SUPPLEMENTARY MATERIAL

April 26, 2021

This material resumes and expands on some of the topics presented in the main review, with the purpose of providing a more formal description of some methods such as the Naive Bayes algorithm, LDA, linear and logistic regression, SVM, perceptron and MLP. As already mentioned in the main paper, because the field is at the intersection of Statistics, Data Science and Engineering, often different names are used interchangeably to indicate the same entity which can confuse the nonspecialists. For this reason an effort will be made to list possible alternative terminologies and the preferred choice will be highlighted in italics. Mostly, descriptions of the methods will draw upon [1], with the purpose of conveying intuitions, skipping the details of their implementation/optimization procedures. Upper case letters will represent generic variables, lower case variables will represent observations: the observed value of X is an NxM matrix with elements $\mathbf{x}_{ij}$, with $i \in 1, ..., N$ and $j \in 1, ..., M$.

The methods described below belong to the class of supervised learning algorithms, which are the most used type of ML and which learn a mapping from input X to output Y where Y is a continuous/*quantitative* or a discrete/categorical/*qualitative* variable. In what follows a realization of X will be an NxM-dimensional matrix of elements $(x_{ij})_{i=1}^{N}{}_{j=1}^{M}$ with N *observations*/measurements (e.g., individuals or samples) and M *features*/predictors/attributes/independent variables (e.g., genetic factors or genomic variables); $(y_i)_{i=1}^{N}$ will be an N-dimensional vector of *output*/outcome/responses/dependent variables assuming continuous or discrete values.

When the output is quantitative, i.e., some output values are bigger than others and outputs close in value are close in nature, the supervised learning problem is known as a *regression problem* (e.g., prediction of a quantitative phenotype such as age). When the outcome is qualitative with no explicit ordering, the prediction problem is called *classification*, and the output (the *class*) is represented by digits (e.g. $0, 1$ as in a case control study or $1, ..., K$ as in cancer type or disease trait classification) or dummy variables ("case", "control", or "cancer A", "cancer B", "cancer C"). A third type of variable is ordered categorical (e.g., "mild", "medium", "severe" as in symptom severity classification): this type is less common and will be skipped here (we recommend [1] for details). Although this naming convention separates regression and classification problems into two separate classes both are tasks in function approximation.

0.1 Naive Bayes

The Naive Bayes (NB) algorithm is a classification algorithm. It belongs to the class of *generative models*, i.e., it builds a joint probability model $P(X, Y)$, therefore a full statistical model for both input and output: given this model the output can be *generated* from the input; also it uses Bayes' rule to infer category labels. It is called *naive* because it makes a very strong but simplifying assumption that all pairs of features are conditionally independent given the labels, an assumption that is generally not true. In mathematical notation, given $Y = k$:

$$f_k(X) = \Pi_{j=1}^{M} f_{kj}(X_j),$$

where $P(Y = k, X = x) = \frac{\pi_k f_k(x)}{\sum_{h=1}^{K} \pi_h f_h(x)}$. To define an NB algorithm each output variable needs to be assigned a prior distribution π_k (the probability of class k) and a marginal distribution f_{kj} for each $k \in 1, ..., K$ and $j \in 1, ..., M$. For continuous features, binary features, categorical unordered features, or features with unknown distribution, the distribution of choice is Gaussian, Bernoulli, Multinomial, or estimated with kernel density, respectively. The parameters of both the prior and the marginal distributions can be set a priori or estimated with non parametric methods from the training set. Naive Bayes classifiers are adopted for their simplicity or as a baseline for a comparison with more complex classifiers. Despite its oversimplistic assumptions, the NB algorithm can outperform more sophisticated alternatives: although the estimator may be *biased*, i.e., make erroneous assumptions, it has low *variance*, i.e., it isn't sensitive to small fluctuations in the training set. Its use it typically recommended when the feature space is large (high M) and density estimations become unfeasible.

0.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) (also called Normal Discriminant Analysis) is a classification algorithm in some way similar to the NB algorithm. Like NB, LDA is a generative model. Differently from NB, it makes the specific assumption that each class density is multivariate Gaussian and all classes $k \in 1, ..., K$ share the same variance $\mathbf{\Sigma}$:

$$f_k(x) \sim \mathcal{N}(\mu_k, \mathbf{\Sigma_k}) = \mathcal{N}(\mu_\mathbf{k}, \mathbf{\Sigma}).$$

The parameters of the model are the overall variance Σ, π_k (the class prior), and μ_k for class k, and are estimated directly from the training data. It can be shown that the *decision boundary* estimated by LDA, i.e., *the line or margin that separates the classes*, is a function of $\mathbf{\Sigma}$ and of the position of the class centroids: the LDA algorithm assigns a new observation to the class of the closest centroid, in a transformed space obtained from a factorization of $\mathbf{\Sigma}$, modulo π_k. Also LDA decision boundaries are flat in the space of the input features: a line in two dimensions, a hyperplane in three or more dimensions. *Regularized* versions of LDA, i.e., that *add information in order to prevent overfitting*, also exist for example with shrinkage of $\mathbf{\Sigma}$.

0.3 Linear regression, regularization strategies and overfitting

So far we have given two examples of generative models. While generative models make missing value estimation straightforward (as they can generate missing data drawing from the model distributions), they don't directly try to maximize the quality of the output on the training set $P(Y = k|X = x)$ but instead try to model the more general joint distribution. In what follows we will describe *discriminative models*, which instead directly try to maximize the quality of the output on the training set. Typically, discriminative methods use an additional *regularization* term in the training cost function. The most popular discriminative models are Linear Regression for a continuous output and Logistic Regression for a binary output.

Linear Regression minimizes the negative loglikelihood or, equivalently, the Residual Sum of Squares (RSS):

$$\sum_{i=1}^{N}(y_i - \theta^t x_i)^2 \tag{1}$$

where y_i is the i-th output, $x_i = (x_{ij})_{j=1}^M$ are the M features of the i-th observation ($x_{i0} = 1$), $\theta = (\theta_j)_{j=0}^M$ are the parameters of the model or regression coefficients, one for each input feature j plus an intercept θ_0 for $j = 0$, and $\theta^t x_i$ is the scalar product between vector θ and vector x_i. *When the number of features is too large*, a penalty/*regularization* term can be added to the model, for having too many features in the model:

$$\sum_{i=1}^{N}(y_i - \theta^t x_i)^2 + \lambda B(\theta).$$

For increasing values of the parameter λ, the amount of penalty increases (the bias increases and the variance decreases), vice versa for decreasing values of λ. *Regularization prevents overfitting*, i.e. fitting a too complex model to the data so that the model fits very well the training data but predictions of testing data is very bad. Depending on the functional form of B (L_1 penalty $\sum_{j=1}^M |\theta_j|$, L_2 penalty $\sum_{j=1}^M \theta_j^2$, or a linear combination of the two - to mention the most common strategies), the algorithm is called LASSO, Ridge, or Elastic Net linear regression, respectively. Ridge Regression shrinks the coefficients θ_j of a feature j towards zero, but doesn't set any of them exactly to zero. LASSO [2] instead forces some of the coefficients to zero and can thus be seen as a *variable selection* method. For this reason, LASSO produces simpler and therefore more interpretable models compared to Ridge regression. Elastic Net combines properties of both regressors by both shrinking coefficients (like in Ridge regression) and setting some of them to zero (as in LASSO). *Cross-validation methods can be used to identify which of these techniques performs better on a specific dataset.*

0.4 Logistic Regression and Softmax Regression

Despite its name, Logistic Regression (LR) (also called Binomial Regression) is not a regression algorithm but a binary (or two-class) classification algorithm (LR is in fact a generalized linear model with a logistic link function, but this goes beyond the scope of this review). Its extension to the multiclass classification problem is called Multinomial Logistic Regression or Softmax Regression (the Binomial distribution is replaced by the Multinomial distribution). LR minimizes the negative loglikelihood loss or *Log Loss*/binomial deviance

$$\sum_{i=1}^{N} \log(1 + e^{-2y_i f(x_i)})$$

where output $y_i \in \{+1, -1\}$, $f(x) = g(\theta^t x) = \frac{1}{1+e^{-\theta^t x}}$, and $g(z) = \frac{1}{1+e^{-z}}$ is the *Logistic*/Sigmoid function. As for linear regression, a regularization term can be added to the Log Loss to obtain Ridge, LASSO, and Elastic Net LR.

It can be shown that the decision boundary of LR and its multinomial version is flat (a hyperplane) in the space of the input features, as in LDA. However the parameters defining the hyperplane are different, as LDA makes additional assumptions on the distribution of the features. Because of those assumptions, LDA parameters have lower variance but can incur in a bias when the assumptions are not valid. For example, if outliers are present, e.g., observations that are far from the decision boundary, they will contribute to the estimation of the LDA covariance matrix and therefore to the estimation of the optimal parameters, while in LR they will be down-weighted. Therefore LR is more robust to outliers. Often though, LR and LDA give similar results.

Although LR can be extended to generate non linear decision boundaries with the use of a kernel, this would result in a high computational cost. A different discriminative model, namely SVM (shown below), can be extended from its linear version to use different kernels at a low computational cost.

0.5 Support Vector Machine

The Support Vector Machine (SVM) shares many similarities with LR. For two-class classification with output $y_i \in \{-1, +1\}$, SVM minimizes a loss function called *Hinge Loss* (HL) plus a penalty term $B = \frac{1}{2} \|\theta\|^2$ (L_2 penalty):

$$C \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{1}{2} \|\theta\|^2 , \tag{2}$$

where $f(x) = \theta^t x$, the subscript "+" indicates positive part, C is the regularization parameter (for an analogy with LDA $C = 1/\lambda$), and $y \cdot f(x)$ is the *margin*, which is analogous to the residual $y - f(x)$ in linear regression (1): observations with positive margins $y_i f(x_i) \geq 0$ are classified correctly, observations with negative margin are misclassified. The name *support vector* arises from the fact that many of the input observations x_i (vectors in this context) do not play a big role in defining the SVM optimal boundary, the few that contribute are called support vectors. More precisely *observations that are inside their class boundary* ($1 - y_i f(x_i) < 0$ or equivalently $y f(x_i) < 1$, see Figure 4) *do not play a (big) role in shaping the boundary*: their individual contribution to the HL is 0 (from (2)) and as the optimal boundary is defined by the set of θ-s that minimizes (2), their contribution to the definition of the optimal value of θ, i.e., to the shape of the boundary, is null. Observations x_i on the wrong side of the margin, i.e., the support vectors ($1 - y_i f(x_i) > 0$), contribute to the loss proportionally to their distance from the margin $f(x_i) = \theta^t x_i$.

Because the HL depends only on a subset of the input observations, optimal parameters can be efficiently estimated. Extensions of the SVM with nonlinear kernels $k(x_i, x_j)$, the most popular being the Gaussian kernel (other non linear kernels are rarely used), allow for nonlinear boundaries. Some versions of the SVM use a smoothed HL (as this is not differentiable and therefore difficult to optimize).

0.6 The perceptron learning algorithm and Artificial Neural Networks

The perceptron learning algorithm minimizes the loss function:

$$\sum_{i=1}^{N} -max(0, y_i f(x_i)) \tag{3}$$

with $f(x_i) = \theta^t x_i$. From this definition, it follows that observations x_i that fall inside their class boundary ($y_i f(x_i)$<0) do not contribute to the loss. In other words, *the perceptron algorithm is designed to find a decision boundary that has minimal distance from misclassified points*. Furthermore the contribution of an observation to the loss (or equivalently to the definition of the decision boundary) depends on the size of the violation $f(x_i)$. The decision boundary of the perceptron defined by (3) is nonlinear. A perceptron is the simplest example of a neural network with only one node.

Artificial neural networks (ANNs) are computational networks inspired by the animal brains that can learn from known examples and generalize to unknown cases. An ANN is composed by a collection of connected nodes called artificial neurons (AN). After receiving an input signal, ANs process and pass it to the connected neurons. The connections are called edges. A weight modulates the strength of the signal that is passed from input to output through the edges. These weights represent the neural synapses. Formally, each neuron has M inputs $x_1, \ldots, x_j, \ldots, x_M$, and each input is assigned a weight w_j: if $w_j > 0$ the input is excitatory, otherwise it is inhibitory. The weighted sum of the inputs is passed to the output y through the *activation*/transfer function F:

$$y = F\left(\sum_{j=1}^{M} x_j w_j\right) . \tag{4}$$

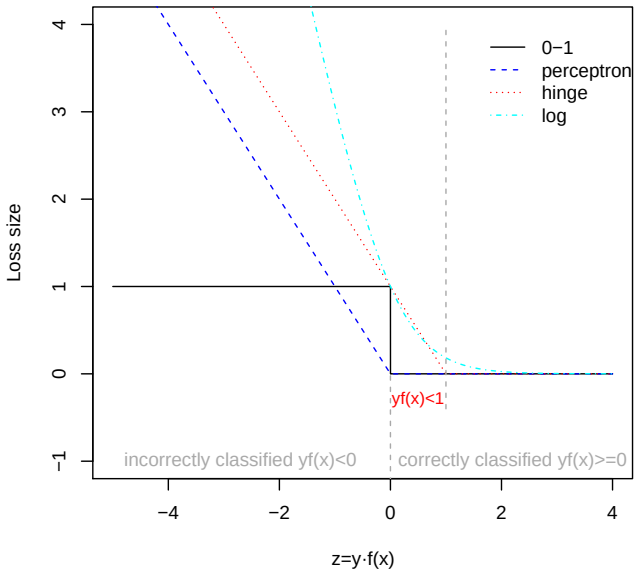

Figure 1: Comparison of different loss functions. Losses are designed to penalize negative margins (or incorrectly classified observations) more than positive margins.

A neuron may have a threshold θ (called Bias) such that only if the aggregate signal crosses the threshold the signal is passed to the output:

$$y = F\left(\sum_{j=1}^{M} x_j w_j - \theta\right). \tag{5}$$

Figure 2 shows a schematic view of an AN.

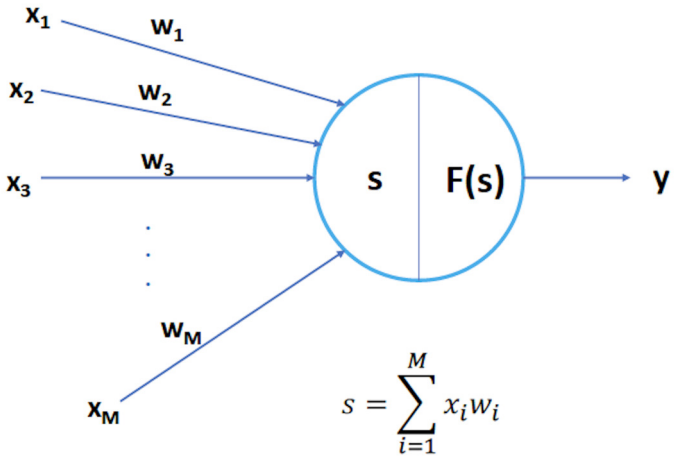

Figure 2: Schematic view of an artificial neuron: x_1,..., x_M are the input nodes, y is the output node, F is the activation function.

The most used activation functions are the Sigmoid (or logistic function seen before), the Hyperbolic Tangent and the ReLu (rectified linear unit). Figure 3 shows a comparison of different activation functions. Typically, neurons are

aggregated into layers: input, hidden, and output layer. The number of neurons and layers are some of the parameters that need to be set to define a network architecture.

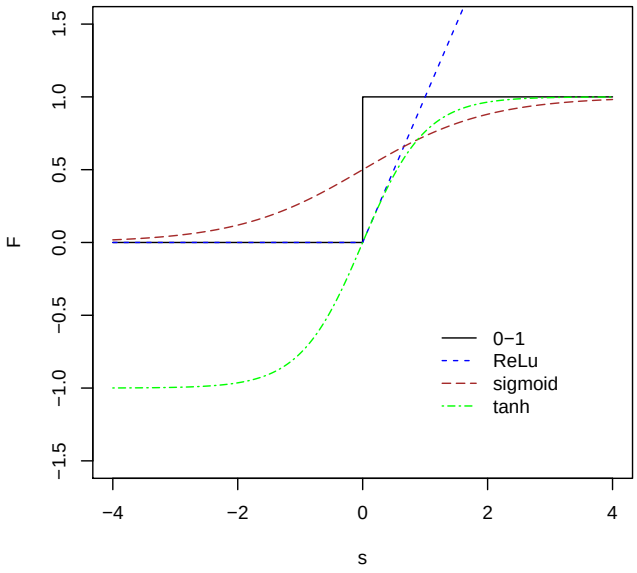

Figure 3: Comparison of different activation functions.

Given the large number of possible architectures, ANNs are often not easy to tune. The most commonly used ANNs are feedforward networks. In these kinds of artificial networks: (i) no computation takes place in the input layer; (ii) the signal always travels forward from the input to the output; (iii) connections between the nodes do not form a cycle. Multilayer perceptron networks (MLPs) are an example of feedforward networks, and are composed of multiple layers of fully connected nodes (each node in one layer is connected to all nodes in the next layer). Internal nodes of an MPL are perceptrons with threshold activation: the weights $(w_j)_{j=1}^M$ in (5) correspond to the coefficients $(\theta)_{j=1}^M$ in (3) and the Bias θ in (5) to $-\theta_0$; the ReLu activation function is analogous to the perceptron loss function in (3).

When the activation function is nonlinear, a two-layer ANN can be proven to be a universal function approximator [3] (Universal Approximation Theorem). For this reason, ANNs with nonlinear activation functions can solve complex problems using only a small number of nodes; multilayer ANNs with identity activation function instead are equivalent to a single layer.

1 Use case: age and gender prediction by human RNAseq data

1.1 Gender classification

In Figure 5 we show the classification accuracy of the three models through a 5-fold cross validation procedure. Figure 6 reports the contingency tables for the three implemented models: Random Forest, Multilayer Perceptron and linear model.

1.2 Age regression

Performances obtained by means of the applied algorithms are shown in Figures 7 and 8.

References

[1] Hastie, T., Tibshirani, R., Friedman, J. (2001), The Elements of Statistical Learning , Springer New York Inc. , New York, NY, USA

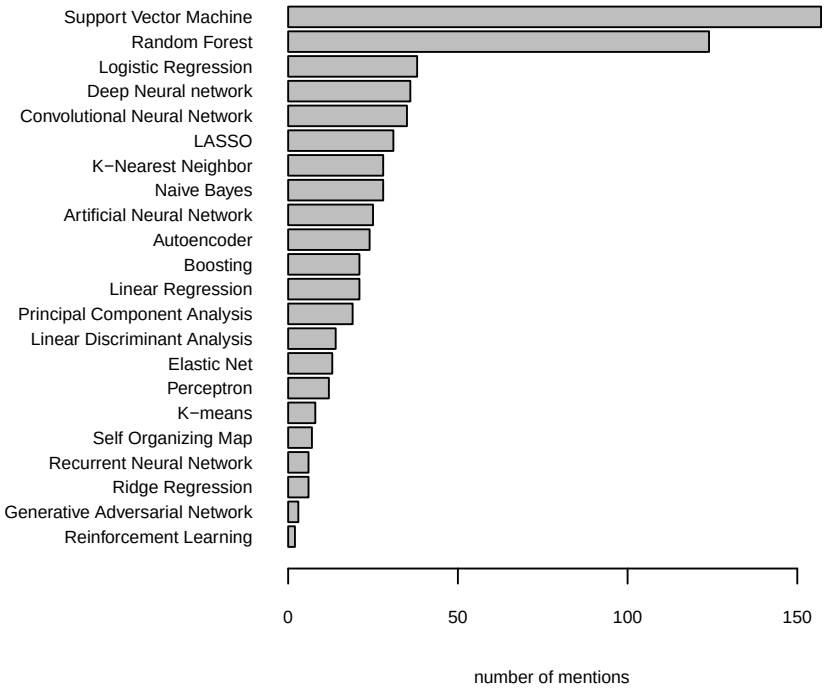

Figure 4: Number of times each algorithm has been mentioned in the title or the abstract of papers in the PubMed database, in the last ten years.

[2] Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288

[3] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. 1993. Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function, Neural Networks, 6(6), 861-867.

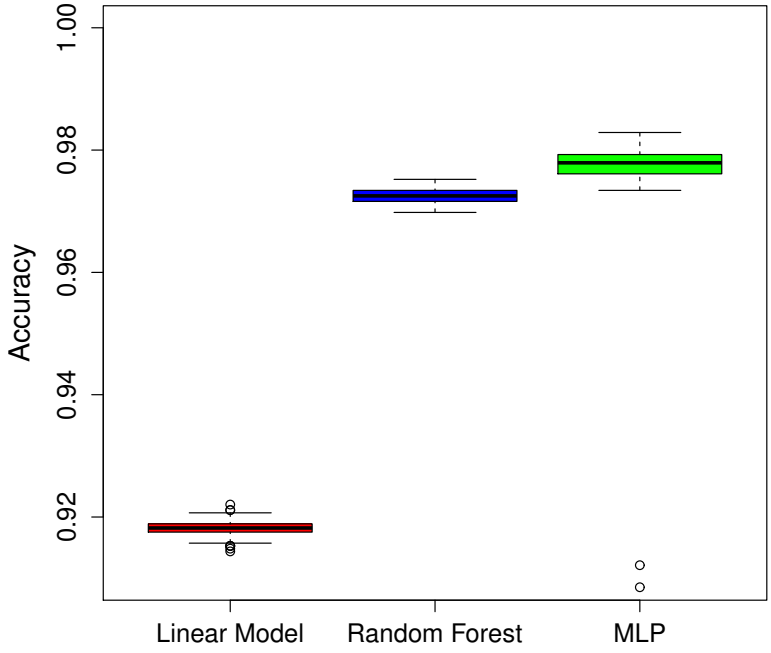

Figure 5: Box plots of classification accuracy of the implemented learning models. Distributions are obtained by means of a 5-fold cross validation procedure repeated 100 times.

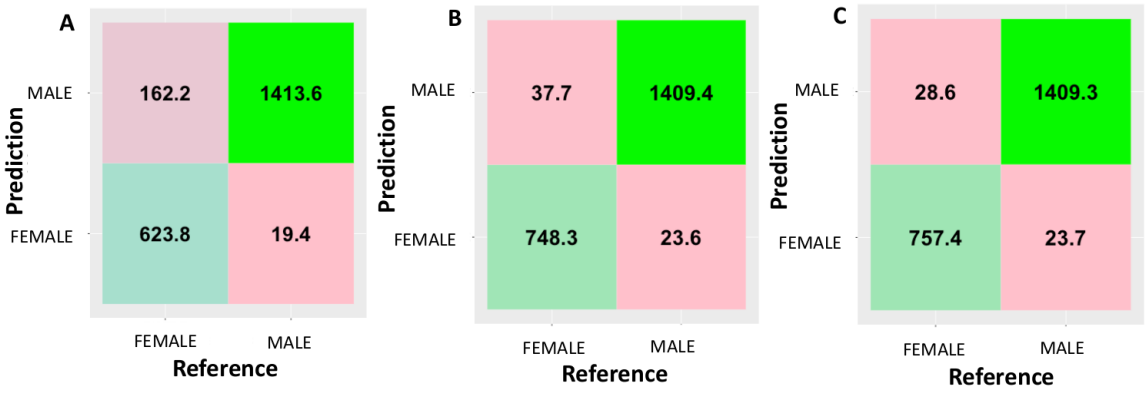

Figure 6: Contingency tables obtained for linear model (LM) (panel A), Random Forest (RF) (panel B) and MLP (panel C) averaged over 100 rounds of 5-fold cross validation.

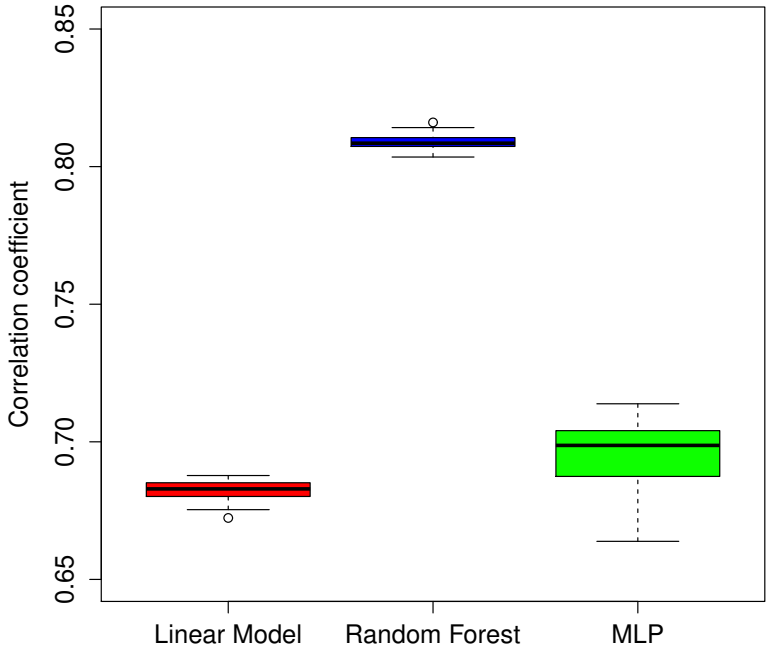

Figure 7: Box plots of the Pearson's correlation between the actual values and the predicted values by means of the three implemented learning algorithms.

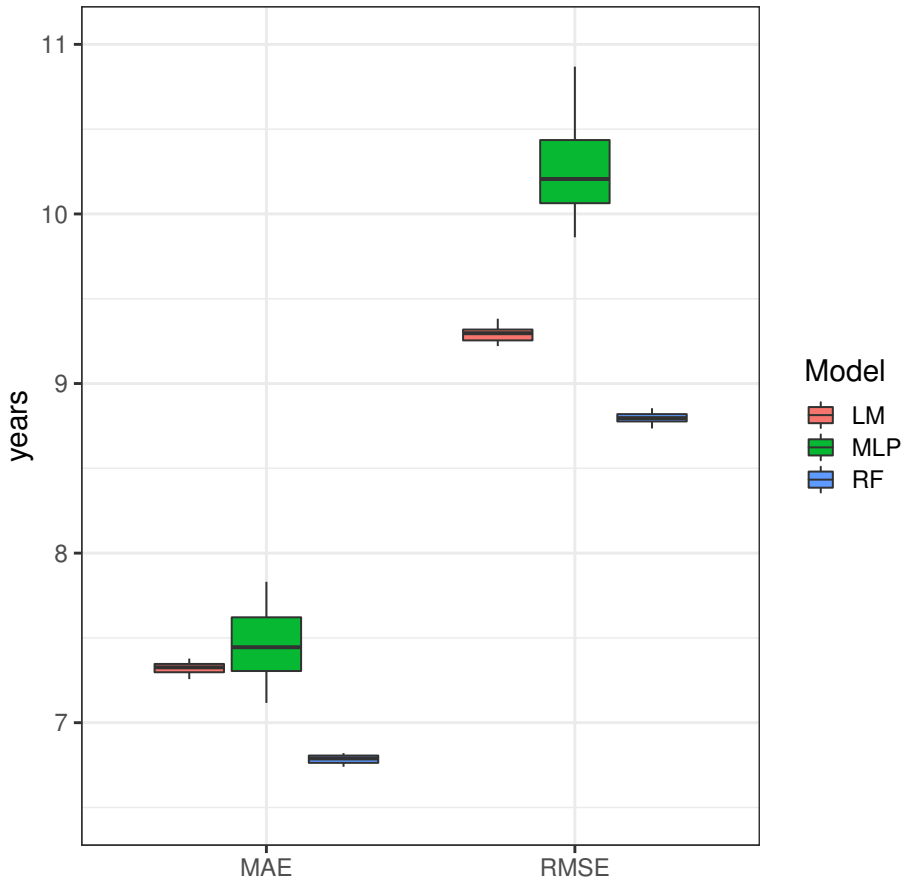

Figure 8: Box plots of regression errors (RMSE, MAE) for the implemented learning models. Distributions are obtained by means of a 5-fold cross validation procedure repeated 100 times.