ability of soil to supply nutrients to plants. Fertility of agricultural soil is depleting due to intensive cultivation practices and inadequate or excesive use of chemical fertilizers. To attenuate these soil problems, there is a need of knowledge about soil physical and chemical status. The village-wise fertility indices for several major (OC and P_2O_5) and micro (Fe, Mn, Zn) soil nutrients are not only helpful to choose the right

fertilizer and dose, but also to know about inherent excess and deficiency in them, i.e., in order to balance nutrients up to critical levels. The OC is very relevant for the biological activity of the soil and for crop productivity (Reeves, 1997), while P_2O_5 is necessary for cell signaling, phosphorylation and bioenergetics in plants. On the other hand, Fe and Mn are used by chlorophyll during photosynthesis to absorb energy from light. Finally, Zn contributes to the production of plant growth hormones and proteins, being responsible for plant root development as well as carbohydrate and chlorophyll formation (Arunachalam et al., 2013). The Zn affects the crop yield and soil quality[1], being the most deficient micro-nutrient in Indian soils by nearly 50% of the required amount. The agriculture planning of the Indian Government requires to determine the village-wise soil fertility indices (N_I) for the previous nutrients and to quantify their levels as low, medium and high. Inspired by the previous ideas, the present work applies a collection of regression techniques to automatically predict village-wise soil fertility indices for the previous nutrients using several chemical parameters of the soil. Rammoorthy and Bajaj (1969) defined the procedure to calculate the village-wise soil fertility index for a nutrient. First, each cultivation lands is evaluated, according to its fertility for the corresponding nutrient, as low, medium or high using the limits listed in Table 1. Second, the numbers N_L, N_M and N_H of cultivation lands with low, medium and high fertility levels, respectively, for the nutrient, and the total number of lands $N_T = N_L + N_M + N_H$, are determined for each village. Finally, the fertility index N_I is calculated using the following formula:

$$N_I = \frac{N_L + 2N_M + 3N_H}{N_T} \tag{1}$$

The value of N_I is a weighted average of the numbers of cultivation lands with low, medium and high fertility indices, so its value is restricted to the range $1 \leq N_I \leq 3$. Values of N_I near to 1 mean that low fertility fields are predominant

[1] `http://www.zinc.org.in/zinc-nutrient-initiative-in-india`

for that nutrient and village; N_I values about 2 are associated to medium fertility index; and N_I values about 3 correspond to high fertility indices. The index value is the same for all the patterns in the village, whose cultivation lands share the same fertility index for every soil nutrient.

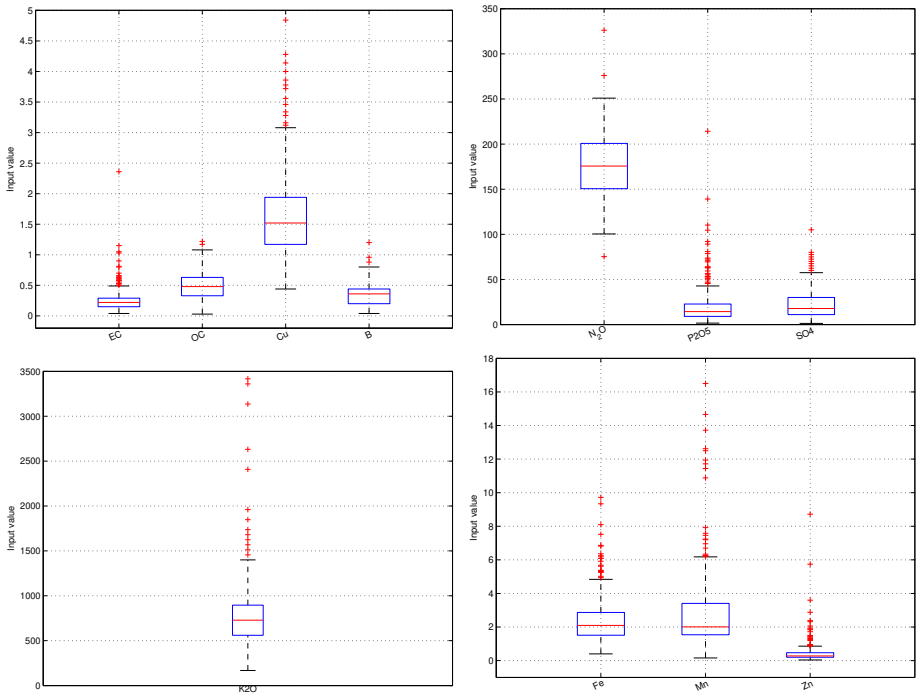

Figure 1: Boxplot of the 11 inputs used for the prediction of the 5 village-wise fertility indices.

4. Material and methods

The current section describes the materials of the current work, which includes the datasets used to predict the soil fertility indices (subsection 4.1), and the collection of regression methods used to develop this prediction (subsection 4.2).

4.1. Soil data

The objective of the current work is to predict the village-wise fertility indices of five important soil nutrients: organic carbon (OC), phosphorus pentoxide (P_2O_5), iron (Fe), manganese (Mn), and zinc (Zn). Our study does not consider village-wise fertility indices for other soil nutrients as nitrous oxide (N_2O), potassium oxide (K_2O) nor Cu because the available datasets for these nutrients only include patterns of one fertility level (low, medium or high). Each nutrient corresponds to a different prediction (or regression) problem, where the nutrient is the output that must be predicted by the regressor. This prediction is developed using input data with 11 physical and chemical parameters of the soil: EC, OC, N_2O, P_2O_5, K_2O, sulfate (SO_4), Cu, Fe, Mn, Zn and boron (B). The values of the inputs are expressed in Kg/ha for EC, N_2O, P_2O_5 and K_2O; in parts per million for SO_4, Cu, Fe, Mn, Zn and B; and in % for OC. Figure 1 shows the boxplots of these inputs. The upper and lower ends of each blue box are the 25% and 75% quartiles, respectively; the red line inside the box is the input median; finally, the upper and lower black wiskers (resp. the red crosses) are the extreme values not considered (resp. considered) outliers. The input ranges are very different: EC, OC, B and Zn are almost bounded between 0 and 1, while Cu is between 1 and 2; Fe and Mn are between 1 and 4; P_2O_5 and SO_4 are between 10 and 30; N_2O is between 150 and 200; and K_2O is aproximately between 500 and 900, with outliers until 3500. The input values are equal for each regresion problem, so the the only difference among the five regression problems is the output to be predicted. The dataset collection was acquired from the Marathwada region by the State Government of Maharashtra (India) during years 2011 to 2015, and it includes 372 patterns, each one corresponding to a cultivation field located in a certain village. Figure 2 shows the boxplots of the five outputs: since they are calculated using eq. 1, their values are bounded between 1 and 3, which correspond to low and high fertility indices, respectively. However, there are big differences among boxplots: OC-F, Fe-F and Zn-F are between 1 and 2 (fertility levels low and medium), while P_2O_5-F is between 1 and 3 (levels low, medium and high), and Mn-F is between 2 and

3 (levels medium and high).

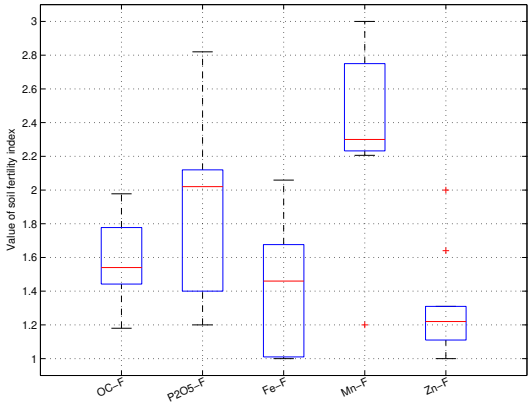

Figure 2: Boxplot of the village-wise fertility indices for the five nutrients to be predicted.

4.2. Regression methods

We use a wide collection of 76 regressors belonging to 20 families, which are briefly described in the list below, grouped by families of related methods. The majority of them (72 regressors) are implemented in the R language for statistical computing (R Core Team, 2008), and they were selected from the list of models[2] provided by the Classification and Regression Training (`caret`) package, implemented by Kuhn (2016). Caret provides an interface for the execution of many classification and regression methods, implemented by different R packages. In the current work, instead of using the interface provided by `caret` through its `train` function, we run the regressors directly using the corresponding R packages (see the detailed list below), in order to control the execution of each single model, and to allow the execution of other 4 popular regression methods which are implemented in other platforms: 1) deep learning neural network (which we call dlkeras in the regressor list below), using the library Keras with Theano interface in the Python programming language; 2) support vector regression, using the LibSVM library (Chang and Lin, 2011)

[2] `http://topepo.github.io/caret/train-models-by-tag.html`

accesed the via C++ library interface (named svr); 3) generalized regression neural network (named grnn) included in the Matlab neural network toolbox (Matlab, 2011); and 4) extreme learning machine with Gaussian kernels (named elm-kernel), also programmed in Matlab.

Most regressors in our collection have tunable hyperparameters, i.e., parameters which must be specified previously to training, whose values often influence the regressor performance. In these cases, it is necessary to try several values for each hyperparameter in a trial-and-error procedure, and to select the value which provides the best performance. In order to optimize each regressor, its tunable hyperparameters and the list of values for each hyperparameter should be known. For the regressors which are not implemented in R, we directly specify the list of tunable hyperparameters and the values used for tuning, which can be extracted from the regressor documentation. For the regressors implemented in R, the `caret` regressor list (see the link above) already specifies, for each regressor, the list of its hyperparameters, Besides, `caret` also provides the `getModelInfo` function, which returns for each regressor a list of reasonable values which should be used for tuning each hyperparameter. This utility of the `caret` package is very useful, because it avoids the need to analyze the documentation of every regressor in order to know proper values which can be tried in the hyperparameter tuning. The list of values provided by the `getModelInfo` function is different for each dataset used for training the regressor. However, in our case the prediction of the five village-wise soil fertility indices shares the same 11 inputs (subsection 3), so the list of hyperparameter values returned by `getModelInfo` is the same for the five regression problems. The hyperparameter values are specified in the following regressor list. The notation `a:b:c` means a list of values from `a` to `c` with step `b` (where the step is missing, its value is indented to be 1).

I. Linear regression

1. **lm** is the linear model provided by the stats package (Bates and Chambers, 1992), which develops multivariate linear regression.

2. **rlm** implements robust linear model (MASS package), fitted using iteratively re-weighted least squares with maximum likelihood type estimation (Huber, 1981). The only hyperparameter is the Ψ function, whose values can be `huber`, which provides a convex optimization problem, `hampel` and Tukey `bisquare`, both with local minima.

II. Generalized linear regression (GLM)

3. **glm** is the generalized linear model implemented by the `stats` package (Dobson, 1990).
4. **penalized** is the penalized linear regression (`penalized` package). The regression is regularized by weighting two penalties: L1 penalty, also called least absolute shrinkage and selection operator (LASSO), is the sum of absolute values of coefficients; and L2, also called ridge penalty, is the sum of squared coefficients. The weights of both penalties are tunable hyperparameters (λ_1 and λ_2 arguments in the `penalized` R function) with values 1, 2, 4, 8 and 16 each one (Goeman, 2010).
5. **glmnet** is the LASSO and elastic-net regularized GLM (Simon et al., 2011) provided by the `glm` package. The mixing percentage (α argument in the `glmnet` function) is tuned with values 0.1:0.1:1, including α=1, which corresponds to LASSO penalty, and α <1 for elastic-net penalty (α=0 corresponds to ridge regression penalty).
6. **glmStepAIC** is the generalized linear model with stepwise feature selection (Ripley, 2002) using the Akaike information criterion (`stepAIC` function in the `MASS` package).

III. Least squares

7. **nnls** is the non-negative least squares regression (`nnls` package), which finds $\arg\min_{\mathbf{x}} |\mathbf{Ax} - \mathbf{b}|$ subject to $\mathbf{x} \geq 0$ using the method proposed by Lawson and Hanson (1995).
8. **krlsRadial** is the radial basis function kernel regularized least squares regression (`KRLS` package), which uses Gaussian radial basis functions (Hain-

mueller and Hazlett, 2013). The regularization parameter (λ), which specifies the tradeoff between model fit and complexity, is 0.1 and the only tunable hyperparameter is the kernel spread (σ), with values $\{10^i\}_{-7}^{2}$.

IV. Partial least squares (PLS)

9. **spls** is the sparse partial least squares (Chun and Keles, 2010) regression, implemented by the `spls` package. The hyperparameters are the number of latent components (`K`), with values 1:11, and the threshold (η), with values 0:0.1:1.
10. **simpls** fits a PLS regression model with the simpls method (`plsr` function in the `pls` package, with `method = simpls`). This regressor (Jong, 1993) directly calculates the PLS factors as linear combinations of the inputs maximizing a covariance criterion with orthogonality and normalization constraints. The only hyperparameter is the number of components used by the simpls model, with values 1:10.
11. **kernelpls** is the PLS regression with `method = kernelpls` (Jong, 1994) in the same function and package as spls and simpls, tuning the number of components with values 1:6.
12. **enpls.fs** is an ensemble of sparse partial least squares regressors provided by the `enpls` package (Xiao et al., 2016), with maximum number of components (`maxcomp`) equal to 3.
13. **plsRglm** is the PLS generalized linear model (`plsRglm` package) with `modele = pls-glm-family`. The hyperparameters are the number of extracted components (`nt`) and the input significance level (`alpha.pvals.expli`), with values 1:5 and $\{10^i\}_{-2}^{2}$, respectively (Bertrand et al., 2014).

V. Least absolute shrinkage and selection operator (LASSO)

14. **lasso** is the LASSO regression, using the `enet` function in the `elasticnet` package with $\lambda = 0$.
15. **relaxo** is relaxed LASSO regression (`relaxo` package), which generalizes the LASSO shrinkage method for linear regression (Meinshausen, 2007).

The relaxation and penalty hyperparameters ϕ and λ are tuned with 10 and 5 values in the ranges 0.1–0.9, and 1.34–163, respectively.

VI. Ridge (or Tikhonov) regression

16. **ridge** (`elasticnet` package) uses the least angle regression-elastic net (LARS-EN) algorithm to compute the elastic net regression model (Zou and Hastie, 2005). The only hyperparameter is the quadratic penalty (or regularization) parameter λ, with values 0.01, 0.03, 0.05, 0.07 and 0.1.
17. **spikeslab** is the spike and slab regression (Ishwaran et al., 2010), which computes weighted generalized ridge regression estimators using Bayesian spike and lab model (`spikeslab` package). The only hyperparameter is the maximum number of inputs (`max.val`) considered in the final model, with values 2 and 11 because there are 11 inputs.
18. **foba** develops ridge regression with forward, backward and sparse input selection (Zhang, 2011), implemented by the `foba` package. The hyperparameters are regularization (λ) for ridge regression, with 10 values in the range 10^{-5}–0.1, and the number of selected inputs (`k`) for prediction, with values 2 and 11.

VII. Neural networks

19. **mlpWD** is the multi-layer perceptron with one hidden layer and weight decay (`mlp` function in the `RSNNS` package, with `learnFunc =Backprop-WeightDecay`, called `mlpWeightDecay` in the `caret` model list). The size of the hidden layer, with values 1:2:19, and the weight decay, with 5 values in the range 0–0.1, are the tunable hyperparameters.
20. **mlpWDml** is the same network but with 3 hidden layers, tuning their sizes (with values 5:5:15 each one) and the weight decay (5 values between 0 and 0.1), called `mlpWeightDecayML` in the `caret` model list).
21. **avNNet** is the model averaged neural network (`caret` package), a committee of 5 neural networks of the same size trained using different random

seeds, whose outputs are averaged. The hyperparameters are the network size, with values 1:2:29, and the weight decay, with values $0, \{10^{-i}\}_2^4$.

22. **rbf** is the radial basis function network (`RSNNS` package) which does a linear combination of basis functions centered around a prototype (Zell, 1998). The only hyperparameter is the size of the hidden layer, with values 1:2:19.

23. **grnn** is the generalized regression neural network (Specht, 1991) implemented by the Matlab neural network toolbox. The Gaussian spread is a hyperparameter, tuned with 10 values in the range 0.001–2. Large (resp. small) spread values lead to smooth (resp. close) approximations.

24. **elm** (`elmNN` package) is the extreme learning machine (Huang et al., 2012). The tunable hyperparameters are the number of hidden neurons, with values 1:2:39, and the activation function of the neuron, which can be sinus, radial basis, linear and hyperbolic tangent.

25. **elm-kernel** is the ELM neural network but with Gaussian kernel (Huang et al., 2012) using the publicly available Matlab code[3]. The hyperparameters are the regularization parameter C and kernel spread with values $\{2^i\}_{-5}^{14}$ and $\{2^i\}_{-16}^{8}$, respectively.

26. **pcaNNet** is a multi-layer perceptron neural network with one hidden layer trained on the PCA-mapped training patterns (`caret` and `nnet` packages). The tunable hyperparameters are the size of the hidden layer and the weight decay, with values 1:2:39 and $0, \{10^{-i}\}_1^4$, respectively.

27. **bdk** (`kohonen` package) is the supervised bi-directional Kohonen network (Melssen et al., 2006). The hyperparameters are the sizes of both maps, with values 1:5 and 2:6, respectively, and the initial weight given to the input map in the distance calculation for the output map, and vice versa (values 0.5:0.1:0.9).

[3] `http://www.extreme-learning-machines.org`

VIII. Deep learning neural networks

28. **dlkeras** is the deep learning neural network using the Keras module (Chollet, 2015), written in the Python programming language. This network has three hidden layers tuned with 50 and 75 neurons for each layer (27 combinations). The deep learning methods (Hinton et al., 2006; Liu et al., 2017) are very popular, specially for image classification, and we included them in this comparison for regression tasks.
29. **dnn** (`DeepNet` package) implements a deep belief network (DBN) in R. It uses three hidden layers and initializes weights using the DBN method, tuning their numbers of neurons with 3 values each one.

IX. Support vector machines

30. **svr** is the epsilon-support vector regression with Gaussian kernel, using the C++ interface to the LibSVM library (Chang and Lin, 2011) . The regularization hyperparameter C and the kernel spread γ are tuned with the same values as elm-kernel (see above).
31. **svmRadial** is another implementation of SVR with Gaussian kernel (`ksvm` function in the `kernlab` package) for regression (`type = eps-svr`), using sequential minimal optimization (SMO) proposed by Platt (1998) to solve the quadratic SVM problem. The kernel spread and regularization parameter are tuned with 6 values in the range 0.03–0.24 and with values $\{2^i\}_{-4}^{5}$, respectively.
32. **rvmRadial** is the relevance vector machine (Tipping, 2001) with Gaussian kernel (`kernlab` package). The Gaussian spread is not tuned, but estimated by the `getModelInfo` function in the `caret` package, with value 0.1176.

X. Regression trees

33. **rpart** is the recursive partitioning and regression tree (Breiman et al., 1984) using the `rpart` package. Only the complexity parameter (`cp`) in the `rpart.control` function is tuned with 10 values between 0.013 to 0.34.

34. **nodeHarvest** is a simple interpretable tree-based ensemble for high-dimensional regression (`nodeHarvest` package) with sparse results (Meinshausen, 2010). The hyperparameters are the maximal interaction depth (`maxint`), with values 1:10, and the `mode`, which can be `mean` (weighted group means) or `outbag` (zero values in the smoothing matrix diagonal).
35. **M5** is the model tree (Quinlan, 1992), implemented by Weka and accesed through the `RWeka` package, tuning three flags: `pruned` and `smoothed`, with values `yes` and `no` each one, and `rules/trees`, a flag to select between a tree of a rule set (`tree` worked better in our experiments).
36. **ctree2** is the conditional inference tree (`ctree` function in the `party` package), which estimates the output using inference after a recursive partitioning the input space (Hothorn et al., 2006). The hyperparameters, specified in the `ctree_control` function, are the threshold `mincriterion` for $1 - p$ in order to do a split (p is the p-value of the Bonferroni statistical test, used by default), with values between 0.01 and 0.99, and the maximum tree depth (`maxdepth`), with values 1:10.
37. **partDSA** develops partitioning using deletion (D), substitution (S), and addition (A), implemented by the `partDSA` package (Molinaro et al., 2010). The only tunable hyperparameter is the maximum number of terminal partitions (`cut.off.grow`), with values 1:10. The `vfold` argument of the `DSA.control` function is set to 1.
38. **evtree** is the evolutionary regression tree (`evtree` package), called "tree model from genetic algorithms" in the `caret` model list (Grubinger et al., 2014). It uses genetic algorithms to learn globally optimal regression trees. The only hyperparameter is the complexity of the cost function (α), which weights negatively large tree sizes, tuned with 10 values between 1 and 3.

XI. Bagging ensembles

39. **bag** is the bagging ensemble of conditional inference regression trees (Breiman, 1996), implemented by the `caret` package. The output for a test pattern is the average of the outputs over all the base regression trees.

40. **bagEarth** is the bagged multivariate adaptive regression splines (MARS). Is is a bagging ensemble of MARS base regressors (see the family "other methods" below), implemented by the `caret` and `earth` packages. The only hyperparameter is the maximum number of terms (`nprune`) in the pruned regression model, with 10 values between 2 and 17.
41. **treebag** is the bagged classification and regression tree (CART), a bagging ensemble of regression trees implemented by the `ipredbagg` function in the `ipred` package.

XII. Boosting ensembles and gradient boosting machines

42. **randomGLM** is a boosting ensemble of generalized linear models provided by the `randomGLM` package (Song et al., 2013). This model uses several bootstrap samples (100 by default) of the training set, randomly selecting inputs and interaction terms among them depending on the maximum interaction order (hyperparameter `maxInteractionOrder`, tuned with values 1:3).
43. **BstLm** is the gradient boosting machine (Friedman, 2001) with linear base regressors (`bst` function in the homonymous package, with `learner = ls`). The only hyperparameter is the number of boosting iterations (`mstop`), with values 50:50:500.
44. **bstSm** (`bst` package) is the gradient boosting with smoothing splines (`learner = sm`) as base regressors, tuning the number of boosting iterations (`mstop`) equally to BstLm.
45. **bstTree** is the gradient boosting with regression base trees (`bst` package). The hyperparameters are the number of boosting iterations (`mstop`) with values 50:50:250, and the maximum depth of nodes (`maxdepth`) in the final tree, specified by the `rpart.control` function in the `rpart` package, tuned with values 1:5.
46. **glmboost** is the gradient boosting ensemble with GLM base learners (`glmboost` function in the `mboost` package), tuning the number of boosting iterations (`mstop`) equal to BstLm.

47. **gbm** is the generalized boosting regression model, called stochastic gradient boosting in the `caret` list (`gbm` package). The hyperparameters are the maximum depth of input interactions (`interaction.depth`), with values 1:5, and the number of trees for prediction (`n.trees`), with values 50:50:250. We use a Gaussian distribution and `shrinkage`=0.1.
48. **blackboost** is the gradient boosting (`blackboost` function in the `mboost` package) with conditional inference regression trees as base learners and arbitrary loss functions (Buehlmann and Hothorn, 2007). The hyperparameters are the maximum tree depth (`maxdepth`), with values 1:10, and the number of boosting iterations (`mstop`), tuned as bstTree.
49. **xgbTree** is the extreme gradient boosting (Friedman, 2001), which uses the `xgb.train` function in the `xgboost` package with `booster = gbtree` and linear regression as objective function. The hyperparameters are the maximum depth of the tree (`max_depth`), with values 1:7, the maximum number of boosting iterations (`nrounds`), with values 50:50:150, and the learning rate (η), with values 0.3 and 0.4.
50. **xgbLinear** is the same extreme gradient boosting, but with `booster = gblinear` (`xgboost` package). Its hyperparameters are the L2 (square loss) regularization term on weights (λ) and bias (α), both with values 0, 0.1 and 0.0001, and the number of iterations (`nrounds`), with values 50:50:150.

XIII. Random forests (RF)

51. **rf** is the random forest ensemble of random regression trees provided by the `randomForest` package (Breiman, 2001), whose output is the average of the regression trees outputs. Its only hyperparameter is the number of inputs randomly selected at each tree (`mtry` parameter), with values 2 and 11, which is the number of inputs.
52. **Boruta** (Kursa and Rudnicki, 2010) is the random forest ensemble with additional feature selection (`Boruta` package). The only hyperparameter is `mtry`, tuned as rf.
53. **RRF** is the regularized RF, which uses regularization to select inputs in

random forest (`RRF` package). The hyperparameters are `mtry`, with values 2, 6 and 11, and the regularization coefficient (`coefReg`), with values 0.01, 0.505 and 1.

54. **cforest** (`party` package) is a random forest ensemble of conditional inference trees (Breiman, 2001), each one fitting one bootstrap sample. The only hyperparameter is `mtry`, tuned as rf.

55. **qrf** is the quantile regression forest (`quantregForest` package), a tree-based ensemble which generalizes RF in order to estimate conditional quantile functions. The `mtry` parameter is tuned as rf. The quantile prediction threshold (`what` argument in the `predict.quantregForest` function) is set to 0.5.

56. **extraTrees** (Geurts et al., 2006) is the ensemble of extremely randomized regression trees (`extraTrees` package). Its tunable hyperparameters are `mtry` (same values as rf), and the minimum sample size to split a node (`numRandomCuts`), with values 1:10.

XIV. Prototype models

57. **kknn** (Hechenbichler and Schliep, 2004) performs weighted k-nearest neighbors regression (`kknn` package). The only hyperparameter is the number of neighbors (`k`), with values 5:2:23. The default optimal kernel and Minkowski distance are used.

58. **cubist** (`Cubist` package) learns a M5 rule-based model with corrections based on nearest neighbors in the training set (Quinlan, 1993). Its hyperparameters are the number of training committees, with values 1, 10 and 20, and the number of neighbors for prediction, with values 0, 5 and 9.

XV. Bayesian models

59. **bayesglm** is the Bayesian GLM (`arm` package). It uses the expectation maximization method to update the β values in GLM at each iteration, representing the prior information with an augmented regression (Gelman

et al., 2009). The coefficients are calculated using a student-t prior distribution.

60. **brnn** (Foresee and Hagan, 1997) is the Bayesian regularized neural network (`brnn` package). The Bayesian regularization (MacKay, 1992) determines the weights of two terms (squared error and squared sum of network weights) based on inference techniques. The weights are not normalized, and the number of hidden neurons is a hyperparameter tuned with values 1:20.

61. **bartMachine** (Kapelner and Bleich, 2016) is the Bayesian additiveregression tree (`bartMachine` package). The tunable hyperparameters are the prior boundary (`k`), with values 2, 3 and 5, and the base value in tree prior to decide if a node is terminal or not (α), with values 0.9, 0.945 and 0.99.

XVI. Principal component regression

62. **pcr** develops principal component regression (`pls` package). This method models the output using classical linear regression with coefficients estimated with PCA, i.e., using the principal components instead of the original inputs (Mevik and Cederkvist, 2004). The number of components (`ncomp`) is tuned with values 1 and 2.

63. **icr** is the independent component regression (`caret` package). It fits a linear regression model using independent component analysis (ICA), implemented by the `fastICA` package, instead of the original inputs (Hyvarinen and Oja, 2000). The only hyperparameter is the number of independent components (`n.comp`), with values 1:11.

64. **superpc** (Bair and Tibshirani, 2004) is the supervised PCA (`superpc` package). The number of principal components, with values 1:3, and the threshold for retaining the input scores, with values 0.1 and 0.9, are the tunable hyperparameters.

XVII. Generalized additive models (GAM)

65. **gam** (`mgcv` package) is the generalized additive model using splines (Wood, 2011). The only hyperparameter is `select`, a boolean flag which adds, when true, an extra penalty term to each function penalizing its wiggliness (waving).

66. **gamboost** (`mboost` package) is the boosted generalized additive model (Buehlmann and Yu, 2003). The only hyperparameter is the number of initial boosting iterations (`mstop`), with values 50:50:500.

XVIII. Gaussian processes

67. **gaussprLinear** implements Gaussian process regression with linear kernel, called `vanilladot` in the `gausspr` function of the `kernlab` package.

68. **gaussprRadial** uses the same function (with `kernel = rbfdot` for a Gaussian kernel) and package. By default, the kernel spread is calculated automatically, with value 0.1145.

69. **gaussprPoly** is the same method with polynomial kernel (`polydot`), tuning the kernel hyperparameters `degree` and `scale`, with values 1:3 and $\{10^{-i}\}_1^3$, respectively.

XIX. Quantile regression

70. **rqlasso** develops quantile regression with LASSO penalty, using the `rq.lasso.fit` function in the `rqPen` package. This method fits a quantile regression model with the LASSO penalty (Mizera and Koenker, 2014), tuning the regularization hyperparameter (λ), with 10 values from 0.1 to 10^{-4}.

71. **rqnc** performs non-convex penalized quantile regression, with the `rq.nc.fit` function in the same `rqPen` package. This regressor performs penalized quantile regression using local linear approximation (Zou and Li, 2008) to maximize the penalized likelihood for non-convex penalties. The two hyperparameters are λ, with the same 10 values as rqlasso, and the penalty

type, which can be `MCP` (minimax concave penalty) or `SCAD` (smoothly clipped absolute deviation).

72. **qrnn** (Cannon, 2011) is the quantile regression neural network (`qrnn` package). The hyperparameters are number of hidden neurons and the penalty for weight decay regularization, with values 1:2:19 and $0, \{10^{-i}\}_1^4$ respectively.

XX. Other methods

73. **lars** (Efron et al., 2004) is the least angle regression (`Lars` package). The `lasso` type and `fraction` mode are specified for training and prediction respectively, and the fraction hyperparameter `s` is tuned with 10 values from 0.05 to 1.

74. **earth** (Friedman, 1991) is the multivariate adaptive regression spline (MARS), implemented by the `earth` package. The maximum number of terms in the model (`nprune`) is tuned with values 2:17.

75. **ppr** performs the projection pursuit regression (Friedman and Stuetzle, 1981), implemented by the `stats` package. The coefficients are iteratively calculated to minimize a projection pursuit (fitting criterion, given by the fraction of unexplained variance which is explained by each function) until it falls below a predefined threshold. The only hyperparameter is the number of terms to be included in the final model (`nterms`), tuned with values 1:10.

76. **sbc** (frbs package) is the subtractive clustering and fuzzy c-means rules (Chiu, 1996). This method iteratively selects the cluster centers as training patterns with a high potencial function, which increases with the number of nearby neighbors. Once all the centers are selected, they are optimized using fuzzy C-means. The only hyperparameter is the neighborhood radius (`r.a` argument of the control list in the `frbs.learn` function), tuned with 7 values between 0 and 1.

5. Results and discussion

In order to evaluate the quality of the regressors in the prediction of the five village-wise soil fertility indices (OC, P_2O_5, Fe, Mn and Zn), we used a variant of the well-known cross-validation methodology (Kohavi, 1995) which is specially adequate for methods with tunable hyperparameters. This variant uses three datasets (training, validation and test), instead of classical cross-validation, which only uses training and test datasets. The reason to use three sets instead of two is the existence of tunable hyperparameters. With only two datasets, we only have the test set: 1) to evaluate the model performance with each hyperparameter value in order to select the best one; and 2) to evaluate the performance of the final model with the selected value. However, doing this the final performance is optimistically biased, because it is measured in the same test set where the model, trained with the selected value of the hyperparameter, achieved its best performance. Thus, the performance of the final model for other datasets would be expected to be lower. The need to select an "optimal value" for the tunable hyperparameters requires to evaluate the model, trained for each hyperparameter value, in a dataset different to the test set, where the performance of the final model, trained with the best hyperparameter values, will be tested. Hence the need of validation datasets. In order to develop the experiments, we created ten random partitions of the datasets. Remember from subsection 4.1 that the five regression problems include 372 patterns, each one corresponding to a cultivation land, with 11 inputs. All the inputs and outputs are pre-processed (standarized) in order to have zero mean and standard deviation one. For each partition, 50% of the patterns are selected for training, 25% for validation (which is used to evaluate the performance of the model trained with each hyperparameter value during tuning) and 25% for test (where the model trained with the selected hyperparameter value, which maximizes the average performance over the validation sets, is tested). In our case, there are 186 training, 93 validation and 93 test patterns. Each regressor is trained on the 10 training partitions for each combination of its hyperparameter values

(regressors may have zero, one, two or three tunable hyperparameters), and tested on its corresponding validation partition. The performance measures used are the root mean square error (RMSE) and the coefficient of determination (R^2), defined as the proportion of the input variance which is explained by the regressor output, which is equal to the square of the correlation coefficient R:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - d_i)^2}, \qquad R^2 = \frac{\left[\sum_{i=1}^{N}(y_i - \bar{y})\left(d_i - \bar{d}\right)\right]^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2 \sum_{i=1}^{N}\left(d_i - \bar{d}\right)^2} \tag{2}$$

where N is the number of test patterns, y_i and d_i are the predicted (by the regressor) and true output, respectively, and $\bar{y}$ and $\bar{d}$ are the averages of y_i and d_i, with $i = 1, \ldots, N$, respectively. For each combination of hyperparameter values, the average RMSE over the 10 validation sets is calculated, and the combination with the lowest RMSE is selected for testing. Finally, the regressor is trained, using this selected combination of its hyperparameter values, on the 10 training partitions, and tested on the 10 test partitions. The average RMSE and R^2 over the 10 test sets are used as the final performance measures.

5.1. Analysis by soil dataset

Table 2 reports the R^2 values for the prediction of OC village-wise soil fertility index, sorted by decreasing values. The random forest with feature selection (Boruta) achieves the highest R^2 value (0.69815), followed by regularized random forests (RRF), random forest (rf) and extremely randomized regression trees (extraTrees), whose R^2 value are above 0.69. In order to evaluate the meaning of these values, we can use the classical definition of Colton (1974) for the correlation intervals and their significance. These intervals, translated into values of determination coefficient (R^2), are the following: a R^2 value in the range 0–0.0225 means that the two vectors under comparison (in our case, these vectors are the true and predicted values of the village-wise soil fertility index for each nutrient) are not correlated at all; R^2 between 0.0225 and

0.25 mean bad to moderate correlation between them; R^2 between 0.25 and 0.5625 mean moderate to good correlation; and $R^2 > 0.5625$ mean very good to excellent correlation. The values of R^2 achieved by Boruta and the following regressors are about 0.69, which correspond to a very good to excellent correlation with the true OC fertility index. Other regressors with R^2 about 0.66 are gradient boosting of regressor trees (bstTree), gradient boosted machine (gbm), Bayesian additive regression tree (bartMachine) and simple interpretable tree-based ensemble for high-dimensional regression (nodeHarvest). The remaining regressors achieve R^2 below 0.65. Note that all the regressors exhibit R^2 above 0.2 excepting the last 6 regressors (from mlpWDml to partDSA), which are not able to learn the regression problem. Figure 3 depicts the output of the best

Regressor	R^2	Regressor	R^2	Regressor	R^2	Regressor	R^2
Boruta	0.69815	cforest	0.60614	spikeslab	0.50615	gaussprLinear	0.43974
RRF	0.69667	treebag	0.59523	lars	0.50369	glm	0.43889
rf	0.69660	brnn	0.58261	glmStepAIC	0.50252	lm	0.43889
extraTrees	0.69189	blackboost	0.56922	rqlasso	0.50040	lasso	0.43889
bstTree	0.67383	grnn	0.56327	spls	0.49312	gam	0.43889
gbm	0.66724	penalized	0.55751	rqnc	0.49116	pcaNNet	0.43857
bartMachine	0.66485	ppr	0.54940	mlpWD	0.48570	enpls.fs	0.43491
nodeHarvest	0.66388	rbf	0.54533	foba	0.48423	rpart	0.43102
qrf	0.65764	avNNet	0.54273	xgbLinear	0.48143	randomGLM	0.42962
cubist	0.64978	bagEarth	0.54094	earth	0.47925	evtree	0.39976
svr	0.63567	SBC	0.53885	bdk	0.47728	BstLm	0.36877
krlsRadial	0.61692	xgbTree	0.53665	elm	0.47169	superpc	0.34595
gaussprRadial	0.61548	kernelpls	0.52951	rlm	0.46442	nnls	0.28929
svmRadial	0.61537	simpls	0.52951	plsRglm	0.46375	mlpWDml	0.05533
gamboost	0.61451	dlkeras	0.52746	ridge	0.45061	pcr	0.00280
rvmRadial	0.61086	relaxo	0.52545	M5	0.44782	icr	0.00149
bstSm	0.61072	qrnn	0.51859	gaussprPoly	0.44605	dnn	0.00124
kknn	0.60791	bag	0.51410	ctree2	0.44506	glmnet	0.00000
elm-kernel	0.60701	glmboost	0.51175	bayesglm	0.44234	partDSA	0.00000

Table 2: Values of R^2 achieved by each regressor for the prediction of OC village-wise soil fertility index, sorted decreasingly.

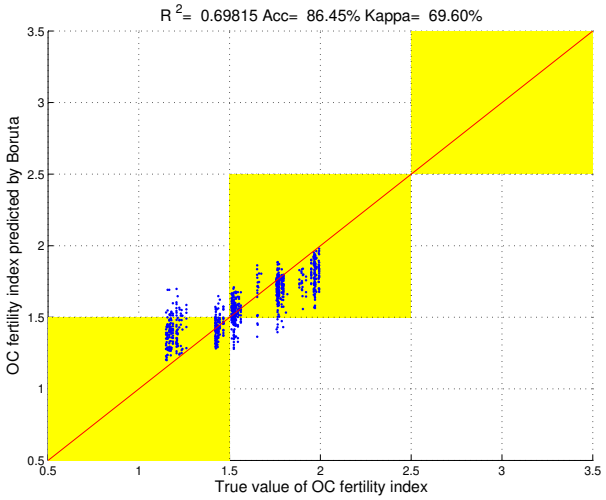

Figure 3: Scatter plot of true and predicted *OC* fertility index using the best regressor (Boruta). The yellow squares are the areas where the predicted indices (vertical coordinates of the blue dots) are rounded to the same integer as the true index (horizontal coordinates of the blue dots).

True/Predicted	Low	Medium
Low	**248**	49
Medium	77	**556**

Table 3: Confusion matrix achieved by Boruta rounding the true and predicted *OC* village-wise soil fertility index to their nearest integer values.

regressor (Boruta) in the vertical axis, and the right value of *OC* soil fertility index in the horizontal axis, for all the test patterns (930, because there are 10 test partitions with 93 patterns each one). In order to evaluate the quality of the prediction, both values are plotted in their original scales, before data standarization. Thus, we can evaluate how much the predicted *OC* fertility index differs from the true value. Since the *OC* fertility of the Marathwada soils is not high, the true fertility index ranges between 1 and 2 (low and medium fertilities, respectively), and the indices predicted by Boruta are also in the same range. In fact, if we round the true and predicted indices to the nearest integer, both roundings agree for 86.45% of the test patterns. This is the accuracy that

Boruta would achieve if we transform the regression problem in a classification problem where the classes are the OC fertility indices, rounded to their nearest integer (see the accuracy in the figure title). The yellow squares in the plot show the places where the patterns are well classified. The blue points that are placed inside (resp. outside) these squares are correctly (resp. wrong) classified, because the rounded true and predicted indices agree (resp. disagree), and only few patterns are located outside the yellow squares. Table 3 reports the confusion matrix of Boruta for this classification problem. From this matrix, the Cohen kappa (κ), which measures the agreement between the classifier and the true class labeling discarding the probability of agreement by chance (Viera and Garrett, 2005), is also high (69.6%), although always lower than the accu-

Regressor	R^2	Regressor	R^2	Regressor	R^2	Regressor	R^2
extraTrees	0.60301	gaussprRadial	0.52148	pcaNNet	0.37141	glmboost	0.12920
RRF	0.60115	brnn	0.51527	bagEarth	0.35866	enpls.fs	0.10175
qrf	0.59993	nodeHarvest	0.51320	mlpWD	0.35857	pcr	0.08860
gbm	0.59991	blackboost	0.49348	bdk	0.34899	bstSm	0.08298
Boruta	0.59981	M5	0.45782	randomGLM	0.33879	rlm	0.05088
rf	0.59684	bag	0.45738	nnls	0.33878	glmStepAIC	0.03505
svr	0.59588	ppr	0.44975	rqnc	0.33383	spikeslab	0.03259
cubist	0.59252	xgbTree	0.44588	rqlasso	0.33280	xgbLinear	0.02643
bstTree	0.58502	SBC	0.44563	gamboost	0.28844	ridge	0.02539
bartMachine	0.57282	rbf	0.44437	evtree	0.27929	bayesglm	0.01853
svmRadial	0.56714	qrnn	0.44347	foba	0.26488	gaussprLinear	0.01670
krlsRadial	0.55043	penalized	0.43679	dlkeras	0.25324	glm	0.01623
avNNet	0.53810	elm	0.40851	BstLm	0.23718	lm	0.01623
rvmRadial	0.53588	rpart	0.39155	plsRglm	0.20345	lasso	0.01623
cforest	0.53568	relaxo	0.38994	earth	0.17010	gam	0.01623
elm-kernel	0.53475	spls	0.38386	lars	0.15086	gaussprPoly	0.01211
kknn	0.53335	kernelpls	0.38350	icr	0.15069	partDSA	0.00986
treebag	0.52475	simpls	0.38350	superpc	0.14513	dnn	0.00575
grnn	0.52401	ctree2	0.38011	mlpWDml	0.14044	glmnet	0.00084

Table 4: Values of R^2 achieved by the regressors for the prediction of P_2O_5 village-wise soil fertility index.

racy. These values show that the performance of Boruta is enough for a reliable prediction of this fertility index.

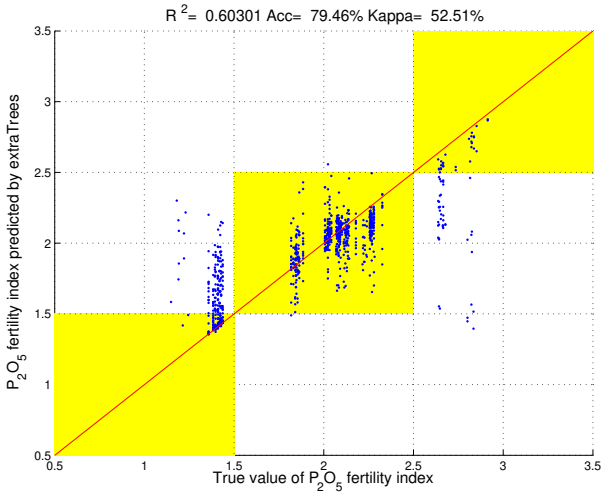

Figure 4: Scatter plot of true values (horizontal axis) and values predicted by extraTrees (vertical axis) of the P_2O_5 soil fertility index.

True/Predicted	Low	Medium	High
Low	**125**	145	0
Medium	1	**586**	1
High	3	41	**28**

Table 5: Confusion matrix achieved by extraTrees rounding the true and predicted P_2O_5 village-wise soil fertility indices to their nearest integer.

With respect to P_2O_5 village-wise soil fertility index, Table 4 reports the R^2 achieved by each regressor. In this case, extraTrees achieves the best R^2 (0.60301, very good to excellent in the Colton scale) alongside with RRF, quantile regression forest (qrf), gbm, Boruta, rf, support vector regression (svr) and M5 rule model corrected by nearest neighbors (cubist), all with values about 0.60, less than for OC fertility index. Figure 4 is the corresponding scatter plot of the original P_2O_5 soil fertility indices and the values predicted by extraTrees. In this case, the values are in the range between 1 and 3, including patterns (i.e.,

cultivation lands) with low, medium and high P_2O_5 fertility indices. The plot shows that prediction is right (accuracy 79.46% and κ= 52.51%) for most of the patterns with medium fertility (placed in the center yellow square). However, for most patterns with low and high indices the predicted values are middle, so they fall outside the squares. Table 5 reports the confusion matrix for such a classification problem, where the terms true low/predicted medium (in the first row and second column) and true high/predicted medium (third row and second column) are higher than the corresponding diagonal terms.

Regressor	R^2	Regressor	R^2	Regressor	R^2	Regressor	R^2
extraTrees	0.66652	grnn	0.51967	penalized	0.33675	plsRglm	0.06026
rf	0.65923	xgbTree	0.50750	elm	0.32940	glmboost	0.03743
RRF	0.65703	avNNet	0.47499	relaxo	0.27555	dnn	0.03734
Boruta	0.65093	earth	0.47386	nnls	0.26595	gaussprPoly	0.03163
bstTree	0.62280	bag	0.45247	rqnc	0.26566	enpls.fs	0.02601
gbm	0.62250	blackboost	0.44578	dlkeras	0.26485	spikeslab	0.01620
qrf	0.62056	brnn	0.44328	mlpWD	0.25959	rlm	0.01175
cubist	0.60060	gamboost	0.43894	foba	0.24628	randomGLM	0.00347
nodeHarvest	0.58885	SBC	0.43771	kernelpls	0.24505	xgbLinear	0.00292
bartMachine	0.58080	rbf	0.42042	simpls	0.24505	glmnet	0.00159
svr	0.57207	bagEarth	0.41911	rqlasso	0.23354	glmStepAIC	0.00157
krlsRadial	0.57181	bdk	0.41487	superpc	0.22517	bayesglm	0.00141
svmRadial	0.55790	bstSm	0.41329	icr	0.19926	ridge	0.00139
elm-kernel	0.55696	qrnn	0.40408	pcr	0.17496	gaussprLinear	0.00094
gaussprRadial	0.55238	ppr	0.39852	M5	0.17016	glm	0.00092
treebag	0.54950	ctree2	0.38632	lars	0.11877	lm	0.00092
cforest	0.54369	rpart	0.37844	BstLm	0.07228	lasso	0.00092
rvmRadial	0.53953	pcaNNet	0.37068	spls	0.06965	gam	0.00092
kknn	0.53278	evtree	0.35403	mlpWDml	0.06801	partDSA	0.00078

Table 6: Values of R^2 for the prediction of Fe village-wise soil fertility index.

Again, the best regressor for the prediction of Fe village-wise soil fertility index (Table 6) is extraTrees, achieving R^2= 0.66652 (very good to excellent in the Colton scale). The R^2 values of the following regressors fall very fast: rf and RRF (about 0.659), Boruta (0.650) and a group of regressors (bstTree, gbm

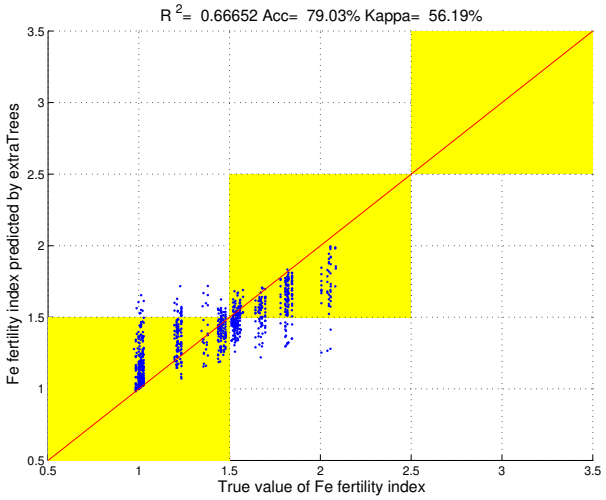

Figure 5: Scatter plot of true Fe soil fertility index and values predicted by extraTrees.

and qrf) about 0.622. The cubist achieves 0.600, and the remaining are already below 0.58. Figure 5 shows the true and predicted Fe fertility indices for the test sets, which belong to levels low and medium. The accuracy (79.03%), or percentage of patterns which fall inside the yellow squares, is lower than for OC fertility, which gives a smaller κ (56.19%). However, the confusion matrix (Table 7) shows that the number of low patterns which are classified as medium is relatively low (10.2% of the low patterns), while more medium patterns are classified as low (34.9% of the medium patterns).

True/Predicted	Low	Medium
Low	**473**	54
Medium	141	**262**

Table 7: Confusion matrix achieved by extraTrees rounding the true and predicted Fe village-wise soil fertility index.

For the prediction of Mn village-wise soil fertility index (Table 8), extraTrees achieves the best R^2 (0.57499), which according to the Colton scale can be considered as very good to excellent. The following five regressors (RRF, bstTree, rf, Boruta and gbm) work sensibly worse, with R^2 about 0.54. Afterwards, the

Regressor	R^2	Regressor	R^2	Regressor	R^2	Regressor	R^2
extraTrees	0.57499	avNNet	0.47010	elm	0.31200	xgbLinear	0.11012
RRF	0.54935	grnn	0.46975	kernelpls	0.30838	rlm	0.10505
bstTree	0.54631	treebag	0.46004	simpls	0.30838	foba	0.10295
rf	0.54572	earth	0.44231	bdk	0.30613	ridge	0.09627
Boruta	0.54305	xgbTree	0.42529	rpart	0.30403	mlpWDml	0.09540
gbm	0.54285	pcaNNet	0.42044	nnls	0.28009	pcr	0.08889
svr	0.53824	bagEarth	0.41290	glmboost	0.27325	bayesglm	0.08209
cubist	0.53317	rbf	0.39635	rqnc	0.26635	gamboost	0.08036
svmRadial	0.52417	mlpWD	0.39625	evtree	0.25813	gaussprLinear	0.07573
qrf	0.52347	SBC	0.39298	rqlasso	0.25330	glmStepAIC	0.07511
rvmRadial	0.51616	bag	0.38973	BstLm	0.21673	glm	0.07458
krlsRadial	0.51495	blackboost	0.38479	lars	0.20178	lm	0.07458
gaussprRadial	0.50518	M5	0.38474	plsRglm	0.19425	lasso	0.07458
nodeHarvest	0.50315	brnn	0.38272	spls	0.19379	gam	0.07458
bartMachine	0.50297	penalized	0.35816	enpls.fs	0.18956	bstSm	0.05226
cforest	0.49539	ppr	0.33826	gaussprPoly	0.18410	dnn	0.04960
kknn	0.49437	qrnn	0.33438	superpc	0.18124	icr	0.02950
elm-kernel	0.49368	relaxo	0.31610	randomGLM	0.17986	glmnet	0.00001
dlkeras	0.49090	ctree2	0.31319	spikeslab	0.17081	partDSA	0.00000

Table 8: Values of R^2 for the prediction of Mn village-wise soil fertility index.

following regressors fall very fast: svr and cubist (about 0.53), svmRadial and qrf (0.52), rvmRadial and krlsRadial (0.52) and so on. The Mn soil fertility indices belong to levels low, medium and high (Figure 6), and the reason of the poor R^2 is the amount of low and high patterns for which the predicted values are near to medium. However, the proportion of patterns assigned to the correct indices is high (accuracy 86.13%), and κ is also high (71.08%). The confusion matrix (Table 9) also reports that almost all the low patterns are assigned to medium levels (just 1 of 11 low patterns is predicted as low), while a relatively reduced percentage (15.95%) of high patterns are predicted as medium.

Finally, the best R^2, achieved by extraTrees, is fairly high (0.70712) for the prediction of Zn fertility index (Table 10). In this case, the difference to the following regressor (cubist, whose $R^2 = 0.630$) is huge. Like the previous

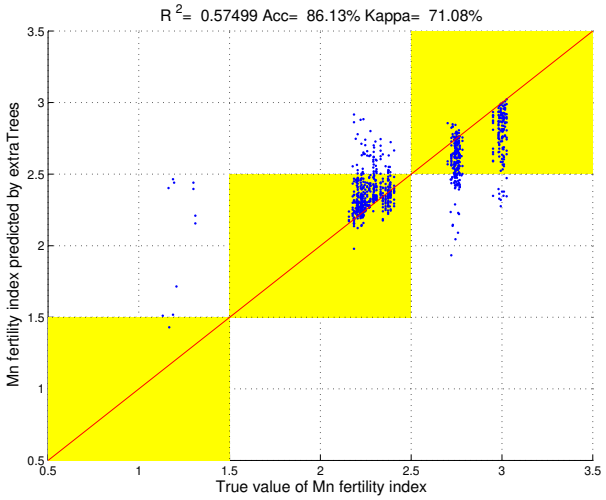

Figure 6: Scatter plot of true Mn soil fertility indices and values predicted by extraTrees.

True/Predicted	Low	Medium	High
Low	**1**	10	0
Medium	0	**505**	63
High	0	56	**295**

Table 9: Confusion matrix created by rounding the true Mn soil fertility indices and predicted values by extraTrees.

indices, the R^2 of the following regressors decreases very quickly: commitee of neural networks (avNNet, 0.62); svr and gbm (0.61); qrf, rf and Boruta (0.60). The remaining regressors are already about 0.59 and so on. Figure 7 plots the corresponding scatter plot for extraTrees: there are only low and medium test patterns, and the percentage of patterns whose rounded predicted index is correct (i.e., inside the yellow squares) is very high (accuracy 97.63%), which gives a high κ (81.03%). The confusion matrix (Table 11) reports the extremely reduced percentage of low patterns predicted as medium, although more medium patterns are predicted as low (29.2%).

Regressor	R^2	Regressor	R^2	Regressor	R^2	Regressor	R^2
extraTrees	0.70712	SBC	0.46920	bagEarth	0.23394	rlm	0.08637
cubist	0.63069	brnn	0.46774	rbf	0.21590	lars	0.08461
avNNet	0.62473	pcaNNet	0.45139	kernelpls	0.18168	plsRglm	0.08374
svr	0.61531	treebag	0.44831	simpls	0.18168	elm	0.07797
gbm	0.61373	cforest	0.43893	BstLm	0.11358	gaussprPoly	0.07442
qrf	0.60643	nodeHarvest	0.43840	xgbLinear	0.10238	rqnc	0.07356
rf	0.60526	evtree	0.41917	bayesglm	0.09802	mlpWDml	0.06757
Boruta	0.60106	dlkeras	0.41058	glm	0.09578	nnls	0.06307
RRF	0.59920	rvmRadial	0.40722	lm	0.09578	earth	0.03920
krlsRadial	0.59675	blackboost	0.38481	lasso	0.09578	foba	0.03078
bstTree	0.59590	qrnn	0.38024	gam	0.09578	partDSA	0.01036
svmRadial	0.59582	xgbTree	0.35661	gaussprLinear	0.09567	icr	0.00970
bartMachine	0.59327	rpart	0.33419	glmStepAIC	0.09534	dnn	0.00714
elm-kernel	0.55800	bag	0.33220	spikeslab	0.09492	glmnet	0.00537
kknn	0.53341	penalized	0.29990	ridge	0.09394	randomGLM	0.00240
grnn	0.51415	M5	0.29530	rqlasso	0.09286	gamboost	0.00075
gaussprRadial	0.50238	bdk	0.27797	spls	0.09153	pcr	0.00026
mlpWD	0.48543	ctree2	0.27698	enpls.fs	0.08895	bstSm	0.00017
ppr	0.47525	relaxo	0.24021	glmboost	0.08858	superpc	0.00000

Table 10: Values of R^2 for the prediction of Zn village-wise soil fertility index.

True/Predicted	Low	Medium
Low	**857**	1
Medium	21	**51**

Table 11: Confusion matrix of extraTrees for Zn soil fertility index.

6. Global discussion

Considering the results over all the soil datasets, extraTrees achieves the most accurate prediction of soil fertility indices for four of five nutrients (P_2O_5, Fe, Mn and Zn), being the fourth for OC, very near (difference 0.007) to the best regressor (Boruta). Besides, other four regressors of the random forest family (rf, RRF, Boruta and qrf) are among the first ten regressors for the six nutrients. Therefore, this family of regressors can be considered the best for

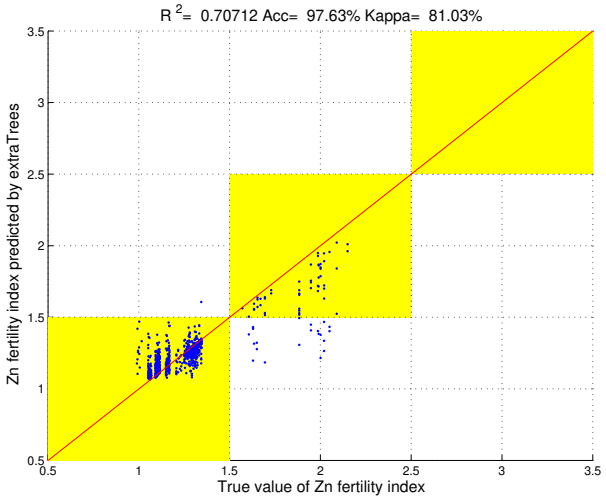

Figure 7: Scatter plot of the true Zn soil fertility indices (horizontal axis) and predicted values by extraTrees (vertical axis).

these datasets, confirming the good result of random forests for soil classification in our previous work (Sirsat et al., 2017). The svr, two gradient boosting ensembles (bstTree and gbm) and M5 rule with nearest neighbors (cubist) are also included among the ten bests regressors.

Tables 12 and 13 report for each regressor the Friedman ranking (García et al., 2007), alongside with the average value, of R^2 over the five soil fertility indices. This ranking compares the regressors by sorting, for each dataset, the regressors by decreasing R^2; and averaging, for each regressor, its positions over all the datasets. This average position is the Friedman rank of each regressor, which decreases with its performance. As an example, extraTrees achieves a Friedman rank of 1.6, which means that, in average over all the fertility indices, it is in the position 1.6, i.e., it achieves a R^2 value between the first and second bests. ExtraTrees also achieves the best average R^2 over all the indices (0.64871). Other four regressors of the random forest family (RRF, rf, qrf, and Boruta) are placed in the first positions, but their average R^2 is about 0.62, far from extraTrees. Two gradient boosting ensembles (bstTree and gbm), support vector regression (svr) and cubist (a prototype based model) are placed in po-

Pos.	Regressor	Rank	Avg. R^2	Pos.	Regressor	Rank	Avg. R^2
1	extraTrees	1.6	0.64871	20	grnn	20.0	0.51817
2	RRF	3.6	0.62068	21	treebag	20.0	0.51556
3	rf	4.4	0.62073	22	brnn	24.6	0.47832
4	Boruta	4.6	0.61860	23	blackboost	26.2	0.45562
5	gbm	5.4	0.60925	24	xgbTree	26.8	0.45439
6	bstTree	6.6	0.60477	25	SBC	27.0	0.45688
7	qrf	7.0	0.60161	26	ppr	28.0	0.44224
8	cubist	7.2	0.60135	27	bag	29.8	0.42918
9	svr	8.0	0.59143	28	rbf	30.4	0.40448
10	bartMachine	11.0	0.58294	29	penalized	32.6	0.39782
11	krlsRadial	11.6	0.57017	30	bagEarth	32.8	0.39311
12	svmRadial	11.8	0.57208	31	qrnn	33.0	0.41615
13	gaussprRadial	15.6	0.53938	32	dlkeras	34.8	0.38941
14	nodeHarvest	15.6	0.54150	33	mlpWD	35.4	0.39711
15	elm-kernel	16.2	0.55008	34	relaxo	37.0	0.34945
16	avNNet	17.2	0.53013	35	pcaNNet	37.2	0.41050
17	kknn	17.2	0.54036	36	kernelpls	39.2	0.32962
18	rvmRadial	17.4	0.52193	37	M5	39.6	0.35117
19	cforest	18.4	0.52397	38	bdk	40.0	0.36505

Table 12: Friedman ranking and average of R^2 over the five fertility indices (continued in Table 13).

sitions 5–10, with R^2 about 0.60. Figure 8 plots the RMSE and R^2 Friedman ranks in the vertical and horizontal axis, respectively, for the 20 best regressors, showing the high agreement between both rankings (excepts perhaps for the last five regressors).

Figure 9 shows the boxplots of the R^2 achieved by the 20 best regressors in the Friedman rank over the 5 fertility indices. ExtraTrees exhibits the highest box, whose median and 75% quartile are much above the other regressors. The boxes of RRF, rf and Boruta have similar 75% quartiles, at the same level as the extraTree median, and their upper wiskers are near extraTrees (about 0.7). The remaining regressors (gbm, bstTree, qrf, cubist, svr, bartMachine, krlsRadial and svmRadial) already have 75% quartiles below 0.65. This quartile is also

Pos.	Regressor	Rank	Avg. R^2	Pos.	Regressor	Rank	Avg. R^2
39	simpls	40.2	0.32962	58	glmStepAIC	58.0	0.14192
40	ctree2	40.8	0.36034	59	enpls.fs	58.8	0.16824
41	rpart	41.8	0.36785	60	rlm	58.8	0.14370
42	earth	42.6	0.32094	61	bayesglm	60.4	0.12848
43	elm	44.4	0.31991	62	randomGLM	60.4	0.19083
44	evtree	45.2	0.34207	63	ridge	60.6	0.13352
45	gamboost	45.4	0.28460	64	gaussprPoly	61.0	0.14966
46	rqlasso	47.8	0.28258	65	superpc	61.2	0.17950
47	spls	48.2	0.24639	66	mlpWDml	62.2	0.08535
48	rqnc	48.2	0.28611	67	gaussprLinear	62.6	0.12575
49	bstSm	51.4	0.23189	68	glm	62.8	0.12528
50	lars	51.4	0.21194	69	lm	63.8	0.12528
51	glmboost	51.4	0.20804	70	pcr	64.2	0.07110
52	nnls	53.0	0.24744	71	icr	64.4	0.07813
53	BstLm	53.2	0.20171	72	lasso	64.8	0.12528
54	foba	53.6	0.22582	73	gam	65.8	0.12528
55	plsRglm	54.6	0.20109	74	dnn	70.4	0.02021
56	spikeslab	55.0	0.16413	75	glmnet	72.8	0.00156
57	xgbLinear	56.0	0.14465	76	partDSA	74.0	0.00420

Table 13: Continuation of Table 12.

above 0.6 for NodeHarvest, but it exhibits the tallest box, which means an unstable behavior compared to the previous regressors.

It is interesting to analyze the best regressors of the remaining families (subsection 4.2). The best least squares regressor is krlsRadial (Table 12), which is located in position 11 with $R^2 = 0.57$. The position 13 is for gaussprRadial (Gaussian process). The best performing neural network is elm-kernel (position 15), although avNNet and generalized regression neural network (grnn) also achieve good results (positions 16 and 20, respectively). The best regression tree is nodeHarvest (position 14), while treebag is the best bagging ensemble (position 21). Two regressors included in "other methods" achieve good positions: substractive clustering and fuzzy C-means (SBC, position 25), projection pursuit regression (ppr, position 26). The best GLM is penalized (position 29),

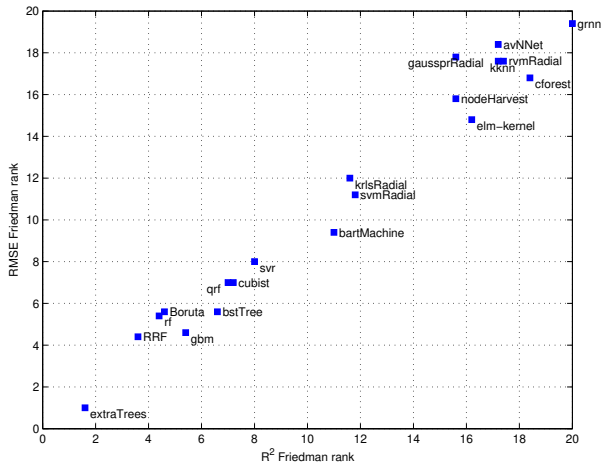

Figure 8: Friedman rank of the RMSE vs. the Friedman rank of R^2 for the 20 best regressors in Table 12.

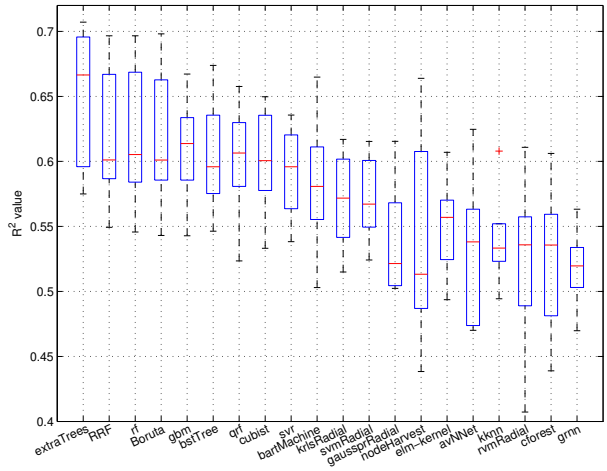

Figure 9: Boxplot of the R^2 achieved by the 20 best regressors on the five fertility indices.

and position 31 is for quantile regression neural network (qrnn), followed by dlkeras (deep learning). The LASSO regressor relaxo is located in position 34 and the PLS regressor kernelpls is in position 36. Finally, the best regressors of several classical families achieve poor results (positions after 50 with $R^2 < 0.2$): linear regression (regressor rlm, position 60), ridge regression (position 63), principal component regression (superpc, position 65) and generalized additive

Pos.	Regressor	p-value	Pos.	Regressor	p-value	Pos.	Regressor	p-value
1	rf	0.54762	26	enpls.fs	0.00794	51	gaussprPoly	0.00794
2	Boruta	0.42063	27	grnn	0.00794	52	rqnc	0.00794
3	gbm	0.42063	28	blackboost	0.00794	53	penalized	0.00794
4	RRF	0.42063	29	glmnet	0.00794	54	icr	0.00794
5	qrf	0.30952	30	earth	0.00794	55	relaxo	0.00794
6	bstTree	0.30952	31	rpart	0.00794	56	bag	0.00794
7	cubist	0.22222	32	bagEarth	0.00794	57	M5	0.00794
8	svr	0.15079	33	kernelpls	0.00794	58	rbf	0.00794
9	bartMachine	0.09524	34	simpls	0.00794	59	SBC	0.00794
10	krlsRadial	0.05556	35	rlm	0.00794	60	glmboost	0.00794
11	nodeHarvest	0.05556	36	ridge	0.00794	61	elm	0.00794
12	svmRadial	0.05556	37	nnls	0.00794	62	spikeslab	0.00794
13	elm-kernel	**0.03175**	38	lars	0.00794	63	pcaNNet	0.00794
14	bstSm	0.03175	39	pcr	0.00794	64	ctree2	0.00794
15	avNNet	0.03175	40	lasso	0.00794	65	plsRglm	0.00794
16	gaussprRadial	0.03175	41	ppr	0.00794	66	bdk	0.00794
17	rvmRadial	0.03175	42	glmStepAIC	0.00794	67	xgbLinear	0.00794
18	kknn	0.03175	43	xgbTree	0.00794	68	mlpWD	0.00794
19	cforest	0.03175	44	foba	0.00794	69	randomGLM	0.00794
20	gamboost	0.03175	45	superpc	0.00794	70	partDSA	0.00794
21	treebag	0.01587	46	gam	0.00794	71	evtree	0.00794
22	brnn	0.01587	47	gaussprLinear	0.00794	72	BstLm	0.00794
23	glm	0.00794	48	rqlasso	0.00794	73	mlpWDml	0.00794
24	lm	0.00794	49	bayesglm	0.00794	74	qrnn	0.00794
25	dlkeras	0.00794	50	spls	0.00794	75	dnn	0.00794

Table 14: List of the p-values, sorted decreasingly, achieved by the Wilcoxon signed rank test comparing R^2 of extraTrees and the remaining 75 regressors over the five soil fertility indices. In bold, is the first regressor whose comparison to extraTrees is statistically significant ($p < 0.05$).

models (gamboost, position 45).

Table 14 reports the p-values for a Wilcoxon signed rank test (Wilcoxon, 1945) comparing the R^2 achieved by extraTrees, which achieves the lowest Friedman rank, sorted by descending order. The value in bold corresponds to the regressor (elm-kernel, position 13 of 75) from which the difference with respect to

extraTrees is statistically significant for a 5%-confidence level ($p < 0.05$). This difference is not statistically significant until position 13 (elm-kernel). However, the comparison of extraTrees and the second regressor (rf) exhibits a p-value (0.54762) much lower than 1, so the difference in R^2 between extraTrees and rf, and the following regressors, is high.

Pos.	Regressor	Rank	Time	Pos.	Regressor	Rank	Time
1	relaxo	3.20	1.044	20	glmStepAIC	21.60	1.802
2	nnls	4.40	1.106	21	krlsRadial	21.80	1.794
3	kknn	4.60	1.145	22	blackboost	22.00	1.799
4	pcr	6.20	1.176	23	icr	23.00	1.941
5	ridge	7.20	1.168	24	mlpWDml	23.20	1.904
6	lasso	8.00	1.223	25	pcaNNet	24.20	2.204
7	glmnet	8.20	1.251	26	avNNet	24.40	1.993
8	rqnc	10.00	1.343	27	xgbTree	24.60	2.097
9	M5	11.80	1.386	28	gaussprPoly	24.80	2.036
10	rpart	12.60	1.388	29	cforest	25.80	2.053
11	BstLm	13.40	1.430	30	extraTrees	26.40	2.206
12	gbm	13.60	1.514	31	bstSm	28.60	2.263
13	partDSA	15.00	1.495	32	bayesglm	29.00	2.349
14	ctree2	15.60	1.509	33	bstTree	32.80	2.692
15	mlpWD	15.80	1.562	34	bagEarth	34.00	2.755
16	rqlasso	17.00	1.623	35	randomGLM	34.20	3.311
17	spls	17.20	1.692	36	bag	35.00	3.080
18	svmRadial	17.40	1.577	37	qrf	35.20	3.005
19	plsRglm	18.20	1.624	38	xgbLinear	39.00	5.531

Table 15: Friedman ranking and average time (over the five fertility indices), in seconds, for each regressor (continued in Table 16).

Considering the speed of the different regressors, Tables 15 and 16 report the Friedman ranking of elapsed times spent by each regressor, and the average time, over all the datasets for each regressor, sorted by increasing rank. These times include the training and testing stages after the parameter tuning, without considering dependences of the regressors with the number of tunable hyperparameters and values. The fastest regressor is relaxo, followed by nnls

and kknn, which is in position 17 in the R^2 rank (Table 12). Besides, gbm is other regressor with good positions in the time and R^2 rankings (12 and 5, respectively), and its average time (1.514 s.) is only slightly slower than relaxo (1.044 s.). Even higher positions in the R^2 ranking correspond to times only slightly largers than the fastest regressors. Specifically, the best performing regressor (extraTrees), which is in position 30 of the time Friedman rank, spends an average of 2.206 s., which is only 2.11 times slower than the fastest regressor (relaxo). This difference can be considered small, so extraTrees is not only the most accurate regressor, but also very fast. Also note that, although the average times of most regressors are low (the 60 fastest regressors have average times below 20 s.), the last regressors of the table are very slow, e.g. RRF, dlkeras, Boruta and brnn (Bayesian regression neural network), with times above 100 s.,

Pos.	Regressor	Rank	Time	Pos.	Regressor	Rank	Time
39	bartMachine	39.20	5.634	58	penalized	58.20	16.781
40	evtree	40.20	7.008	59	dnn	58.80	17.713
41	SBC	40.60	6.491	60	glmboost	60.20	20.355
42	glm	42.60	9.554	61	elm	60.80	21.245
43	foba	43.60	10.010	62	elm-kernel	62.40	23.807
44	lm	43.80	9.888	63	rbf	63.00	27.254
45	rlm	45.20	10.373	64	grnn	63.60	27.725
46	rvmRadial	45.40	10.351	65	cubist	65.00	34.171
47	gaussprLinear	46.80	10.738	66	bdk	66.00	43.362
48	lars	48.20	10.885	67	spikeslab	67.00	45.654
49	enpls.fs	48.60	11.096	68	rf	68.00	54.220
50	simpls	48.60	11.040	69	svr	69.00	70.999
51	kernelpls	49.20	11.126	70	gamboost	70.00	78.804
52	gaussprRadial	49.60	11.200	71	RRF	71.00	114.762
53	superpc	51.20	11.659	72	dlkeras	72.00	193.679
54	gam	53.40	12.574	73	Boruta	73.00	513.723
55	earth	54.80	13.409	74	brnn	74.00	637.354
56	ppr	56.40	15.023	75	nodeHarvest	75.20	13078.245
57	treebag	56.60	14.879	76	qrnn	75.80	14109.732

Table 16: Continnuation of Table 15.

and nodeHarvest and qrnn, with times above 10,000 s.

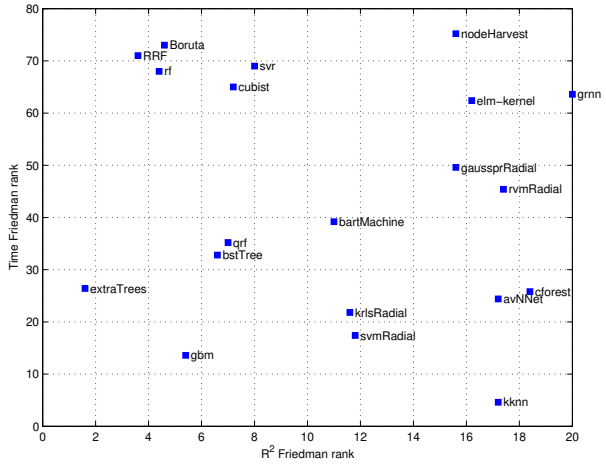

Figure 10: Friedman rank of the time (vertical axis) against the Friedman rank of the R^2 (horizontal axis) for the 20 best regressors over the five soil fertility indices.

Figure 10 plots the Friedman rank of the times against the Friedman rank of R^2 for the 20 best performing regressors in Table 12. ExtraTrees is placed on the left end of the plot (R^2 rank about 1-2), being the 7^{th} fastest in the plot after kknn, gbm, svmRadial, krlsRadial, avNNet and cforest, which however perform much worse than extraTrees. Among the other best regressors, rf, RRF and Boruta exhibit slightly lower R^2 than extraTrees, being much slower (time rank above 70). BstTree and qrf are slightly slower (higher time rank) than extraTrees, while svr, cubist and elm-kernel are much slower than extraTrees (time rank above 60-70) with much higher R^2 ranks (about 8 and 16, respectively).

7. Conclusions

Agriculture is a pillar of the Indian economy, but it is extremely dependend on factors such as soil quality, weather condition and fertilizer management. The application of the right fertilizers in the right amounts is a very relevant issue which requires detailed information about the fertility levels of several nutrients for each village (e.g., which nutrient is defficient or excessive). The

creation of maps for different nutrients with their fertility indices in different villages is very important for an adequate fertilizer application, to avoid soil degradation and to optimize the crop yield. The automatic prediction of these village-wise soil fertility indices for several nutrients, namely organic carbon, P_2O_5, Fe, Mn and Zn, would be very useful for the Indian Government in the creation of such fertility maps. Besides, the prediction of the fertility indices for these nutrients from measurements of soil N_2O, P_2O_5, K_2O, SO_4 and electrical conductivity, among others, would reduce the cost of the chemical analysis and save time for specialized technicians in the creation of fertility maps. The current paper uses regression techniques to automatically predict village-wise soil fertility indices for the previous nutrients in soils of the Indian state of Maharashtra. We compare 76 regressors belonging to 20 families including neural networks, deep learning, support vector regression, random forests, partial least squares, bagging and boosting, quantile regression and generalized additive models, among many others. Globally, the ensemble of extremely randomized regression trees (extraTrees) achieves the best performance, both in terms of root mean squared error (RMSE) and determination coefficient (R^2), followed by regularized random forest, random forests, and random forest with feature selection (Boruta), with similar performance but low speed. Other regressors with good performance are: gradient boosting of regression trees (bstTree) and generalized boosting regression (gbm); quantile random forest, M5 rule-based model with corrections based on nearest neighbors (cubist) and support vector regression (svr). The prediction quality achieves R^2 values between 0.57 and 0.70 (correlation values about 0.75-0.83), depending on the nutrient, which according to Colton (1974) correspond to relations between true and predicted fertility indices from very good to excellent. Rounding the values of fertility index to the standard fertility levels defined by the Indian Government (low, medium and high), these results corresponds to accuracy and Cohen kappa values in the ranges 79%-97% and 52%-81%, respectively, which can be considered as fairly accurate. The extraTrees also provide the best trade-off between performance and execution time, being in average just twice slower than the fastest

regressor in the whole collection. In the future work, we expect to collect additional data for those nutrients whose available patterns only belong to one fertility level, and to include the fertility indices of new nutrients of interest.

Funding

This research is supported by the Erasmus Mundus Euphrates programme [project number 2013-2540/001-001-EMA2], by the Xunta de Galicia (Centro singular de investigación de Galicia, accreditation 2016-2019) and by the European Union (European Regional Development Fund - ERDF).

References

Arunachalam, P., Kannan, P., Prabukumar, G., Govindaraj, M., 2013. Zinc deficiency in Indian soils with special focus to enrich Zinc in peanut. African J. Agri. Res. 8, 6681–6688.

Bair, E., Tibshirani, R., 2004. Semi-supervised methods to predict patient survival from gene expression data. PLoS Biol 2, 511–522.

Bates, D., Chambers, J., 1992. Linear models. Wadsworth & Brooks/Cole, Pacific Grove, CA. chapter 4.

Bertrand, F., Maumy-Bertrand, M., Meyer, N., 2014. Partial least squares regression for generalized linear models. R package version 1.1.1.

Bouma, J., 1989. Using soil survey data for quantitative land evaluation. Adv. in Soil Sci. 9, 177–213.

Breiman, L., 1996. Bagging predictors. Machine Learning 24, 123–140.

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and regression trees. Wadsworth and Brooks, Monterey, CA.

Buehlmann, P., Hothorn, T., 2007. Boosting algorithms: regularization, prediction and model fitting. Stat. Sci. 22, 477–505.

Buehlmann, P., Yu, B., 2003. Boosting with the L2 loss: regression and classification. J. Am. Stat. Assoc. 98, 324–339.

Cannon, A., 2011. Quantile regression neural networks: implementation in R and application to precipitation downscaling. Computers & Geosciences 37, 1277–1284.

Chang, C., Lin, C., 2011. LIBSVM: a library for support vector machines. ACM Trans. Intel. Syst. and Technol. 2, 27:1–27:27.

Chiu, S., 1996. Method and software for extracting fuzzy classification rules by subtractive clustering, in: Fuzzy Inf. Proc. Soc., NAFIPS, pp. 461–465.

Chollet, F., 2015. Keras. `https://keras.io`.

Chun, H., Keles, S., 2010. Sparse partial least squares for simultaneous dimension reduction and variable selection. J. R. Stat. Soc. 72, 3–25.

Colton, T., 1974. Statistical in medicine. Little Brown and Co., New York, NJ.

Research Council, N., 1989. Alternative agriculture. Technical Report. National Academy of Sciences. Washington DC.

Department of Agriculture & Cooperation, 2011. Methods manual. Soil testing in India. Technical Report. Ministry of Agriculture, Government of India. New Delhi, India.

Dobson, A., 1990. An introduction to generalized linear models. Chapman and Hall, London, UK.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., 2004. Least angle regression. Ann. Stat. 32, 407–499.

Foresee, F., Hagan, M.T., 1997. Gauss-Newton approximation to Bayesian regularization, in: Intl. J. Conf. on Neural Netw., pp. 1930–1935.

Friedman, J., 1991. Multivariate adaptive regression splines. Ann. Stat. 19, 1–141.

Friedman, J., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 29, 1189–1232.

Friedman, J., Stuetzle, W., 1981. Projection pursuit regression. J. Am. Stat. Assoc. 76, 817–823.

García, S., Fernández, A., Benítez, A., Herrera, F., 2007. Statistical comparisons by means of non-parametric tests: a case study on genetic based machine learning, in: Proc. of the II Congreso Español de Informática (CEDI 2007), pp. 95–104.

Gelman, A., Jakulin, A., Pittau, M., Su, Y., 2009. A weakly informative default prior distribution for logistic and other regression models. Ann. Appl. Stat. 2, 1360–1383.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63, 3–42.

Goeman, J., 2010. L-1 penalized estimation in the cox proportional hazards model. Biometrical J. 52, 70–84.

Grubinger, T., Zeileis, A., Pfeiffer, K., 2014. Evtree: evolutionary learning of globally optimal classification and regression trees in R. J. Stat. Softw. 61, 1–29.

Gruhn, P., Goletti, F., Yudelman, M., 2000. Integrated nutrient management, soil fertility, and sustainable agriculture: current issues and future challenges. Intl. Food Policy Res. Inst.

Hainmueller, J., Hazlett, C., 2013. Kernel regularized least squares: reducing misspecification bias with a flexible and interpretable machine learning approach. Political Analysis 22, 143–168.

Hechenbichler, K., Schliep, K., 2004. Weighted k-nearest-neighbor techniques and ordinal classification. Technical Report. Ludwig-Maximilians University Munich.

Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554.

Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. J. of Comp. and Graph. Stat. 15, 651–674.

Huang, G.B., Zhou, H., Ding, X., Zhang, R., 2012. Extreme learning machine for regression and multiclass classification. IEEE Trans. Systs., Man, and Cybern.-Part B: Cybern. 42(2), 513–529.

Huber, P., 1981. Robust statistics. Wiley, New York, NJ.

Hyvarinen, A., Oja, E., 2000. Independent component analysis: algorithms and applications. Neural netw. 13, 411–430.

Ishwaran, H., Rao, J., Kogalur, U., 2010. Spikeslab : prediction and variable selection using spike and slab regression. The R Journal 2, 68–73.

Jain, S.K., Singh, V.P., Genuchten, M.T.V., 2004. Analysis of soil water retention data using artificial neural networks. J. Hydrol. Eng. , 415–420.

Jia, H.Y., Chen, J., Yu, H.L., Liu, D.Y., 2010. Soil fertility grading with Bayesian network transfer learning, in: Proc. Intl. Conf. on Mach. Learn. and Cybern., Qingdao, China. pp. 1159–1163.

Jong, S., 1993. SIMPLS: an alternative approach to partial least squares regression. Chemom. and intel. lab. systs. 18, 251–263.

Jong, S., 1994. Comment on the PLS kernel algorithm. J. Chemom. 8, 169–174.

Kapelner, A., Bleich, J., 2016. bartMachine: machine learning with Bayesian additive regression trees. J. of Stat. Softw. 70, 1–40.

Katyal, J., Rattan, R., 2003. Secondary and micronutrients: research gaps and future needs. Fertil. News 48, 9–20.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Intl. J. Conf. on Artif. Intel. (IJCAI), Montreal (Canada). pp. 1137–1143.

Kuhn, M., 2016. Caret: Classification and regression training. URL: `https://CRAN.R-project.org/package=caret`. R package version 6.0-70.

Kursa, M., Rudnicki, W., 2010. Feature selection with the Boruta package. J. Stat. Softw. 36, 1–13.

Lamorski, K., Pachepsky, Y., Slawiński, C., Walczak, R.T., 2008. Using support vector machines to develop pedotransfer functions for water retention of soils in Poland. Soil Sci. Soc. Am. J. 72, 1243–1247.

Lawson, C., Hanson, R., 1995. Solving least squares problems. volume 15 of *Classics in Appl. Math.* Soc. for Industrial and Appl. Math. (SIAM), Philadelphia, PA.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F., 2017. A survey of deep neural network architectures and their applications. Neurocomputing 234, 11–26.

MacKay, D., 1992. Bayesian interpolation. Neural Computation 4, 415–447.

Matlab, 2011. version 7.12 (R2011a). Natick (MA). URL: `https://es.mathworks.com/help/nnet`.

Meinshausen, N., 2007. Relaxed lasso. Comput. Stat. and Data Anal. , 374–393.

Meinshausen, N., 2010. Node harvest. Ann. Appl. Stat. 4, 2049–2072.

Melssen, W., Wehrens, R., Buydens, L., 2006. Supervised Kohonen networks for classification problems. Chemom. Intell. Lab. Syst. 83, 99–113.

Mevik, B., Cederkvist, H., 2004. Mean squared error of prediction (MSEP) estimates for principal component regression (PCR) and partial least squares regression (PLSR). J. Chemom. 18, 422–429.

Minasny, B., McBratney, A.B., Bristow, K.L., 1999. Comparison of different approaches to the development of pedotransfer functions for water-retention curves. Geoderma 93, 225–253.

Mizera, I., Koenker, R., 2014. Convex optimization in R. J. Stat. Softw. 60, 1–23.

Molinaro, A., Lostritto, K., van der Laan, M., 2010. PartDSA: deletion/substitution/addition algorithm for partitioning the covariate space in prediction. Bioinformatics 26, 1357–1363.

Mucherino, A., Papajorgji, P., Pardalos, P., 2009. A survey of data mining techniques applied to agriculture. Oper. Res. 9, 121–140.

Muhr, G., Datta, N., Shankara, S., Dever, F., Lecy, V., Donahue, R., 1965. Soil testing in india. U.S. Agency for International Development, Mission to India.

Obade, V., Lal, R., 2016. Towards a standard technique for soil quality assessment. Geoderma 265, 96–102.

Pachepsky, Y., Rajkai, K., Tóth, B., 2015. Pedotransfer in soil physics: trends and outlook - a review. Agrokémia és talajtan 64, 339–360.

Platt, J., 1998. Sequential minimal optimization: a fast algorithm for training support vector machines. Technical Report MSR-TR-98-14. Microsoft Research.

Quinlan, R., 1992. Learning with continuous classes, in: Proc. Australian J. Conf. on Artif. Intel., pp. 343–348.

Quinlan, R., 1993. Combining instance-based and model-based learning, in: Proc. Intl. Conf. on Mach. Learn., pp. 236–243.

R Core Team, 2008. R: A language and environment for statistical computing. Vienna, Austria. URL: https://www.R-project.org. ISBN 3-900051-07-0.

Rammoorthy, B., Bajaj, J., 1969. Available N, P and K status of Indian soils. Fertilizer news 14(8), 24–26.

Reeves, D., 1997. The role of soil organic matter in maintaining soil quality in continuous cropping systems. Soil and Tillage Research 43, 131–167.

Ripley, B., 2002. Modern applied statistics with S. Springer, New York, NJ.

Sheela, P., Sivaranjani, K., 2015. A brief survey of classification techniques applied to soil fertility prediction, in: Int. Conf. Eng. Trends in Sci. and Hum., pp. 80–83.

Simon, N., Friedman, J., Hastie, T., Tibshirani, R., 2011. Regularization paths for cox's proportional hazards model via coordinate descent. J. Stat. Softw. 39, 1–13.

Sirsat, M., Cernadas, E., Fernández-Delgado, M., Khan, R., 2017. Classification of agricultural soil parameters in India. Comput. Electron. Agric. 135, 269–279.

Song, L., Langfelder, P., Horvath, S., 2013. Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics 14, 1–22.

Specht, D., 1991. A general regression neural network. IEEE Trans. on Neural Netw. 2, 568–576.

Terhoeven-Urselmans, T., Spaargaren, O., Shepherd, K.D., 2010. Prediction of soil fertility properties from a globally distributed soil mid-infrared spectral library. Nutrient Management and Soil and Plant Analysis 74, 1792–1799.

Tipping, M., 2001. Sparse Bayesian learning and the relevance vector machine. J. Mach. Learn. Res. 1, 211–244.

Viera, A., Garrett, J., 2005. Understanding interobserver agreement: the kappa statistic. Family Med. 37(5), 360–363.

Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics Bulletin 1, 80–83.

Wood, S., 2011. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. J. R. Stat. Soc. 1, 3–36.

Xiao, N., Cao, D., Li, M., Xu, Q., 2016. Enpls: an R package for ensemble partial least squares regression. arXiv preprint .

Zell, A., 1998. SNNS Stuttgart Neural Network Simulator User Manual, Version 4.2. Technical Report. IPVR, University of Stuttgart and WSI, University of Tübingen.

Zhang, T., 2011. Adaptive forward-backward greedy algorithm for learning sparse representations. IEEE Trans. Inf. Theor. 57, 4689–4708.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. 67, 301–320.

Zou, H., Li, R., 2008. One-step sparse estimates in nonconcave penalized likelihood models. Ann. Stat. 36, 1509–1533.