



UNIVERSIDADE DE SANTIAGO DE COMPOSTELA  
Centro de Investigación en Tecnoloxías da Información (CiTIUS)

Tesis doctoral

**APPLICATION OF MACHINE LEARNING TO AGRICULTURAL  
SOIL DATA**

Presentada por:

**Manisha Sanjay Sirsat**

Dirigida por:

**Eva Cernadas García**

**Manuel Fernández Delgado**

Santiago de Compostela, Julio de 2017



**Dr. Eva Cernadas García**, Profesora Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

**Dr. Manuel Fernández Delgado**, Profesor Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

**HACEN CONSTAR:**

Que la memoria titulada **APPLICATION OF MACHINE LEARNING TO AGRICULTURAL SOIL DATA** ha sido realizada por **Manisha Sanjay Sirsat** bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información de la Universidad de Santiago de Compostela, y constituye la Tesis que presenta para optar al título de Doctora.

Santiago de Compostela, Julio de 2017

**Eva Cernadas García**  
Directora de la tesis

**Manuel Fernández Delgado**  
Co-director de la tesis

**Manisha Sanjay Sirsat**  
Autora de la tesis



*Education is the manifestation of the perfection  
already in man.*

Swami Vivekananda



## Acknowledgments

I would like to express my gratitude and respect for my thesis advisors, Dr. Eva Cernadas García and Dr. Manuel Fernández Delgado for their guidance and support which contributed greatly to the timely completion of this project.

I also extend my gratitude to Centro de Investigación en Tecnoloxías da Información (CiTIUS), University of Santiago de Compostela. My gratitude also goes to Dr. Razaullah Khan and Dr. Balaji Aglave for their advise and motivation.

I deeply thank to my family Sakharam, Antika and Sanjay for their heart-warming kindness, timely encouragement, and endless patience. Finally, I am thankful to Erasmus Mundus Euphrates programme [project number 2013-2540/001-001-EMA2] for giving me wonderful opportunity to pursue my research work internationally.

Santiago de Compostela, July 2017



# Resumen

El sector agrícola constituye uno de los principales pilares de la economía en la India, representando en torno al 47% del empleo (datos de 2013–2014) y el 11% del producto interior bruto de este país, que dedica el 60% de su territorio a labores agrícolas y ganaderas. El funcionamiento de este sector económico está condicionado por factores climáticos, así como por las prácticas agrícolas y las características del suelo. De hecho, la sobre-explotación y el uso excesivo, o incorrecto, de fertilizantes afecta a un porcentaje significativo del suelo agrícola indio, dos tercios del cual se puede calificar como degradado. La medición de parámetros del suelo tales como los índices de fertilidad de las diferentes localidades para distintos nutrientes, tipo de suelo, acidez y niveles de nutrientes, entre otros, resulta fundamental de cara a planificar el uso de fertilizantes y la selección de cultivos, pero requiere un considerable esfuerzo por parte de personal especializado en análisis químicos. Esta tesis evalúa la capacidad de los métodos de aprendizaje automático (*machine learning*) para la predicción de estos parámetros a partir de datos químicos del suelo publicados por los gobiernos de la India y del estado de Maharashtra. Estos datos permiten la predicción de los siguientes 12 parámetros del suelo de interés para la agricultura:

- 1-6. Índices locales de fertilidad para 6 nutrientes del suelo: carbono orgánico ( $OC$ ), pentóxido de fósforo ( $P_2O_5$ ), óxido de potasio ( $K_2O$ ), hierro ( $Fe$ ), manganeso ( $Mn$ ) y cinc ( $Zn$ ). Para un nutriente y para cada terreno agrícola de la localidad (que se corresponde con un patrón para las tareas de aprendizaje automático), se cuantifica el nivel del nutriente, que es un dato de tipo numérico, en bajo, medio o alto, usando para ello los niveles definidos por el gobierno de la India. El índice local de fertilidad se calcula promediando de forma ponderada el número de terrenos de la localidad con niveles bajos, medios y altos ( $N_l$ ,  $N_m$  y  $N_h$ ) con pesos 1, 2 y 3 respectivamente, mediante la fórmula  $(N_l + 2N_m + 3N_h)/N_t$ , sendo  $N_t$  el número total de terrenos en la localidad. Este índice

es único para cada localidad, de modo que todos los terrenos de una localidad tienen el mismo índice de fertilidad.

- 7-9. Niveles de nutrientes del suelo: nitrógeno, fósforo y potasio ( $N$ ,  $P$  y  $K$ ), presentes en forma de óxido nitroso ( $N_2O$ ), pentóxido de di-fósforo ( $P_2O_5$ ) y óxido de potasio ( $K_2O$ ) respectivamente. Estos niveles resultan necesarios para la recomendación de fertilizantes y sus cantidades adecuadas.
10. Acidez del suelo o  $pH$ , definido como el logaritmo de la concentración de iones  $H^+$ , que para los datos disponibles adopta valores entre 6 y 9 aproximadamente. Valores inferiores (resp. superiores) a 7 se consideran ácidos (resp. básicos o alcalinos).
11. Cultivo recomendado para el terreno bajo consideración. Los cultivos considerados en los datos disponibles son *bajra*, algodón de regadío y seco, o soja.
12. Tipo de suelo, otro dato también categórico que en los datos disponibles puede adoptar los valores: “ligero”, correspondiente a un suelo arenoso, con capacidad para absorber agua y niveles bajos de los nutrientes; o “medio”, correspondiente a un suelo fértil, con alto contenido de humus, adecuado para el cultivo.

En una primera aproximación (capítulo 2), hemos cuantificado los valores de estos parámetros, la mayoría de los cuales son de carácter numérico, usando niveles estándar definidos por el gobierno de la India, con el objetivo de transformar la predicción de sus valores en problemas de clasificación. Los índices para  $K_2O$  y  $Zn$  no se han podido tratar como problemas de clasificación ya que, dados los rangos considerados para cada clase, los patrones disponibles para ambos problemas pertenecen a una única clase. Para resolver estos problemas de clasificación hemos aplicado una colección de 20 técnicas, implementadas en C++, Matlab, R y Weka, pertenecientes a las siguientes 7 familias:

1. Árboles de decisión: específicamente, se incluyen en esta familia los clasificadores J48, *random tree*, *recursive partitioning* y el ensemble *Decorate*, integrado por clasificadores base del tipo J48.
2. Clasificadores basados en reglas, incluyendo el clasificador híbrido DTNB (*decision table-naïve Bayes*) y RIPPER (*repeated incremental pruning to produce error reduction*).

3. *Ensembles* de tipo *bagging*, con varios clasificadores base: árboles de decisión, clasificadores FDA (*flexible discriminant analysis*) y PART (*pruned partial C4.5 decision trees*).
4. *Ensembles* de tipo *adaboost*, usando el algoritmo *Adaboost.M1*.
5. Clasificador *K-nearest neighbors* (KNN).
6. Redes neuronales, incluyendo *extreme learning machine* (ELM) y ELM con núcleo gaussiano, *multi-layer perceptron* (MLP), *radial basis function* (RBF) *neural network*, y *probabilistic neural network* (PNN).
7. *Support vector machine* (SVM) con núcleo gaussiano.
8. *Random forest* con árboles de decisión del tipo *random tree*, y *rotation forest* ensemble con clasificadores base del tipo J48.

La metodología experimental emplea 4 grupos formados cada uno por tres conjuntos (de entrenamiento, validación y test), con porcentajes del 50%, 25% y 25% de los patrones, respectivamente. La medida de calidad empleada en los experimentos de clasificación es el parámetro Kappa ( $\kappa$ ) de Cohen, que evalúa el rendimiento descartando la probabilidad de acierto por casualidad. Para cada clasificador, se realiza un entrenamiento por cada combinación de valores de sus hiperparámetros sintonizables y para cada uno de los 4 conjuntos de entrenamiento. El clasificador entrenado se valida sobre el conjunto de validación correspondiente. Para cada combinación de valores de los hiperparámetros sintonizables, se promedia el  $\kappa$  sobre los 4 conjuntos de validación. La combinación de valores seleccionada es la que alcanza el mayor  $\kappa$  promedio. El  $\kappa$  final obtenido por el clasificador es el promedio, sobre los 4 conjuntos de test, del clasificador entrenado en los 4 conjuntos de entrenamiento con la combinación seleccionada para los valores de sus hiperparámetros.

El clasificador random forest alcanza los mejores resultados para 6 de los 10 problemas de clasificación (índices locales de fertilidad para *OC* y *Fe*, nutrientes *N<sub>2</sub>O* y *K<sub>2</sub>O*, *pH* y cultivo recomendado). En los restantes problemas, random forest ha estado muy cerca de ser el mejor, ya que en todos los casos supera el 90% del mejor  $\kappa$  alcanzado por algún otro clasificador. En estos problemas, los mejores resultados son los de adaboost, en 2 problemas (índice local de fertilidad y nivel de *P<sub>2</sub>O<sub>5</sub>*), SVM y Decorate (ambos en un problema). Se han constatado importantes diferencias entre los resultados en la clasificación de los distintos parámetros, ya

que  $\kappa$  supera el 60% para los índices locales de fertilidad (90% para  $OC$  y 85% para  $P_2O_5$ ), tipo de suelo (97%) y cultivo recomendado (88%); alcanza valores intermedios (69% y 65%) para los índices de fertilidad de  $Fe$  y  $Mn$ ; y es inferior al 35% para los nutrientes  $N_2O$ ,  $P_2O_5$  y  $K_2O$ . Dado que las clases se corresponden a niveles de cuantificación de parámetros del suelo, estos problemas constituyen ejemplos de clasificación ordinal, en los que existe una relación de orden entre las clases. Esto se manifiesta en las matrices de confusión de los mejores clasificadores en cada problema, donde los errores cometidos por asignar patrones a clases contiguas son más elevados que los correspondientes a clases no contiguas. También hemos comprobado que, para algunos parámetros (índices locales de fertilidad para  $N_2O$ ,  $Mn$  y  $Fe$ ), algunos clasificadores entrenados con datos de una región de Maharashtra resultan válidos para clasificar índices locales de fertilidad de otras regiones del mismo estado, es decir, que existe una cierta compatibilidad entre regiones. Específicamente, entrenando y testeando con datos de las regiones de North-Maharashtra y Paschim-Maharashtra respectivamente, bagging alcanza buenos resultados para índices de fertilidad de  $P_2O_5$  y  $Mn$ , y lo mismo ocurre intercambiando las regiones de entrenamiento y test. El árbol de decisión J48 y SVM alcanzan también buenos resultados entrenando y testeando con cualquiera de las dos regiones anteriores. Asimismo, random forest funciona bien entrenando y testeando con Marathwada y Paschim-Maharashtra respectivamente. También existen compatibilidades bi-direccionales parciales entre Marathwada y Paschim-Maharashtra para la clasificación de  $pH$ , usando los clasificadores rotation forest y KNN, y con el índice de fertilidad de  $Fe$  usando árboles de decisión del tipo *recursive partitioning*.

La predicción directa de valores numéricos para los parámetros del suelo, más allá de la clasificación de sus valores cuantificados, se ha realizado (capítulo 3) empleando una colección de 76 técnicas de regresión, implementadas en su mayoría en R, con 2 técnicas en Matlab, una en Python y otra en C++, y pertenecientes a las siguientes 20 familias (para cada familia se indican sus regresores):

1. Regresión lineal: incluye los regresores *linear model* (lm) y *robust linear model* (rlm).
2. *Generalized linear models* (GLM): incluye el modelo clásico (glm), *penalized GLM* (penalized), *elastic-net regularized GLM* (glmnet) y *GLM with stepwise feature selection using Akaike Information Criterion* (glmStepAIC).
3. *Least squares*: incluye *non-negative least squares* (nnls) y *radial basis function kernel regularized least squares regression* (krlsRadial).

4. *Partial least squares* (PLS): incluye *sparse partial least squares* (splS); PLS usando el método SIMPLS (simplS); PLS usando el algoritmo kernel (kernelpls); ensemble de regresores base del tipo splS (enpls.fs) y *partial least squares GLM* (plsRglm).
5. Regresión mediante *least absolute shrinkage and selection operator* (lasso) y *relaxed lasso* (relaxo).
6. Regresión *ridge*, usando el algoritmo LARS-EN (ridge), regresión *spike and slab* (spikeslab) y *ridge regression with forward, backward and sparse input selection* (foba).
7. Redes neuronales: perceptrón multicapa con *weight decay* usando una o varias capas ocultas (mlpWeightDecay y mlpWeightDecayML respectivamente); ensemble de perceptrones multicapa con diferentes inicializaciones (avNNet); redes de base radial (rbf); *generalized regression neural network* (grnn); *extreme learning machine* (elm) y elm con núcleo gaussiano (elm-kernel); perceptrón multicapa con PCA (pcaNNet); *supervised bi-direccional Kohonen network* (bdk).
8. Redes neuronales convolucionales (*deep learning*), usando la implementación en Python proporcionada por la librería Keras (dlkeras) y la versión en R proporcionada por el paquete deepnet (dnn).
9. Máquinas de vectores de soporte: *support vector regression* con núcleo gaussiano en C++ usando LibSVM (svr) y en R usando el paquete kernlab (svmRadial); y *relevance vector machine* con núcleo gaussiano (rvmRadial).
10. Árboles de regresión: *recursive partitioning* (rpart), *simple interpretable tree-based ensemble with sparse results for high-dimensional regression* (nodeHarvest); *model tree* (M5); *conditional inference tree* (ctree2); *partitioning using deletion, substitution and addition* (partDSA); y *model tree with evolutionary algorithms* (evtrees).
11. *Bagging ensembles* variando el regresor base: *conditional inference regression trees* (bag); *multi-variate adaptive regression splines* (bagEarth) y árboles CART (treebag).
12. *Boosting ensembles* y *gradient boosted machines*: incluye boosting con regresores base del tipo GLM (randomGLM); *gradient boosted machine* con regresores base lineales (BstLm), *splines* interpolantes (bstSm), árboles de decisión (bstTree), GLM (glmboost) y *conditional inference regression trees* (blackboost); *generalized boosting regression*

*model* (gbm); y *extreme gradient boosting* con regresores lineales (xgbLinear) y árboles de regresión (xgbTree).

13. *Random forests*: incluye el modelo original (rf); random forest con selección de características (Boruta); *regularized random forest* (RRF); random forest de *conditional inference trees* (cforest); *quantile regression forest* (qrf); y *extremely randomized regression trees* (extraTrees).
14. Modelos basados en prototipos: regresión de  $k$  vecinos más cercanos (kkn) y modelo M5 basado en reglas con correcciones basadas en vecinos más cercanos (cubist).
15. Regresión bayesiana: GLM bayesiano (bayesglm); *Bayesian regularized neural network* (brnn); y *Bayesian additive regression tree* (bartMachine).
16. *Principal component Analysis* (PCA): regresión mediante componentes principales (pcr); regresión mediante componentes independientes (icr); y PCA supervisado (superpc).
17. *Generalized additive models* (gam) y *boosted generalized additive model* (gamboost).
18. Procesos gaussianos: con núcleos lineal (gaussprLinear), gaussiano (gaussprRadial) y polinómico (gaussprPoly).
19. Regresión cuantil: regresión cuantil con penalización lasso (rqlasso); *non-convex penalized quantile regression* (rqnc); y *quantile regression neural network* (qrnn).
20. Otros métodos: *least angle regression* (lars); *multi-variate adaptive regression splines* (earth); *projection pursuit regression* (ppr); y *subtractive clustering and fuzzy C-means rules* (sbc).

Esta colección de regresores ha sido evaluada inicialmente sobre una colección de 66 problemas de regresión seleccionados del UCI *machine learning repository*. El número de entradas y de patrones de estos problemas varían en los rangos 7-439 y 60-2000 respectivamente. En los problemas con más de 2000 patrones sólo se consideran los 2000 primeros patrones, en un intento de evitar conjuntos de datos grandes, dada la extrema lentitud de muchos regresores. Por otra parte, todas las entradas constantes o colineales han sido suprimidas para evitar los errores de ejecución experimentados por algunos regresores en tales casos. La ejecución de todos los regresores sobre todos los conjuntos de datos representa un total de 5016 experimentos. De ellos, tan sólo 46 generaron errores debido a un consumo excesivo

de memoria RAM (más de 128 GB) o de tiempo (más de 150 horas, correspondientes a 6.25 días), lo cual representa un 0.92% del total de experimentos. Las medidas de calidad empleadas son el error cuadrático medio (*root mean squared error*, denotado como RMSE) y el coeficiente de correlación de Pearson, que ofrece una medida absoluta de la calidad en la predicción y, al encontrarse en el mismo rango (-1,+1) para todos los problemas, admite el promediado sobre los mismos, lo cual no ocurre con el RMSE.

Los mejores resultados son obtenidos por la red neuronal extreme learning machine con núcleo gaussiano (elm-kernel), que en el 86.4% de los problemas considerados supera el 90% de la correlación máxima obtenida por algún regresor de la colección. Un tipo de random forest, llamado extremely randomized regression trees (extraTrees), y support vector regression (svr) superan dicho 90% en el 84.8% de los problemas, mientras regularized random forest (RRF) se sitúa en el 81.8%. La mejor correlación promedio sobre todos los conjuntos de datos es relativamente baja (0.79), lo cual denota la complejidad de los datos a predecir, ya que se considera que una predicción aceptable requiere una correlación superior a 0.9. Sin embargo, en 40 de 66 problemas (que representa el 60.6% de los problemas considerados) la mejor correlación obtenida por algún regresor supera 0.9, y sólo en 17 problemas (que representa el 25.7% del total) la correlación es inferior a 0.7, lo cual significa que para una buena parte de los problemas algún regresor obtuvo buenos resultados.

Desde el punto de vista de las familias de regresores, las 10 primeras posiciones corresponden a redes neuronales, SVM, modelos basados en prototipos (cuyo regresor cubist, basado en reglas del tipo M5 con correcciones basadas en vecinos más cercanos, figura en la 3ª posición), random forests, ensembles del tipo *boosting* y *generalized linear regression*, cuyo regresor penalized figura en la posición 9 y es el que obtiene la mejor correlación en más problemas (16), seguido por SVM (en 11 ocasiones), extraTrees (9 veces), cubist (6 veces), avNNet (5 veces) y *quantile regression forest* (4 veces). Otras familias de regresores cuyo mejor regresor figura entre las posiciones 10 y 20 son los ensembles de tipo *bagging* (posición 12), modelos bayesianos (posición 13), árboles de regresión (15), *deep learning* (18), procesos gaussianos (19) y *least squares* (20). Los regresores pertenecientes a las restantes familias alcanzan posiciones posteriores a 20.

En cuanto a los tiempos de ejecución, los experimentos revelan que las técnicas más rápidas ofrecen poca precisión, ya que ningún regresor está simultáneamente en las primeras posiciones de los rankings de Friedman de correlaciones y de tiempos. No obstante, algunas de las técnicas más precisas son relativamente rápidas, a pesar de no figurar entre las más

rápidas de la colección. Es el caso de *extraTrees*, *cubist*, *random forest* y *generalized boosted regression model* (*gbm*), situadas en las 10 primeras posiciones en el ranking de correlación y en posiciones medias del ranking de tiempos (25, 48, 51 y 37 respectivamente).

Esta colección de regresores se ha empleado para predecir los valores numéricos de los parámetros del suelo descritos anteriormente. En este caso, al tratarse de métodos de regresión, se han suprimido el cultivo recomendado y tipo de suelo, ya que al poseer salidas discretas son intrínsecamente problemas de clasificación. Sin embargo, entre los problemas de regresión se han incluido los índices locales de fertilidad para  $K_2O$  y  $Zn$ , que no se consideraron en el capítulo 2 porque los datos disponibles sólo pertenecen a uno de los niveles de cuantificación (clases) definidos por el gobierno de la India. La metodología experimental y medidas de calidad son las mismas que en el capítulo 3.

A diferencia de los experimentos con los datos de la UCI, en los datos de suelo el regresor *extraTrees*, que ya obtenía buenos resultados de correlación y velocidad, alcanza los mejores resultados en 7 de 10 problemas, y en los restantes funciona casi tan bien como el mejor regresor, superando el 90% de la correlación máxima. De hecho, el ranking de Friedman de los regresores sobre todos los problemas sitúa a *extraTrees* en primera posición con un valor de 2.3, lo cual significa que en promedio este regresor figura entre las posiciones 2 y 3 de la colección de 76 regresores. Sin embargo, en términos absolutos la mejor correlación promedio sobre todos los problemas, obtenida por *extraTrees*, es relativamente reducida (0.68), lo cual refleja la complejidad del problema de predicción abordado. Los 5 mejores regresores en términos de correlación pertenecen a la familia *random forest*: *extraTrees*, *regularized random forest*, *random forest*, *quantile regression forest* y *Boruta*. La *svr* y dos regresores de la familia *gradient boosted machines*, *generalized boosted regression model* (*gbm*) y *gradient boosting with regression trees* (*bstTree*) ocupan las siguientes posiciones. Los resultados son peores para *elm-kernel*, que se sitúa en las posiciones 10 y 14 en términos de RMSE y correlación respectivamente, aunque su correlación media (0.62) no es muy inferior a la obtenida por *extraTrees*. Considerando conjuntamente la correlación y la velocidad de los regresores, *extraTrees* proporciona el mejor compromiso entre ambos factores, ya que alcanza el mínimo ranking de Friedman de correlaciones, y el segundo menor ranking de Friedman de tiempos entre los 20 regresores más precisos.

En un análisis individualizado para cada uno de los 10 problemas de regresión, se aprecia una cierta correspondencia con los resultados de las técnicas de clasificación (capítulo 2). En la clasificación de índices locales de fertilidad de *OC* y *Fe*, donde los mejores  $\kappa$  alcanzan

90.65% y 69.35% respectivamente, las mejores correlaciones obtenidas en la regresión son superiores a 0.8. También en el índice local de  $Zn$  la correlación supera 0.8, pero este índice no se considera en la clasificación, por pertenecer todos los patrones disponibles a la misma clase. En los índices de fertilidad de  $P_2O_5$  y  $Mn$ , y en el  $pH$ , donde  $\kappa$  alcanza 85.54%, 64.8% y 47.32% respectivamente, las correlaciones son intermedias (0.776, 0.758 y 0.694 respectivamente). El índice local de  $K_2O$  también es intermedio (0.631), pero este índice tampoco se considera en la clasificación, por la misma razón que el índice de  $Zn$ . Y, finalmente, para los nutrientes  $N_2O$ ,  $P_2O_5$  y  $K_2O$ , donde en la clasificación  $\kappa$  alcanza valores de 33.6%, 35.08% y 31.85% respectivamente, en la regresión se alcanzan las correlaciones más bajas (0.519, 0.487 y 0.517, respectivamente). Estos resultados confirman la dificultad de los problemas analizados, ya que las mejores correlaciones no alcanzan 0.9, y con los tres nutrientes  $N_2O$ ,  $P_2O_5$  y  $K_2O$  no superan 0.6. Se puede decir que la predicción es aceptable sólo en los índices locales de fertilidad de  $OC$ ,  $Fe$  y  $Zn$ , donde la correlación supera 0.8. En el resto de los problemas, la incertidumbre en la predicción es más elevada. Sin embargo, si no resulta imprescindible una aproximación precisa de los parámetros del suelo, las técnicas empleadas proporcionan una predicción aproximada para los valores numéricos mediante técnicas de regresión, y para los valores cuantificados mediante técnicas de clasificación.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Soil fertility index . . . . .	3
1.2	Fertilizer nutrients . . . . .	4
1.3	Soil reaction . . . . .	4
1.4	Cropping cycle . . . . .	5
1.5	Soil texture . . . . .	6
1.6	Related work . . . . .	6
1.7	Outline of the work . . . . .	7
<b>2</b>	<b>Application of classification methods to agricultural soil data</b>	<b>9</b>
2.1	Village-wise $OC$ , $P_2O_5$ , $Mn$ and $Fe$ fertility indices . . . . .	9
2.2	Soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$ . . . . .	10
2.3	Soil $pH$ . . . . .	12
2.4	Crop selection . . . . .	12
2.5	Soil type . . . . .	13
2.6	Classification methods . . . . .	15
2.7	Experimental setup . . . . .	21
2.8	Global discussion of the results . . . . .	22
2.9	Classification of village-wise $OC$ , $P_2O_5$ , $Mn$ and $Fe$ fertility indices . . . . .	26
2.10	Classification of soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$ . . . . .	27
2.11	Classification of soil $pH$ . . . . .	27
2.12	Classification of crop . . . . .	28
2.13	Classification of soil type . . . . .	28
2.14	Comparison among regions . . . . .	29

<b>3</b>	<b>Application of regression methods to general datasets</b>	<b>31</b>
3.1	Regression methods . . . . .	31
3.2	Experimental setup . . . . .	46
3.3	Results and discussion . . . . .	49
<b>4</b>	<b>Application of regression methods to agricultural soil data</b>	<b>65</b>
4.1	Regression problems . . . . .	66
4.2	Prediction of $OC$ , $P_2O_5$ , $K_2O$ , $Fe$ , $Mn$ and $Zn$ village-wise fertility indices . .	68
4.3	Prediction of soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$ . . . . .	74
4.4	Prediction of soil $pH$ . . . . .	76
4.5	Global discussion . . . . .	78
<b>5</b>	<b>Conclusions</b>	<b>85</b>
	<b>Bibliography</b>	<b>89</b>
	<b>List of Figures</b>	<b>101</b>
	<b>List of Tables</b>	<b>103</b>

## CHAPTER 1

# INTRODUCTION

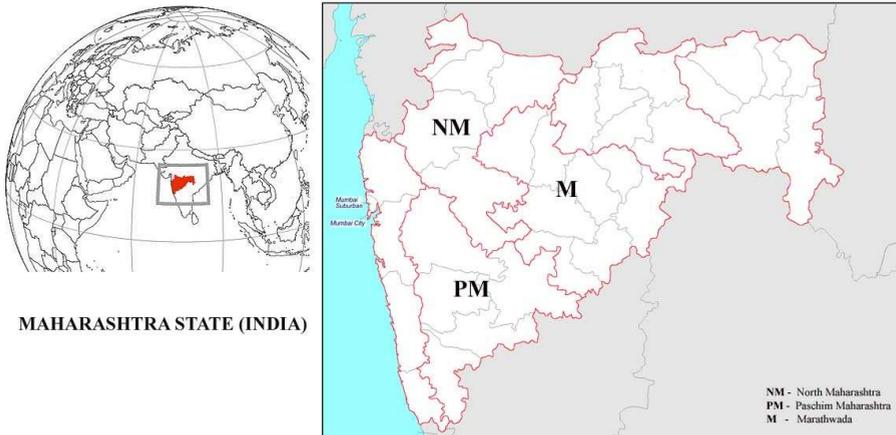
India is second largest country in the farm production and seventh largest in agricultural export [30]. Agriculture is the pillar of the Indian economy, offering 47% employment of total population in 2013-14 and having significant contribution in global food basket. India is the fastest growing exporter of agricultural products over a decade to more than 100 countries, mainly in the Middle East, Southeast Asia, countries of the South Asian Association for Regional Cooperation (SAARC), the European Union and the United States [128]. According to data of year 2011, India devotes 60.5% of its land<sup>1</sup> to agriculture, distributed among arable land (52.8%), land for permanent crops (4.2%) and pastures (3.5%). Share of agriculture and related activities was 11.3% of the Gross State Domestic product (GSDP) in 2013-14. The 11<sup>th</sup> five-year economic plan acknowledges the need of proper soil management in agriculture. Excessive and miscalculated use of fertilizer focused on increase production has led to soil degradation, and today nearly 66.67% of India's agricultural land can be categorized as either degraded or sick [89]. In India, each state and union territory is responsible for the set-up of the soil testing facilities and maintaining the state soil database. Some states, for e.g. Gujarat initiated the "Soil Health Cards Programme" in 2006 to recommend fertilizers, crop rotations and to record the data on a national network which can be used to survey different soil types<sup>2</sup>. The farmers are assisted by several non-governmental organization and community groups, who are responsible for soil sample testing [89]. In year 2013-14, the cultivation areas of major crops were 15 and 57 millions of hector in Kharif and Rabi seasons, respectively [24]. However, agriculture in India is conditioned by the poor fertility of the soil,

---

<sup>1</sup><https://www.cia.gov/library/publications/the-world-factbook/geos/in.html>

<sup>2</sup><http://indiagovernance.gov.in/news.php?id=204>

which depends on the levels of its nutrients. The physical, chemical and biological properties of the soil are useful to evaluate its fertility, to design a cultivation plan and to predict the crop productivity. The information technologies, and specifically Machine Learning (ML), offer new possibilities in the field of agriculture and may help in data evaluation for decision making. Maharashtra ranks second largest state in terms of population and geographical area in India and it is located at  $15^{\circ} 38''$  to  $22^{\circ} 01''$  North and  $72^{\circ} 39''$  to  $80^{\circ} 44''$  East where, total 64.14% of the people are employed in agriculture and allied activities. Agricultural calendar of Maharashtra is governed by monsoon, being the 60% of the farming rain-fed. This state is divided in 9 agro climatic zones, composed by 39% of shallow soils and 42.4% of degraded land. Maharashtra has 5 main regions; Vidarbha, Konkan, Marathwada, Paschim Maharashtra, and North Maharashtra, although only the three latter will be considered in the current study. Marathwada, is one of the most prominent agricultural regions in India, located at  $19^{\circ} 52' 59.88''$  North and  $75^{\circ} 19' 59.88''$  East.



**Figure 1.1:** Geographical representation of 6 regions of Maharashtra (India). Marathwada, Paschim Maharashtra and North Maharashtra are study areas, highlighted by red borders.

The soil of Marathwada is made of basalt rock with scarlet, blackish and yellowish colors and semi-dry plateau, good in iron level, moisture retentive, and poor in nitrogen as well as organic matter, including variable climatic condition. However, temperature is mostly humid throughout the year, maximum and minimum temperature oscillates from  $27^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  and  $14^{\circ}\text{C}$  to  $27^{\circ}\text{C}$ . The classification of soil according to its physical and chemical properties is

useful to maintain and enhance its productivity, to avoid soil degradation problems and to overcome environmental damage. The major challenge is to increase crop yield for solving global food security problem. However, soil quality and crop yield are negatively affected by changing trends of temperature and rainfall, insufficient water and light, agriculture practices and absence of nutrients. It is important to develop an effective nutrient management by means of an adequate soil analysis and a proper application of fertilizers. Hence the relevance of a research effort to classify soil parameters such as the fertility indices for several nutrients: organic carbon (OC), phosphorus pentoxide ( $P_2O_5$ ), manganese (*Mn*) and iron (*Fe*), among others; soil *pH*, soil type, preferred crop and levels of nutrients as the nitrous oxide ( $N_2O$ ), phosphorus pentoxide ( $P_2O_5$ ) and potassium oxide ( $K_2O$ ), which are relevant for fertilizer recommendation. The interest of predicting the levels of these magnitudes with ML techniques is to avoid the need to chemically measure these magnitudes, thus reducing the cost of the analysis and saving time of specialized technicians. The current study tries to enhance the accuracy of soil problem interpretation for Indian agriculture, although similar studies would benefit other nations around the globe.

## 1.1 Soil fertility index

In India, land-man ratio is quickly decreasing, so there is a need increase agricultural production without harm to environment and sustainability. The per capita land in India has decreased from 0.48 to 0.41 ha./person since year 1951 to year 2001, and the prediction is to decrease even more, until 0.10 ha/person by the year 2025. Besides, urbanization and industrialization led to destruction of forest, and to a reduction in the cultivation land and in its quality. As a result, agricultural production is being affected unfavorably. This situation appeals for planning of soil fertility by supplying essential nutrients to the crop in sufficient amount and at right time for its best growth. Therefore, fertilizers are a great significance input for targeting high crop production. Moreover, imbalances in soil quality leads to crop health and lower/higher crop yield [87]. The Indian soil has generally low or medium levels of soil nutrients and organic matters. Consequently, an efficient soil nutrient management is a major concern for maintaining soil fertility and it mainly depends on optimum fertilizer rates as per nutrient demand of crop. The crop needs optimum quantity of various nutrients for its vegetative growth and ultimate yielding. Declining status of soil fertility and mismanagement of soil nutrients may be factors for food crises for the world's population [44]. Generally,

Indian soil fertility data are summarized for block and district level. These data are useful for decision making of application of suitable amount fertilizers, policy of fertilizer distribution and consumption in the view of changes in fertility levels. One of the objectives of the work developed in the current PhD. Thesis is to predict village-wise soil fertility indices, which can be used in preparing a village-wise soil fertility index map. This map would allow to compare levels of soil fertility among villages, and to make fertilizer recommendations.

## 1.2 Fertilizer nutrients

Maharashtra is one of the major fertilizer-consuming states in India with annually around 1.8 to 2.0 million tonnes in terms of  $N_2O$ ,  $P_2O_5$  and  $K_2O$  soil nutrients<sup>3</sup>. Distorted levels of these nutrients might increase fertilizer demand and this can adversely affect soil fertility. The usage of these fertilizers in Maharashtra has rapidly increased from 56.85, 29.74, 16.14<sup>4</sup> to 65.93, 32.13, 17.64<sup>5</sup> (kg/ha) since 2012-13 to 2013-14. The soil  $N$  is crucial nutrient fertilizer for food production moreover, responsible parameters for its availability in soil are its type, texture, soil  $pH$ , climate, etc. Generally, soils contain 0.02 to 0.44 % total  $N$  however, climate of study area is tropical so that soils are poor in  $OC$ , and consequently low in  $N$ . The phosphorus ( $P$ ) plays a key role in substances are used as building blocks for genes and chromosomes, in plant root growth and biochemical processes that involve energy transfer. Generally, the  $P$  content in agricultural crop ranges from 0.1 to 0.5%. The status of  $P$  in Indian soils range between 134.6 to 310.4 kg/ha. There is significant relation between  $OC$  and  $P$  due to creation of sustainable soil environment with soil organic matter [121]. However, deficiency of  $P$  leads to: breakdown of plant cell membranes, which reduces energy transfer; lowering of root ratio; poor seed/fruit setting; decreased disease resistance, and reduced tillering in cereals.

## 1.3 Soil reaction

The soil reaction is another name for the  $pH$  which measures the acidity or alkalinity in the soil. The soil  $pH$  influences the solubility of nutrients, affecting the activity of microorganisms responsible for breaking down organic matter and most chemical transformations in the

---

<sup>3</sup><http://www.moef.nic.in/soer/state/SoE%20report%20of%20Maharashtra.pdf>

<sup>4</sup><http://fert.nic.in/sites/default/files/Indian%20Fertilizer%20SCENARIO-2014.pdf>

<sup>5</sup>[http://fert.nic.in/sites/default/files/Indian%20Fertilizer%20SCENARIO-2014\\_0.pdf](http://fert.nic.in/sites/default/files/Indian%20Fertilizer%20SCENARIO-2014_0.pdf)

soil, affects how plants grow, and modifies the availability of several plant nutrients. The  $pH$  change over time is influenced by factors including chemical composition of the soil, weathering and current agricultural practices, and it also fluctuates through the year. When soil acidity changes, the solubility of metal ions also changes, and the plant growth is really affected by the varying concentration of these metals in solution rather than by the acidity itself. The desirable  $pH$  range for optimum plant growth varies among crops. While some crops grow best between 6.0 and 7.0, others grow well under slightly acidic conditions. Soil properties that influence the need for and response to lime vary by region. Soils become acidic when basic elements such as calcium, magnesium, sodium and potassium held by soil colloids are replaced by hydrogen ions. Soils formed under conditions of high annual rainfall are more acidic than soils formed under more arid conditions. In India, most Southeastern soils are inherently more acidic than soils of the Midwest and far West. Soils formed under low rainfall conditions tend to be basic with soil  $pH$  readings around 7.0. Intensive farming over a number of years with nitrogen ( $N$ ) fertilizers or manures can result in soil acidification. In the wheat-growing regions of Kansas and Oklahoma (USA), for example, which have soil  $pH$  of 5.0 and below, aluminum toxicity in wheat and good response to liming have been documented in recent years. A knowledge of the soil and the crop is important in managing soil  $pH$  for the best crop performance. The  $pH$  levels generally found in soil are listed in the figure 2.1 of the subsection 2.3. Significance of soil  $pH$  classification is to increase the soil quality and crop production.

## 1.4 Cropping cycle

In Maharashtra, uncertain climatic situations, namely rainfall, affects on agricultural productivity that tends to certain type of farming only. This state is highly dependent on monsoon cycle for large crop yields. The major crop produced by Maharashtra is cotton and soybeans. Moreover, farmers generally use a single cropping due to drought conditions, climate change, soil fertility status and unawareness about soil nutrient status. However, crop rotation is an effective technique to control of weeds, pests, diseases, and more economical utilization of soil fertility. An accurate crop classification allows to predict the optimal crop based on available soil nutrients, whereas significance of optimal cropping system is to avoid environmental damage, consequently it improves soil quality, reduces the build-up of pests, spreads

the workload on family labour, mitigates the risk of weather changes, becomes less reliant on agricultural chemicals, and increases the net profit [104].

## 1.5 Soil texture

The soil can be classified in a set of classes accordingly to its behavior, potential uses and productivity. The early system of soil classification was quite simple and practical; for instance: economic classification is the grouping of soils based on their productivity; physical classification is based on soil texture (loamy, sandy, and clayey soils). The chemical classification categorizes the soil in acidic, alkaline, calcareous, gypsiferous soils, etc. Geological classification is based on the nature of underlying parent rock or material e.g basalt, lime stone, sandstone, etc. Physiographic classification is based on the characteristics of landscape e.g levee, basin terrace, mountain, vally, upland and lowland soil, etc. since these system were based on a single character, their utility was limited. Hence, the requirement of more comprehensive system was felt. In the current study, classification of soil type is based on soil physical and chemical properties to understand relationship among several soil parameters. Specifically, we classify the soil type according to its texture as light and medium soil, as we will describe in subsection 2.13.

## 1.6 Related work

Several studies [83] have applied ML techniques to solve soil problems in agriculture, namely to predict soil fertility, defined as the soil ability to supply the required nutrient levels and water for high quality crop yield. The soil fertility was predicted using artificial neural networks (ANNs) with Levenberg-Marquadt based back-propagation [119], and also using partial least squares regression [88] using as input data the soil bulk density, electrical conductivity (*EC*), available water capacity, soil *OC*, pewamo silty clay loam, glywood silt loam, kibbie fine sandy loam, crosby silt loam and crosby celina silt loams soil.

The crop yield has been predicted using the BeeHive and improved k-means clustering techniques [86]. The One-R rule classifier, the J48 decision tree, K-nearest neighbors (KNN) and A priori classifiers have been used to predict wheat yield [114] using as inputs phenotypic plant traits (thousand grain weight, plant height, peduncle length, harvest index, spikelets number, grain number, grain weight and spike fertility). The wheat yield has also been quantified [94] as low, medium and high with supervised Kohonen and counter-propagation neural

networks, and with XY-fusion models using multi-layer soil data:  $pH$ , moisture content, total nitrogen, total carbon, magnesium, calcium, cation exchange capacity, available phosphorus and satellite imagery crop growth characteristics. Decisions about insecticide application (either spray or non-spray) for leafroller pest monitoring on kiwifruit are recommended [50] using decision tree (DT), naive Bayes classifier, random forest (RF), adaboost, support vector machine (SVM) and logistic regression (LR). The generalized regression neural network (GRNN) was used to forecast plant diseases [19] for leaf wetness prediction. The results of GRNN and multiple linear regression were compared based on their prediction accuracy and computation time. The RF provides the best accuracy for mapping the soil class spatial distribution in three semi-arid study areas with different sets of environmental covariates [10], compared to clustering algorithms, discriminant analysis, multinomial logistic regression (MLR), ANN, DT and SVM. The soil has also been classified in 11 orders and 18 great groups from satellite images at 100 m. spatial resolution, using classification and regression trees (CART), bagging with CART base classifiers, RF, KNN, nearest shrunken centroid, ANN, MLR, logistic model trees and SVM [49].

The RF outperformed several regression models, in terms of RMSE, for prediction of  $N_2O$  emission on the basis of local information [97]. Further studies [98] presented statistical random-effects model for predicting  $N_2O$  emissions which response to applied nitrogen fertilizer. The data set contains 985 patterns from 203 publications and every publication accounted number of  $N_2O$  emission calibrations for various  $N$  fertilizer treatments. The prediction error reduced when the study used location-specific  $N_2O$  emission instead of average predictions. The work [108] discussed the state-of-art ensemble techniques such as conventional, decomposition, negative correlation learning, multi-objective optimization, fuzzy, multiple kernel and deep learning based ensemble methods.

## 1.7 Outline of the work

Our work deals with several problems analysed in some of the previous papers, such as classification of soil type (although with a different set of classes) and fertility indices (which in our study are specific for  $OC$ ,  $P_2O_5$ ,  $Fe$  and  $Mn$ ). However, there are some differences with previous studies. In the first place, our work analyses more soil problems than the previous works, including the prediction of ten soil parameters: village-wise fertility indices of  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$ ; soil nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , which allows to develop a fertilizer

recommendation; soil  $pH$  and type; and suitable crop. Classification of these soil parameters allows to save time of specialized technicians developing expensive chemical analysis. We also evaluate the validity of the ML methods trained with data from one region of India to model data from other regions. On the other hand, the available data for the current study do not allow to predict crop yield, insecticide application nor plant diseases, as in the paper [50]. Our data are exclusively chemical measurements (see sections 2.1-2.3), excluding data such as satellite images and phenotypic plant traits, as opposed to some other previous research works. Finally, we use a wider and more diverse collection of ML techniques than previous papers, both for classification, selected due to its good performance in the experimental comparison [28], and for regression, trying a large collection of regressors in the current study. The work has been developed in three stages:

1. **Classification for soil data.** The numeric values (e.g.  $pH$ , OC fertility, etc.) are quantified in several levels (e.g. low, medium, high), defined by the Indian Government, and classification methods are used to predict that level (see chapter 2) for the ten soil problems listed above. Since initially our experience with regression techniques was not very wide, we applied first classification methods, where our experience was higher, and evaluated the performance in order to assess the real utility of using ML techniques to predict the (quantified) soil parameters.
2. **Regression for generic data.** Previously to apply regression techniques to soil data, we developed an experimental comparison of many regressors to a large collection of generic datasets selected from the University of California at Irving (UCI) machine learning repository (see chapter 3). This work provided us the knowledge of new regression methods and the experience to apply them, as well as an idea of the most competitive techniques.
3. **Regression for soil data.** We applied the regressors used in stage 2 to the soil datasets, developing a direct prediction of their numeric values (see chapter 4). The accuracy of the prediction was evaluated for the ten soil problems listed above, as an alternative to the prediction of the quantified values (classification) developed in stage 1.

Finally, the chapter 5 compiles the conclusions of the current PhD. Thesis and the future work.

## CHAPTER 2

# APPLICATION OF CLASSIFICATION METHODS TO AGRICULTURAL SOIL DATA

In the current research we use data collected from the region of Marathwada by the State Government of Maharashtra (India) during year 2011 to 2015. Details about calibration of each input magnitude are publicly available (in Marathi language). The inputs that we use are the following soil parameters:  $N_2O$ , measured by the soil testing laboratories using alkaline permanganate [60, 125];  $OC$ , using carbon spectrophotometric [6];  $pH$ , using  $pH$  meter method [109];  $EC$ , using  $EC$  meter method [109];  $K_2O$ , using flame photometric [29, 57]; and  $P_2O_5$ , using the Olsen's method [90]. Micro nutrients as  $Fe$ , copper ( $Cu$ ), zinc ( $Zn$ ),  $Mn$  and boron ( $B$ ), which are useful to evaluate imbalances in soil nutrients, are measured using atomic absorption spectroscopy [68]. The  $pH$  is expressed as the decimal logarithm of the Hydrogen concentration. The values of  $N_2O$ ,  $K_2O$  and  $P_2O_5$  are expressed in kilograms per hectare (kg/ha), while  $Fe$ ,  $Cu$ ,  $Zn$ ,  $Mn$ ,  $B$  and  $SO_4$  are expressed as parts per million (PPM). The  $EC$  is expressed in milli-siemens per centimeter (mS/cm), while  $OC$  and  $CaCO_3$  are expressed as mass percentages (denoted as %). The following sections describe the ten soil parameters which we want to predict, in this first approach using a classification scheme.

### 2.1 Village-wise $OC$ , $P_2O_5$ , $Mn$ and $Fe$ fertility indices

There are proofs of the interconnection among organic matter, ecosystem sustainability and soil fertility [27], which is important for crop yield. This fertility mainly depends on  $OC$ ,

	Major nutrients		Micro nutrients		Index
	OC (%)	$P_2O_5$ (kg/ha)	Mn (PPM)	Fe (PPM)	
Low <	0.5	10	1	2.5	1.67
Medium	0.5-0.75	10-24.6	1-2	2.5-4.5	1.67-2.33
High >	0.75	24.6	2	4.5	2.33

**Table 2.1:** Intervals defined by the Indian Government [23] for the major and micro nutrients respectively [84, 63], and rate of nutrient index [105].

$N_2O$ ,  $P_2O_5$  and  $K_2O$ , considered soil major nutrients because they appear in large quantities, and also on micro nutrients  $Fe$ ,  $Mn$ ,  $Zn$  and  $Cu$ , which appear in smaller quantities. However, our work is restricted to fertility levels of  $OC$ ,  $P_2O_5$ ,  $Fe$  and  $Mn$ , due to data availability. The  $OC$  is very important for the soil health, biological activity and crop productivity [107], and adequate fertilizers help to keep its level [130]. The  $P_2O_5$  is used by plants for cell signaling, phosphorylation and bioenergetics, while  $Fe$  and  $Mn$  help chlorophyll to absorb light energy for photosynthesis. The agriculture planning of the Indian Government requires to determine the village-wise fertility indices  $N_I$  for the previous nutrients, using the thresholds listed in Table 2.1 to quantify their levels as low, medium and high. For each village and nutrient,  $N_l$ ,  $N_m$  and  $N_h$  are the number of patterns (i.e., cultivation lands) with low, medium and high levels, respectively. The village-wise fertility index  $N_I$  for a nutrient is calculated as  $N_I = (N_l + 2N_m + 3N_h)/N_t$ , being  $N_t$  is the total number of patterns analysed for a village. The value of  $N_I$  (which is the same for all the patterns in the village) is then quantified into low, medium and high levels, according to the threshold values listed in the rightmost column of Table 2.1. The classification of the village-wise fertility index uses the inputs listed in the first four lines of Table 2.2. The labels  $OC$ -F,  $P_2O_5$ -F,  $Mn$ -F and  $Fe$ -F mean village-wise fertility index of  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$ , respectively, whose values are completely different to the inputs  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$ . Our data (see Table 2.2) only include patterns with  $OC$ -F and  $Fe$ -F (resp.  $Mn$ -F) in levels low and medium (resp. medium and high). We could not develop classification of  $K_2O$  nor  $Zn$  village-wise fertility indices because the available patterns for each problem only belong to one class.

## 2.2 Soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$

The direct measurement of soil  $N_2O$  is difficult, but it is largely present in the  $OC$  form (97-99%), so that it can be determined indirectly from the  $OC$ , e.g. using linear regression [106].

Problem	Inputs	#Patterns	Classes and #patterns per class			
			Low	Medium	High	
$OC$ -F	$EC, OC, N_2O$	372	203	169	—	
$P_2O_5$ -F	$P_2O_5, K_2O, SO_4$	372	104	240	28	
$Mn$ -F	$Cu, Fe, Mn$	367	—	183	184	
$Fe$ -F	$Zn, B$	372	276	96	—	
$N_2O$	$EC, OC, P_2O_5, K_2O, SO_4, Cu, Fe, Mn, Zn, B$	372	124	124	124	
$P_2O_5$	$EC, OC, N_2O, K_2O, SO_4, Cu, Fe, Mn, Zn, B$	372	124	124	124	
$K_2O$	$EC, OC, N_2O, P_2O_5, SO_4, Cu, Fe, Mn, Zn, B$	372	124	124	124	
$pH$	$P_2O_5, K_2O, EC, OC, CaCO_3, Cu, Mn, Zn, Fe$	1137	SA 22	N 432	SAL 544	MAL 139
Crop	$P_2O_5, K_2O, pH, EC, OC$	2878	Bajra(R) 185	Cotton(I) 324	Cotton(R) 712	Soybean(R) 1657
Soil	$P_2O_5, K_2O, pH, EC, OC$	1692	Light 482	Medium 1210	Heavy —	

**Table 2.2:** Inputs, number of total patterns (#Patterns) and patterns per class for each classification problem. The labels SA, N, SAL and MAL mean slightly acid, neutral, slightly alkaline and moderately alkaline, respectively. The compounds  $SO_4$  and  $CaCO_3$  are sulfate and calcium carbonate respectively. The symbol — means that patterns of that class are not available.

High  $N_2O$  levels, as in tropical agro-ecosystems, have negative effects on water, air, ecosystem and human health, limiting the crop growth [42] and ecosystem productivity, which also depends on temperature, precipitations and atmospheric  $CO_2$ . The deficiencies of the Indian soils with respect to  $N_2O$  lead to a strong application of suitable fertilizers. The  $P_2O_5$  is also very important for the soil fertility, as we explained in the previous subsection. The  $K_2O$  is involved in crop physiological functions, being strongly deficient in the Indian soils [85], and the corresponding fertilizer (muriate of potash) is a non-renewable resource which can not be synthesized from other chemicals, so it can not be managed in large amounts. The classification of  $N_2O$ ,  $P_2O_5$  and  $K_2O$  is very useful because  $N$ ,  $P$  and  $K$  are the most responsible nutrients for fertilizer recommendation. However, as we mentioned in the previous subsection, using the national limits defined by the Indian Government (Table 2.1) the available data for  $N_2O$  and  $K_2O$  only include patterns of one class. Therefore, we defined the limits in order to have equally populated low, medium and high classes. These classifications use the 10 inputs listed in the lines labeled  $N_2O$ ,  $P_2O_5$  and  $K_2O$  of Table 2.2, and their results

are used to recommend the suitable amounts of fertilizers in the following way. Let  $C(X)$  be the predicted class for nutrient  $X$ , defined as  $C(X) = 1, 2$  or  $3$  for classes low, medium and high, respectively, being the nutrient  $X = N_2O$ ,  $X = P_2O_5$  or  $X = K_2O$ . Let also  $R(Y)$  be the amount, in kg/ha, of fertilizer  $Y$ , being  $Y = \text{uria}$ ,  $Y = \text{super phosphate}$  or  $Y = \text{muriate of potash}$  the recommended fertilizers to correct the level of nutrients  $N_2O$ ,  $P_2O_5$  or  $K_2O$ , respectively. The Mahatma-Phule Agricultural University [71] recommends an amount  $R(X, Y)$  of fertilizer  $Y$  to correct the level of nutrient  $X$  calculated using  $C(X)$  and a pre-defined reference amount  $F(Y)$  of fertilizer  $Y$ , according to an expression proposed by the technicians of the State Government of Maharashtra (see footnote 1):

$$R(X, Y) = \frac{6 - C(X)}{4} F(Y) \quad (2.1)$$

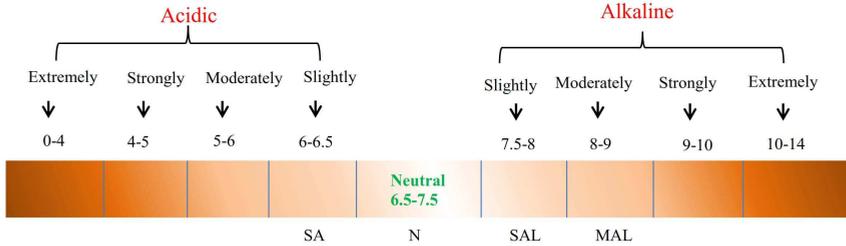
Therefore,  $R(1, Y) = 5F(Y)/4$ ,  $R(2, Y) = F(Y)$  and  $R(3, Y) = 3F(Y)/4$  for  $C(X) = 1, 2$  and  $3$ , respectively, so the recommended amount is 125%, 100% or 75% of  $F(Y)$  when the level of nutrient  $X$  is low, medium and high, respectively. This expression allows to calculate the recommended amount of fertilizer  $Y$  using the predicted class  $C(X)$  for nutrient  $X$ , given the reference amount  $F(Y)$ .

### 2.3 Soil pH

The  $pH$  is the scale of soil acidity or alkalinity, which affects the crop yield and all the soil parameters, because soil acidity is one of its major degradation problems. Specifically, the region of Marathwada has slightly alkaline soil (high  $pH$ ), which leads to nutrient deficiency, low  $OC$  and high  $CaCO_3$  levels, limits the crop growth and reduces the crop yield. The  $pH$  classification uses the inputs listed in the line labeled “ $pH$ ” of Table 2.2. Although usually nine  $pH$  levels are considered (Figure 2.1), but for our purposes it is enough to discriminate between the four middle classes: slightly acidic (labeled SA), neutral (N), slightly alkaline (SAL) and moderately alkaline (MAL). The classification of  $pH$  into levels is useful to decide suitable crops and pesticides, and to evaluate microbial activity, nutrient levels and soil corrosion.

### 2.4 Crop selection

The growth of a given crop needs a balanced supply of important nutrients, whose levels define the best crop for a given soil. The cropping cycle determines the way in which the soil



**Figure 2.1:** Degree of acidity and alkalinity of soil [16], with nine classes (up) and with the four classes considered in this paper (down).

parameters are enhanced: a good cycle is important for optimum yield and improvement of soil quality, erosion and moisture, organic matter and carbon storage [59]. There are several studies about crop classification [127], crop yield forecasting [81], mapping for crop rotation [93] and land cover classification on remotely sensed data using time-series analysis techniques [113]. The RF, linear discriminant analysis (LDA) and SVM were used to classify crops (walnut, table grape, almond and European plum) on four feature sets [96]. The SVM was also used for plant discrimination using satellite land images [45]. Our data includes four crops, which require neutral  $pH$  (in the range 6.5-7.5): bajra(R), cotton(I), cotton(R) and soybean(R), where R and I mean rainfed and irrigated, respectively. The Figure 2.3 (left panel) shows a geographical plot of the crop data, which are distributed across a wide area including the Aurangabad, Jalna, Bid, Osmanabad and Latur districts of the Marathwada region, inside of the state of Maharashtra. In the current paper, crop classification predicts which crop is suitable for the next stage of the cropping cycle using the inputs listed in the line labeled “Crop” of Table 2.2.

## 2.5 Soil type

The soil classification according to its type allows to select the best soil for a particular crop. Soil has been classified [126] using LR, ANN, SVM, KNN, RF and DT in 5 types: 1) coarse loamy, mixed, mesic, lithic xerorthents; 2) fine, mixed, mesic, typic calcixerepts; 3) fine loamy, carbonatic, mesic, typic calcixerepts; 4) fine loamy, mixed, mesic, typic haploxerepts; and 5)

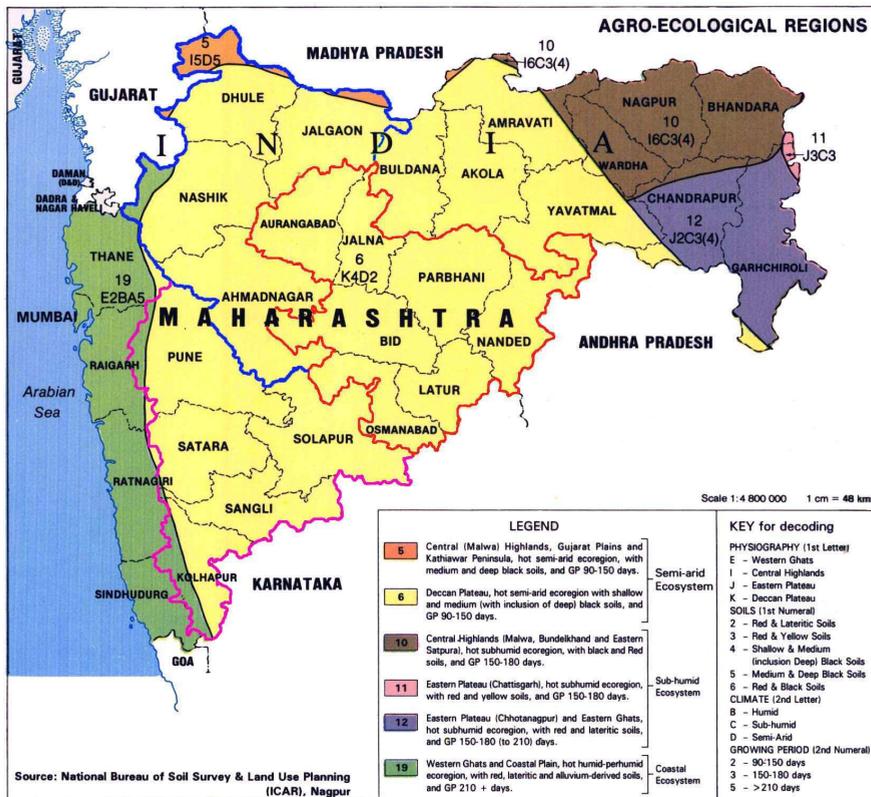
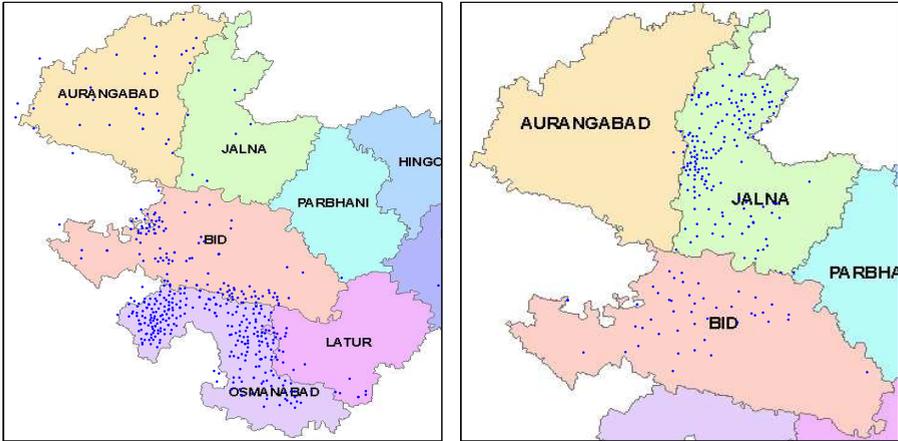


Figure 2.2: Soil map of Maharashtra [118]. The geographical study areas are highlighted with outlines: red (Marathwada), blue (North Maharashtra) and pink (Paschim Maharashtra).

fine, mixed, mesic, typic haploxerepts. The study used 217 patterns collected from Kurdistan Province, North-West Iran, achieving accuracy and Cohen kappa [131] of 71% and 69% using DT and ANN, respectively. Figure 2.2 shows<sup>1</sup> the different types of soil in the state of Maharashtra (label 6 locates the soil type of Marathwada). In our study, three soil classes are considered according to its texture. *Light soil* has large proportion of sand, low parameter levels and ability to hold water. Sandy, peaty and chalky soils are subtypes of light soil. *Medium (loam) soil* contains silt, clay and humus (decayed matter) and it is suitable for several crops, being the predominant type in Marathwada. *Heavy soil* contains more moisture

<sup>1</sup>[http://eussoils.jrc.ec.europa.eu/esdb\\_archive/EuDASM/Asia/images/maps/download/in3010\\_3so.jpg](http://eussoils.jrc.ec.europa.eu/esdb_archive/EuDASM/Asia/images/maps/download/in3010_3so.jpg) (visited March, 29, 2017).



**Figure 2.3:** Maps with the geographical location of crop (left panel) and soil type (right panel) data over several districts in the Marathwada region (red outline in the map of Figure 2.2). Each point locates a different village, from where several patterns are recorded.

and sticky lump, due to the high proportion of silt (slightly larger particles of rock) or clay (small particles of rock). The classification of soil type uses the inputs listed in the last line of Table 2.2, although the available data only contains patterns of classes light and medium. A geographical plot of the soil type data is shown in the Figure 2.3 (right panel). Locations are widely distributed among the Jalna and Bid districts of the Marathwada region.

## 2.6 Classification methods

In order to develop the classification for the ten problems described in the previous section, we used 20 classifiers selected among the ones which provided the best performance in the comprehensive comparison [28]. These classifiers are implemented in the Java programming language using the Weka data mining software [47], in the R statistical computing language [103], in the C++ programming language and in the Matlab platform [72]. Henceforth, the suffix of the classifier name shows the implementation used: `_w`, `_r`, `_c` and `_m` mean Weka, R, C++ and Matlab, respectively. The classifiers, grouped by families, are described in the following enumeration alongside with their tunable metaparameters (the symbol # means ‘number of’, e.g. `#inputs` ‘number of inputs’).

## I. *Decision trees*

1. **j48\_w** is the Java implementation of the C4.5 decision tree [100] provided by the Weka data mining software. This binary decision tree is composed of internal and leaf nodes, where each one does a binary splitting of a given input. Each leaf node gives as output a class label. The C4.5 method selects the input with the highest normalized information gain (also called Kullback-Leibler divergence, which is a measure of the non-symmetric difference between two class probability distributions) to be splitted for a new node of the tree. The method then proceeds by recurrence on the two subsets in which the node splits the training set, and the new nodes (internal or leaf) are added as children of the current node. When all the training patterns in the subset belong to the same class, a leaf node (whose output is that class) is created. When no input provides information gain, or when a training pattern of an unseen class appears, an upper level node is created for that class. For a test pattern, the decision tree starts from the root node and travels down the tree, according to the splittings in each nodes and the input values, until it reaches a leaf node, giving the class associated to this node as output. The only hyperparameter of the J48 tree is the pruning confidence threshold  $C$  with values 10, 20 and 30.
2. **rt\_w** is the random tree provided by Weka. Each node of the tree splits  $\lfloor \log_2(\#inputs) + 1 \rfloor$  randomly chosen inputs at each node of the tree, without any pruning.
3. **rpt\_r** is the recursive partitioning method [9] provided by the rpart R package. This method creates a decision tree by binary splitting the inputs and partitioning the training patterns in subsets in a recursive way.
4. **dj48\_w** is the decorate ensemble of J48 tree base classifiers with high diversity [76], provided by Weka. Decorate is an acronym for diverse ensemble creation by oppositional relabeling of artificial training examples. This method iteratively generates an ensemble by adding trained base classifiers (in our case, J48 trees). The first one is trained on the original training set, but the following ones use a fraction of patterns which are artificially generated. These labels are chosen to be maximally different from the actual ensemble output, in order to increase its diversity. Any new base classifier which reduces the ensemble accuracy is rejected. The iterative classifier addition continues until a specified ensemble size is reached. In our case, this size is a hyperparameter tuned with values 10, 15 and 30.

## II. Rule-based classifiers

5. **dtnb\_w** is the hybrid decision table-naive Bayes classifier provided by Weka [46]. A decision table (DT) is a lookup table which associates observed frequencies of selected inputs (which are expectedly highly discriminative) to class probability estimates. These inputs are selected using forward search using as objective the maximization of cross-validation performance. The DTNB uses two disjoint subsets of inputs: one for DT and other for naive Bayes (NB). At each search iteration, DTNB decides whether split the input in one or the other subset. The decision is oriented to maximize the area-under-curve of DTNB in a cross-validation. Initially, all the inputs are modeled by DT, but during search the selected inputs are modeled using NB, and the remaining ones by DT. The output class for a test pattern is the one with the highest weighted sum of DT and NB probabilities for that class and pattern (the last one divided by the prior class probability). The hyperparameter is the type of cross-validation, tuned with 4 values: leave-one-out, 4-fold, 5-fold and 10-fold.
6. **jrpf\_w**: repeated incremental pruning to produce error reduction (RIPPER), provided by Weka [21]. The RIPPER classifier starts from an empty rule set, and it iteratively: 1) creates a new rule by adding conditions to it until the rule is 100% accurate; 2) tries all the values of the inputs and selects the condition with the highest information gain; and 3) prunes the rule set using a specific metric. Once the rule is created, all the training patterns covered by the rule, both positives and negatives, are discarded. These three steps are repeated until the description length of the rule set reaches a maximum size, until the whole training set has been discarded, or until the error rate overcomes 50%. Finally, an optimized rule set is created in the following way: for each rule, two variants are generated: one from an empty rule using steps 1-2 above, and the other by adding antecedents to the rule. Then, one of the three rules (the original one and the two variants), the one with the smallest description length, is selected to be included in the optimized rule set. Besides, rules that increase the description length of the rule set are removed. The hyperparameters are the number of folds for reduced error pruning (4 values from 3 to 10) and the number of optimization runs (2,3 and 4).

## III. Bagging ensembles

7. **bg\_r** is the bagging ensemble [7] of decision tree base classifiers, provided by the `ipred` (bagging ensemble) and `rpart` (base classifiers) packages. Bagging is an acronym for

bootstrap aggregating, a method to build an ensemble of base learners (classifiers or regressors) based on training several base classifiers (in our case, rpart decision trees) on different random bootstrap samples of the training set. The sampling is with replacement, which means that training patterns can be repeated. A new test pattern is classified using a voting scheme over all the base classifiers. The bagging method, proposed initially for decision tree classifiers, aims to reduce the variance of the ensemble in order to avoid overfitting (i.e., a good learning of the training set which does not mean good learning of unseen test patterns) and to improve the stability of the single classifiers.

8. **bgf\_r** is a bagging ensemble of flexible discriminant analysis (FDA) base classifiers which uses the `bagFDA` and `fda` functions provided by the `caret` and `mda` packages, respectively [48]. The FDA classifier is a generalization of LDA for non-linear regression using optimal class scorings. In our case, we use the multi-variate adaptive resonance splines (MARS), provided by the `earth` package, as basis functions for FDA. The hyperparameters are the polynomial degree (1 and 2) of the polynomial splines and the maximum number of terms to keep in the pruned model (10 values from 2 to 11).
9. **bgp\_w** is a bagging ensemble of pruned partial C4.5 decision trees (PART) provided by Weka [32]. The PART trees combine C4.5 and RIPPER (see `jr_p_w` classifier above) trees. Both create a starting set of rules, which are refined by dropping rules (C4.5) or adjust them (RIPPER) by dropping the tail of a rule based on empirical error on a separate training set until a stop criterion (minimum description length heuristic) is met. Both are two-step methods which do global optimization. On the contrary, the PART creates a rule, removes the training patterns covered by the rule, and continues creating new rules recursively until the whole training set is processed. A partial tree is a tree with branches to undefined subtrees. The training process integrates the tree creation and pruning to find a stable (which can not be simplified) subtree. Initially, a split is selected and the training set is divided in two subsets. Both subsets are expanded, first the one with the smallest average entropy, which probably result in a smaller subtree and a more general rule. The process continues recursively until a subset is expanded into a leaf. When an internal node in the tree has only leaves, the method searches whether the node can be replaced by a leaf. The only hyperparameter is the bag size  $P$ , given as the percentage of the training set, tuned with values 25%, 50%, 75% and 100%.

#### IV. Boosting ensembles

10. **ab\_r** is an Adaboost.M1 [33] ensemble of classification trees implemented by the boosting function in the adabag package [2]. Adaboost is an acronym of adaptive boosting, a training method for classifier ensembles where each base classifier (in our case, classification trees) is biased to learn better those training patterns which have been misclassified by the previous classifiers of the ensemble. This is done by weighting each training pattern with the error on that sample. At each iteration, the weight of the base classifier with the smallest weighted error (sum of weights of the misclassified training patterns) is updated using that error. The weights decay are normalized again in each iteration. The test output is decided by a weighted voting among the base classifiers.

#### V. Nearest neighbors

11. **knn\_r** is the K-nearest neighbor [110] classifier, implemented by the knn function of the class R package. The output class for a test pattern is decided by voting over its  $K$  nearest training patterns. The number  $K$  of neighbors is the only hyperparameter, tuned with 13 values from 1 to 37.

#### VI. Neural networks

12. **elm\_m** is the extreme learning machine (ELM). The ELM is a single hidden layer feed-forward network which assigns random values to the input weights, calculates the hidden neuron outputs  $H$  using the activation function and calculates the output weight matrix  $B$  as the product of the Moore-Penrose pseudo-inverse of the  $H$  matrix multiplied by the desired output [53]. The test output is the product of  $H$  (dependent on the test pattern) times  $B$ . This network does not require iterative training, it does not fall in local minima, it avoids tunable hyperparameters as learning rate or momentum, and its training is much simpler than classical neural networks and support vector machines. We used the publicly available Matlab code<sup>2</sup>, and the tunable hyperparameters are the transfer function (six functions: sinus, signum, hard limit, triangular basis, radial basis and sigmoid) and the number of neurons in the hidden layer (20 values between 3 and 200).
13. **gelm\_m** is the ELM with Gaussian kernel tuning the regularization hyperparameter  $C$  with 20 values in the set  $\{2^i\}_{-5}^{14}$  and the kernel spread with 25 values in the set  $\{2^i\}_{-16}^8$ .

<sup>2</sup><http://extreme-learning-machines.org> (visited March, 29, 2017).

14. **mlp\_m** is the classical multi-layer perceptron neural network implemented by the `train` function in the Matlab Neural Network Toolbox. The network has only one hidden layer, whose number of neurons is a hyperparameter tuned with 11 values from 3 to 30.
15. **rbf\_m** is the radial basis function (RBF) neural network [95], implemented by the `newrb` Matlab function. The RBF is a two-layer network whose hidden layer is composed by radial basis neurons, usually with Gaussian activation, which are iteratively added during training. Starting from an empty layer, the training set is classified and the pattern with the highest root mean square error is selected as weight vector for a new hidden neuron, being the center of the neuron Gaussian activation. Then, the output weights between the hidden and output layers are calculated to minimize the error. The process is repeated, adding new hidden neurons until the error falls below a goal, or the maximum number of hidden neurons (which by default is the number of training patterns) is reached. The only hyperparameter is the spread of the Gaussian activations for the hidden neurons, tuned with 29 values from 0.1 to 70.
16. **pnn\_m** is the probabilistic neural network [123], implemented by the `newpnn` Matlab function. This network has only one hidden layer whose neurons are radial basis functions centered in the training patterns. The weights of the output layer are the class labels for the training patterns. The only hyperparameter is the Gaussian spread, tuned with 10 values between 0.01 and 10.

## VII. Support vector machines

17. **svm\_c** is the support vector classifier [15] with Gaussian kernel implemented by the LIBSVM library<sup>3</sup> in C++, tuning the regularization hyperparameter  $C$  with 20 values in the set  $\{2^i\}_{-5}^{14}$ , and the Gaussian spread with 25 values in the set  $\{2^i\}_{-16}^8$ .

## VIII. Random forests

18. **rf\_r** is the random forest ensemble [8] of random tree base classifiers implemented by the `randomForest` package in R. Each random tree is trained using feature bagging, which randomly selects an input subset with  $\sqrt{\#inputs}$  items at each candidate split. If some input is very useful to predict the output (i.e., the input is very related to the

---

<sup>3</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvm> (visited March, 29, 2017).

output), that input will be selected in many base trees, and they might be highly correlated, which is undesirable to achieve a diverse ensemble of trees. In order to avoid this, and to correct the overfitting of individual trees, they are trained in different bootstrap samples of the training set, and voting is used to give a class output for a test pattern. The base classifiers should not be strongly correlated among them, so that voting is expected to reduce its variance without increasing its bias.

19. **rf\_w** is an alternative implementation of random forest provided by Weka. We tune the number of trees in the forest with values 100, 250, 500 and 750.
20. **rtf\_w** is the rotation forest ensemble [112] of J48 base classifiers, implemented in Weka. The rotation forest simultaneously promotes classifier accuracy and ensemble diversity. The input set is randomly split into a number of subsets. For each input subset, PCA is applied on a bootstrap sample of the training set. Thus, each input subset represents a different rotation of the original input space. Each J48 base tree is trained on the whole collection of training patterns with all the principal components in order to promote accuracy. On the other hand, each classifier uses a different input subset of the principal components, in order to promote diversity. For a test pattern, each base tree gives a probability to assign that pattern to each class, so the ensemble assigns the test pattern to the class with the highest total probability summed over all the trees. The hyperparameters are the percentage of patterns to be removed in the bootstrap sample (5 values from 10% to 75%) and the number of iterations (5 values from 10 to 50).

## 2.7 Experimental setup

In order to apply the classifiers described in subsection 2.6 to the datasets described in subsections 2.1-2.3, we used as quality measure to evaluate the classification performance the Cohen kappa [131], henceforth denoted by  $\kappa$  and measured in %. The Cohen  $\kappa$  evaluates the classification accuracy discarding the probability of classifier success by chance, being defined as:

$$\kappa = 100 \frac{p_a - p_e}{s - p_e}, \quad p_a = \sum_{i=1}^C n_{ii}$$

$$p_e = \frac{1}{s} \sum_{i=1}^C \left( \sum_{j=1}^C n_{ij} \right) \left( \sum_{k=1}^C n_{ki} \right); \quad s = \sum_{i=1}^C \sum_{j=1}^C n_{ij}$$

where  $C$  is the number of classes and  $n_{ij}$  is the number of patterns of class  $i$  assigned by the classifier to class  $j$ , for  $i, j = 1, \dots, C$ . Besides  $\kappa$ , we also use two additional class-specific performance measures: sensitivity (SE) and positive predictivity (PP) of each class  $i$ , defined as  $SE_i = 100n_{ii}/\sum_{j=1}^C n_{ij}$  (percentage of patterns of class  $i$  assigned by the classifier to class  $i$ ) and  $PP_i = 100n_{ii}/\sum_{j=1}^C n_{ji}$  (percentage of patterns assigned by the classifier to class  $i$  which really belong to class  $i$ ). For each classification problem, we developed a 4-fold validation creating four groups of data sets, each composed by a training, a validation and a test set which do not intersect among them. The patterns of each class are randomly shuffled and 50% of them are used for training, 25% for validation and 25% for testing, thus guaranteeing that the same percentages of the pattern set are devoted to training, validation and test sets. For each class, the three sets are rotated among folds: e.g. with  $M$  patterns, the first fold uses patterns from 1st to  $M/2$ -th for training, patterns from  $M/2 + 1$  to  $3M/4$  for validation, and patterns from  $3M/4 + 1$  to  $M$  for testing; the second fold uses patterns from  $M/4 + 1$  to  $3M/4$  for training, from  $3M/4 + 1$  to  $M$  for validation and from 1 to  $M/4$  for testing; and so on. Therefore, the percentages of patterns for training, validation and test are respected also for each single class, so that 50%, 25% and 25% of the patterns of each class are devoted for training, validation and test, respectively. Each classifier is trained on the four training sets using each combination of values of its metaparameters and then it is tested on the validation sets. For each combination of metaparameter values, the average  $\kappa$  over the four validation sets is calculated, and the combination of metaparameter values with the best  $\kappa$  is selected for testing. At this point, each classifier is trained on each training set with the selected metaparameter values, and then it classifies each test set. The final test performance is measured by the average  $\kappa$  over the four test sets. Note that any pattern of the test set is not included in the training nor in the validation sets of the same group, so the test results can not be optimistically biased by overfitting.

## 2.8 Global discussion of the results

Table 2.2 in subsection 2.1 lists the information (number of patterns and inputs) of each classification problem, and the population of each class. Given that the region of Marathwada has soil with low  $OC$ -F and  $Fe$ -F levels, the “low” class is the most populated for both problems. This soil exhibits medium  $P_2O_5$  level predominantly and it is slightly alkaline, so the SAL class is the most populated. Besides, most of the soil is of type medium, and the major

crops are soybean(R) and cotton(R), which rely on rainfall because Marathwada comes under drought area since four years. The data are preprocessed in order to have zero mean and one standard deviation for each input.

Classifier	OC-F	P <sub>2</sub> O <sub>5</sub> -F	Mn-F	Fe-F	N <sub>2</sub> O	P <sub>2</sub> O <sub>5</sub>	K <sub>2</sub> O	pH	Crop	Soil
ab_r	88.50	<b>85.54</b>	59.33	67.45	25.40	<b>35.08</b>	25.81	44.71	85.27	96.65
bg_r	73.84	74.95	52.72	53.58	28.63	27.42	28.63	26.76	82.02	93.47
bgf_r	85.18	78.77	54.95	64.21	26.61	29.03	29.44	40.06	77.79	95.47
bgp_w	82.42	74.97	54.95	64.09	24.60	24.60	20.97	42.13	86.37	95.49
dj48_w	83.03	80.45	49.44	57.18	31.85	28.63	24.60	42.49	85.15	95.37
dtmb_w	81.33	66.78	47.79	44.69	22.18	22.58	5.24	35.93	85.85	<b>97.82</b>
elm_m	75.45	68.99	48.41	49.90	23.39	25.81	12.10	34.61	82.25	91.76
gelm_m	82.41	77.23	57.14	57.85	30.65	30.24	20.16	42.91	85.17	96.20
j48_w	69.25	70.76	37.92	54.42	20.16	21.37	14.92	40.78	82.34	94.76
jr_p_w	71.28	70.34	45.02	52.00	19.35	22.18	12.90	40.44	83.12	94.94
knn_r	74.14	72.56	52.77	54.09	25.81	24.19	15.73	41.55	84.32	94.46
mlp_m	56.95	30.25	38.77	41.46	24.60	17.74	12.10	13.73	47.62	86.18
pnn_m	77.43	73.33	55.56	52.82	23.79	22.98	18.15	40.38	84.52	94.90
rbf_m	48.82	40.04	34.40	43.32	2.82	14.92	2.82	13.30	82.99	92.03
rf_r	87.35	83.01	58.78	<b>69.35</b>	<b>33.06</b>	33.06	<b>31.85</b>	<b>47.32</b>	<b>88.13</b>	96.37
rf_w	<b>90.65</b>	79.58	64.27	65.17	30.24	32.26	26.61	46.85	87.64	96.80
rpt_r	67.05	63.81	46.65	36.23	21.37	22.58	21.37	38.92	80.01	92.89
rt_r	69.15	68.93	51.14	36.03	15.32	21.37	19.35	33.49	78.75	93.39
rtf_w	87.39	75.83	57.70	59.01	28.23	30.65	27.42	46.77	86.34	96.80
svm_c	80.2	82.3	<b>64.8</b>	60.1	29.0	30.2	18.5	43.0	86.1	95.8

**Table 2.3:** Values of  $\kappa$  (in %) achieved by each classifier for each classification problem. The best  $\kappa$  for each problem is in bold.

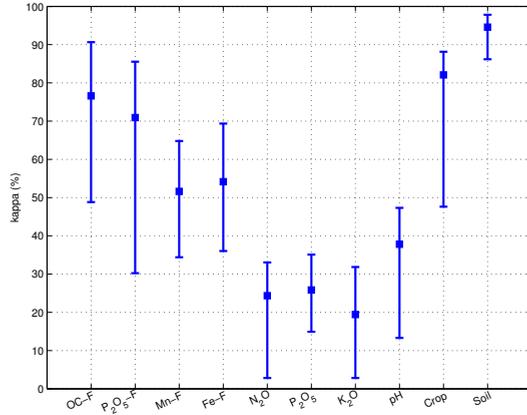
Friedman rank test						Wilcoxon signed rank test					
Pos.	Clasif.	Rank	Pos.	Clasif.	Rank	Pos.	Clasif	<i>p</i> -value	Pos.	Clasif	<i>p</i> -value
1	rf_r	2.200	11	knn_r	11.400	1	—	—	11	dtmb_w	0.384494
2	rf_w	2.950	12	pnn_m	11.400	2	ab_r	0.969839	12	jr_p_w	0.384494
3	ab_r	3.900	13	dtmb_w	12.650	3	rf_w	0.850051	13	elm_m	0.344523
4	rtf_w	4.650	14	jr_p_w	14.300	4	rtf_w	0.623046	14	j48_w	0.307308
5	svm_c	5.900	15	j48_w	14.350	5	svm_c	0.520366	15	elm_m	0.34055
6	gelm_m	6.600	16	elm_m	14.650	6	dj48_w	0.495968	16	rt_w	0.272856
7	dj48_w	7.400	17	rpt_r	15.650	7	bgf_r	0.472509	17	bg_r	0.272675
8	bgf_r	8.050	18	rt_w	16.250	8	bgp_w	0.472342	18	rpt_r	0.240968
9	bgp_w	9.600	19	mlp_m	18.200	9	knn_r	0.427181	19	rbf_m	0.10385
10	bg_r	11.300	20	rbf_m	18.600	10	pnn_m	0.427181	20	mlp_m	<b>0.03114</b>

**Table 2.4:** Columns 1-6: classifier ranking and position according to the Friedman rank test. Columns 7-12: classifiers ordered by decreasing *p*-value of a Wilcoxon signed rank test comparing the best ranked classifier (rf\_r) to the remaining ones (the significant tests are in bold). The label — means that rf\_r can not be compared to itself

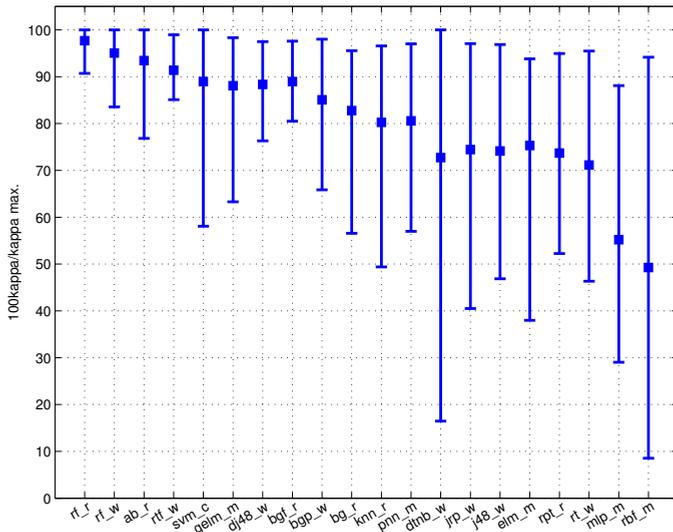
Table 2.3 reports the  $\kappa$  achieved by each classifier for the 10 classification problems: *OC*-F, *P<sub>2</sub>O<sub>5</sub>*-F, *Mn*-F and *Fe*-F (village-wise soil fertility indices of nutrients *OC*, *P<sub>2</sub>O<sub>5</sub>*, *Mn* and *Fe*); nutrients *N<sub>2</sub>O*, *P<sub>2</sub>O<sub>5</sub>* and *K<sub>2</sub>O*; soil *pH*; suitable crop and soil type. Globally, *rf\_r* and *rf\_w* achieve the best  $\kappa$  for 5 and 1 problems, respectively, being very near to the best ones for the other 4 problems. The *ab\_r* is the best in two problems (*P<sub>2</sub>O<sub>5</sub>*-F and *P<sub>2</sub>O<sub>5</sub>*), while *svm\_c* and *dtmb\_w* are the best for one problem each (*OC*-F, *Mn*-F and soil, respectively). The columns 1-6 of Table 2.4 reports the classifiers ordered by their Friedman rank test (decreasing with the classifier performance): the *rf\_r* (random forest in R) is the best with a rank of 2.2, which means that in average it is nearly the second best classifier for all the problems. The RF provided by Weka (*rf\_w*) is the second one, so its metaparameter tuning does not improve the good results of the R version. Adaboost in R (*ab\_r*) and rotation forest of J48 trees in Weka (*rtf\_w*) also achieve good ranks, although far from *rf\_r* (1.7 and 2.4 points higher), and *svm\_c* gets the 5th position (3.7 points higher), followed by *gelm\_m* (Gaussian kernel extreme learning machine in Matlab, 6th position), which works much better than *elm\_m* (16th position). The bagging classifiers *bgf\_r*, *bgp\_w* and *bg\_r* work similarly (positions 8, 9 and 10, respectively), and similarly to *knn\_r* (K-nearest neighbor classifier in R) and *pnn\_m* (probabilistic neural network in Matlab). Considering columns 7-12, the Wilcoxon signed rank tests between *rf\_r* and the other classifiers show that  $p > 0.05$  (i.e., the differences are not statistically significant) except with respect to *mlp\_m*, which gets the worst results. The lack of statistical significance might be due to the low number of measurements (just ten classification problems) for each classifier.

Figure 2.4 shows the  $\kappa$  intervals (minimum, mean and maximum) achieved by the different classifiers for each problem, with large differences in the  $\kappa$  values and in the interval width. The soil type classification problem has the highest maximum  $\kappa$  (upper limit of the interval) and the narrowest interval (i.e., all the classifiers work very well), while *OC*-F, *P<sub>2</sub>O<sub>5</sub>*-F and crop have high maximum  $\kappa$  with wider intervals. The *N<sub>2</sub>O*, *P<sub>2</sub>O<sub>5</sub>*, *K<sub>2</sub>O* and *pH* have lower  $\kappa$  values.

The intervals for each classifier over all the problems are reported by Figure 2.5 but, since the  $\kappa$  values are very different among problems, each interval represents the percentages of the maximums  $\kappa$  in the different problems achieved by the classifier. For example, the maximum  $\kappa$  in the *OC*-F problem is 90.65% achieved by *rf\_w*, so *rf\_w* has 100% of the maximum  $\kappa$ , while *ab\_r* has  $100 \cdot 88.5/90.65=97.62\%$  of the maximum  $\kappa$ . In Figure 2.5, the *rf\_r* has the highest and narrowest interval, being above 90% of the maximum  $\kappa$  for all the problems.



**Figure 2.4:** Intervals of  $\kappa$  (in %) of the different classifiers for each problem (the filled square shows the mean  $\kappa$ ).



**Figure 2.5:** Intervals of percentages of the maximum  $\kappa$  for each problem, for all the problems and for each classifier.

Some other classifiers as `ab_r`, `svm_c` or `dtnb_w` have intervals whose maximum achieves 100%, but their means and minimums are much lower than `rf_r` (e.g. `dtnb_r` achieves about 15% of the maximum  $\kappa$  for some problem). Although almost all the classifiers achieve more

than 90% of the maximum  $\kappa$  for some data set (excepting *mlp\_m*), all the classifiers exhibit minimum percentages below 80% excepting *rf\_r*, *rf\_w*, *rtf\_w* and *bgf\_r*.

	Major nutrients					Micro-nutrients			
	<i>OC-F</i> ( <i>rf_w</i> )		<i>P<sub>2</sub>O<sub>5</sub>-F</i> ( <i>ab_r</i> )			<i>Mn-F</i> ( <i>svm_c</i> )		<i>Fe-F</i> ( <i>rf_r</i> )	
Fertility	L	M	L	M	H	M	H	L	M
L	<b>52.9</b>	1.4	<b>22.8</b>	3	0.3	—	—	<b>66.2</b>	2.8
M	3.3	<b>42.4</b>	1.3	<b>58.5</b>	0.3	<b>36.5</b>	8.5	7.5	<b>16.5</b>
H	—	—	0.3	1.5	<b>5.3</b>	7.5	<b>38.5</b>	—	—
$\kappa$ (%)	90.65		85.54			64.8		69.35	
SE(%)	97.5	92.9	87.5	97.5	75	81.1	83.7	96	68.8
PP(%)	94.2	96.9	93.8	92.9	91.3	83	81.9	89.8	85.7

**Table 2.5:** Confusion matrices (in %) of the best classifiers (showed in the matrix header between brackets) for the classification of soil fertility indices. The symbol “—” means that the class is not available. Labels L, M and H mean low, medium and high respectively.

## 2.9 Classification of village-wise *OC*, *P<sub>2</sub>O<sub>5</sub>*, *Mn* and *Fe* fertility indices

The confusion matrices (in %) of the best classifiers, along with  $\kappa$ , SE and PP are reported by Table 2.5 for the soil fertility indices problems *OC-F*, *P<sub>2</sub>O<sub>5</sub>-F*, *Mn-F* and *Fe-F*. The *rf\_w* achieves the best  $\kappa$  for *OC-F* (90.65%, Table 2.3), followed by *ab\_r* (88.5%), *rtf\_w* (87.39%) and *rf\_r* (87.35%). The confusion matrix of *rf\_w* shows low values outside the diagonal, being the percentage of M patterns assigned to class L (3.3%) twice the percentage of L patterns assigned to class M (1.4%). The classes L and M exhibit high SE and PP. For *P<sub>2</sub>O<sub>5</sub>-F*, the best classifier is *ab\_r* ( $\kappa=85.54\%$ ), followed at certain distance by *rf\_r* (83.01%) and *svm\_c* (82.3%). In this case, the errors in the confusion matrix of *ab\_r* are for adjacent classes, e.g., between L and M, or between M and H, but the error percentages between non-adjacent classes are low (0.3%). The *ab\_r* tends to classify L patterns as M (3%), and in less degree H patterns as M (1.5%) and M patterns as L (1.3%). The best detected class is M, with SE and PP above 93%, while SE of class L (87.5%) and, specially, of class H (75%) are much lower. The results are worse for the classification of *Mn-F* and *Fe-F* problems. For *Mn-F*, the *svm\_c* is the best classifier ( $\kappa=64.8\%$ ), followed by *rf\_w* (64.27%), and the following classifiers are under 60% (*ab\_r* achieves 59.33%). The confusion matrix of *svm\_c* shows diagonal values (above 36%) much higher than outside the diagonal (7-8%), with slightly higher percentage of M patterns classified as H than the opposite. The results for *Fe-F* are slightly better: *rf\_r*

achieves  $\kappa=69.35\%$ , followed by *ab\_r* (67.45%) and *rf\_w* (65.17%). The confusion matrix of *rf\_r* shows a high number of M patterns classified as L (7.5%), while class M, which is much less populated, exhibits a low SE (68.8%).

	$N_2O$ ( <i>rf_r</i> )			$P_2O_5$ ( <i>ab_r</i> )			$K_2O$ ( <i>rf_r</i> )		
	L	M	H	L	M	H	L	M	H
L	<b>21.77</b>	8.33	3.23	<b>22.04</b>	6.99	4.30	<b>19.35</b>	9.41	4.57
M	11.02	<b>13.17</b>	9.14	9.41	<b>14.78</b>	9.14	8.87	<b>14.25</b>	10.22
H	6.45	6.45	<b>20.43</b>	4.57	8.87	<b>19.89</b>	4.57	7.80	<b>20.97</b>
$\kappa(\%)$	33.06			35.08			31.83		
Acc(%)	55.37			56.71			54.56		
SE(%)	65.3	39.5	61.3	66.1	44.4	59.7	58.1	42.7	62.9
PP(%)	55.5	47.1	62.3	61.2	48.2	59.7	59.0	45.3	58.6

**Table 2.6:** Confusion matrices (in %) of the best classifiers for  $N_2O$ ,  $P_2O_5$  and  $K_2O$ . Labels L, M and H mean low, medium and high, respectively. The values in the matrix diagonal are in bold.

## 2.10 Classification of soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$

Table 2.6 reports the best confusion matrices for the classification of  $N_2O$  and  $K_2O$ , achieved by *rf\_r* (with  $\kappa=33.06\%$  and  $31.83\%$ , respectively), and for the classification of  $P_2O_5$ , achieved by *ab\_r* (for which  $\kappa=38.08\%$ ), results which are worse than the corresponding to the classification of the village-wise soil fertility indices. Of course, the values of the accuracies are higher (about 55% for the three problems). For the three nutrients the class M has the lowest SE and PP (between 39-48%), while classes L and H exhibit SE and PP values about 60%. Since the classes represent quantization levels, they can be considered as ordinal classification problems, with an ordering relation among the classes. This explains that classes L and H are better recognized, while the middle class M wins external or loses internal patterns from/to L and H. Besides, the three matrices exhibit the highest non-diagonal values in positions adjacent to the diagonal, because the classification errors are more probable between adjacent classes representing contiguous levels of quantization.

## 2.11 Classification of soil $pH$

The *rf\_r* achieves the best performance for this problem, with  $\kappa=47.32\%$  (Table 2.3) and the confusion matrix of Table 2.7, for which the accuracy is 69.63%. Again, the only high values outside the diagonal correspond to adjacent classes: SA and N, with values 0.64% and

0.75% where the diagonal value is 0.37%; N and SAL, being more probable to classify N patterns as SAL (10.45%) than the opposite (6.66%); SAL and MAL, with more MAL patterns (7.78%) classified as SAL than MAL patterns correctly classified (2.67%). The remaining non-diagonal values are below 1%. The SE and PP are only acceptable (above 68%) for the most populated classes N and SAL, while the less populated classes SA and MAL exhibit worse results (SE about 20-30%).

<i>pH</i>	SA	N	SAL	MAL	SE(%)	PP(%)
SA	<b>0.37</b>	0.64	0.27	0	29.2	26.9
N	0.75	<b>26.92</b>	10.45	0.27	70.1	74.6
SAL	0.21	6.66	<b>39.66</b>	1.44	82.7	68.2
MAL	0.05	1.87	7.78	<b>2.67</b>	21.6	61.0

**Table 2.7:** Confusion matrix (in %) of the best classifier (rf\_r, with  $\kappa=47.32\%$  and accuracy=69.63%) for soil *pH* classification. Labels SA, N, SAL and MAL mean slightly acidic, neutral, slightly alkaline and moderately alkaline respectively.

## 2.12 Classification of crop

The rf\_r achieves the best  $\kappa$  (88.13%, with accuracy above 90%) also for crop classification (Table 2.3), followed by rf\_w (87.64%), bgp\_w (86.37%) and rtf\_w (86.34%). The ab\_r and svm\_c, which achieved good results in the previous problems, also work well. The confusion matrix of the rf\_r for this problem (Table 2.8) has almost all the values outside the diagonal under 1%. Only the percentage (3.34%) of bajra(R) patterns classified as cotton(R) is higher than the corresponding diagonal term (2.89%), which reduces the SE of class bajra(R) to 45.1%. The reason is that the usual crop rotation in Marathwada is between both classes, so their patterns are very similar. Besides, cotton(R) is more important and populated than bajra(R), whose patterns tend to be classified as cotton(R). All the remaining classes have SE above 90% and PP above 80%.

## 2.13 Classification of soil type

The best soil type classifier is dtnb\_w (hybrid classifier of decision table and naive Bayes), with  $\kappa=97.82\%$ , followed by the group of methods which were the best in the previous classification problems: rf\_w and rtf\_w (96.80%), ab\_r (96.65%), rf\_r (96.37%) and svm\_c

Crop	Bajra(R)	Cotton(I)	Cotton(R)	Soybean(R)	SE(%)	PP(%)
Bajra(R)	<b>2.89</b>	0.04	3.34	0.14	45.1	79.8
Cotton(I)	0	<b>10.14</b>	0.35	0.76	90.1	95.7
Cotton(R)	0.74	0.07	<b>23.14</b>	0.81	93.5	85.3
Soybean(R)	0	0.34	0.32	<b>56.90</b>	97.1	97.1

**Table 2.8:** Confusion matrix (in %) of rf\_r for crop classification ( $\kappa=88.13\%$ , accuracy=93.09%). Labels R and I mean rainfed and irrigated respectively.

(95.8%). The confusion matrix of the two best classifiers (Table 2.9) shows similarly low non-diagonal values for classes L and M, with SE and PP values above 97%.

Soil	dtnb_w				rf_w			
	L	M	SE(%)	PP(%)	L	M	SE(%)	PP(%)
L	<b>27.96</b>	0.47	98.33	99.42	<b>27.73</b>	0.71	97.50	97.17
M	0.41	<b>71.15</b>	98.54	99.34	0.59	<b>70.97</b>	97.91	99.01

**Table 2.9:** The confusion matrix (in %) of the two best classifiers (dtnb\_w and rf\_w,  $\kappa= 97.82\%$  and  $96.80\%$  respectively) for soil type classification. Labels L and M mean light and medium respectively.

## 2.14 Comparison among regions

As well as the data from the region of Marathwada, we also have data available from regions Paschim-Maharashtra and North-Maharashtra, in the same state of Maharashtra. An interesting issue is how valid are the classifiers, trained using data from one region, to test data from different regions. In other words, what is the representativity and generalization ability of the trained classifiers with respect to regions? We developed experiments training and tuning the metaparameters of the classifiers with patterns from one region, and then testing the trained and tuned classifier with data from different regions. Specifically, we have data from Marathwada, North Maharashtra and Paschim Maharashtra (henceforth labeled as M, NM and PM, respectively) highlighted in Figure 2.2, for village-wise  $OC$ -F,  $P_2O_5$ -F,  $Mn$ -F,  $Fe$ -F and  $pH$  problems. The corresponding experiments for  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , crop and soil type classification could not be developed due to the lack of data for regions NM and PM.

Table 2.10 reports the best classifier for each classification problem and combination of training and test regions, and the  $\kappa$  that it achieves (e.g. M-PM in the leftmost column means training and test using data from Marathwada and Paschim-Maharashtra, respectively). No

Regions	OC-F		$P_2O_5$ -F		Mn-F		Fe-F		pH	
	Best	$\kappa$	Best	$\kappa$	Best	$\kappa$	Best	$\kappa$	Best	$\kappa$
M-NM	rpt_r	17.21	bgp_r	15	bg_r	65.98	rpt_r	66.78	rtf_w	23.97
M-PM	elm_m	9.66	rf_r	100	bg_r	66.62	rpt_r	70.52	rtf_w	66.22
NM-M	rtf_w	32.60	rf_r	34.10	rt_w	45.70	svm_c	42.50	—	—
NM-PM	pnn_m	14.99	bg_r	100	bg_r	100	j48_w	100	—	—
PM-M	rt_w	12.73	bgf_r	34.57	bgf_r	45.15	elm_m	44.33	knn_r	69.61
PM-NM	pnn_m	11.13	bg_r	100	bg_r	100	svm_c	100	rf_r	48.23

**Table 2.10:** Values of  $\kappa$  (in %) achieved by the best classifier training and testing with patterns of different regions (first column, see text for region labels). The symbol ‘—’ means that data are not available.

classifier achieves good results for OC-F, no matter the combination of regions used for training and testing, and the best result is  $\kappa=32.6\%$  with rtf\_w. The results for  $P_2O_5$ -F are very good ( $\kappa=100\%$ ) for NM-PM and PM-NM (both with bg\_r) and for M-PM (with rf\_r), but much worse (between 15% and 35%) for the remaining combinations. This is surprising, because M-PM works well, but PM-M works much worse, suggesting that data from region M are somehow representative for data from PM, but the opposite is not true. For Mn-F and Fe-F problems, both NM-PM and PM-NM work well (100% with bg\_r for Mn-F, and with j48\_w and svm\_c for Fe-F), so the data seem to be valid between NM and PM. The performance of cases M-NM and M-PM are about 66-71%, so the data from M are somehow valid for the other two regions, but the inverse combinations (NM-M and PM-M) are about 42-46%, so NM and PM data are not so valid for region M. As a conclusion: 1) the data for OC-F classification are not valid among regions; 2) the data for  $P_2O_5$ -F, Mn-F and Fe-F classifications are valid only between North Maharashtra and Paschim Maharashtra, and from Maharashtra to Paschim-Maharashtra; 3) the data for  $P_2O_5$ -F classification are compatible from Marathwada to Paschim Maharashtra, but not the inverse; and 4) the data for Mn-F and Fe-F classifications are slightly compatible (about 66-70%) from Marathwada to the other regions, but not the inverse.

## CHAPTER 3

# APPLICATION OF REGRESSION METHODS TO GENERAL DATASETS

Previously to apply regression methods on the different datasets of the soil problem, we developed an experimental comparative of the most popular regression methods in order to know which are the best methods for non-specific datasets. For this work we used the benchmark datasets of the University of California at Irving (UCI) machine learning repository. The current chapter presents and discusses this comparison.

### 3.1 Regression methods

We have applied a wide collection of 76 regressors which belong to several families. The majority of them (68 regressors) are selected from the caret model list<sup>1</sup> and implemented in R. Instead of using the interface provided by caret (train function), we run the regressors directly using the corresponding R packages (see the detailed list below), in order to have control on the execution of each single model, and to avoid the execution of some regressors not included in the caret list. In fact, we also included other four popular methods implemented in other platforms: deep learning neural network, using the module dlkeras in Python (named dlkeras); support vector regression, using the LibSVM library in C++ (named svr); generalized regression neural network and extreme learning machine with Gaussian kernels in Matlab (named grnn and elm-kernel respectively). Some regression models on the caret list gave errors (list

---

<sup>1</sup><http://topepo.github.io/caret/train-models-by-tag.html> (visited March, 29, 2017).

them). The model operation is optimized by tuning the set of hyperparameters specified in the caret regressor list. The hyperparameter values used are specified by the caret package (`getModelInfo` function), and they are different for each dataset. For each regressor, we use the list of tunable hyperparameters specified by the caret package in the previous link, and a number of values used for tuning them. These values are listed in the regressor description below. Note that for some regressors (e.g. `gaussprRadial`) and datasets, the caret function `getModelInfo(...)` returns a value list with less items than the number specified in `values.txt`, and even sometimes just one value is used, so although the caret website specifies that hyperparameter as tunable, in the practice it uses only one value, so it is not tuned at all. The regressors in other languages use pre-specified values equal for all datasets. The 76 regressors are described in the following list, grouped by families.

### **I. *Linear regression***

1. **lm** is the linear model provided by the stats package [14]. Collinear inputs exhibit undefined regression coefficients (as returned by `lm`), so they are discarded for `lm` and many other regressors in the list.
2. **rlm** implements robust linear model (MASS package), fitted using iteratively re-weighted least squares with maximum likelihood type estimation, which is robust to outliers in the output although not in inputs [54]. The only hyperparameter is the psi functions: Huber, which provides a convex optimization problem; Hampel and Tukey bisquare, both with local minima.

### **II. *Generalized linear regression***

3. **glm** is the generalized linear model provided by the stats package [25], which combines a probability distribution (e.g. Gaussian, binomial, Poisson, etc.), a linear predictor and the link function corresponding to the distribution (which may be non-Gaussian), which relates the output mean and the inputs. They are used to model positive values, categorical or ordinal data.
4. **penalized** is the penalized linear regression (penalized package), which fits generalized linear models with a combination of L1 and L2 penalties. The L1 penalty, also called least absolute shrinkage and selection operator (lasso), penalizes the sum of absolute values of the coefficients, thus reducing the coefficients of inputs which are not relevant

similarly to input selection. The L2 penalty (also called ridge) penalizes the sum of squared coefficients, reducing the consequences of input collinearity. The regression is regularized by weighting both penalties [41], whose weights (given by hyperparameters `lambda1` and `lambda2`, with 5 and 4 values respectively) are tuned.

5. **glmnet** is the lasso and elastic-net regularized GLM provided by the `glm` package [120]. The `glmnet` model uses penalized maximum likelihood to fit a GLM for the lasso and elastic-net non-convex penalties. The mixing percentage `alpha` is tuned with 5 values, including `alpha=1` (resp. `alpha<1`), which corresponds to lasso penalty, and `alpha<1` for elastic-net penalty (`alpha=0` corresponds to ridge regression penalty). The `glmnet` function already tries a number of values (100 by default) for the regularization parameter `lambda`, so it was not tuned despite of being included among the `glm` hyperparameters in the `caret` list.
6. **glmStepAIC** is the generalized linear model with stepwise feature selection [111] using the Akaike information criterion (`stepAIC` function in the `MASS` package).

### III. Least squares

7. **npls** is the non-negative least squares regression (`npls` package), it solves for  $\mathbf{x}$  the optimization problem  $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$  subject to  $\mathbf{x} \geq 0$  using the Lawson-Hanson NNLS method [67].
8. **krlsRadial** is the radial basis function kernel regularized least squares regression (KRLS package), which uses Gaussian radial basis functions to learn the best fitting function minimizing the squared loss of a Tikhonov regularization problem. The `krls` method learns a closed form function so interpretable as ordinary regression models. The only hyperparameter is the kernel spread (10 values). The `krls` determines the trade-off between model fit and complexity (`lambda` parameter) by minimizing the sum of squared leave-one-out errors, so the `caret` `getModelInfo` function does not provide values for it, despite being listed as a tunable parameter.

### IV. Partial least squares

9. **spls** is the sparse partial least squares regression (`spls` package), which introduces sparse linear combinations of the inputs in the dimensionality reduction of PLS in order to

avoid lack of consistency of PLS with high dimensional patterns [20]. The hyperparameters are the number of latent components ( $K$ ) and the threshold  $\eta$  (3 and 7 values, respectively).

10. **simpls** fits a PLS regression model with the `simpls` method (`pls` function in the `pls` package, with `method=simpls`). The PLS method projects the inputs and the output to a new space and searches the direction in the input space which explains the maximum output variance, being particularly useful when there are more inputs than patterns and inputs are collinear. The `simpls` method [115] 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 (10 values).
11. **kernelpls** is the PLS regression with `method=kernelpls` [116] in the same function and package, using the same hyperparameter setting.
12. **enpls.fs** is an ensemble of sparse partial least squares regressors provided by the `enpls` package [134]. The number of components (1 value) is specified by the `caret` function `getModelInfo` for each dataset, but the `threshold` argument, specified as a hyperparameter by the `caret` list, is missing in the `enpls.fit` function.
13. **plsRglm** is the partial least squares generalized linear model (`plsRglm` package) with `modele=pls-glm-family` [4]. The hyperparameters are the number of extracted components and (4 values) and the input significance level (`alpha.pvals.expli`, 5 values).

## V. *Least absolute shrinkage and selection operator (lasso)*

14. **lasso** does lasso regression, using the `enet` function in the `elasticnet` package similarly to `ridge`, but using the `lambda` parameter equal to zero to obtain the lasso solution.
15. **relaxo** develops relaxed lasso (`relaxo` package), which generalizes the lasso shrinkage method for linear regression [73]. This method overcomes the trade-off between speed and convergence, specially for sparse high-dimensional patterns, in the L2-loss function of the regular lasso, providing sparser solutions with better prediction error. The relaxation and penalty hyperparameters  $\phi$  and  $\lambda$  are tuned using 7 and 3 values respectively.

## VI. Ridge (or Tikhonov) regression

16. **ridge** (elasticnet package), which uses the LARS-EN algorithm to compute the elastic net [136] regression model. Elasticnet provides a model for regularization and input selection, grouping together the inputs which are strongly correlated, being more useful when the number of inputs is higher than the number of patterns, as opposed to lasso models (see below). The only hyperparameter is the quadratic penalty (or regularization) parameter (10 non-zero values).
17. **spikeslab** implements the spike and slab regression [56] uses the spikeslab package to compute weighted generalized ridge regression estimators using Bayesian spike and slab models. This model combines filtering for dimensionality reduction, model averaging using Bayesian model averaging, and variable selection using the gnet estimator.
18. **foba** develops ridge regression with forward, backward and sparse input selection [135] (foba package). We use the adaptive forward-backward greedy version of foba (default value 'foba' for the 'type' argument of the foba function), which does a backward step when the ridge penalized risk increases in less than the parameter nu (0.5 by default) multiplied by the ridge penalized risk reduction in the previous forward step. The hyperparameters are regularization for ridge regression and the number of selected inputs (sparsity) for the prediction (10 and 2 values, respectively).

## VII. Neural networks

19. **mlpWeightDecay** is the multi-layer perceptron with one hidden layer and weight decay (mlp function in the RSNNS package, with learnFunc=BackpropWeightDecay). The size of the hidden layers and weight decay are tunable hyperparameters (5 values each one).
20. **mlpWeightDecayML** is the same network with three hidden layers, tuning their sizes (3 values each one) and the weight decay (5 values).
21. **avNNet** is the model averaged neural network provided by the caret package. A committee of 5 neural networks [110] of the same size is trained using different random seeds, being averaged to give an output. The hyperparameters are the network size and the weight decay (7 and 3 values, respectively).

22. **rbf** is the radial basis function network (RSNNS package) which does a linear combination of basis functions centered around a prototype [1]. The information is locally codified (opposed to globally in the MLP), the training should be faster and the network is more interpretable, although the output might be undefined if a test pattern does not activate any prototype. The only hyperparameter is the size of the hidden layer (10 values).
23. **grnn** is the generalized regression neural network [124] implemented by the Matlab neural network toolbox. The GRNN is a special type of RBF network: after a clustering of the training set, the nodes of the hidden layer store the cluster centers (the Matlab implementation uses so many clusters as training patterns). The output for a test pattern is a weighted sum of the Gaussian functions centered in the cluster centers, scaled by the cluster populations. During training, whenever a pattern is assigned to that cluster the weight of the Gaussian function corresponding to that cluster is updated using the desired output. The Gaussian spread is a hyperparameter (14 values): large (resp. small) values lead to smooth (resp. close) approximations.
24. **elm** is the extreme learning machine, which has been already introduced in the section 2.6. For the regression problem we use the elmNN package [53]. The only hyperparameters are the number of hidden neurons (20 values) and the activation function (sinus, radial basis, linear and hyperbolic tangent).
25. **elm-kernel** is the ELM neural network but with Gaussian kernel [53] using the publicly available Matlab code<sup>2</sup>. The hyperparameters are regularization  $C$  and kernel spread with values  $2^{-5}..2^{14}$  and  $2^{-16}..2^8$  (20 and 25 values, respectively).
26. **pcaNNet** is a multi-layer perceptron neural network with one hidden layer trained on the PCA-mapped training patterns, using the caret and nnet packages. The principal components which account for more than 95% of the data variance are used for training. With a test pattern, it is mapped to the principal component space and the trained pcaNNet model gives an output. Tunable hyperparameters are the size of the hidden layer and the weight decay of the network (7 and 3 values, respectively).
27. **bdk** is the supervised bi-directional Kohonen network, using the kohonen package [75]. The bdk combines Kohonen maps and counterpropagation networks, using two maps,

---

<sup>2</sup><http://www.extreme-learning-machines.org> (visited March, 29, 2017).

for inputs and output respectively. In each iteration, the direct (resp. inverse) pass updates only the weights of the input (resp. output) map, using a weighted similarity measurement (Euclidean distance for regression) which involves both maps, leading to a bi-directional updating. The test output is the weight of the winner node of the output map. The hyperparameters are the sizes of both maps (3 values each one) and the initial weight given to the input map in the distance calculation for the output map, and vice versa (2 values).

### VIII. *Deep learning neural networks*

28. **dlkeras** is the deep learning neural network using the Keras module [18] in Python, with three hidden layers tuned with 50 and 75 neurons for each layer (27 combinations). The deep learning methods [51, 69] are very popular, specially for image classification, and we included them in this comparison for regression tasks.
29. **dnn** implements a deep belief network in R using the DeepNet package using only one hidden layer and tuning the number of neurons with values from 5 to 60 with step 2. We tried with several hidden layers but the results are worse. The weights are initialized using stacked autoencoder (SAE), which gave better results than deep belief network (DBN).

### IX. *Support vector machines*

30. **svr** is the support vector machine for regression, with Gaussian kernel using the LibSVM library [15] with the C++ interface. We tuned the regularization hyperparameter  $C$  and the kernel spread  $\gamma$  with values  $2^{-5}..2^{14}$  and  $2^{-16}..2^8$ .
31. **svmRadial** is another implementation of SVR with Gaussian kernel (ksvm function in the kernlab package) for regression (type=eps-svr), using SMO to solve the quadratic SVM problem and tuning the same parameters as svr (5 and 4 values respectively).
32. **rvmRadial** is the relevance vector machine [129] with Gaussian kernel (same package). The RVM has the same functional form as the SVM, but using a Bayesian learning framework which reduces the number of basis functions, compared to the SVM, while keeping an accurate prediction. It also avoids tunable (e.g. regularization) hyperparameters of the SVM, but uses a method similar to Expectation-Maximization which

may fall in local minima, unlike SMO in the SVM. The Gaussian spread is fixed by the `getModelInfo` caret function.

## X. Regression trees

33. **rpart** is the recursive partitioning and regression tree [9] using the `rpart` package. Only the complexity parameter is tuned (10 values).
34. **nodeHarvest** is a simple interpretable tree-based ensemble for high-dimensional regression (`nodeHarvest` package) with sparse results [74]. A starting tree of few thousand nodes is randomly generated. For a test pattern assigned to a node, the output is the mean of its training outputs; when the test pattern is assigned to several nodes, the output is the weighted average of their means. The selection of the nodes and node weights requires to solve a quadratic programming problem with linear inequality constraints. Only few nodes with non-zero weights are selected, so the solution is sparse. The hyperparameters are the maximal interaction depth (10 values) and mode: mean (weighted group means) or outbag (zero values in the smoothing matrix diagonal).
35. **M5** is the model tree [102] using the `RWeka` package, tuning the flags `pruned` and `smoothed` (yes/no), and `rules/trees` of the Weka M5 implementation.
36. **ctree2** is the conditional inference tree (`party` package), which estimates the output using inference after a recursive partitioning the input space [52]. It tests the null hypotheses of statistical independence between any input and the output, and stop if it can not be rejected. Otherwise, it selects the input most related to the output, measured by the  $p$ -value of the partial test of independence between the output and that input. Then, it does a binary splitting of the selected input. Then, it recursively repeats the two previous steps. The hyperparameters are the threshold for  $1 - p$  in order to do a split (4 values) and the maximum tree depth (5 values).
37. **partDSA** develops partitioning using deletion, substitution, and addition [82], implemented by the `partDSA` package. This method recursively partitions the space considering that multiple inputs jointly influence the output, predicting a piecewise constant estimation through a parsimonious model of and/or conjunctions. The only hyperparameter is the maximum number of terminal partitions (`cut.off.grow`) is tuned with 10 values, and the parameter `vfold=1`.

38. **evtree** is a tree model from genetic algorithms [43] uses evolutionary algorithms to learn globally optimal regression trees. It chooses splits for the recursive partitioning in the forward stepwise search in order to optimize a global cost function. The only hyperparameter is the complexity parameter *alpha* of the cost function, with 10 values between 1 (default) and 3, which weights negatively large tree sizes.

## **XI. Bagging ensembles**

39. **bag** is the bagging ensemble of conditional inference regression trees [7] provided 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 MARS (bagEarth package), a bagging ensemble of MARS (see below) base regressors, provided by the earth package, which learns an Earth model for each bootstrap sample (25 samples are used by default). The hyperparameter is the maximum number of terms in the pruned regression model (10 values).
41. **trebag** is a bagged CART (function `ipredbag` in the `ipred` package), with 25 bootstrap replications (default) and base tree regressors in the `rpart` package.

## **XII. Boosting ensembles and gradient boosting machines**

42. **randomGLM** is a boosting ensemble of generalized linear models [122] provided by the `randomGLM` package. 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 tuned with 3 values). For each sample, inputs are ranked by its correlation with the output, and a predefined number of them are selected to create, using forward selection to create a multivariate GLM. For a test pattern, the predicted value is the average of the GLM outputs.
43. **BstLm** is the gradient boosting machine with linear regressors as base learners, provided by the `bst` package. Gradient boosting optimizes arbitrary differentiable loss functions defining the fitting criteria [35]. Boosting combines weak base regressors into a strong ensemble by iteratively adding base regressors, and in each iteration the new regressor is learned to fit the error (residual) of the previous ensemble. Since the error can be viewed as the negative gradient of the squared error loss function, we see that gradient boosting is a gradient descent method. The `BstLm` uses the `bst` function

with linear base regressors and Gaussian family, since squared error loss is used. The only hyperparameter is the number of boosting iterations (10 values).

44. **bstSm** is the gradient boosting with smoothing splines (`learner=smoothing spline`) as base regressors, tuning the number of boosting iterations (10 values). It is provided by the `bst` package.
45. **bstTree** is the gradient boosting with regression trees as base learners, provided by the `bst` package. The hyperparameters are the number of boosting iterations (4 values) and maximum depth of nodes in the final tree (parameter of `rpart.control`, in the `rpart` package, 5 values).
46. **glmboost** is the gradient boosting ensemble with GLMs as base learners (`glmboost` function in the `mboost` package), tuning the number of boosting iterations (10 values).
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 and number of trees for prediction (5 values each one). We use a Gaussian distribution and shrinkage equal to 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 [11], being more flexible than `gbm`. The only hyperparameter is the maximum tree depth (5 values).
49. **xgbTree** is the extreme gradient boosting [35] using the `xgb.train` function in the `xgboost` package with `booster=gtree` and linear regression as objective function. The hyperparameters are the tree maximum depth, maximum number of boosting iterations (3 values each one) and learning rate (2 values).
50. **xgbLinear** is the extreme gradient boosting with `booster=gblinear`, also provided by the `xgboost` package. Here the hyperparameters are the L2 (square loss) regularization term on weights (`lambda`, 3 values) and bias (`alpha`, 2 values), and the number of iterations (3 values).

### XIII. *Random forests*

51. **rf**, random forest (RF) ensemble of random regression trees provided by the randomForest package [8], but in this case random regression trees are used and the outputs of the base regressors are averaged, opposed to chapter 2. The only hyperparameter is the number of randomly selected inputs (mtry, 10 values).
52. **Boruta** uses RF with additional feature selection (Boruta package). An input is removed when a statistical test proves that it is less relevant than a shadow random input, created by shuffling the original ones [65]. Conversely, inputs that are significantly better than shadowed ones are confirmed. The iterative search stops when only confirmed inputs are retained, or after a maximum number of iterations (100 by default), case in which non-confirmed inputs remain unless the iterations or the test p-value (0.01 by default) are increased. In our case, if the number of inputs is less than 2, the RF works on the whole input set. The only hyperparameter is the mtry value of RF, with 10 values.
53. **RFR** is the regularized RF (homonymous package) uses regularization for input selection in RF, penalizing the selection of a new input for splitting when its Gini information gain is similar to the inputs included in the previous splits. The hyperparameters are the RF mtry parameter (2 values) and the regularization coefficient (coefReg, 2 values).
54. **cforest** is a RF ensemble of conditional inference trees [8], each one fitting one bootstrap sample, using the party package. Each ctree fits one bootstrap sample. The only hyperparameter (mtry, 2 values fixed by the caret getModelInfo function) of the conditional trees.
55. **qrf** is the quantile regression forest (quantregForest package) is a tree-based ensemble which generalizes RF in order to estimate conditional quantile functions. It grows several RFs, storing all the training patterns associated to each node in each tree; for each test pattern, its weight for each training pattern is calculated averaging its weight for all the training patterns and trees; using these weights, the distribution function of each output value, and the conditional quantiles, are estimated. The only hyperparameter is the RF mtry (2 values). The quantile prediction threshold (what) is set to 0.5.
56. **extraTrees** is the ensemble of extremely randomized regression trees [39] using the extraTrees package. It randomizes the input and cut-point of each split (or node in

the tree), using a parameter which tunes the randomization strength. It uses the full training set instead of a bootstrap replica. It is expected that explicit randomization of input and cut-point splittings combined with ensemble averaging should reduce strongly the variance than other methods. Three parameters are the number of inputs randomly selected at each node (`mtry`, tuned with 2 values), number of trees (500 by default) and the minimum sample size to split a node (`numRandomCuts`, tuned with 10 values).

#### XIV. *Prototype models*

57. **kknn** performs weighted k-nearest neighbors regression [61] using the `kknn` package, weighting the neighbors decreasingly to their similarities (calculated using the distances) to the test pattern. The only hyperparameter is the number `k` of neighbors (10 values). The default kernel (optimal) and distance (Minkowski) are used.
58. **cubist** learns a M5 rule-based model with corrections based on nearest neighbors in the training set [101]. It is provided by the `Cubist` package. A tree structure is created and translated to a collection of rules, which are pruned and combined, and each rule gives a regression model, which applies on the patterns which accomplish that rule. `Cubist` adds boosting with more than one training committees (hyperparameter tuned with 3 values). The number of neighbors (other hyperparameter also with 3 values) is used to correct the rule-based prediction. Hyperparameters are the number of training committees and the number of neighbors for prediction (3 values each one).

#### XV. *Bayesian models*

59. **bayesglm** is the Bayesian GLM (`arm` package): uses Expectation Maximization to update the betas in GLM at each iteration representing the prior information with an augmented regression [38]. The coefficients are calculated using a Student-t prior distribution.
60. **brnn** is the Bayesian regularized neural network (`brnn` package) is a network with one hidden layer trained using Gauss-Newton optimization minimizing a combination of squared error and a regularization term which uses the squared network weights [31]. The Bayesian regularization [70] determines the weights of the two terms based on inference techniques, which requires to iteratively compute the Hessian matrix (or its Gauss-Newton approximation) of the performance w.r.t. the weights and biases until a

goal is met or a maximum number of iterations is reached. The weights are not normalized, and the number of hidden neurons is a hyperparameter tuned with 15 values.

61. **bartMachine** is the Bayesian additive regression tree (bartMachine package). This model [62] consists of a sum of decision trees and a regularization process developed on the parameters of the tree set. We used the default number of trees (50, the unique value used by the caret tuning), and the hyperparameters are  $k$  (prior boundary, 3 values) and  $\alpha$  (base value in tree prior to decide if a node is terminal or not).

## XVI. *Principal component analysis*

62. **pcr** develops principal component regression, and it is provided by the pls package. The pcr models the output using classical linear regression with coefficients estimated with PCA, i.e., using the principal components as inputs in three stages [78]: perform PCA and select a subset of the principal components; use ordinary least squares to model the output vector using linear regression on the selected components; calculate the final pcr estimator transforming the modeled output vector to the original space using the eigenvectors corresponding to the selected components, and estimate the regression coefficients for the original outputs. The number of components is tuned with 10 values or, in some low-dimensional datasets, the number determined by the getModelInfo caret function.
63. **icr** is the independent component regression (caret package), i.e., a linear regression model fitted using independent component analysis (ICA), instead of PCA, provided by the fastICA package, instead of the original inputs [55]. In short, the input data are considered a linear combination of a number of independent and non-Gaussian components (sources), so the training set matrix is written as the product of the source matrix and a linear mixed matrix (coefficients of the linear combination). The ICA estimates a “separating” matrix, which multiplied by the original data provides the sources. This matrix must maximize the non-gaussianity of the sources, measured by the neg-entropy. The only hyperparameter is the number of independent components (10 values, always below the number of inputs).
64. **superpc** develops supervised PCA: it uses supervised principal component analysis retaining only a subset of the components which are correlated to the output. The

number of principal components (1-3) and the threshold for retaining the input scores (2 values). It is provided by the `superpc` package.

## XVII. *Generalized additive models*

65. **gam** is the generalized additive model [133] using splines with the `mgcv` package. The GAM is a GLM whose linear predictor is a sum of smooth functions (penalized regression splines) of the covariates. The estimation of the spline parameters uses by default the generalized cross validation criterion. The hyperparameter is the `select` parameter with values `true` and `false`: in the former case, an extra penalty term is added to each function penalizing its wiggleness (waving).
66. **gamboost** is the boosted generalized additive model [12] using the `mboost` package. This method is a gradient boosting ensemble which minimizes (computing its negative gradient) a weighted sum of the loss function evaluated at the training patterns. The base regressors are component-wise models (P-splines with a B-spline base, by default). The only hyperparameter is the number of initial boosting iterations (`mstop`), with 10 values.

## XVIII. *Gaussian processes*

67. **gaussprLinear** implements Gaussian process regression with linear (`vanilladot`) kernel using the `kernlab` package (function `gausspr`).
68. **gaussprRadial** uses the same function with Gaussian (`rbfdot`) kernel, with spread automatically calculated (option `kpar=1`).
69. **gaussprPoly** is the same method with polynomial (`polydot`) kernel, tuning the kernel hyperparameters degree and scale (3 values each one).

## XIX. *Quantile regression*

70. **rqlasso** develops quantile regression with LASSO penalty, using the `rq.lasso.fit` in the `rqPen` package. The quantile regression models optimize the so-called quantile regression error, which uses the tilted absolute value instead the square as RMSE. This tilted function applies asymmetric weights to positive/negative errors, computing conditional quantiles of the predictive distribution. The `rqlasso` method fits a quantile regression model with the LASSO penalty [80], tuning the `lambda` hyperparameter (10 values).

71. **rqnc** performs non-convex penalized quantile regression, with the `rq.nc.fit` function in the previous package. This regressor performs penalized quantile regression using local linear approximation [137] to maximize the penalized likelihood for non-convex penalties. The two hyperparameters are `lambdas` (10 values) and 2 penalty types: MCP (minimax concave penalty) and SCAD (smoothly clipped absolute deviation).
72. **qrnn** is the quantile regression neural network [13], with the `qrnn` package. A QRNN is a neural network where the transfer function is a ramp, being the error function the quantile regression error. The hyperparameters are number of hidden neurons and the penalty for weight decay regularization (7 and 3 values respectively).

## XX. Other methods

73. **lars** is the least angle regression [26] using the `Lars` package. `Lars` is a model selection method which is less greedy than the typical forward selection methods. It starts with zero coefficients for all the inputs and finds the input  $i$  most correlated with the output, increasing step-by-step its coefficient until another input  $j$  has high correlation with the current residual (error, or difference between desired and real output). `Lars` increases the coefficients of inputs  $i$  and  $j$  in the equiangular direction between inputs  $i$  and  $j$  until some other input  $k$  is so correlated with the residual as input  $j$ . Then, it proceeds in the equiangular direction among  $i$ ,  $j$  and  $k$ , which is the “least angle direction”, and so on until all the coefficients are non-zero (i.e., all the inputs are in the model). The `lasso` and `fraction` options are specified for training and prediction respectively, and the `fraction` hyperparameter is tuned with 10 values. The number of terms is tuned with 10 values.
74. **earth** is the multivariate adaptive regression spline (MARS) using the `earth` package [34]. It uses an expansion of product spline functions to model non-linear data and interactions among inputs. The spline number and parameters are automatically determined from the data using recursive partitioning, and distinguishing between additive contributions of each input and interactions among them. The functions are added iteratively to reduce maximally the residual, until its change is too small or a number of iterations is reached. The maximum number of terms in the model is tuned with 15 values.
75. **ppr** performs the projection pursuit regression [36], in the `stats` package. The `ppr` models the output as a sum of averaging functions (mean, median, etc.) of linear combina-

tions of the inputs. 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.

76. **sbc** is subtractive clustering and fuzzy c-means rules [17], using the `frbs` package. This method uses SBC to get the cluster center of a fuzzy rule-based system for classification or regression. Initially, each training pattern is weighted by a potential function which decreases with its distances to the remaining centers. The center with the highest potential is selected as a cluster center, and the potential of the remaining centers are updated. The only hyperparameter is the neighborhood radius, tuned with 7 values usually between 0 and 1. The selection of new cluster center and potential updating is repeated until the potentials of the remaining patterns are below a pre-specified fraction of the potential of the first cluster center. Once all the centers are selected, they are optimized using fuzzy C-means.

## 3.2 Experimental setup

We run the 76 previous regressors on a collection composed by 66 regression datasets selected from the UCI Machine Learning Repository [3]. For the current study, we selected 44 out of the 63 datasets labeled as “Regression” by the UCI repository<sup>3</sup>, plus other two datasets (Breast cancer Wisconsin original and prognostic, included in the “Classification” UCI list), which are listed in Table 3.1. For space reasons, the table only lists the first two words of the original name of each dataset. The remaining 26 datasets were discarded due to diverse reasons. The “Amazon Access Samples” dataset does not explain what is the output. The “Educational Process Mining (EPM): A Learning Analytics Data Set” has time-series data with a very complex format. The “Challenger USA Space Shuttle O-Ring” has too few patterns (23) and inputs (3). The data are missing in the “Improved Spiral Test Using Digitized Graphics Tablet for Monitoring Parkinson’s Disease” dataset. The “NoisyOffice” dataset is difficult to read because it is composed by printed text images. We were unable to decide the output for the “Tennis Major Tournament Match Statistics” dataset. For large datasets, only the first 2000 patterns are used. The reason is that many regressors are not able to train and test with large datasets, and they give memory errors or simply they do not ever finish.

---

<sup>3</sup><http://archive.ics.uci.edu/ml> (visited March, 29, 2017).

Although we selected 46 UCI datasets, some of them generated several datasets, one for each data column which can be used as output for regression. This is the reason why some original datasets in column 1 give several datasets in column 2 (e.g., the Air quality dataset gives five datasets). Therefore, we achieved a list of 66 datasets. All the constant, repeated and collinear inputs, whose NA coefficients in a linear regression model using the `lm(...)` function in the R stats package, are removed from the dataset. For example, the “Blog feedback” dataset reduced its inputs from 280 to 13. On the contrary, those inputs with discrete values, they are replaced by dummy (also called indicator) inputs. For each discrete input with  $n$  values, it is replaced by  $n - 1$  dummy inputs with values zero or one: the first value of the original discrete input is codified as zero values for the  $n - 1$  dummy inputs; the second value is codified as 1 in the first dummy variable and zero in the remaining ones; the third value is codified as 1 in the second dummy input, and so on. Therefore, those datasets with discrete inputs increase the number of inputs, e.g. dataset “Student perf. mat.” increases its inputs from 32 to 77. In the Table 3.1 those #inputs column where appear two numbers (i.e. 8/23), the first one is the original #inputs, and the second one is the number of inputs used effectively by the regressors, after removing constant, repeated and collinear inputs, and after replacing discrete inputs by their corresponding dummy variables.

Ten random partitions are generated for each dataset, using the 50% for training, 25% for validation (in hyperparameter tuning) and 25% for test. Each regressor is trained on the ten training partitions for each combination of its hyperparameter values, and tested on its corresponding validation partition [64]. The performance measure that we use is the root mean square error (RMSE), defined as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - d_i)^2} \quad (3.1)$$

where  $N$  is the number of test patterns, being  $y_i$  and  $d_i$  the regressor and the desired (right) outputs respectively. For each combination, the average RMSE over the ten partitions is calculated, and the one with the lowest RMSE is selected for testing. Finally, the regressor is trained, using this selected combination of its hyperparameter values, on the ten training partitions, and tested on the ten test partitions. The average RMSE over the ten test sets is used as the final quality measure. Some regressors which are specially sensitive to collinear inputs are trained, for each partition, using only those inputs which are not collinear. Although collinear inputs have been removed from the dataset in the initial preprocessing, sometimes it happens that for certain partition the training patterns have several collinear inputs, which are

Original	Datasets	#Patterns	#Inputs	Original	Datasets	#Patterns	#Inputs
Air quality	CO NMHC NO2 NOx O3	1230	8	GPS trajectories		163	10
				Greenhouse gas		2000	326
				Housing		452	13
				Individual household		2000	6/5
				Insurance company		2000	85/439
Auto_MPG		398	8/23	Istanbul stock		536	8
Automobile		205	26/66	KEGG reaction		2000	27/25
Bike sharing	day hour	731 2000	13/30 14/42	KEGG relation		2000	22/17
				Online news		2000	59/55
Blog feedback		2000	280/13	Online video	video-trans.	2000	20/8
Breast Wisc.	wbc	699	9/81	Parkinson speech		1040	26
B. W. diagnosis	wdbc	569	30	Parkinsons tel.	motor total	2000	16
B. W. prognostic	wdbc	198	33/32				
Buzz social	buzz-twitter	2000	77	Physicochem. prop.		2000	9
Com. & crime	com-crime	1994	122	Relative CT slices		2000	385/355
C. & C. unnorm.		2000	124/126	Servo		167	4/15
Computer hardware	com-hd	209	7	Skillcraft master		2000	18
Concrete compressive	cond-compress	1030	8	SML2010		2000	21/18
Concrete slump	slump comp flow	103	9 7	Student perf.	mat por	395 649	32/77 32/56
				Tamilnadu electr.	elec.hour.	2000	4/3
				Twin gas sensor		2000	8
Condition based	compress turbine	2000	16/13	UJIIndoorLoc	floor lat long	1999	528/373
Energy efficiency	cool heat	768	8/7				
Fertility		100	9/16	UJIIndoorLoc-Mag	curve-lat curve-long line-lat line-long	2000	9 10
Forestfires		517	12/39				
Gas sensor drift	gas-drift	2000	129				
Gas sensor flow	gas-flow	58	438/57				
Geogr. origin	geo-lat geo-long geo-music-lat geo-music-long	1059	116/72	wiki4HE		913	52/240
				Wine quality	red white	1599	11
			Yacht hydro.				2000
			68			308	6

**Table 3.1:** Collection of 66 datasets from the UCI repository: original name in the UCI repository; datasets created from the original one; number of patterns and inputs, before and after preprocessing.

not collinear considering the whole dataset. That is the reason why these inputs are discarded for these regressors, which otherwise give errors. This happens for regressors earth, evtree, foba, glm, icr, krlsRadial, lasso, lm, nnls, nodeHarvest, plsRglm, qrnn, ridge, SBC and spls. Obviously, discarding collinear inputs by partitions only happens for certain datasets. All the inputs and the output are pre-processed to have zero mean and standard deviation one. Some regressors gave errors for some datasets. In these cases, the programs are configured to give the mean of the desired outputs, which is zero. Most regressors have one, two, three or four

tunable hyperparameters. As well as the RMSE, another performance measure that we also use is the correlation coefficient, defined as:

$$\text{Correlation} = \frac{\left\{ \sum_{i=1}^N (y_i - \bar{y}) \right\} \left\{ \sum_{i=1}^N (d_i - \bar{d}) \right\}}{\sigma(y)\sigma(d)} \quad (3.2)$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i, \quad \bar{d} = \frac{1}{N} \sum_{i=1}^N d_i \quad (3.3)$$

$$\sigma(y) = \sqrt{\frac{1}{N} \left( \sum_{i=1}^n (y_i - \bar{y})^2 \right)}, \quad \sigma(d) = \sqrt{\frac{1}{N} \left( \sum_{i=1}^n (d_i - \bar{d})^2 \right)} \quad (3.4)$$

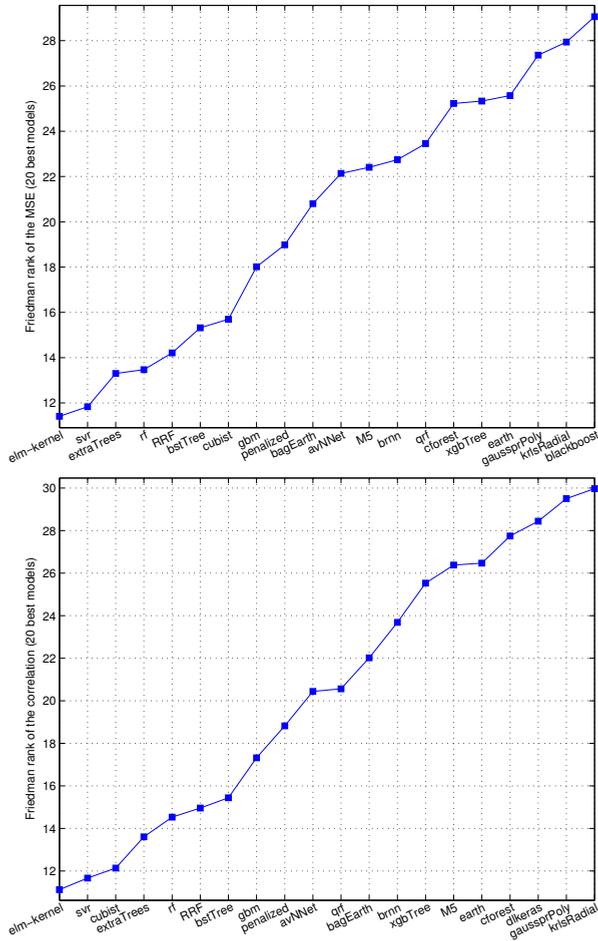
where  $\bar{y}$  and  $\sigma(y)$  are the mean and standard deviation of  $y_1, \dots, y_N$ , respectively, and the same for  $d_1, \dots, d_N$ .

Regressor	#Datasets	Datasets
nodeHarvest	27	bike-day, UJ-lat, twin-gas-sensor, buzz-twitter, air-quality-NMHC, video-transcode, com-crime-unnorm, geo-lat, wine-red, air-quality-O3, skill-craft, KEGG-relation, blog-feedback, gas-drift, SML2010, UJ-MagLine-lat, wine-white, air-quality-NO2, com-crime, UJ-MagLine-long, air-quality-NOx, bike-hour, physico-protein, KEGG-reaction, park-total-UPDRS, park-motor-UPDRS and UJ-long
qrmn	6	greenhouse-net, CT-slices, UJ-long , UJ-floor , UJ-lat, insurance-coil
brnn	5	wiki4HE, insurance-coil, UJ-long , UJ-floor , UJ-lat
randomGLM	3	gas-drift, insurance-coil and com-crime-unnorm
rqlasso	3	wiki4HE , wbc , insurance-coil
rqnc	2	wiki4HE, insurance-coil

**Table 3.2:** List of regressors (and datasets) whose execution did not finish within 150 h. or required more than 128 GB RAM.

### 3.3 Results and discussion

We run a collection of 76 regressors over 66 datasets, developing a total of 5016 experiments. These experiments were developed on the CiTIUS cluster, using Intel Xeon E5-2650L microprocessors with 64 GB RAM. As commented above, those regressors which gave errors for some partitions or combinations of the tunable hyperparameters are evaluated as if they give the mean output, which is zero because the desired output is preprocessed to have zero mean. Additionally, some regressors did not finish for certain datasets (see Table 3.2), either due to a huge memory consumption (more than 128 GB) or computation time (an upper limit



**Figure 3.1:** Friedman rank of RMSE (upper panel) and correlation (lower panel) for the 20 best regressors.

of 150 hours or 6.25 days was set). These combinations of regressor-dataset which did not finish are 46 of 5016, which represents 0.92% of the total experiments. The nodeHarvest had a specially bad behavior, because it did not finished for 27 datasets, which represents 40.9% of the datasets, and randomGLM overcame the memory limit for 3 datasets. The remaining regressors did not finish within the 150 hours deadline.

In order to develop a comparison of the different regressors over the collection of datasets, we used the Friedman rank [37] of the RMSE for the whole regressor collection. This rank

Order	RMSE rank		Correlation rank		Avg. correl.	<i>p</i> -value
	Regressor	Rank	Regressor	Rank		
1	elm-kernel	11.4	elm-kernel	10.5	0.79198	–
2	svr	11.8	svr	11.7	0.79432	1
3	extraTrees	13.4	cubist	12.1	0.78843	1
4	rf	13.4	extraTrees	13.7	0.79071	1
5	RRF	14.3	rf	14.6	0.78632	1
6	bstTree	15.3	RRF	15.0	0.78611	1
7	cubist	15.5	bstTree	15.6	0.77836	1
8	gbm	17.8	gbm	17.4	0.77304	1
9	penalized	19.0	penalized	18.6	0.77478	0.99999
10	bagEarth	20.6	avNNet	20.4	0.76830	0.99001
11	avNNet	22.1	qrf	20.7	0.77053	0.98697
12	M5	22.6	bagEarth	21.9	0.75025	0.83450
13	brnn	22.7	brnn	23.7	0.70599	0.56961
14	qrf	23.7	xgbTree	25.1	0.75767	0.20598
15	earth	25.2	earth	26.4	0.74607	0.05424
16	xgbTree	25.3	M5	26.7	0.75375	0.08772
17	cforest	25.4	cforest	27.8	0.73645	<b>0.02051</b>
18	gaussprPoly	27.4	dlkeras	28.4	0.72999	0.01010
19	krlsRadial	27.9	gaussprPoly	29.5	0.69821	0.00290
20	blackboost	29.1	krlsRadial	29.9	0.65848	0.00129
21	treebag	30.5	ppr	30.4	0.72542	0.00084
22	bag	32.6	treebag	31.1	0.74875	0.00035
23	grnn	32.7	blackboost	31.7	0.68776	0.00011
24	kknnc	33.1	rvmRadial	32.2	0.72535	0.00008
25	lars	33.3	kknnc	33.3	0.73088	0.00001
26	foba	33.5	svmRadial	33.4	0.73238	0.00001
27	ppr	33.8	foba	33.8	0.68832	0.00001
28	spikeslab	33.9	grnn	33.9	0.72847	0
29	dlkeras	34.1	lars	34.2	0.71300	0
30	rvmRadial	34.2	gaussprRadial	34.7	0.72400	0
31	glmboost	35.2	simpls	36.1	0.70851	0
32	gaussprRadial	35.5	bag	36.3	0.69490	0
33	svmRadial	36.3	glmboost	36.8	0.67369	0
34	simpls	36.8	spikeslab	36.9	0.67195	0
35	kernelpls	38.2	ridge	37.4	0.66220	0
36	BstLm	38.7	kernelpls	37.7	0.70214	0
37	spls	39.6	lasso	39.1	0.70074	0
38	icr	39.7	plsRglm	39.2	0.67618	0

**Table 3.3:** Friedman rank of the RMSE (left) and of the correlation (right), average correlation and *p*-value of the Posthoc Friedman Nemenyi test comparing the best regressor to the remaining ones. Continued in Table 3.4.

evaluates the position in which each regressor is, in average over all the datasets, when the measure is sorted by decreasing performance (in our case, by increasing RMSE). This rank is calculated as follows: let  $r_{ij} = 0$ , for  $i = 1, \dots, n$  and  $j = 1, \dots, m$ , being  $n$  and  $m$  the

RMSE rank			Correlation rank		
Order	Regressor	Rank	Regressor	Rank	Avg. correl.
39	plsRglm	39.7	bayesglm	39.4	0.67650
40	ctree2	40.3	rqlasso	39.6	0.68176
41	evtree	41.2	glm	39.8	0.66703
42	SBC	41.2	spls	39.9	0.68100
43	rqlasso	41.6	BstLm	40.1	0.67797
44	rpart	41.6	SBC	40.8	0.66650
45	lasso	41.7	gaussprLinear	40.8	0.66061
46	ridge	42.0	lm	40.8	0.66703
47	glmStepAIC	42.7	glmStepAIC	41.1	0.66918
48	bayesglm	43.0	icr	41.6	0.67397
49	glm	43.2	rlm	41.8	0.66732
50	randomGLM	43.3	rqnc	42.0	0.68974
51	relaxo	43.4	rpart	42.2	0.70422
52	lm	44.2	gam	42.5	0.65881
53	gaussprLinear	44.3	evtree	42.5	0.67557
54	nodeHarvest	44.3	randomGLM	42.8	0.51610
55	rqnc	44.4	ctree2	43.6	0.68371
56	elm	44.7	relaxo	45.3	0.53902
57	rlm	44.8	nodeHarvest	46.5	0.39834
58	gamboost	45.1	elm	47.9	0.66668
59	gam	45.3	gamboost	48.2	0.34728
60	rbf	48.1	xgbLinear	48.9	0.54951
61	bstSm	48.3	pcaNNet	49.1	0.63487
62	xgbLinear	49.3	qrnn	50.1	0.34677
63	npls	50.6	npls	50.4	0.59542
64	qrnn	51.0	mlpWeightDecay	50.6	0.64055
65	enpls.fs	53.1	bstSm	51.3	0.26817
66	partDSA	56.3	rbf	51.9	0.55881
67	pcaNNet	57.6	enpls.fs	55.1	0.44590
68	mlpWeightDecay	57.8	partDSA	58.6	0.55021
69	pcr	59.0	bdk	61.1	0.48553
70	mlpWeightDecayML	61.2	superpc	61.5	0.39352
71	Boruta	61.4	pcr	64.0	0.39539
72	dnn	61.6	mlpWeightDecayML	64.0	0.46150
73	superpc	62.1	dnn	65.9	0.36021
74	bdk	62.5	Boruta	67.7	0.01491
75	bartMachine	72.2	bartMachine	73.8	-0.03007
76	glmnet	75.8	glmnet	74.7	-0.04054

Table 3.4: Continuation of Table 3.3.

number of regressors and datasets, respectively. For each dataset  $j$ , the RMSE values of all the regressors are sorted increasingly. For each regressor  $i$  let  $r_{ij} = k$ , where  $k$  is the position of the regressor  $i$  for dataset  $j$ . The Friedman rank of regressor  $i$  is the average of  $r_{ij}$  for  $j = 1, \dots, m$ , i.e., the average position of regressor  $i$  over all sorting of RMSE for the

different datasets. For example, a rank of 5 means that the regressor is, in average over all the datasets, the 5th best regressor.

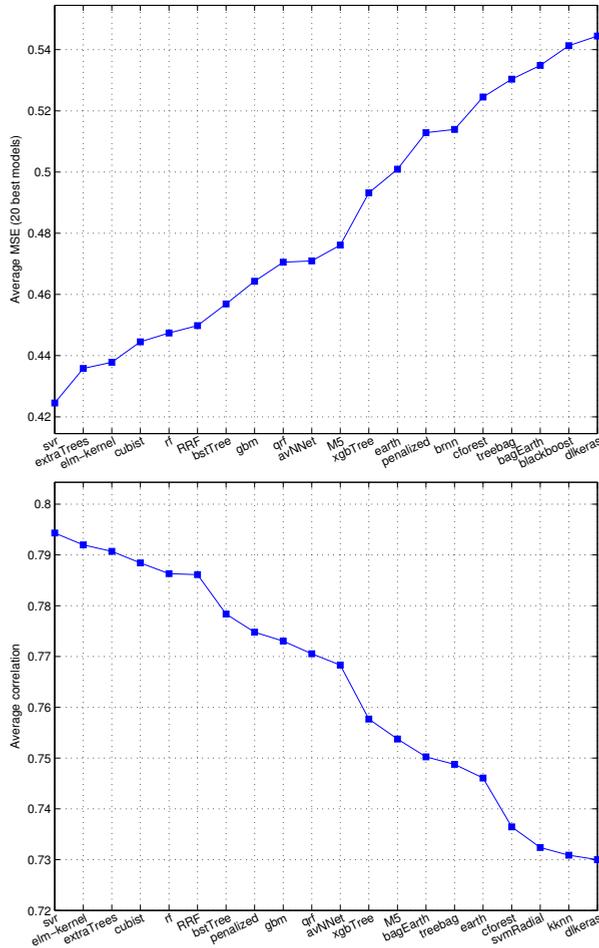
The Figure 3.1 plots the Friedman rank for the RMSE and correlation (in this case, the correlation must be sorted decreasingly) for the best 20 regressors. The extreme learning machine (elm-kernel) achieves the best rank, followed by the support vector regression (svr) with LibSVM and Gaussian kernel, extraTrees (a forest of extremely randomized regression trees) and the random forest (rf). Both the RMSE and correlation ranks are very similar, specially in the first positions. The Tables 3.3 and 3.4 report the complete Friedman ranks for RMSE and correlation for the 76 regressors. The column “Avg. correl.” reports the average correlation of each regressor over all the datasets. We do not report the average RMSE of each regressor over all the datasets because this measure has different ranges for each dataset, depending on the data complexity, so we can not average them directly. However, the correlation coefficient ranges between -1 and +1 for any dataset, so it can be averaged.

RMSE				Correlation			
Regressor	#times best	Regressor	#times best	Regressor	#times best	Regressor	#times best
penalized	17	SBC	2	penalized	16	SBC	2
svr	10	brnn	2	svr	11	brnn	2
extraTrees	9	qrnn	2	extraTrees	9	bstTree	2
cubist	6	elm-kernel	1	cubist	6	qrnn	2
avNNet	4	grnn	1	avNNet	5	elm-kernel	1
gbm	4	rf	1	qrf	4		
qrf	4	bstTree	1	glm	3		
bagEarth	2			gbm	3		

**Table 3.5:** List of the regressors which achieve the best RMSE (left part) or correlation (right part) for some dataset.

Following the suggestions in [22], we also developed a post-hoc Friedman-Nemenyi test, using the PMCMR package [99] provided by the R language, in order to evaluate the statistical significance of the differences in the correlation coefficients of the different regressors. The last column of Table 3.3 reports the  $p$ -values of the comparisons between the best regressor (elm-kernel) and the remaining ones. The  $p$ -values of the 9 best regressors are almost 1, and it decreases very fast in such a way that for cforest (position 17) the  $p$ -value is already under 0.05 (the threshold value). Therefore, the differences between elm-kernel and regressors under position 17 are statistically significant.

Given the meaning of the Friedman rank, it is also interesting to see, for each regressor, how many datasets it achieves the best result. The Table 3.5 lists the regressors which achieve



**Figure 3.2:** Average RMSE (upper panel) and correlation (lower panel) over all the datasets for the 20 best regressors.

the lowest RMSE (left columns) and the highest correlation (right columns) for some dataset. Although both lists are not exactly equal, three regressors overcome on both lists: penalized, with 16-17 hits, svr (10-11 hits) and extraTrees (9 hits). In fact, the three regressors represent 36 of 66 datasets. Other regressors which are the best for some datasets are cubist (a M5 rule-based regressor with corrections based on nearest neighbors in the training set, which belongs to the family of prototype models), avNNet, gbm and qrf. The elm-kernel, which is globally

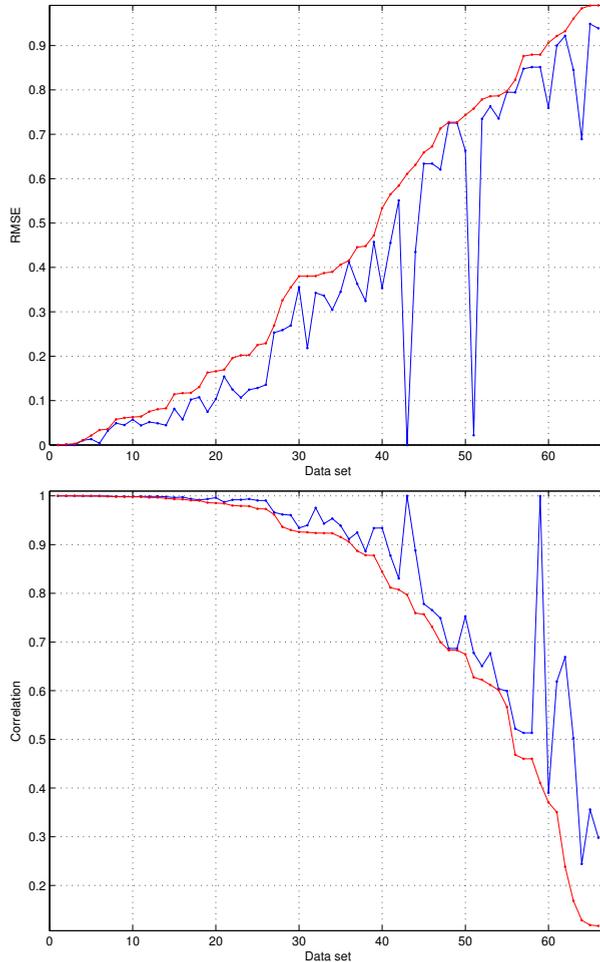
the best one, is only the best for one dataset. This is not surprising, because a regressor may be the best for several datasets and however work poorly in other cases, so a global evaluation is required. In fact, a regressor may be very near to the best result for several datasets and never be the best.

The Figure 3.2 plots, for the 20 best regressors according to the Friedman rank, the average RMSE and correlation over all the datasets. Note that averaging RMSE is not statistically acceptable, because the RMSE takes values in ranges which are different for each dataset. However, this does not happen with correlation, so averaging correlation (right panel) is statistically correct. Therefore, we only include the average RMSE plot for illustrative purposes, while the most relevant one is the right plot with the average correlation. Anyway, both plots show results similar to the Friedman rank, with the difference that *svr* is slightly better than *elm-kernel*. The absolute values of the correlation are also relevant in order to see how well, in average, behave the best regressors. We see that the highest average correlation is not so high (0.8), so the datasets are not so easy to predict.

The Figure 3.3 plots the best correlation achieved by some regressor, which in general is different, for each dataset sorted decreasingly (38 regressors in each panel). For 41 of 66 datasets, the correlations achieved by some regressor are above 0.9, which is a very good result. Among the 25 remaining datasets, 8 have correlations between 0.7 and 0.9, which are moderate, but for the last 17 datasets no regressor was able to learn correctly the data, with correlation values under 0.7 up to 0.2.

In order to see how well the best regressor (*elm-kernel*) behaves, it is interesting to compare, for each dataset, the best regressor (according to RMSE or correlation) and the result (RMSE or correlation) achieved by *elm-kernel* (see Figure 3.4). With respect to RMSE (upper panel), the *elm-kernel* follows the best value, excepting two datasets whose best RMSE is near zero (*gas-flow-mod*, where *qrf* achieves  $RMSE=0$  while *elm-kernel* achieves  $RMSE=0.54$ , and *greenhouse-net*, where *penalized* achieves  $RMSE=0.02$  while *elm-kernel* achieves  $RMSE=0.75$ ) and another one for what the difference is about 0.2. The third dataset where *elm-kernel* is clearly below the best is *wpsc*, where *penalized* and *elm-kernel* achieve  $RMSE=0.69$  and  $RMSE=0.98$ , respectively. Remember that *penalized* is in the 9th position of the Table 3.3 with a Friedman rank of 18.6, which means that in average over all the datasets, *penalized* is the 18th best regressor. Considering correlation (lower panel of Figure 3.4), the differences between *elm-kernel* and the best result seem to be reduced with respect to RMSE: for the 37 datasets in the left, the correlation of *elm-kernel* is above 0.9, and the differences for the 29

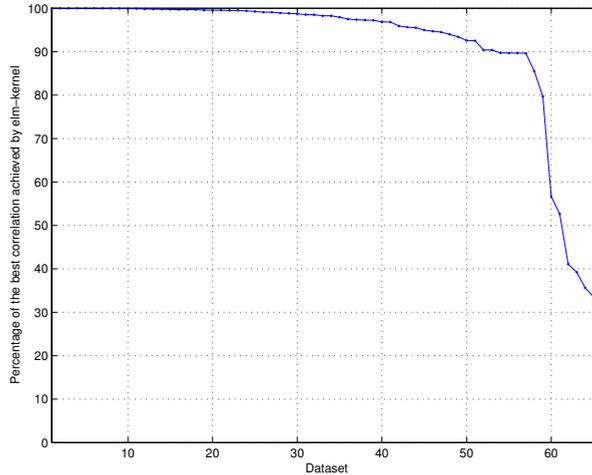




**Figure 3.4:** Upper panel: best RMSE, in blue, and RMSE achieved by elm-kernel (in red) for each dataset, sorted by increasing elm-kernel RMSE. Lower panel: the analogous plot for correlations, sorted by decreasing values.

remaining datasets are not very high, excepting the three datasets that we mentioned for the RMSE. However, for dataset *gas-flow-mod*, the correlation of elm-kernel is 0.8, which is not too low. For dataset *greenhouse-net* the difference in RMSE also reflects in correlation (0.91 and 0.47 for penalized and elm-kernel, respectively). The correlation values of elm-kernel are

low on the right end of the plot, but the best correlations are also bad (e.g. 0.67 and 0.24 for penalized and elm-kernel), so the behavior of elm-kernel is expectable.



**Figure 3.5:** Percentage of the best correlation achieved by the elm-kernel, for each dataset, sorted by decreasing values.

The Figure 3.5 plots the percentage of the best correlation achieved by the elm-kernel, for all the datasets. For most datasets, the elm-kernel achieves a correlation which is above 90% of the best correlation achieved by any regressor. Specifically, for 57 of 66 datasets, which represents 86.4% of the datasets, elm-kernel is above or about 90% of the best correlation, only for 9 datasets it is below 90%, and it never decreases under 30%.

The Table 3.6 reports, for each one of the 20 regressor families, the best regressor of that family, its position in the Friedman rank for correlations (see Tables 3.3 and 3.4), its Friedman rank and its average correlation. The families are sorted by the position of their best regressor in the Friedman rank. The neural networks and support vector machines achieve the two best results, but there are many other families with good results, as prototype models, whose best regressor cubist (M5 rule-based model with corrections based on nearest neighbors) achieves the 3rd position in the Friedman rank, random forests (4th position), boosting (gradient boosting with regression trees, 7th position) and generalized linear regression (penalized linear regression, 9th position).

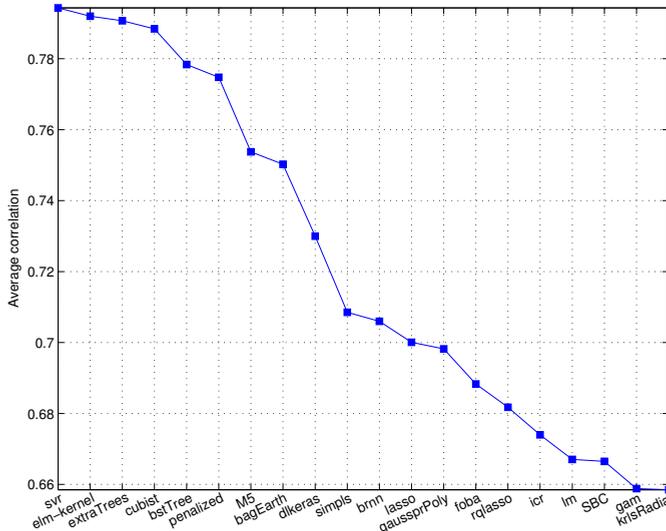
The Figure 3.6 shows a slightly different family sorting, by decreasing average correlation. The plot shows a group of families, composed by the four bests (svr, elm-kernel, extraTrees

Order	Family	Best regressor	Position	Friedman rank	Avg. correl.
1	Neural networks	elm-kernel	1	11.1	0.79198
2	Support vector machines	svr	2	11.7	0.79432
3	Prototype models	cubist	3	12.1	0.78843
4	Random forests	extraTrees	4	13.6	0.79071
5	Boosting ensembles	bstTree	7	15.4	0.77836
6	Generalized linear regression	Penalized	9	18.8	0.77478
7	Bagging ensembles	bagEarth	12	22.0	0.75025
8	Bayesian models	brnn	13	23.7	0.70599
9	Regression trees	M5	15	26.4	0.75375
10	Deep learning	dlkeras	18	28.4	0.72999
11	Gaussian processes	gaussprPoly	19	29.5	0.69821
12	Least squares	krlsRadial	20	30.0	0.65848
13	Ridge	foba	28	34.0	0.68832
14	Partial least squares	simpls	31	36.1	0.70851
15	Lasso	lasso	39	39.1	0.700074
16	Quantile regression	rqlasso	40	39.6	0.68176
17	Other methods	SBC	44	40.8	0.66650
18	Linear regression	lm	45	41.0	0.66703
19	Principal component analysis	icr	48	41.7	0.67397
20	Generalized additive models	gam	53	42.4	0.65881

**Table 3.6:** Regressor families sorted by the Friedman rank of its best regressor according to the correlation coefficient (its average value is reported in the last column).

and cubist) with average correlations above 0.78. A second group composed by bstTree and penalized are about 0.78, and a third group composed by M5 (family regression trees) and bagEarth (family bagging) are about 0.75. After these regressors, the average correlations decrease very fast: dlkeras (family deep learning) is about 0.73, and the following four families are already about 0.70: simpls (partial least squares), brnn (Bayesian models), lasso and gaussprPoly (Gaussian processes). The last seven regressors are clearly under 0.7: foba, rqlasso, icr, lm, SBC, gam and krlsRadial, which belong to families ridge, quantile regression, principal component analysis, linear regression, other methods, generalized additive models and least squares, respectively.

The Table 3.7 reports the positions in the Friedman rank of the regressors within families, excluding the best regressor of each family. We would highlight some issues. 1) For *neural networks*, the good results of avNNet (ensemble of multi-layer perceptrons) and grnn (generalized regression neural networks), and the bad results of the elm (extreme learning machine, much worse than elm-kernel) and rbf (radial basis functions). The case of elm suggests that it is competitive mainly with Gaussian kernel. 2) With *support vector machines*, the rvmRadial (relevance vector machine with Gaussian kernel) works even slightly better than svmRadial.



**Figure 3.6:** Average correlation (sorted decreasingly) of the best regressors of each family.

Since the `svmRadial` in the `kernlab` package seems to use the same `LibSVM` implementation as `svm`, perhaps the cause of the difference between them is the reduced collection of hyperparameter values for `svmRadial`, which are generated by the `getModelInfo` function of the `caret` package. 3) All the *random forest* regressors work well (within the first 17 positions), except `Boruta`, which suggests that the feature selection is not working properly. 4) Among *boosting* regressors, the `gbm` (position 8, just one below `bstTree`) and `xgbTree` are competitive. 5) The R implementation of the *deep learning* (`dnn`) is not competitive to the `Keras` module in Python.

The discussion about elapsed time spent by each regressor leads us to develop a Friedman rank of these times over all the datasets. We can not simply average the times of each regressor over all the datasets due to the large difference between the times spent for different datasets, whose number of patterns and inputs are very different. The measurements include training and test times for test stage, divided by the number (10) of test trials. We do not include the time spent for hyperparameter tuning, because it is obviously conditioned by the number of tunable parameters and the number of values tried for each hyperparameter. The Table 3.8 reports the Friedman rank for the times over all the datasets. The first conclusion is that, unfortunately, the fastest regressors are also worst ones: `dnn`, `Boruta`, `pcr`, `glm` and others.

Family	Regressor	Pos.	Regressor	Pos.	Regressor	Pos.	Regressor	Pos.
Neural networks	avNNet	10	grnn	27	elm	58	pcaNNet	61
	mlpwd	64	rbf	66	bdk	69	mlpwdML	72
Support vector machines	rvmRadial	24	svmRadial	26				
Prototype models	kknn	25						
Random forests	rf	5	RRF	6	qrf	11	cforest	17
	Boruta	74						
Boosting ensembles	gbm	8	xgbTree	14	blackboost	23	glmboost	33
	BstLm	43	randomGLM	54	xgbLinear	60	bstSm	65
Generalized linear regression	glm	42	glmStepAIC	46	glmnet	76		
Bagging ensembles	treebag	22	bag	32				
Bayesian models	bayesglm	37	bartMachine	75				
Regression trees	rpart	51	evtree	52	ctree2	55	nodeHarvest	57
	partDSA	68						
Deep learning	dnn	72						
Gaussian processes	gaussprRadial	30	gaussprLinear	47				
Least squares	kernelpls	35	plsRglm	38	spls	41	enpls.fs	67
Ridge	spikeslab	34	ridge	36				
Partial least squares	npls	63						
Lasso	relaxo	56						
Quantile regression	rqnc	49	qrnn	62				
Other methods	lars	29	earth	16	ppr	21		
Linear regression	lm	45	rlm	50				
Principal component analysis	superpc	70	pcr	71				
Generalized additive models	gamboost	59						

**Table 3.7:** Positions of the regressors, grouped by families, in the Friedman rank (data extracted from Tables 3.3-3.4), in the same order as Figure 3.6. The acronym mlpwd means mlpWeightDecay.

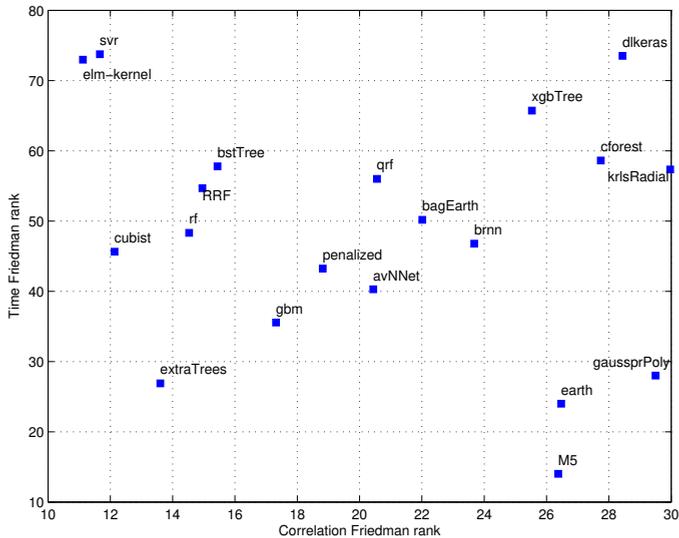
Conversely, the two slowest regressors are the best ones: elm-kernel and svr. For illustrative purposes, the average times spent by elm-kernel and svr are 1772 seconds and 5705 seconds respectively, while the fastest regressor (dnn) spent only 3.21 seconds. Actually, the elm-kernel is about 5 times faster than svr. Among the 20 regressors with the highest correlations, the M5 exhibits the best time rank, being in the position 16 of the correlation rank (average time 4.13 seconds) and in the position 6 of the time rank. Besides, regressor earth is in the positions 15 and 17 of the correlation and time ranks, respectively (average time 6.0 seconds). They are the only regressors which are among the 20 first of both ranks.

Since there is no clear trade-off between correlation and time, we can discuss the times of the 20 best regressors according to correlation. The Figure 3.7 plots the time rank against the correlation rank for these regressors. The two best regressors elm-kernel and svr are very high on the left, which means low correlation rank (good performance) and high time rank (low speed). Perhaps the best trade-off between correlation and time might be provided by

Pos.	Regressor	Rank	Pos.	Regressor	Rank
1	dnn	2.55	39	gaussprLinear	36.09
2	Boruta	5.53	40	rqnc	38.82
3	pcr	7.89	41	icr	39.14
4	glm	12.11	42	avNNet	40.29
5	ridge	12.52	43	plsRglm	40.48
6	M5	14.02	44	treebag	42.30
7	glmnet	16.98	45	penalized	43.24
8	superpc	17.00	46	gamboost	44.79
9	kknn	19.55	47	glmStepAIC	45.17
10	rvmRadial	20.70	48	cubist	45.64
11	foba	20.91	49	brnn	46.79
12	gam	21.29	50	partDSA	47.12
13	kernelpls	21.91	51	rf	48.32
14	rpart	23.20	52	bag	49.00
15	rqlasso	23.73	53	qrnn	49.79
16	lm	23.74	54	bagEarth	50.17
17	earth	24.00	55	BstLm	51.11
18	lars	24.05	56	enpls.fs	51.17
19	elm	24.55	57	xgbLinear	52.61
20	simpls	24.92	58	randomGLM	52.91
21	spls	25.55	59	spikeslab	53.08
22	glmboost	26.09	60	mlpWeightDecay	53.20
23	ctree2	26.11	61	rbf	53.27
24	nnls	26.33	62	bartMachine	54.52
25	extraTrees	26.89	63	RRF	54.68
26	gaussprPoly	28.00	64	mlpWeightDecayML	54.74
27	rlm	28.23	65	qrf	56.00
28	bdk	28.80	66	krlsRadial	57.35
29	relaxo	30.32	67	evtrees	57.64
30	ppr	30.58	68	bstTree	57.79
31	svmRadial	31.35	69	cforest	58.62
32	gaussprRadial	31.62	70	xgbTree	65.73
33	lasso	31.95	71	SBC	67.73
34	bayesglm	34.05	72	nodeHarvest	67.83
35	bstSm	35.38	73	grnn	69.58
36	pcaNNet	35.47	74	elm-kernel	72.97
37	gbm	35.56	75	dlkeras	73.52
38	blackboost	35.71	76	svr	73.74

**Table 3.8:** Friedman rank of the elapsed time spent by the regressors over all the datasets.

extraTrees, whose positions are 5 and 25 in the correlation and time ranks respectively. In fact, extraTrees spent an average time of 7.17 seg., while the fastest regressor (dnn) spends 3.21 s. The cubist regressor provides an even better correlation rank (about 12), but at the cost of a much higher time rank (about 35) compared to extraTrees. There is a group of three regressors (rf, RRF and bstTree) with similar correlation rank (about 15) and medium-high



**Figure 3.7:** Friedman rank of the time (vertical axis) against the Friedman rank of the correlation for the 20 best regressors in Table 3.3.

time rank (about 50-60). Another group, composed by `gbm`, `penalized`, `avNNet`, `qrf`, `bagEarth` and `brnn` is in the middle of both ranks (correlation about 17-24, time about 35-60). Finally, there are other two groups: the first one, composed by `M5`, `earth` and `gaussprPoly`, has low time and correlation (lower right corner); the second group, composed by `xgbTree`, `cforest`, `krlsRadial` and `dlkeras`, is placed in the upper right corner, with high time and low correlation.

Considering globally the results of the current comparative, the best results are provided by the neural networks and support vector machines, specifically the extreme learning machine (position 1) and the support vector regression (position 2), both with Gaussian kernels. `Cubist` and `extraTrees` also achieve average correlations above 0.78, and can be considered also as very competitive. The `extraTrees` provides the best trade-off between correlation and elapsed time, being the third fastest regressor among the 20 more accurate ones. Other regressor achieve average correlations about 0.77, such as `bstTree`, a gradient boosting ensemble of regression trees and penalized linear regression, or about 0.75, such as the bagging ensemble of multivariate adaptive regression splines (MARS) and the regression tree model (M5). None of the two implementations of deep learning neural networks achieves good results, but the Keras module of Python provides an acceptable performance (about 0.72). Finally, other re-

gressors as the Bayesian regularized neural network (brnn), Gaussian processes, least squares and partial least squares, ridge and lasso regression, quantile methods, linear regression, principal component analysis and generalized additive models achieve correlations between 0.70 and 0.65, and can be considered not so competitive.

## CHAPTER 4

# APPLICATION OF REGRESSION METHODS TO AGRICULTURAL SOIL DATA

After the comparison of regression methods over the UCI machine learning repository datasets developed in chapter 3, in the current chapter we will apply them to the prediction of soil parameters described in chapter 2. An automatic prediction of soil parameters should be accurate enough to be used instead of a direct measurement of these parameters. Most of the literature about prediction of soil parameters uses the concept of pedotransfer function (PTF): a function or algorithm which uses soil measurements to predict or estimate certain soil parameters whose measurement is time-consuming or expensive [5]. The PTF can be formulated using data mining, exploration and machine learning regression methods. Examples of PTF were developed by [40] and [66] for prediction of soil total nitrogen using global soil data and water retention of soil, respectively. The latter paper demonstrated that SVM outperformed ANN for this task. Multiple-linear regression and ANN were used [77] to predict soil hydraulic parameters such as: field capacity, permanent wilting point, available water capacity, saturated hydraulic conductivity and soil water content. The inputs were basic soil properties such as sand, silt, clay, bulk density and number of pores of various diameters. The multiple-linear regression method outperformed ANN, although the difference was not statistically significant. The performance of PTF is limited by the accuracy of the prediction methods, and by spatial and temporal variations of soil parameters. Several studies [58] focused on PTFs for the determination of water retention, and the saturated and unsaturated hydraulic conductivity in order to solve the groundwater problem, properties which are expensive and difficult to measure.

Extended nonlinear regression, MLR and ANN were used [79] to estimate water-retention PTFs on Australian soil datasets. The ROSETTA software<sup>1</sup> is a user friendly tool [117] to access five PTFs to estimate hydraulic properties. Likewise, the soil inference systems (SINFERS) allows to select the PTFs with the minimum variance using knowledge rules [92]. Consequently, research about PTFs is being demanded because they are used in all branches of soil science for describing any mathematical relationships among soil properties, and they allow to estimate missing soil parameters [91]. After reviewing the literature about PTFs, our objective is to use regression techniques as PTFs that predict village-wise soil fertility indices for  $OC$ ,  $K_2O$ ,  $P_2O_5$ ,  $Fe$ ,  $Mn$ , and  $Zn$ , as well as soil  $pH$  and nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , using data from the Indian region of Marathwada. Prediction of soil and crop type can not be included into this regression framework because the outputs are discrete, so they can only be treated as classification problems.

	$K_2O$ (kg/ha)	$Zn$ (PPM)
Low <	0.6	108
Medium	0.6-1.2	108-280
High >	1.2	280

**Table 4.1:** Intervals defined by the Indian Government [23] for the calculation of village-wise soil fertility indices of  $K_2O$  and  $Zn$  nutrients [84, 63].

## 4.1 Regression problems

Good farm practice targets to maintain the several soil parameters that are responsible to optimize the yields of crops in Eco-friendly ways. There is need of sustainable land management practices for maintaining yield potential of agricultural crops. The soils of Marathwada are intensively cultivated for crop production by introducing novel practices. However, application of heavy doses of chemicals fertilizers deteriorates the soil health. 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 use of chemical fertilizers. To solve soil problems, there is a need of knowledge about soil physical and chemical status. The village-wise soil fertility indices for  $OC$ ,  $K_2O$ ,  $P_2O_5$ ,  $Fe$ ,  $Mn$  and  $Zn$  are not only helpful to choose correct fertilizer doses, but also to know about inherent excess and deficiency in them. The predicted values can be used to balance soil

<sup>1</sup><https://www.ars.usda.gov/pacific-west-area/riverside-ca/us-salinity-laboratory/docs/rosetta-model>

nutrients up to critical level in soil. Inspired by the previous concept, we apply the collection of 76 regression techniques described in chapter 3, using the experimental setup described in section 3.2, to predict village-wise soil fertility indices of major ( $OC$ ,  $P_2O_5$ ,  $K_2O$ ) and micro ( $Fe$ ,  $Mn$ ,  $Zn$ ) nutrients, denoted as  $OC$ -F,  $K_2O$ -F,  $P_2O_5$ -F,  $Fe$ -F,  $Mn$ -F and  $Zn$ -F, which were already considered in chapter 2 for classification. We also include two additional village-wise soil fertility indices (for  $K_2O$  and  $Zn$ ), which are missing in chapter 2 because the available patterns belong just to one of the classes. For the eight previous regression problems, there are 372 patterns with 11 inputs:  $EC$ ,  $OC$ ,  $N_2O$ ,  $P_2O_5$ ,  $K_2O$ ,  $SO_4$ ,  $Cu$ ,  $Fe$ ,  $Mn$ ,  $Zn$  and  $Boron$ . In order to calculate the  $K_2O$  and  $Zn$  village-wise soil fertility indices (see chapter 2), we used the formula defined in [105] and the limits listed in Table 4.1. Our study does not consider soil  $N_2O$  and  $Cu$  village-wise soil fertility indices because the available data only include patterns of one soil nutrient level.

Fertilizer	$a$	Crop	$b$	Nutrient
Uria	3.31	Bajra(R)	0.38	$N_2O$
	3.38		4.11	
	1.65		0.068	
Super-phosphate	13.1	Cotton(R), (I)	0.75	$P_2O_5$
	6.83		2.84	
	8.57		0.18	
Murate of potash	6.86	Soybean(R)	0.68	$K_2O$
	6.17		4.46	
	3.96		0.13	

**Table 4.2:** Values of parameters  $a$  and  $b$  for the prediction of different soil nutrients and crops.

In order to maintain the soil quality and to get an optimal crop yield, it is necessary to apply a suitable type and amount of fertilizer which avoid excess of deficiency of soil nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ . The prediction of the levels of these nutrients is therefore useful to calculate the right fertilizer dose, without the need of direct measurements in soil testing laboratory. This prediction will be also useful for developing field applications which recommend amounts of specific fertilizers based on the predicted nutrients levels and the target crop. Our available data contain the levels of the three nutrients for years 2011 to 2015. In our study, we use the formula developed by the Mahatma Phule Agriculture University (India) to recommend the amount of nutrient fertilizer adequate for a particular crop. The amount  $A$  of fertilizer to apply is calculated as  $A = aE - bP$ , being  $E$  the expected crop production (an input data) and  $P$  the predicted nutrient level. The values  $A$  and  $P$  are measured in kg/ha, while  $E$  is measured in quintal/ha. Depending on the nutrient and crop, the parameters  $a$  and

$b$  take different values (see Table 4.2) and the fertilizer is different. The values of  $a$  and  $b$  in Table 4.2 are set for black soil, being different for other soil types. For nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , the corresponding fertilizer is different: uria (which contains 46% of  $N_2O$ ), super phosphate (46% of  $P_2O_5$ ) and muriate of potash (60-62% of  $K_2O$ ), respectively.

The following subsections discuss the results achieved by the collection of regressors for the village-wise soil fertility indices, for the soil nutrients and for soil  $pH$ . The last section discusses globally the results.

## 4.2 Prediction of $OC$ , $P_2O_5$ , $K_2O$ , $Fe$ , $Mn$ and $Zn$ village-wise fertility indices

The Table 4.3 reports the RMSE for the  $OC$  village-wise soil fertility index, sorted by increasing values. The extremely randomized regression trees (extraTrees) achieves the lowest RMSE (0.564), although regularized random forests (RRF), standard random forest (rf), random forest with feature selection (Boruta), gradient boosting of regressor trees (bstTree) and gradient boosted machine (gbm) also achieve values about 0.57. The remaining regressors are about 0.6, including elm-kernel and svr, which are the two best regressors in chapter 3. On the opposite side of the table, the worst regressor is icr, with RMSE=1.17. These values can be considered high, because the output is normalized to have zero mean and standard deviation one, so the RMSE should be low (e.g. under 0.5). The Figure 4.1 (upper panel) shows the correlations achieved by the 20 best regressors for soil  $OC$ -F problem. The four best regressors are the same as in Table 4.3, with changes in their positions: Boruta, RRF, rf and extraTrees achieved high correlation above 0.83 for the soil  $OC$ -F problem, while bstTree and gbm are about 0.82. However cforest, elm-kernel and kknn and 6 more regressors achieved correlations below 0.77. The lower panel of Figure 4.1 presents the scatter plot of the best regressor, Boruta, with RMSE=0.573 and correlation 0.8355.

The results for  $P_2O_5$  village-wise soil fertility index are reported by the Table 4.4. Again, the extraTrees achieves the best RMSE alongside with RRF, gbm, quantile random forest (qrf), Boruta, svr and rf (about 0.63). The elm-kernel works even worse (0.683), and penalized, which also achieves good results in chapter 3, is about 0.75. The gaussprPoly is the worst regressor, with RMSE = 8.41, alongside with ridge, gam, glm, lasso and lm, with RMSE about 1.93, which also correspond to poor performances. The regressors from glmnet to gaussprPoly got the lowest performances with RMSE values above 1.

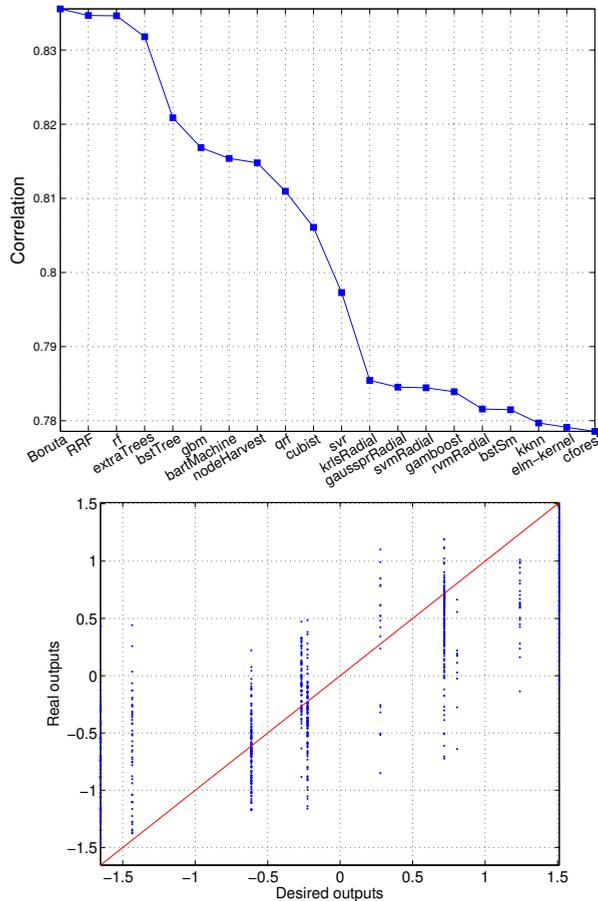
Regressor	RMSE	Regressor	RMSE	Regressor	RMSE	Regressor	RMSE
extraTrees	0.564	treebag	0.641	lars	0.709	earth	0.772
RRF	0.57	bstSm	0.645	rqlasso	0.71	gam	0.772
rf	0.571	brnn	0.65	glmStepAIC	0.711	glm	0.772
Boruta	0.573	grnn	0.666	rqnc	0.718	lasso	0.772
bstTree	0.573	penalized	0.667	spls	0.722	enpls.fs	0.772
gbm	0.578	blackboost	0.671	foba	0.728	lm	0.772
bartMachine	0.584	rbf	0.677	dlkeras	0.729	randomGLM	0.776
qrf	0.592	ppr	0.677	xgbLinear	0.732	evtree	0.795
cubist	0.599	xgbTree	0.684	elm	0.735	superpc	0.811
nodeHarvest	0.601	simpls	0.689	plsRglm	0.748	mlpWeightDecay	0.836
svr	0.606	kernelpls	0.689	rlm	0.75	BstLm	0.838
krlsRadial	0.62	avNNet	0.689	gaussprPoly	0.759	npls	0.845
svmRadial	0.623	SBC	0.689	bdk	0.759	pcaNNet	0.848
gamboost	0.626	relaxo	0.691	ridge	0.765	mlpWeightDecayML	0.975
elm-kernel	0.629	bagEarth	0.697	M5	0.766	dnn	1
cforest	0.632	qrnn	0.699	ctree2	0.769	pcr	1
rvmRadial	0.632	bag	0.7	bayesglm	0.769	glmnet	1.01
gaussprRadial	0.633	glmboost	0.702	gaussprLinear	0.771	partDSA	1.01
kknn	0.633	spikeslab	0.706	rpart	0.772	icr	1.17

**Table 4.3:** RMSE for the prediction of OC village-wise soil fertility index.

Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
extraTrees	0.631	gaussprRadial	0.7	randomGLM	0.814	dnn	1.01
RRF	0.634	brnn	0.704	npls	0.814	glmboost	1.01
gbm	0.634	nodeHarvest	0.707	rqlasso	0.825	partDSA	1.01
qrf	0.635	blackboost	0.723	rqnc	0.837	dlkeras	1.08
Boruta	0.635	M5	0.741	foba	0.859	enpls.fs	1.21
svr	0.636	bag	0.744	gamboost	0.864	earth	1.27
rf	0.637	SBC	0.745	evtree	0.865	bstSm	1.33
cubist	0.643	xgbTree	0.746	BstLm	0.88	spikeslab	1.58
bstTree	0.647	rbf	0.751	bdk	0.893	rlm	1.61
bartMachine	0.654	penalized	0.752	bagEarth	0.899	xgbLinear	1.7
svmRadial	0.659	ppr	0.756	mlpWeightDecay	0.899	glmStepAIC	1.73
krlsRadial	0.672	elm	0.779	pcaNNet	0.904	ridge	1.76
elm-kernel	0.683	relaxo	0.782	icr	0.929	bayesglm	1.86
cforest	0.683	spls	0.786	superpc	0.939	gaussprLinear	1.91
rvmRadial	0.684	simpls	0.786	mlpWeightDecayML	0.952	gam	1.93
avNNet	0.686	kernelpls	0.786	pcr	0.958	glm	1.93
kknn	0.687	qrnn	0.797	lars	0.964	lasso	1.93
grnn	0.691	rpart	0.806	plsRglm	0.99	lm	1.93
treebag	0.691	ctree2	0.81	glmnet	1	gaussprPoly	8.41

**Table 4.4:** RMSE for the prediction of P<sub>2</sub>O<sub>5</sub> village-wise soil fertility index.

The Table 4.5 reports the RMSE for the prediction of K<sub>2</sub>O village-wise soil fertility index. The best RMSE (0.768 using extraTrees) in this table is higher than the two previous indices, so this dataset is more difficult. Boruta achieves RMSE (0.78), while RRF, rf and qrf are about



**Figure 4.1:** Twenty best correlation values (upper panel) and scatter plot of Boruta (lower panel) for the prediction of the *OC* village-wise soil fertility index in the state of Marathwada.

0.79. There is also a group of regressors with RMSE about 0.80, composed by *elm-kernel*, *bstTree* (gradient boosting of regression trees), *svr*, *gaussprRadial* (Gaussian processes with radial kernel) and *nodeHarvest* (simple interpretable tree ensemble for high-dimensional regression), which in the soil problems does not exhibit problems related to very slow execution. The remaining regressors are about 0.81 until the worst value (1.58), achieved by *rlm* (robust linear regression).

Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
extraTrees	0.768	cubist	0.851	evtree	0.941	spls	1.13
Boruta	0.78	avNNet	0.86	blackboost	0.945	rqnc	1.18
RRF	0.793	grnn	0.861	ppr	0.946	spikeslab	1.19
rf	0.795	earth	0.861	ctree2	0.966	plsRglm	1.21
qrf	0.795	bstSm	0.863	dnn	0.986	gamboost	1.27
elm-kernel	0.8	xgbTree	0.887	mlpWeightDecayML	0.986	foba	1.29
bstTree	0.805	SBC	0.888	glmnet	0.988	enpls.fs	1.31
svr	0.806	rbf	0.891	partDSA	0.989	randomGLM	1.34
gaussprRadial	0.806	mlpWeightDecay	0.894	npls	0.991	xgbLinear	1.39
nodeHarvest	0.807	pcaNNet	0.897	pcr	0.991	M5	1.41
bartMachine	0.815	brnn	0.901	icr	1.01	ridge	1.41
krlsRadial	0.82	qrnn	0.903	dlkeras	1.02	glmStepAIC	1.41
treebag	0.826	simpls	0.903	gaussprPoly	1.04	bayesglm	1.47
gbm	0.826	kernelpls	0.903	superpc	1.05	gaussprLinear	1.5
svmRadial	0.834	bag	0.913	lars	1.06	gam	1.51
cforest	0.835	relaxo	0.917	rqlasso	1.06	glm	1.51
rvmRadial	0.839	elm	0.921	BstLm	1.06	lasso	1.51
kknn	0.844	rpart	0.926	bdk	1.07	lm	1.51
penalized	0.849	bagEarth	0.941	glmboost	1.12	rlm	1.58

**Table 4.5:** RMSE for the prediction of  $K_2O$  village-wise soil fertility index.

With respect to the prediction of  $Fe$  village-wise soil fertility index (see Table 4.6), extraTrees achieves the lowest RMSE (0.606), followed by a group of three random forests (rf, RRF and Boruta) about 0.61. The gradient boosting ensembles bstTree and gbm are about 0.63, and cubist (M5 rule model with nearest neighbors) is about 0.65. The remaining regressors are already above 0.67. The Figure 4.2 (upper panel) shows the 20 highest correlations for this problem, exhibiting almost the same order as in Table 4.6. The best correlation (0.816 using extraTrees; the lower panel of Figure 4.2 plots its scatter plot for this dataset) corresponds with a certain accuracy in prediction. The rf, RRF and Boruta achieve correlations above 0.80, while bstTree, gbm and quantile regression forest (qrf) are about 0.79.

The extraTrees also achieves the best RMSE (0.657) for the prediction of  $Mn$  village-wise soil fertility index (see Table 4.7). The following six regressors (bstTree, RRF, gbm, rf, svr and Boruta, which are the same as in the previous indices) are about 0.68. The cubist, svmRadial and qrf are about 0.69, and the remaining regressors are above 0.7 until bstSm, with RMSE of 1.78. The deep learning neural network (dlkeras) achieves RMSE about 0.755 and is placed in the 22nd position.

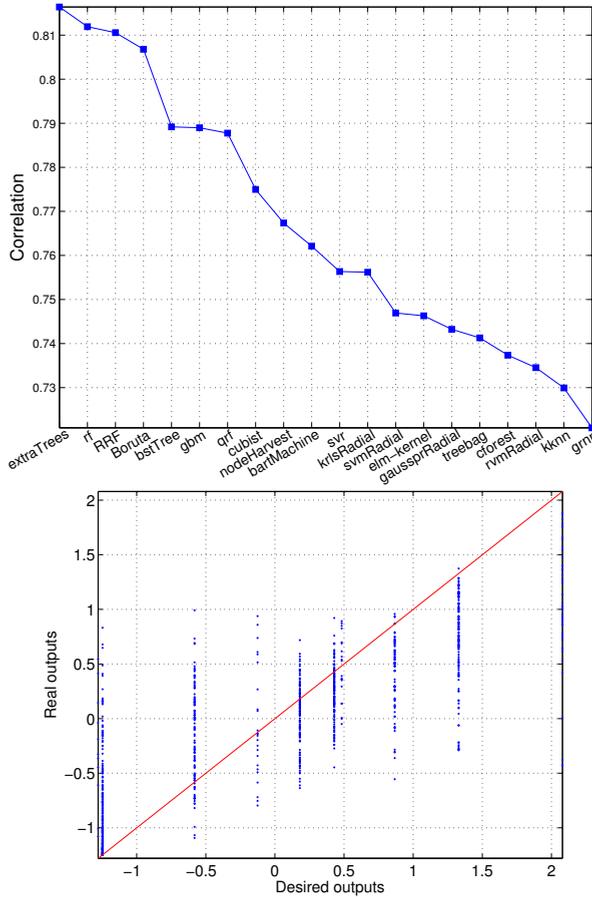
Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
extraTrees	0.606	grnn	0.714	bdk	0.854	BstLm	1.06
rf	0.614	xgbTree	0.725	relaxo	0.873	dlkeras	1.08
RRF	0.616	earth	0.745	npls	0.879	spls	1.11
Boruta	0.618	avNNet	0.749	foba	0.891	plsRglm	1.13
bstTree	0.632	bag	0.76	simpls	0.892	M5	1.14
gbm	0.632	SBC	0.77	kernelpls	0.892	glmboost	1.18
qrf	0.636	gamboost	0.771	pcaNNet	0.904	enpls.fs	1.33
cubist	0.651	brnn	0.78	superpc	0.907	spikeslab	1.4
bartMachine	0.67	rbf	0.784	rqlasso	0.907	rlm	1.74
svr	0.672	blackboost	0.786	rqnc	0.918	xgbLinear	1.88
nodeHarvest	0.672	bstSm	0.786	icr	0.92	randomGLM	2.01
krplsRadial	0.672	ppr	0.8	mlpWeightDecay	0.925	bayesglm	2.05
svmRadial	0.683	qrnn	0.803	pcr	0.939	ridge	2.09
elm-kernel	0.685	ctree2	0.814	lars	0.967	gaussprLinear	2.13
treebag	0.692	rpart	0.829	mlpWeightDecayML	0.995	gam	2.14
cforest	0.696	bagEarth	0.831	gaussprPoly	1.01	glm	2.14
rvmRadial	0.699	evtrees	0.839	dnn	1.02	lasso	2.14
gaussprRadial	0.703	penalized	0.84	glmnet	1.03	lm	2.14
kknn	0.711	elm	0.847	partDSA	1.03	glmStepAIC	2.22

**Table 4.6:** RMSE for the prediction of *Fe* village-wise soil fertility index.

Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
extraTrees	0.657	treebag	0.737	mlpWeightDecay	0.847	bdk	0.958
bstTree	0.677	avNNet	0.745	npls	0.851	randomGLM	0.97
RRF	0.679	dlkeras	0.755	elm	0.854	dnn	0.997
gbm	0.679	earth	0.757	ctree2	0.855	glmnet	1
rf	0.681	xgbTree	0.759	glmboost	0.855	partDSA	1
svr	0.683	bagEarth	0.774	rpart	0.868	xgbLinear	1.11
Boruta	0.683	bag	0.782	rqnc	0.88	foba	1.11
cubist	0.687	rbf	0.783	evtrees	0.885	rlm	1.14
svmRadial	0.692	brnn	0.79	rqlasso	0.891	ridge	1.17
qrf	0.692	M5	0.791	BstLm	0.903	bayesglm	1.22
krplsRadial	0.7	SBC	0.796	lars	0.912	gaussprLinear	1.25
rvmRadial	0.701	penalized	0.803	superpc	0.912	gam	1.26
bartMachine	0.706	blackboost	0.822	gaussprPoly	0.912	glm	1.26
gaussprRadial	0.712	qrnn	0.826	plsRglm	0.944	lasso	1.26
elm-kernel	0.713	relaxo	0.828	spls	0.945	lm	1.26
nodeHarvest	0.715	ppr	0.829	enpls.fs	0.947	glmStepAIC	1.27
cforest	0.718	simpls	0.835	mlpWeightDecayML	0.954	icr	1.32
kknn	0.719	kernelpls	0.835	spikeslab	0.955	gamboost	1.52
grnn	0.734	pcaNNet	0.842	pcr	0.957	bstSm	1.78

**Table 4.7:** RMSE for the prediction of *Mn* village-wise soil fertility index.

The best RMSE decreases to 0.582, achieved again by extraTrees, for the prediction of the *Zn* village-wise soil fertility index (see Table 4.8), which corresponds to a certain level in the prediction accuracy. However, in this case the regressors in the first positions are different:



**Figure 4.2:** Twenty best correlations (upper panel) and scatter plot of extraTrees (lower panel), which achieves the best results for the prediction of Fe village-wise soil fertility index.

the Boruta, random forests rf, and RRF (about 0.66) are replaced by cubist and the avNNet ensemble of multi-layer perceptrons, while gbm, svr and bstTree achieve RMSE above 0.64, much higher than extraTrees.

Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
extraTrees	0.582	treebag	0.776	bdk	0.965	elm	2.09
cubist	0.628	cforest	0.786	bagEarth	0.99	spikeslab	2.46
avNNet	0.638	nodeHarvest	0.79	mlpWeightDecayML	0.997	foba	2.51
gbm	0.64	evtree	0.795	gaussprPoly	1	xgbLinear	2.55
svr	0.645	brnn	0.813	icr	1.02	earth	2.66
bstTree	0.654	blackboost	0.814	glmnet	1.03	spls	2.71
svmRadial	0.657	xgbTree	0.826	dnn	1.03	glmStepAIC	2.76
bartMachine	0.657	rvmRadial	0.842	pcr	1.03	bayesglm	2.77
krlsRadial	0.663	bag	0.846	partDSA	1.04	plsRglm	2.82
qrf	0.664	rpart	0.856	rqlasso	1.05	ridge	2.83
Boruta	0.664	penalized	0.86	M5	1.1	gaussprLinear	2.87
rf	0.666	qrnn	0.861	superpc	1.16	gam	2.89
RRF	0.668	mlpWeightDecay	0.862	lars	1.22	glm	2.89
elm-kernel	0.685	pcaNNet	0.877	dlkeras	1.23	lasso	2.89
kknn	0.715	relaxo	0.896	rqnc	1.32	lm	2.89
grnn	0.724	ctree2	0.899	BstLm	1.33	enpls.fs	3.02
ppr	0.753	rbf	0.924	npls	1.49	gamboost	3.05
SBC	0.758	simpls	0.935	rlm	1.64	bstSm	3.17
gaussprRadial	0.772	kernelpls	0.935	glmboost	1.82	randomGLM	6.44

**Table 4.8:** RMSE for the prediction of Zn village-wise soil fertility index.

### 4.3 Prediction of soil nutrients $N_2O$ , $P_2O_5$ and $K_2O$

Penalized linear regression, rf and elm-kernel are the optimal regressors for the prediction of soil nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , respectively. However, the RMSE values achieved by all regressors are worse (0.863, 0.807 and 0.818) than for village-wise soil fertility indices. The Table 4.9 reports the RMSE values (between 0.863 and 5.32) from penalized to bstSm for the prediction of soil  $N_2O$  nutrient. The random forest of conditional inference trees (cforest) and svr achieve RMSE values about 0.87, while extraTrees, elm-kernel, RRF, rf and treebag are about 0.88, and the following regressors after quantile random forest (qrf) are above 0.89. The Figure 4.3 (upper panel) plots the low correlations (under 0.5192, achieved by penalized) of the 20 best regressors for the prediction of soil  $N_2O$ , and the scatter plot of penalized (lower panel).

The Table 4.10 shows that random forest (rf) achieves the best RMSE (0.807) for the prediction of soil  $P_2O_5$  level, followed by RRF and extraTrees (about 0.81). These RMSE values are the highest among the three soil nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , and the worse regressor (bagEarth) recorded the highest RMSE (6.76) over all the soil regression problems.

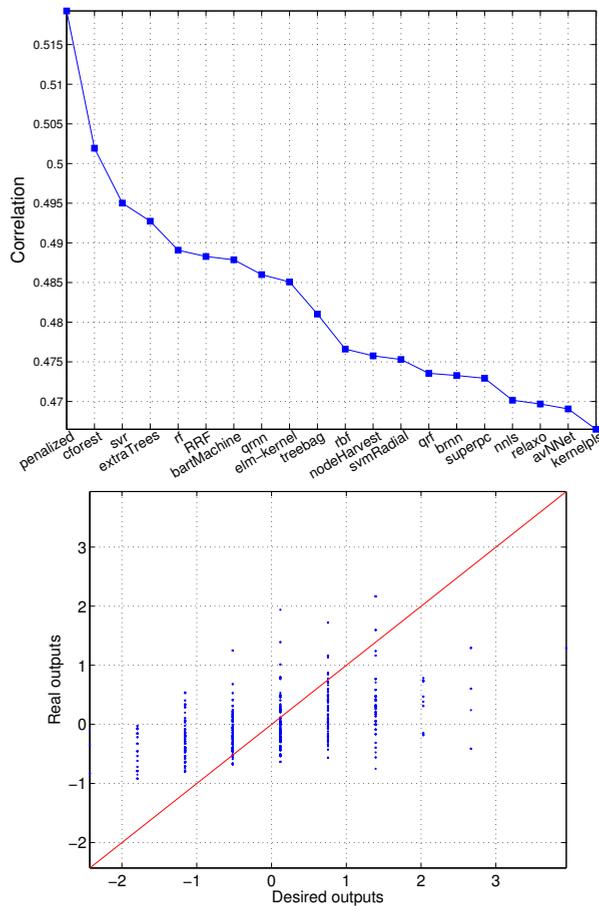
Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
penalized	0.863	npls	0.892	earth	0.916	rpart	0.974
cforest	0.874	svmRadial	0.894	spikeslab	0.917	xgbLinear	0.984
svr	0.877	avNNet	0.894	foba	0.918	evtree	0.987
extraTrees	0.878	rqlasso	0.895	ppr	0.92	mlpWeightDecayML	0.987
RRF	0.88	bag	0.897	bagEarth	0.92	SBC	0.989
rf	0.88	Boruta	0.898	blackboost	0.932	ridge	1
bartMachine	0.881	bstTree	0.898	xgbTree	0.935	glmnet	1.01
treebag	0.884	glmboost	0.899	M5	0.936	dnn	1.01
elm-kernel	0.886	BstLm	0.899	bayesglm	0.942	partDSA	1.01
brnn	0.888	cubist	0.9	gam	0.945	randomGLM	1.02
qrnn	0.889	spls	0.902	glm	0.945	bdk	1.09
superpc	0.889	lars	0.902	lasso	0.945	gaussprPoly	1.11
qrf	0.889	krlsRadial	0.903	lm	0.945	plsRglm	1.11
rbf	0.89	rqnc	0.906	gaussprLinear	0.946	dlkeras	1.19
nodeHarvest	0.89	glmStepAIC	0.906	pcaNNet	0.95	rlm	1.25
relaxo	0.89	gaussprRadial	0.912	grnn	0.952	enpls.fs	1.54
simpls	0.891	kknn	0.912	mlpWeightDecay	0.952	icr	1.86
kernelpls	0.891	rvmRadial	0.912	ctree2	0.959	gamboost	4.41
gbm	0.892	elm	0.914	pcr	0.97	bstSm	5.32

**Table 4.9:** RMSE values for the prediction of  $N_2O$  nutrient level.

Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
rf	0.807	bag	0.862	rpart	0.922	npls	3.24
RRF	0.808	rvmRadial	0.862	dnn	0.922	spls	3.3
extraTrees	0.81	grnn	0.865	mlpWeightDecayML	0.922	plsRglm	3.36
bstTree	0.815	brnn	0.871	qrnn	0.923	enpls.fs	4.1
treebag	0.82	Boruta	0.876	ctree2	0.927	glmboost	4.42
penalized	0.821	rbf	0.877	ppr	0.951	dlkeras	4.59
nodeHarvest	0.826	M5	0.881	SBC	0.974	foba	4.8
gbm	0.831	relaxo	0.883	rqlasso	0.993	spikeslab	4.96
bartMachine	0.832	blackboost	0.884	avNNet	1.03	bayesglm	5.26
cforest	0.835	pcr	0.89	gamboost	1.05	ridge	5.33
qrf	0.845	xgbTree	0.901	bstSm	1.07	randomGLM	5.37
simpls	0.854	superpc	0.906	rlm	1.18	gaussprLinear	5.38
kernelpls	0.854	pcaNNet	0.908	bdk	1.19	gam	5.41
svr	0.855	elm	0.911	icr	1.19	glm	5.41
gaussprRadial	0.857	evtree	0.912	gaussprPoly	1.2	lasso	5.41
kknn	0.859	cubist	0.914	lars	1.23	lm	5.41
elm-kernel	0.86	mlpWeightDecay	0.914	xgbLinear	2.06	glmStepAIC	5.44
svmRadial	0.862	glmnet	0.921	rqnc	2.22	earth	6.69
krlsRadial	0.862	partDSA	0.921	BstLm	2.73	bagEarth	6.76

**Table 4.10:** RMSE values for the prediction of  $P_2O_5$  nutrient level.

In the prediction of soil  $K_2O$  nutrient, the Table 4.11 shows that elm-kernel and gaussprRadial achieve the best RMSE (0.818), with a high different with respect to extraTrees (0.827), rf



**Figure 4.3:** Correlation (upper panel) and scatter plot of penalized (lower panel) for the prediction of  $N_2O$  level in the state of Marathwada.

and krlsRadial (0.829) and RRF (0.83). The highest RMSE (1.72) is achieved by independent component regression (icr).

#### 4.4 Prediction of soil $pH$

ExtraTrees achieves the best RMSE (0.711) for the prediction of soil  $pH$  (see Table 4.12), followed with minor differences by cubist, RRF and rf (about 0.74), Boruta and qrf (about

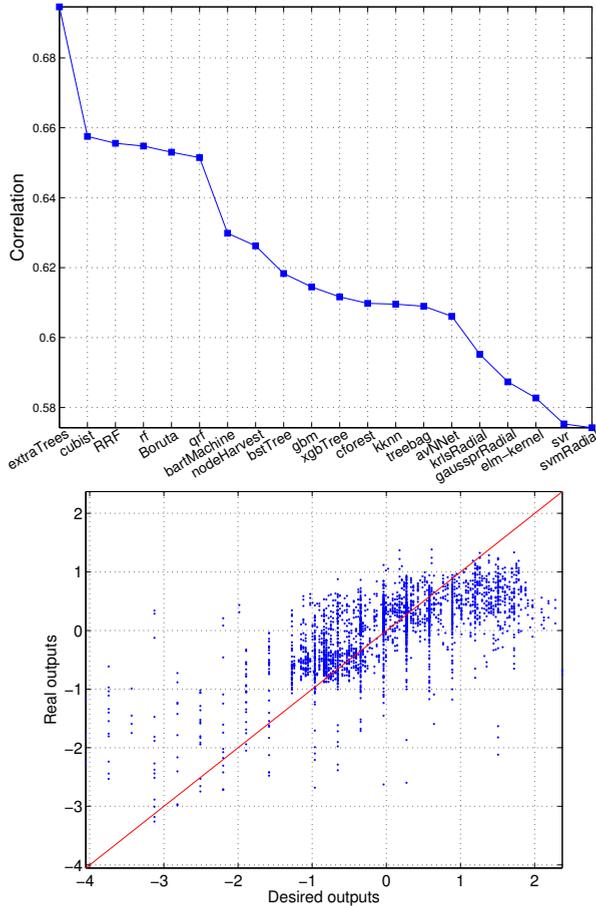
Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.	Regressor	RMSE.
elm-kernel	0.818	pcaNNet	0.876	foba	0.93	ppr	1.09
gaussprRadial	0.818	M5	0.878	rqnc	0.931	gamboost	1.12
svr	0.819	mlpWeightDecay	0.883	SBC	0.936	bdk	1.21
extraTrees	0.827	relaxo	0.884	blackboost	0.943	qrnn	1.24
rf	0.829	rpart	0.889	gaussprPoly	0.949	rlm	1.24
krlsRadial	0.829	bagEarth	0.899	pcr	0.949	enpls.fs	1.29
RRF	0.831	brnn	0.899	spikeslab	0.952	glmStepAIC	1.29
kknn	0.835	earth	0.901	ctree2	0.953	spls	1.3
treebag	0.837	simpls	0.904	glmnet	0.954	plsRglm	1.3
qrf	0.841	kernelpls	0.904	mlpWeightDecayML	0.954	xgbLinear	1.36
nodeHarvest	0.842	BstLm	0.904	partDSA	0.955	ridge	1.36
svmRadial	0.843	grnn	0.907	dnn	0.956	bayesglm	1.44
bartMachine	0.847	elm	0.907	evtree	0.962	gaussprLinear	1.47
Boruta	0.848	glmboost	0.908	cubist	0.968	gam	1.48
penalized	0.857	rbf	0.911	avNNet	0.97	glm	1.48
rvmRadial	0.859	npls	0.913	lars	0.972	lasso	1.48
bstTree	0.865	bag	0.916	superpc	1	lm	1.48
cforest	0.867	xgbTree	0.918	dlkeras	1.02	bstSm	1.56
gbm	0.872	rqlasso	0.925	randomGLM	1.05	icr	1.72

**Table 4.11:** RMSE values for the prediction of  $K_2O$  nutrient level.

Regressor	RMSE	Regressor	RMSE	Regressor	RMSE	Regressor	RMSE
extraTrees	0.711	svmRadial	0.812	gamboost	0.894	superpc	1.03
cubist	0.746	M5	0.815	bstSm	0.903	bdk	1.06
RRF	0.747	rvmRadial	0.817	relaxo	0.904	BstLm	1.13
rf	0.748	evtree	0.823	simpls	0.908	glmboost	1.15
Boruta	0.749	earth	0.829	kernelpls	0.908	randomGLM	1.16
qrf	0.751	grnn	0.831	pcaNNet	0.913	enpls.fs	1.28
bartMachine	0.767	brnn	0.848	rqlasso	0.94	spikeslab	1.43
nodeHarvest	0.774	bag	0.85	spls	0.941	ridge	1.45
bstTree	0.776	bagEarth	0.859	rqnc	0.953	xgbLinear	1.55
gbm	0.78	SBC	0.861	mlpWeightDecay	0.953	rlm	1.55
xgbTree	0.782	rpart	0.862	lars	0.956	gaussprLinear	1.55
kknn	0.782	ppr	0.864	gaussprPoly	0.959	gam	1.56
treebag	0.783	rbf	0.868	pcr	0.976	glm	1.56
cforest	0.785	blackboost	0.871	mlpWeightDecayML	0.982	lasso	1.56
krlsRadial	0.797	qrnn	0.874	dnn	0.986	bayesglm	1.56
avNNet	0.798	elm	0.876	plsRglm	0.986	lm	1.56
gaussprRadial	0.801	ctree2	0.876	glmnet	0.99	npls	1.57
elm-kernel	0.802	penalized	0.885	partDSA	0.99	glmStepAIC	1.59
svr	0.811	dlkeras	0.887	icr	0.991	foba	1.66

**Table 4.12:** RMSE values for the prediction of soil pH.

0.75). The remaining regressors are already above 0.77. The Figure 4.4 (upper panel) plots the low correlation values achieved by the 20 best regressors, started by extraTrees (0.6945, see its scatter plot on the right panel) and followed by a group about 0.65 (cubist, RRF, rf,



**Figure 4.4:** Correlation of the twenty best regressors (upper panel) and scatter plot of extraTrees (lower panel), which achieves the best correlation for the prediction of soil  $pH$ .

Boruta and qrf). The right panel of Figure 4.4 shows the scatter plot of extraTrees for the prediction of soil  $pH$ .

## 4.5 Global discussion

Considering the results over all the soil datasets (see Tables 4.3-4.12), extraTrees achieved the best RMSE for 7 of 10 soil problems ( $OC$ ,  $P_2O_5$ ,  $K_2O$ ,  $Fe$ ,  $Mn$  and  $Zn$  village-wise soil

fertility indices, and soil  $pH$ ). Besides, penalized linear regression, random forest and elm-kernel are the best regressors for soil  $N_2O$ ,  $P_2O_5$  and  $K_2O$  nutrients, respectively. Perhaps the most relevant conclusion that we can draw from the results is that, besides being extraTrees the best regressors for more than a half of the datasets, other four regressors of the random forest family (rf, RRF, Boruta and qrf) are among the best five regressors for almost all the soil datasets. Therefore, this family can be considered the best for these datasets, confirming the good result of both random forests in the soil classification problems (see chapter 2). The svr and two gradient boosting ensembles (bstTree and gbm) are also among the ten best regressors for almost all the fertility indices, as well as for  $P_2O_5$  and  $pH$ . The M5 rule with nearest neighbors (cubist) is among the ten best regressors for 6 datasets, being the second best for  $Zn$  village-wise soil fertility index and prediction of  $pH$ .

OC-F		$P_2O_5$ -F		$K_2O$ -F		Fe-F		Mn-F	
Regressor	Corr.	Regressor	Corr.	Regressor	Corr.	Regressor	Corr.	Regressor	Corr.
Boruta	0.835	extraTrees	0.776	extraTrees	0.631	extraTrees	0.816	extraTrees	0.758
RRF	0.834	RRF	0.775	Boruta	0.619	rf	0.812	RRF	0.741
rf	0.834	qrf	0.774	RRF	0.607	RRF	0.810	bstTree	0.739
extraTrees	0.832	gbm	0.774	rf	0.601	Boruta	0.807	rf	0.739
bstTree	0.821	Boruta	0.774	qrf	0.597	bstTree	0.789	Boruta	0.737
gbm	0.817	rf	0.772	elm-kernel	0.586	gbm	0.789	gbm	0.737
bartMachine	0.815	svr	0.772	bstTree	0.585	qrf	0.788	svr	0.734
nodeHarvest	0.815	cubist	0.770	gaussprRadial	0.581	cubist	0.775	cubist	0.730
qrf	0.811	bstTree	0.765	nodeHarvest	0.579	nodeHarvest	0.767	svmRadial	0.724
cubist	0.806	bartMachine	0.757	svr	0.578	bartMachine	0.762	qrf	0.723
Zn-F		$N_2O$		$P_2O_5$		$K_2O$		$pH$	
Regressor	Corr.	Regressor	Corr.	Regressor	Corr.	Regressor	Corr.	Regressor	Corr.
extraTrees	0.841	penalized	0.519	qrf	0.487	gaussprRadial	0.517	extraTrees	0.694
cubist	0.794	cforest	0.502	rf	0.476	svr	0.516	cubist	0.657
avNNNet	0.790	svr	0.495	RRF	0.473	elm-kernel	0.514	RRF	0.655
svr	0.784	extraTrees	0.493	extraTrees	0.470	krlsRadial	0.511	rf	0.655
gbm	0.783	rf	0.489	gbm	0.464	extraTrees	0.500	Boruta	0.653
qrf	0.779	RRF	0.488	bartMachine	0.460	rf	0.495	qrf	0.651
rf	0.778	bartMachine	0.488	bstTree	0.458	RRF	0.492	bartMachine	0.630
Boruta	0.775	qrnn	0.486	treebag	0.446	kkn	0.486	nodeHarvest	0.626
RRF	0.774	elm-kernel	0.485	penalized	0.444	qrf	0.483	bstTree	0.618
krlsRadial	0.772	treebag	0.481	nodeHarvest	0.434	treebag	0.477	gbm	0.614

**Table 4.13:** Ten bests correlations for the prediction of each soil dataset.

The Table 4.13 reports the ten best regressors according to correlation coefficient for the ten soil datasets. ExtraTrees achieves the best correlations for six of ten datasets ( $P_2O_5$ -F,  $K_2O$ -F, Fe-F, Mn-F, Zn-F and  $pH$ ), being the 4th or 5th in the remaining four datasets. The best regressors for the remaining datasets are Boruta, penalized, qrf and gaussprRadial for

Pos.	RMSE rank		Correlation rank		Avg. correl.
	Regressor	Rank	Regressor	Rank	
1	extraTrees	1.7	extraTrees	2.3	0.68132
2	RRF	4.2	RRF	4.0	0.66519
3	rf	4.7	rf	4.3	0.66520
4	qrf	7.5	qrf	7.0	0.65677
5	svr	8.5	Boruta	8.4	0.64915
6	bstTree	9.2	svr	8.5	0.64234
7	gbm	9.5	gbm	9.7	0.64419
8	bartMachine	9.5	bstTree	9.7	0.64580
9	Boruta	9.8	bartMachine	9.9	0.64228
10	elm-kernel	12.9	nodeHarvest	12.9	0.62539
11	nodeHarvest	13.0	svmRadial	13.3	0.62436
12	cforest	14.1	krlsRadial	13.5	0.62748
13	svmRadial	14.1	cubist	13.5	0.62658
14	treebag	14.3	elm-kernel	14.0	0.62206
15	krlsRadial	14.7	treebag	15.6	0.61491
16	gaussprRadial	16.6	cforest	15.8	0.60960
17	cubist	17.3	gaussprRadial	16.4	0.61405
18	kknn	17.9	kknn	18.4	0.60867
19	rvmRadial	19.3	avNNet	19.3	0.59096
20	penalized	23.1	rvmRadial	19.4	0.59719
21	brnn	24.0	penalized	25.3	0.54830
22	avNNet	24.9	brnn	25.6	0.55449
23	grnn	25.3	grnn	27.1	0.56332
24	xgbTree	27.5	xgbTree	29.0	0.54622
25	bag	28.1	rbf	29.6	0.51867
26	rbf	28.1	ppr	30.5	0.52761
27	relaxo	31.3	bag	31.2	0.52418
28	kernelpls	31.8	qrnn	32.3	0.50870
29	blackboost	32.2	SBC	32.5	0.51816
30	simpls	32.8	kernelpls	34.3	0.48163
31	SBC	33.2	relaxo	34.3	0.48478
32	qrnn	33.5	blackboost	34.4	0.49932
33	ppr	35.7	pcaNNet	34.4	0.52631
34	M5	37.2	dlkeras	34.9	0.49958
35	bagEarth	37.4	simpls	35.3	0.48163
36	earth	38.5	mlpWeightDecay	36.2	0.49583
37	elm	39.3	bagEarth	37.5	0.48584
38	rpart	39.8	rpart	37.9	0.50542

**Table 4.14:** Friedman rank of the RMSE (left) and of the correlation (right), and average correlation. Continued in Table 4.15.

$OC-F$ ,  $N_2O$ ,  $P_2O_5$  and  $K_2O$ , respectively. The random forests achieve very good results: RRF is the 2nd or 3rd best for 7 datasets; rf is the 4th regressor in 4 datasets; Boruta is among the 5 bests for 6 datasets; and qrf is among the 10 bests for 9 of 10 datasets. The boosting regressors gbm and bstTree are among the 10 bests for 7 of 10 datasets; the svr, bartMachine and cubist

Order	RMSE rank		Correlation rank		Avg. correl.
	Regressor	Rank	Regressor	Rank	
39	pcaNNet	41.4	M5	39.8	0.47795
40	rqlasso	42.3	earth	40.0	0.45912
41	evtree	42.4	elm	41.4	0.45311
42	mlpWeightDecay	42.6	ctree2	42.3	0.46354
43	ctree2	43.5	bdk	42.9	0.45099
44	rqnc	46.2	rqlasso	44.4	0.41581
45	lars	46.9	evtree	44.8	0.44705
46	superpc	47.2	rqnc	47.3	0.39901
47	npls	47.6	spls	48.2	0.36216
48	gamboost	48.4	BstLm	49.1	0.35185
49	glmboost	48.6	gamboost	49.6	0.33209
50	BstLm	48.8	glmboost	50.3	0.33887
51	dlkeras	49.8	bstSm	50.4	0.31987
52	bstSm	50.3	lars	51.1	0.34588
53	spls	50.8	npls	51.6	0.35760
54	pcr	51.4	superpc	51.9	0.32258
55	mlpWeightDecayML	52.1	spikeslab	54.2	0.29352
56	foba	53.3	foba	54.5	0.33074
57	bdk	53.3	plsRglm	55.2	0.31830
58	gaussprPoly	53.4	randomGLM	57.7	0.28005
59	dnn	54.4	glmStepAIC	58.2	0.25896
60	spikeslab	54.8	bayesglm	58.8	0.24860
61	plsRglm	58.4	ridge	59.0	0.25092
62	icr	59.3	gaussprPoly	59.4	0.26222
63	glmStepAIC	60.2	enpls.fs	59.5	0.27277
64	randomGLM	60.6	xgbLinear	59.7	0.22684
65	rlm	61.4	rlm	60.4	0.25463
66	xgbLinear	61.4	gaussprLinear	61.0	0.24405
67	enpls.fs	61.8	glm	61.2	0.24355
68	bayesglm	63.9	lm	62.2	0.24355
69	ridge	64.8	pcr	63.0	0.18370
70	gaussprLinear	65.6	lasso	63.2	0.24355
71	glm	66.4	icr	63.9	0.17313
72	lm	67.4	gam	64.2	0.24355
73	lasso	68.4	mlpWeightDecayML	65.0	0.18838
74	partDSA	69.0	dnn	71.9	0.06922
75	gam	69.4	glmnet	75.2	-0.03778
76	glmnet	76.0	partDSA	75.3	-0.04306

Table 4.15: Continuation of Table 4.14.

for 6 of them; and nodeHarvest (5 datasets). Considering the best correlation values, they only overcome 0.8 for datasets *OC-F*, *Fe-F* and *Zn-F*, being between 0.6 and 0.8 in four datasets (*P<sub>2</sub>O<sub>5</sub>-F*, *K<sub>2</sub>O-F*, *Mn-F* and *pH*), and below 0.6 for *N<sub>2</sub>O*, *P<sub>2</sub>O<sub>5</sub>* and *K<sub>2</sub>O* nutrients. The low correlations for these three nutrients confirms the higher difficulty of these datasets, where the classification experiments (see chapter 2) already achieved Cohen  $\kappa$  lower than the village-

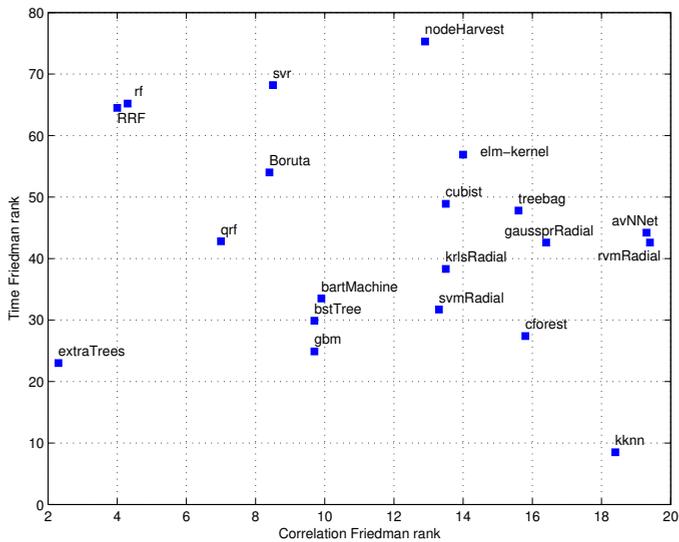
wise fertility indices, crop, soil type and  $pH$ . For some datasets the difference between the best correlation and the following values is relatively high:  $K_2O$ -F, where extraTrees and Boruta achieve 0.631 and 0.619, respectively (difference 0.012),  $Zn$ -F, with difference 0.097 between extraTrees (0.891) and cubist (0.794);  $N_2O$ , difference 0.017 between penalized and cforest; and  $pH$ , with difference 0.037 between extraTrees and cubist.

Pos.	Regressor	$p$ -value	Pos.	Regressor	$p$ -value	Pos.	Regressor	$p$ -value
1	extraTrees	—	27	qrnn	<b>0.0451546</b>	53	randomGLM	0.00131494
2	rf	0.73373	28	ppr	0.0376353	54	lars	0.000768539
3	RRF	0.62318	29	bag	0.0376353	55	glmStepAIC	0.000768539
4	qrf	0.57075	30	blackboost	0.031209	56	superpc	0.000768539
5	svr	0.52052	31	pcaNNet	0.031209	57	spikeslab	0.000768539
6	Boruta	0.47268	32	mlpWeightDecay	0.0257481	58	lm	0.00058284
7	gbm	0.42735	33	evtrees	0.0257481	59	enpls.fs	0.00058284
8	cubist	0.38467	34	rpart	0.0211339	60	r1m	0.00058284
9	bstTree	0.38467	35	bagEarth	0.0211339	61	ridge	0.00058284
10	elm-kernel	0.34470	36	earth	0.0172575	62	lasso	0.00058284
11	krlsRadial	0.34470	37	r1f	0.0172575	63	gam	0.00058284
12	gaussprRadial	0.30749	38	ctree2	0.0172575	64	gaussprLinear	0.00058284
13	svmRadial	0.30749	39	M5	0.0140193	65	bayesglm	0.00058284
14	bartMachine	0.30749	40	bdk	0.0140193	66	gaussprPoly	0.00058284
15	cforest	0.27304	41	relaxo	0.0091085	67	plsRglm	0.00058284
16	kknn	0.24132	42	elm	0.0091085	68	BstLm	0.00058284
17	nodeHarvest	0.24132	43	rqnc	0.00728456	69	glm	0.00058284
18	rvmRadial	0.18588	44	gamboost	0.00728456	70	xgbLinear	0.00058284
19	treebag	0.18588	45	kernelpls	0.00579536	71	glmnet	0.000182672
20	avNNet	0.16197	46	simpls	0.00579536	72	dnn	0.000182672
21	grnn	0.12122	47	bstSm	0.00458639	73	pcr	0.000182672
22	SBC	0.07566	48	npls	0.00458639	74	icr	0.000182672
23	brnn	0.07566	49	rqlasso	0.00458639	75	partDSA	0.000182672
24	xgbTree	0.05390	50	foba	0.00282727	76	mlpWeightDecayML	0.000182672
25	penalized	0.05390	51	spls	0.00170625			
26	dlkeras	0.05390	52	glmboost	0.00170625			

**Table 4.16:** List of the  $p$ -values achieved by the Wilcoxon signed rank test comparing the correlations of extraTrees and the remaining 75 regressors over the 10 soil regression problems.

The Tables 4.14 and 4.15 report the Friedman ranks for the RMSE and correlation, and the average correlation, for all the regressors over all the ten soil datasets. The extraTrees regressor achieves the first position, with a rank of 1.7 and 2.3 for RMSE and correlation respectively. This means that, in average over the ten datasets, extraTrees is between positions 1-2 for RMSE and 2-3 for correlation. However, the highest average correlation (0.681), achieved by extraTrees, reflects that the prediction is not very accurate, because an accurate prediction would require correlations about 0.9-0.95. Four regressors of the random forest family (RRF, rf, qrf, and Boruta) are placed in the first positions, alongside with the svr and

the two gradient boosting ensembles (bstTree and gbm). The elm-kernel is in position 10, with average correlation about 0.625, which is far from the best results. Other regressors with good results in some soil datasets (e.g. cubist) or in the UCI datasets (see chapter 3), e.g. penalized and avNNet, are in positions 10-20 on this ranking. The last positions of the ranking (Table 4.15) are for lm (linear regression), bstSm (gradient boosting with smoothing splines), glm (generalized linear models), icr (independent component regression), randomGLM (boosting ensemble of GLM), glmStepAIC (GLM with stepwise feature selection and the Akaike information criterion) and lasso (regression by least absolute shrinkage and selection operator).



**Figure 4.5:** Friedman rank of the time (vertical axis) against the Friedman rank of the correlation (horizontal axis) for the 20 best regressors over the ten soil data sets.

The Table 4.16 reports the  $p$ -values for a Wilcoxon signed rank test [132] comparing the correlations achieved by extraTrees, which is the best regressor on the soil datasets according both to RMSE and correlation, to the correlations of the remaining regressors, sorted by decreasing order. The value in bold corresponds to the regressor (qrnn, position 27 of 76) from which the difference with respect to extraTrees is statistically significant for a 5%-confidence level (i.e.,  $p < 0.05$ ). Since the difference extraTrees and the first regressors in the list is only statistically significant after position 27 (qrnn) of 76 regressors, it is clear that differences are not very high, in fact even the best regressors do not exhibit an excellent performance.

We also measured the elapsed time for each regressor and dataset. Since a simple averaging of times over datasets is not statistically acceptable, because times vary in different ranges for each dataset, we created a Friedman rank of the elapsed times for each regressor over all datasets. The Figure 4.5 plots the time against correlation (both in terms of Friedman rank) for the 20 best regressors in the correlation rank of Table 4.14. The figure shows that the best regressor (extraTrees) in terms of correlation, because it is placed on the left end of the plot (correlation rank about 2), is also the second fastest one because it is placed on the lower end (time rank 23), being only slower than kkn (time rank about 8), which however works much worse (correlation rank about 18). Among the other best regressors in chapter 3, the rf and RRF exhibit slightly lower correlation than extraTrees (they are on its right), but they are much slower (time rank above 60). BstTree and gbm are slightly slower (upper time rank) than extraTrees, but their correlation is much worse (correlation rank about 10). Finally, the elm-kernel and svr are much more slower than extraTrees (time rank above 55) with much lower correlation (ranks about 8 and 14, respectively).

## CHAPTER 5

# CONCLUSIONS

Agriculture is a major sector in the Indian economy, which is affected by changing trends in temperature and rainfall, insufficient water, agriculture practices and nutrient deficiencies. Adequate soil parameters and proper application of fertilizers may help to attenuate these problems. The current research supports the Indian Government to make decisions about improving soil quality and crop production. The soil quality depends on its type and  $pH$ , village-wise fertility indices of  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$ , and on the selected crop. Thus, an automatic prediction of their values from measurements of  $N_2O$ ,  $P_2O_5$ ,  $K_2O$ ,  $SO_4$  and  $EC$ , among others, would reduce the cost of the chemical analysis and save time for specialized technicians. The prediction of levels for the soil nutrients  $N_2O$ ,  $P_2O_5$  and  $K_2O$  would also be very useful for the recommendation of suitable fertilizers. The work developed in this PhD. Thesis is oriented to use machine learning techniques to automatically predict these values for soils of the Indian state of Maharashtra. The results of this study might contribute to design agriculture strategies of the Indian Government to manage the soil fertility degradation, crop productivity and usage of fertilizers.

Despite of being examples of regression problems, our first approach was to quantify the values of the magnitudes to be predicted into low, medium and high levels using thresholds defined by the Indian Government, transforming them into classification problems. We applied a wide and diverse collection of classifiers including decision trees, rule-based classifiers, bagging and boosting ensembles, random forests, neural networks, support vector machines and nearest neighbors classifiers. We achieved values of the Cohen  $\kappa$  about 97% and 90% for soil classification and village-wise  $OC$  fertility index, respectively, above 85% for  $P_2O_5$

fertility index and crop classification, and about 65% for the classification of *Mn* and *Fe* fertility indices. The  $\kappa$  is much lower, about 47%, for *pH* classification, and about 35% for  $N_2O$ ,  $P_2O_5$  and  $K_2O$  classification, with accuracies about 55%. The random forest is the first in a Friedman rank test, being the best in 6 of 10 problems and overcoming 90% of the maximum  $\kappa$  in all the problems, although the difference with almost all the remaining classifiers is not statistically significant in a Wilcoxon signed rank test. Other classifiers with good results are adaboost, rotation forest of J48 base classifiers, the LibSVM support vector machine and the extreme learning machine, both with Gaussian kernels. The remaining methods work much worse. We studied the model validity across three regions (Marathwada, North Maharashtra and Paschim Maharashtra) of the Indian state of Maharashtra, training and testing each classifier with data from different regions, finding compatibilities between North Maharashtra and Paschim Maharashtra for village-wise  $P_2O_5$ , *Mn* and *Fe* fertility indices classification. The data from Marathwada are compatible for  $P_2O_5$  and only relatively compatible ( $\kappa$  about 66%) with Paschim Maharashtra for *Mn* and *Fe* fertility indices and for *pH*.

After finding a relative ability to predict the quantified soil parameters using classification techniques, we tried to solve the regression problem starting by the development of an experimental comparison of regression techniques over a wide collection of general purpose datasets from the UCI machine learning repository. We compared 76 regressors belonging to 20 regressor families: linear regression, generalized linear regression, least squares, partial least squares, lasso regression, ridge regression, neural networks, deep learning, support vector machines, regression trees, bagging ensembles, boosting, random forests, prototype models, Bayesian regression, principal component analysis, generalized additive models, Gaussian processes, quantile regression and other methods. The dataset collection includes 66 regression datasets selected from UCI repository, and some of them provide several datasets, one for each output. The extreme learning machine and support vector regression, both with Gaussian kernels, and the extremely randomized forest (extraTrees) achieve the best results (correlations above 0.79). Specifically, the best regressor (elm-kernel) overcomes 90% of the best correlation for 86.4% of the data sets, while the svr achieves the best correlation for 11 datasets. Some regressors which also are competitive, with correlations above 0.77, are: nearest neighbors (cubist), other random forest (RF and regularized RF), gradient boosting machines (bstTree and gbm) and penalized linear regression (which achieves the best result for 17 datasets). Other families with intermediate results are the bagging ensemble of MARS regressors (correlation 0.75), the Bayesian regularized neural network (0.70) and

model regression tree (M5, 0.75). The deep learning network using Keras only achieves 0.72. Considering time, the elm-kernel and svr are very slow (500 and 1700 times slower than the fastest regressor), but the extremely randomized forest (extraTrees) provides the best trade-off between performance (correlation above 0.79) and speed (just twice slower than the fastest regressor).

In the third stage of the work we applied the collection of 76 regressors to ten soil problems, excluding soil and crop types, which are by nature classification problems, and adding  $K_2O$  and  $Zn$  village-wise fertility indices, not considered in the classification approach because the available data belong to just one class. The extremely randomized forest (extraTrees) achieves the best correlation for six of ten datasets ( $P_2O_5$ ,  $K_2O$ ,  $Fe$ ,  $Mn$  and  $Zn$  village-wise fertility indices and for  $pH$ ), while random forest with feature selection (Boruta), penalized, quantile random forest (qrf) and Gaussian process with radial kernel (gaussprRadial) are the bests for  $OC$  village-wise fertility index, and for  $N_2O$ ,  $P_2O_5$  and  $K_2O$  nutrients, respectively. The results of elm-kernel and svr are worse than in the comparison with generic data sets, being among the first 10 position for 2 and 6 datasets, respectively. Globally, extraTrees achieves the best performance and the second highest speed over all the datasets, followed by random forests and regularized random forest, with similar correlation but low speed, quantile random forest, and by boosting ensembles– gradient boosting of regression trees (bstTree) and generalized boosting regression machine (gbm), with similar time but lower correlation–. One of the main conclusions is that the correlation takes low values: above 0.8 for  $OC$ ,  $Fe$  and  $Zn$  village-wise fertility indices; above 0.75 for  $P_2O_5$  and  $Mn$  fertility indices; above 0.6 for  $K_2O$  index and  $pH$ ; and about 0.5 for  $N_2O$ ,  $K_2O$  and  $P_2O_5$ . Therefore, a prediction with some level of accuracy (correlation above 0.8) is only available for  $OC$ ,  $Fe$  and  $Zn$  village-wise fertility indices, with high levels of uncertainty for the remaining problems. However, if an accurate prediction is not strictly required, an approximated prediction of the quantified levels is available with certain accuracy, as shown in the chapter 2 where the Cohen  $\kappa$  overcomes 60% for  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$  fertility indices, as well as crop and soil type classification.

The future work includes to review in depth the data for the problems with a low correlation by a soil expert in order to make them easier to predict by the available regression methods. We also plan to create village-wise soil fertility maps for the  $OC$ ,  $P_2O_5$ ,  $Mn$  and  $Fe$  nutrients using the classifiers which achieved the best  $\kappa$ , above 65%, for these problems.



# Bibliography

- [1] A. Zell *et al.* SNNS Stuttgart Neural Network Simulator User Manual, Version 4.2. Technical report, IPVR, University of Stuttgart and WSI, University of Tübingen, 1998.
- [2] E. Alfaro, M. Gámez, and N. García. Multiclass corporate failure prediction by Adaboost.M1. *Int. Advances in Economic Res.*, 13:301–312, 2007.
- [3] K. Bache and M. Lichman. UCI machine learning repository, 2013.
- [4] F. Bertrand, M. Maumy-Bertrand, and N. Meyer. *Partial least squares regression for generalized linear models*, 2014. R package version 1.1.1.
- [5] J. Bouma. Using soil survey data for quantitative land evaluation. *Advances in Soil Science*, 9:177–213, 1989.
- [6] R.A. Bowman, W.D. Guenzi, and D.J. Savory. Spectroscopic method for estimation of soil organic carbon. *Soil Sci. Soc. of Am. J.*, 55:563–566, 1991.
- [7] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [8] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
- [9] L. Breiman, J. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees*. Wadsworth and Brooks, 1984.
- [10] C.W. Brungard, J.L. Boettinger, M.C. Duniway, S.A. Wills, and T.C. Edwards Jr. Machine learning for predicting soil classes in three semi-arid landscapes. *Geoderma*, 239–240:68–83, 2015.

- [11] P. Buehlmann and T. Hothorn. Boosting algorithms: regularization, prediction and model fitting (with discussion). *Statistical Science*, 22(4):477–505, 2007.
- [12] P. Buehlmann and B. Yu. Boosting with the  $l_2$  loss: regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.
- [13] A.J. Cannon. Quantile regression neural networks: implementation in R and application to precipitation downscaling. *Computers & Geosciences*, 37:1277–1284, 2011.
- [14] J.M. Chambers. *Linear models*, chapter 4. J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole, 1992.
- [15] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Trans. on Intel. Syst. and Technol.*, 2:27:1–27:27, 2011.
- [16] W. Chesworth. *Encyclopedia of Soil Science*. Springer, 2008.
- [17] S. Chiu. Method and software for extracting fuzzy classification rules by subtractive clustering. In *Fuzzy Information Processing Society, NAFIPS*, pages 461–465, 1996.
- [18] F. Chollet. Keras. <https://github.com/fchollet/keras> (visited March, 29, 2017)., 2015. <https://keras.io>.
- [19] Y. Chtiouia, S. Panigrahaia, and L. Francl. A generalized regression neural network and its application for leaf wetness prediction to forecast plant disease. *Chem. and Intel. Lab. Sys.*, 48:47–58, 1999.
- [20] H. Chun and S. Keles. Sparse partial least squares for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society*, 72:3–25, 2010.
- [21] W.W. Cohen. Fast effective rule induction. In *Int. Conf. on Mach. Learn.*, pages 115–123, 1995.
- [22] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.
- [23] Department of Agriculture & Cooperation. Methods manual. Soil testing in India. Technical report, Ministry of Agriculture, Government of India, New Delhi, 2011.

- [24] Directorate of Economics and Statistics. Economic survey of Maharashtra. Technical report, Planning Department, Maharashtra Government, Mumbai, 2015.
- [25] A.J. Dobson. *An introduction to generalized linear models*. Chapman and Hall, 1990.
- [26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.
- [27] C. Feller, E. Blanchart, M. Bernoux, R. Lal, and R. Manlay. Soil fertility concepts over the past two centuries: the importance attributed to soil organic matter in developed and developing countries. *Archives of Agronomy and Soil Science*, 58(S1):S3–S21, 2012.
- [28] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real classification problems? *J. Mach. Learn. Res.*, 15:3133–3181, 2014.
- [29] C.L. Ford. Determination of sodium and potassium oxides by flame photometry. *Anal. chem.*, 26:1578–1581, 1954.
- [30] Foreign Agricultural service. India’s Agricultural Exports Climb to Record High. Technical report, United States Department of Agriculture, USDAFAS, Office of Global Analysis, Global Policy Analysis Division, 2014.
- [31] F.D. Foresee and M. T. Hagan. Gauss-Newton approximation to Bayesian regularization. In *International Joint Conference on Neural Networks*, pages 1930–1935, 1997.
- [32] E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In *Int. Conf. on Mach. Learn.*, pages 144–151, 1998.
- [33] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Syst. Sci.*, 55:119–139, 1997.
- [34] J.H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.
- [35] J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

- [36] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.
- [37] S. García, A. Fernández, A.D. Benítez, and F. Herrera. Statistical comparisons by means of non-parametric tests: A case study on genetic based machine learning. In *Proceedings of the II Congreso Español de Informática (CEDI 2007). V Taller Nacional de Minería de Datos y Aprendizaje (TAMIDA)*, pages 95–104, 2007.
- [38] A. Gelman, A. Jakulin, M.G. Pittau, and Y.S. Su. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383, 2009.
- [39] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.
- [40] M. J. Glendining, A. G. Dailey, D. S. Powlson, G. M. Richter, J.A. Catt, and A. P. Whitmore. Pedotransfer functions for estimating total soil nitrogen up to the global scale. *European Journal of Soil Science*, 62:13–22, 2011.
- [41] J.J. Goeman. L-1 penalized estimation in the cox proportional hazards model. *Biometrical Journal*, 52:70–84, 2010.
- [42] J.W. Groenigen, D. Huygens, P. Boeckx, Th.W. Kuyper, I.M. Lubbers, T. Rütting, and P.M. Groffman. The soil N cycle: new insights and key challenges. *Soil*, 1:235–256, 2015.
- [43] T. Grubinger, A. Zeileis, and K.P. Pfeiffer. Evtree: evolutionary learning of globally optimal classification and regression trees in R. *Journal of Statistical Software*, 61(1):1–29, 2014.
- [44] P. Gruhn, F. Goletti, and M. Yudelman. *Integrated Nutrient Management, Soil Fertility, and Sustainable Agriculture: Current Issues and Future Challenges*. International Food Policy Research Institute, 2000.
- [45] J.M. Guerrero, G. Pajares, M. Montalvo, J. Romeo, and M. Guijarro. Support vector machines for crop/weeds identification in maize fields. *Expert Syst. Appl.*, 39:11149–11155, 2012.

- [46] M. Hall and E. Frank. Combining naive Bayes and decision tables. *Proc. Artif. Intel. Soc. Conf.*, pages 318–319, 2008.
- [47] M. Hall, E. Frank, G.Ho., B. Pfahringer, P. Reutemann, and I.H. Witten. The Weka data mining software: an update. *SIGKDD Explorations*, 11:10–18, 2009.
- [48] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270, 1993.
- [49] B. Heung, H. Chak Ho, J. Zhang, A. Knudby, C.E. Bulmer, and M.G. Schmidt. An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping. *Geoderma*, 265:62–77, 2016.
- [50] M.G. Hill, P.G. Connolly, P. Reutemann, and D. Fletcher. The use of data mining to assist crop protection decisions on kiwifruit in New Zealand. *Comput. Electron. Agric.*, 108:250–257, 2014.
- [51] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.
- [52] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.
- [53] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Sys., Man, and Cybern.-Part B: Cybern.*, 42(2):513–529, 2012.
- [54] P.J. Huber. *Robust statistics*. Wiley, 1981.
- [55] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13:411–430, 2000.
- [56] H. Ishwaran, J.S. Rao, and U.B. Kogalur. Spikeslab : prediction and variable selection using spike and slab regression. *The R Journal*, 2:68–73, 2010.
- [57] M.L. Jackson. *Soil chemical analysis*. Prentice Hall of India Pvt. Ltd., New Delhi, 1958.

- [58] S. K. Jain, V. P. Singh, and M. Th. V. Genuchten. Analysis of Soil Water Retention Data Using Artificial Neural Networks. *Journal of Hydrologic Engineering*, pages 415–420, 2004.
- [59] M.E. Jakubauskas, D.R. Legates, and J.H. Kastens. Crop identification using harmonic analysis of time-series AVHRR NDVI data. *Comput. Electron. Agric.*, 37:127–139, 2002.
- [60] C.H. Jones. Activity of organic nitrogen as measured by the alkaline permanganate method. *J. Ind. Eng. Chem.*, 24:438–441, 1912.
- [61] K. Hechenbichler and K.P. Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. Technical report, Ludwig-Maximilians University Munich, 2004.
- [62] A. Kapelner and J. Bleich. bartMachine: machine learning with Bayesian additive regression trees. *Journal of Statistical Software*, 70(4):1–40, 2016.
- [63] J.C. Katyal and R.K. Rattan. Secondary and micronutrients: research gaps and future needs. *Fertil. News*, 48:9–20, 2003.
- [64] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.
- [65] M.B. Kurşa and W.R. Rudnicki. Feature selection with the Boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.
- [66] K. Lamorski, Y. Pachepsky, C. Slawiński, and R. T. Walczak. Using Support Vector Machines to Develop Pedotransfer Functions for Water Retention of Soils in Poland. *Soil Sci. Soc. Am. J.*, 72:1243–1247, 2008.
- [67] C.L. Lawson and R.J. Hanson. *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), 1995.
- [68] Leggeit and D.P. Argyle. The DTPA-extractable iron, manganese, copper, and zinc from neutral and calcareous soils dried under different conditions. *Soil Sci. Soc. Am. J.*, 47(3):518–522, 1983.

- [69] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F.E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.
- [70] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.
- [71] Mahatma Phule Agricultural University. Krishi Darshani. Parbhani, Maharashtra, India, 2016. Page 18 (in Marathi language).
- [72] Matlab. *version 7.14 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012.
- [73] N. Meinshausen. Relaxed lasso. *Computational Statistics and Data Analysis*, pages 374–393, 2007.
- [74] N. Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010.
- [75] W. Melssen, R. Wehrens, and L. Buydens. Supervised Kohonen networks for classification problems. *Chemom. Intell. Lab. Syst.*, 83:99–113, 2006.
- [76] P. Melville and R.J. Mooney. Creating diversity in ensembles using artificial data. *Inform. Fusion: special issue on diversity in multiclassifier systems*, 6(1):99–111, 2004.
- [77] H. Merdun, O. Cinar, R. Meral, and M. Apan. Comparison of artificial neural network and regression pedotransfer functions for prediction of soil water retention and saturated hydraulic conductivity. *Soil and Tillage Research*, 90:108–116, 2006.
- [78] B.H. Mevik and H.R. Cederkvist. Mean squared error of prediction (msep) estimates for principal component regression (pcr) and partial least squares regression (plsr). *Journal of Chemometrics*, 18(9):422–429, 2004.
- [79] B. Minasny, A. B. McBratney, and K. L. Bristow. Comparison of different approaches to the development of pedotransfer functions for water-retention curves. *Geoderma*, 93:225–253, 1999.
- [80] I. Mizera and R. Koenker. Convex optimization in r. *Journal of Statistical Software*, 60(5):1–23, 2014.

- [81] M.S. Mkhabela, P. Bullock, S. Raj, S. Wang, and Y. Yang. Crop yield forecasting on the Canadian prairies using MODIS NDVI data. *Agric. For. Meteorol.*, 151:385–393, 2011.
- [82] A.M. Molinaro, K.Lostritto, and M.J. van der Laan. Partdsa: deletion/substitution/addition algorithm for partitioning the covariate space in prediction. *Bioinformatics*, 26(10):1357–63, 2010.
- [83] A. Mucherino, P. Papajorgji, and P.M. Pardalos. A survey of data mining techniques applied to agriculture. *Oper. Res.*, 9:121–140, 2009.
- [84] G.R. Muhr, N.P. Datta, S.N. Shankara, F. Dever, V.K. Lecy, and R.R. Donahue. *Soil testing in India*. U.S. Agency for International Development, Mission to India, 1965.
- [85] L.G. Naidu, V. Ramamuthy, G.S. Sidhu, and D. Sarkar. Emerging deficiency of potassium in soils and crops of india. *Karnataka J. Agric. Sci.*, 24:12–19, 2011.
- [86] U.P. Narkhede and K.P. Adhiya. A study of clustering techniques for crop prediction - a survey. *Am. Int. J. of Res. in Sci., Technol., Eng. and Math.*, 5(1):44–48, 2014.
- [87] National Research Council. *Alternative agriculture*. Technical report, National Academy of Sciences, Washington, DC, 1989.
- [88] V.P. Obade and R. Lal. Towards a standard technique for soil quality assessment. *Geoderma*, 265:96–102, 2016.
- [89] Planning Commission Government of India, editor. *Eleventh Five Year Plan 2007-2012: Volume I: Inclusive Growth; Volume II: Social Sector Services; Volume III: Agriculture, Rural Development, Industry, Services, and Physical Infrastructure*. Oxford University Press, New Delhi, 2008.
- [90] S.R. Olsen. *Estimation of available phosphorus in soils by extraction with sodium bicarbonate*. Circular series. U.S. Dept. of Agriculture, 1954.
- [91] Y. Pachepsky, K. Rajkai, and B. Tóth. Pedotransfer in soil physics: Trends and outlook - A review. *AGROKÉMIA ÉS TALAJTAN*, 64:339–360, 2015.
- [92] Y.A. Pachepsky and W.J. Rawls. Preface: Status of pedotransfer functions. *Development fo Pedotransfer Functions in Soil Hydrology*, 30:7–16, 2004.

- [93] S. Panigrahy and S.A. Sharma. Mapping of crop rotation using multirate Indian remote sensing satellite digital data. *ISPRS J. of Photogrammetry and Remote Sens.*, 52:85–91, 1997.
- [94] X.E. Pantazi, D. Moshou, T. Alexandridis, R.L. Whetton, and A.M. Mouazen. Wheat yield prediction using machine learning and advanced sensing techniques. *Comput. Electron. Agric.*, 121:57–65, 2016.
- [95] J. Park and I.W. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.
- [96] M.A. Peña and A. Brenning. Assessing fruit-tree crop classification from Landsat-8 time series for the Maipo Valley, Chile. *Remote Sens. Environ.*, 171:234–244, 2015.
- [97] A. Philibert, C. Loyce, and D. Makowski. Prediction of N<sub>2</sub>O emission from local information with random forest. *Environmental Pollution*, 177:156–163, 2013.
- [98] A. Philibert, C. Loyce, and D. Makowski. Predicting nitrous oxide emissions with a random-effects model. *Environmental Modelling and Software*, 61:12–18, 2014.
- [99] T. Pohlert. *The pairwise multiple comparison of mean ranks package (PMCMR)*, 2014. R package.
- [100] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
- [101] R. Quinlan. Combining instance-based and model-based learning. In *Proc. Intl. Conf. on Machine Learning*, pages 236–243, 1993.
- [102] R.J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.
- [103] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.
- [104] P. Ramesh, N. R. Panwar, A. B. Singh, S. Ramana, S.K. Yadhav, R. Shrivastava, and A.S. Rao. Status of organic farming in india. *Current Science*, 98(9):1190–1194, 2010.

- [105] B. Rammoorthy and J.C. Bajaj. Available *N*, *P* and *K* status of Indian soils. *Fertilizer news*, 14(8):24–26, 1969.
- [106] M. Rashidi and M. Seilsepour. Modeling of soil total nitrogen based on soil organic carbon. *ARN J. of Agric. and Biol. Sci.*, 4(2):1–5, 2009.
- [107] D.W. Reeves. The role of soil organic matter in maintaining soil quality in continuous cropping systems. *Soil and Tillage Research*, 43:131–167, 1997.
- [108] Y. Ren, L. Zhang, and P. Suganthan. Ensemble classification and regression recent developments, applications and future directions. *IEEE Computational intelligence magazine*, pages 41–53, 2016.
- [109] L.A. Richards, L.E. Allison, L. Bernstein, C.A. Bower, J.W. Brown, M. Fireman, J.T. Hatcher, H.E. Hayward, G.A. Pearson, R.C. Reeve, A. Richards, and L.V. Wilcox. Diagnosis and improvement of saline and alkaline soils. *Science*, 12:800, 1954.
- [110] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge Univ. Press, 1996.
- [111] B.D. Ripley. *Modern applied statistics with S*. Springer, 2002.
- [112] J.J. Rodríguez and L.I. Kuncheva. Rotation forest: A new classifier ensemble method. *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 28(10):1619–1630, 2006.
- [113] J. Rogan, J. Franklin, D. Stow, J. Miller, C. Woodcock, and D. Roberts. Mapping land-cover modifications over large areas: A comparison of machine learning algorithms. *Remote Sens. Environ.*, 112:2272–2283, 2008.
- [114] J.R. Romero, P.F. Roncallo, P.C. Akkiraju, I. Ponzoni, V.C. Echenique, and J.A. Carballido. Using classification algorithms for predicting durum wheat yield in the province of Buenos Aires. *Comput. Electron. Agric.*, 96:173–179, 2013.
- [115] S. De Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems*, 18:251–263, 1993.
- [116] S. De Jong. Comment on the PLS kernel algorithm. *Journal of Chemometrics*, 8:169–174, 1994.

- [117] M.G. Schaap, F. J. Leij, and M. Th. V. Genuchten. ROSETTA: a computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. *Journal of Hydrology*, 251:163–176, 2001.
- [118] J.L. Sehgal. *Agro-ecological Regions of India*. Technical bulletin (National Bureau of Soil Survey & Land Use Planning). Indian Council of Agricultural Research, 1990.
- [119] P.J. Sheela and K. Sivaranjani. A brief survey of classification techniques applied to soil fertility prediction. In *Int. Conf. Eng. Trends in Sci. and Hum.*, pages 80–83, 2015.
- [120] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for cox’s proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011.
- [121] R. P. Singh and S. K. Mishra. Available macro nutrients (n, p, k, and s) in the soils of chiraigaon block of district varanasi (u.p) in relation to soil characteristics. *Indian J.Sci.Res.*, 3(1):97–100, 2012.
- [122] L. Song, P. Langfelder, and S. Horvath. Random generalized linear model: a highly accurate and interpretable ensemble predictor. *BMC Bioinformatics*, 14(1):1–22, 2013.
- [123] D.F. Specht. Probabilistic neural networks. *Neural Netw.*, 3:109–118, 1990.
- [124] D.F. Specht. A general regression neural network. *IEEE Trans. on Neural Networks*, 2:568–576, 1991.
- [125] B.V. Subbaiah and G.L. Asija. A rapid procedure for the estimation of available nitrogen in soil. *Current Sci.*, 25:259–260, 1956.
- [126] R. Taghizadeh-Mehrjardi, K. Nabiollahi, B. Minasny, and J. Triantafyllis. Comparing data mining classifiers to predict spatial distribution of USDA-family soil groups in Baneh region, Iran. *Geoderma*, 253–254:67–77, 2015.
- [127] K. Tatsumi, Y. Yamashiki, M.A. Canales Torres, and C.L. Ramos Taípe. Crop classification of upland fields using random forest of time-series Landsat 7 ETM+ data. *Comput. Electron. Agric.*, 115:171–179, 2015.

- [128] The Agricultural and Processed Food Products Export Development Authority (APEDA). Indian Agro and Food Industry. Technical report, Incredible India, New Delhi, 2016.
- [129] M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, 2001.
- [130] M.-S. Turmel, A. Speratti, F. Baudron, N. Verhulst, and B. Govaerts. Crop residue management and soil health: A systems analysis. *Agric. Systs*, 134:6–16, 2015.
- [131] A.J. Viera and J.M. Garrett. Understanding interobserver agreement: the kappa statistic. *Family Medicine*, 37(5):360–363, 2005.
- [132] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
- [133] S.N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society*, 1(73):3–36, 2011.
- [134] N. Xiao, D.S. Cao, M.Z. Li, and Q.S. Xu. Enpls: an R package for ensemble partial least squares regression. *arXiv preprint*, 2016.
- [135] T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Trans. Inf. Theor.*, 57(7):4689–4708, 2011.
- [136] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67:301–320, 2005.
- [137] H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, 36(4):1509–1533, 2008.

# List of Figures

Fig. 1.1	Geographical representation of 6 regions of Maharashtra (India). Marathwada, Paschim Maharashtra and North Maharashtra are study areas, highlighted by red borders. . . . .	2
Fig. 2.1	Degree of acidity and alkalinity of soil [16], with nine classes (up) and with the four classes considered in this paper (down). . . . .	13
Fig. 2.2	Soil map of Maharashtra [118]. The geographical study areas are highlighted with outlines: red (Marathwada), blue (North Maharashtra) and pink (Paschim Maharashtra). . . . .	14
Fig. 2.3	Maps with the geographical location of crop (left panel) and soil type (right panel) data over several districts in the Marathwada region (red outline in the map of Figure 2.2). Each point locates a different village, from where several patterns are recorded. . . . .	15
Fig. 2.4	Intervals of $\kappa$ (in %) of the different classifiers for each problem (the filled square shows the mean $\kappa$ ). . . . .	25
Fig. 2.5	Intervals of percentages of the maximum $\kappa$ for each problem, for all the problems and for each classifier. . . . .	25
Fig. 3.1	Friedman rank of RMSE (upper panel) and correlation (lower panel) for the 20 best regressors. . . . .	50
Fig. 3.2	Average RMSE (upper panel) and correlation (lower panel) over all the datasets for the 20 best regressors. . . . .	54
Fig. 3.3	Best correlation achieved by some regressor for each dataset. . . . .	56

Fig. 3.4	Upper panel: best RMSE, in blue, and RMSE achieved by elm-kernel (in red) for each dataset, sorted by increasing elm-kernel RMSE. Lower panel: the analogous plot for correlations, sorted by decreasing values. . . . .	57
Fig. 3.5	Percentage of the best correlation achieved by the elm-kernel, for each dataset, sorted by decreasing values. . . . .	58
Fig. 3.6	Average correlation (sorted decreasingly) of the best regressors of each family. . . . .	60
Fig. 3.7	Friedman rank of the time (vertical axis) against the Friedman rank of the correlation for the 20 best regressors in Table 3.3. . . . .	63
Fig. 4.1	Twenty best correlation values (upper panel) and scatter plot of Boruta (lower panel) for the prediction of the <i>OC</i> village-wise soil fertility index in the state of Marathwada. . . . .	70
Fig. 4.2	Twenty best correlations (upper panel) and scatter plot of extraTrees (lower panel), which achieves the best results for the prediction of <i>Fe</i> village-wise soil fertility index. . . . .	73
Fig. 4.3	Correlation (upper panel) and scatter plot of penalized (lower panel) for the prediction of <i>N<sub>2</sub>O</i> level in the state of Marathwada. . . . .	76
Fig. 4.4	Correlation of the twenty best regressors (upper panel) and scatter plot of extraTrees (lower panel), which achieves the best correlation for the prediction of soil <i>pH</i> . . . . .	78
Fig. 4.5	Friedman rank of the time (vertical axis) against the Friedman rank of the correlation (horizontal axis) for the 20 best regressors over the ten soil data sets. . . . .	83

# List of Tables

Tabla 2.1	Intervals defined by the Indian Government [23] for the major and micro nutrients respectively [84, 63], and rate of nutrient index [105]. . . . .	10
Tabla 2.2	Inputs, number of total patterns (#Patterns) and patterns per class for each classification problem. The labels SA, N, SAL and MAL mean slightly acid, neutral, slightly alkaline and moderately alkaline, respectively. The compounds $SO_4$ and $CaCO_3$ are sulfate and calcium carbonate respectively. The symbol — means that patterns of that class are not available. . . . .	11
Tabla 2.3	Values of $\kappa$ (in %) achieved by each classifier for each classification problem. The best $\kappa$ for each problem is in bold. . . . .	23
Tabla 2.4	Columns 1-6: classifier ranking and position according to the Friedman rank test. Columns 7-12: classifiers ordered by decreasing $p$ -value of a Wilcoxon signed rank test comparing the best ranked classifier (rf_r) to the remaining ones (the significant tests are in bold). The label — means that rf_r can not be compared to itself . . . . .	23
Tabla 2.5	Confusion matrices (in %) of the best classifiers (showed in the matrix header between brackets) for the classification of soil fertility indices. The symbol “—” means that the class is not available. Labels L, M and H mean low, medium and high respectively. . . . .	26
Tabla 2.6	Confusion matrices (in %) of the best classifiers for $N_2O$ , $P_2O_5$ and $K_2O$ . Labels L, M and H mean low, medium and high, respectively. The values in the matrix diagonal are in bold. . . . .	27

Tabla 2.7	Confusion matrix (in %) of the best classifier (rf_r, with $\kappa= 47.32\%$ and accuracy= 69.63%) for soil <i>pH</i> classification. Labels SA, N, SAL and MAL mean slightly acidic, neutral, slightly alkaline and moderately alkaline respectively. . . . .	28
Tabla 2.8	Confusion matrix (in %) of rf_r for crop classification ( $\kappa=88.13\%$ , accuracy=93.09%). Labels R and I mean rainfed and irrigated respectively.	29
Tabla 2.9	The confusion matrix (in %) of the two best classifiers (dtnb_w and rf_w, $\kappa= 97.82\%$ and 96.80% respectively) for soil type classification. Labels L and M mean light and medium respectively. . . . .	29
Tabla 2.10	Values of $\kappa$ (in %) achieved by the best classifier training and testing with patterns of different regions (first column, see text for region labels). The symbol ‘—’ means that data are not available. . . . .	30
Tabla 3.1	Collection of 66 datasets from the UCI repository: original name in the UCI repository; datasets created from the original one; number of patterns and inputs, before and after preprocessing. . . . .	48
Tabla 3.2	List of regressors (and datasets) whose execution did not finish within 150 h. or required more than 128 GB RAM. . . . .	49
Tabla 3.3	Friedman rank of the RMSE (left) and of the correlation (right), average correlation and <i>p</i> -value of the Posthoc Friedman Nemenyi test comparing the best regressor to the remaining ones. Continued in Table 3.4. . . . .	51
Tabla 3.4	Continuation of Table 3.3. . . . .	52
Tabla 3.5	List of the regressors which achieve the best RMSE (left part) or correlation (right part) for some dataset. . . . .	53
Tabla 3.6	Regressor families sorted by the Friedman rank of its best regressor according to the correlation coefficient (its average value is reported in the last column). . . . .	59
Tabla 3.7	Positions of the regressors, grouped by families, in the Friedman rank (data extracted from Tables 3.3-3.4), in the same order as Figure 3.6. The acronym mlpwd means mlpWeightDecay. . . . .	61
Tabla 3.8	Friedman rank of the elapsed time spent by the regressors over all the datasets.	62
Tabla 4.1	Intervals defined by the Indian Government [23] for the calculation of village-wise soil fertility indices of $K_2O$ and Zn nutrients [84, 63]. . . . .	66

Tabla 4.2 Values of parameters  $a$  and  $b$  for the prediction of different soil nutrients and crops. . . . . 67

Tabla 4.3 RMSE for the prediction of  $OC$  village-wise soil fertility index. . . . . 69

Tabla 4.4 RMSE for the prediction of  $P_2O_5$  village-wise soil fertility index. . . . . 69

Tabla 4.5 RMSE for the prediction of  $K_2O$  village-wise soil fertility index. . . . . 71

Tabla 4.6 RMSE for the prediction of  $Fe$  village-wise soil fertility index. . . . . 72

Tabla 4.7 RMSE for the prediction of  $Mn$  village-wise soil fertility index. . . . . 72

Tabla 4.8 RMSE for the prediction of  $Zn$  village-wise soil fertility index. . . . . 74

Tabla 4.9 RMSE values for the prediction of  $N_2O$  nutrient level. . . . . 75

Tabla 4.10 RMSE values for the prediction of  $P_2O_5$  nutrient level. . . . . 75

Tabla 4.11 RMSE values for the prediction of  $K_2O$  nutrient level. . . . . 77

Tabla 4.12 RMSE values for the prediction of soil  $pH$ . . . . . 77

Tabla 4.13 Ten bests correlations for the prediction of each soil dataset. . . . . 79

Tabla 4.14 Friedman rank of the RMSE (left) and of the correlation (right), and average correlation. Continued in Table 4.15. . . . . 80

Tabla 4.15 Continuation of Table 4.14. . . . . 81

Tabla 4.16 List of the  $p$ -values achieved by the Wilcoxon signed rank test comparing the correlations of extraTrees and the remaining 75 regressors over the 10 soil regression problems. . . . . 82

