Journal of Data Science 19 (3), 409–428
July 2021

DOI: 10.6339/21-JDS1014
Statistical Data Science

Random Machines: A Bagged-Weighted Support Vector Model with Free Kernel Choice

Anderson Ara[1,*], Mateus Maia[1], Francisco Louzada[2], and Samuel Macêdo[3]

[1]*Department of Statistics, Federal University of Bahia, Salvador-BA, Brazil*
[2]*Institute of Mathematical and Computer Sciences, University of São Paulo, São Carlos-SP, Brazil*
[3]*Department of Natural Sciences and Mathemathics, Federal Inst. of Pernambuco, Recife-PE, Brazil*

Abstract

Improvement of statistical learning models to increase efficiency in solving classification or regression problems is a goal pursued by the scientific community. Particularly, the support vector machine model has become one of the most successful algorithms for this task. Despite the strong predictive capacity from the support vector approach, its performance relies on the selection of hyperparameters of the model, such as the kernel function that will be used. The traditional procedures to decide which kernel function will be used are computationally expensive, in general, becoming infeasible for certain datasets. In this paper, we proposed a novel framework to deal with the kernel function selection called Random Machines. The results improved accuracy and reduced computational time, evaluated over simulation scenarios, and real-data benchmarking.

Keywords *bagging; kernel functions; support vector machines*

1 Introduction

Currently, the application and development of statistical learning methods is an important research topic in the academic and industry community. The supervised machine learning techniques have been applied at numerous classification tasks ranging from cancer diagnostics and prediction (Sato et al., 2019), speech recognition (Mokgonyane et al., 2019), text classification (Burdisso et al., 2019; Kim et al., 2005) and financial fraud detection (Dighe et al., 2018). The variety of methods that have been used in the field is huge, and the Support Vector Machine (SVM) plays an important role among them. The SVM (Cortes and Vapnik, 1995) is the youngest well established and successful in traditional learning methods. Smola et al. (2000) presented some great proprieties of this learning algorithm, including good generalization capacity, high efficiency in prediction tasks and, the convexity of the objective function which guarantees a global minimum. Some works present the superiority of the SVM when compared with other supervised learning benchmarking techniques, highlighting favorable accuracy results (Cueto-López et al., 2019; Thanh Noi and Kappas, 2018; Shah and Issac, 2018).

Concurrently, the ensemble methods have been arising as a tool to improve the accuracy in classification models. The combination of singular models can enhance predictive power and increase its generalization capacity (Van Wezel and Potharst, 2007). There are two main classes of ensemble algorithms: bagging (Breiman, 1996) that uses independent bootstrap samples to create multiple models and built a final classifier combining them and boosting algorithms

*Corresponding author. Email: alsouzara@gmail.com.

Received December 9, 2020; Accepted April 28, 2021

(Freund et al., 1999) that built sequential models to assign different weights relying on their performance.

The literature already proposed bagging methods jointly with the support vector machine algorithm (Kim et al., 2002) as a methodology of increasing its accuracy (ACC). Wang et al. (2009) realized an empirical study of Bagged SVM and showed that the technique performs as well or better than other methods with a relatively higher generality. Moreover, different applications of bagged SVM are reported, e.g. breast cancer prediction (Huang et al., 2017; Wang et al., 2018), credit score modelling (Zhou et al., 2010), gene detection (Tong et al., 2013), spatial prediction of landslides (Pham et al., 2018), bacterial transcription start sites prediction (Gordon et al., 2005), text speech recognition (Lei et al., 2006) and membership authentication (Pang et al., 2003).

Despite the diverse number of works that present the bagging based on support vector machine classifiers, none of them presents an optimal framework to choose which kernel function will be used in the ensemble classifier. The choice of the kernel function, as their hyperparameters, has a crucial impact on the accuracy of the technique (Jebara, 2004). Generally, this selection is supported by a grid search that runs all functions and their parameter combinations to select which one has the lowest generalization error rate. Random Search (Bergstra and Bengio, 2012) is another approach to tuning the hyperparameters, where the parameter configurations are randomly chosen until a budget B is exhausted. Besides these, Tree-Structured Parzen Estimator (Bergstra et al., 2011), and Simulated Annealing (Kirkpatrick et al., 1983) are optimization structures used in tuning workflow too. However, all of them can be computationally expensive and slow, making them infeasible to use.

The kernel methods, *e.g.:* Kernelized Support Vector Machines, can be considered as non-parametric machine learning models which are useful to capture the non-linear behavior, beyond their strong theoretical properties. However, they have some problems to be applied to large-scale data sets since their time and memory demand, that is at least n^2, where n is the number of observations. Recent works solve the problem of computational limitations through the use of Nyström method (Williams and Seeger, 2001; Smola and Schölkopf, 2000) or random features. Both of them have their specific versions for support vector machine (Sun et al., 2018; Li et al., 2016), and represents a solid advance in those techniques.

This work introduces a novel method that presents a solution for the choice of kernel function to be used in the bagged supported vector machine, using an alternative to the open problem of hyperparameters' selection which has an adequate computational time and robust accuracy power, hereafter, the Random Machines (RM). The method received this name because it uses random kernel choice for each model that composes the bagged support vector machine method, besides proposing weights to these classifiers, increasing the accuracy and lowering the correlation of the final model. The result was validated over simulation studies and on 27 different benchmarking datasets.

The following paper is organized on the ensuing outline. Section 2 presents a theoretical description about the support vector machine method, proposed by (Cortes and Vapnik, 1995), the challenges on the selection of hyperparameters and some traditional kernel functions; Section 3 presents a general description of the bagging algorithm and bagged SVM; Section 4 presents how the proposed random machines approach works in detail, followed by the simulations studies in Section 5, and applications in real data in Section 6. Section 7 shows an empirical justification of why the method works that proves the consistency of the technique. Finally, in Section 8, final considerations regarding the improvements and limitations that can be explored in this novel approach.

2 Support Vector Machines

Support vector machines (Cortes and Vapnik, 1995) have been introduced for solving classification problems. The overall idea of the technique is to calculate a hyperplane that separates observations between two classes, maximizing the distance between the support vectors.

Supposing a training sample given by $\{\mathbf{x_i}, y_i\}$ with $i = 1, \ldots, n$ observations, where $\mathbf{x_i}$ the independent variable and y_i the dependent one, given by $y_i = \{-1, 1\}$, where $y_i = 1$ represents that the observation belongs to a positive class, while $y_i = -1$ is the negative one. If the training sample is linear separable, the hyperplane that separate these two classes is represented by

$$\mathbf{w} \cdot \mathbf{x_i} + b = 0$$

where $\mathbf{w}$ is the vector of weighted parameters, $\mathbf{x}$ is the vector of input variables and b the offset parameter.

In order to find such optimal hyperplane the estimation of $\mathbf{w}$ and b is taken in order to maximize the distance between the support vectors (Boser et al., 1992; Cortes and Vapnik, 1995), following the restrictions of $y_i(w \cdot \mathbf{x_i} + b) \geqslant 1$, if y_i belongs to the positive class, therefore, $y_i = 1$, and $y_i(w \cdot \mathbf{x_i} + b) \geqslant -1$, otherwise $y_i = -1$. These equations are expressed by

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 \geqslant 0 \tag{1}$$

where dot product represents the projection of each instance in the hyperplane.

The distance is given by $\frac{2}{||\mathbf{w}||}$, to maximize it is necessary to solve a convex problem given by

$$\min \frac{1}{2}||\mathbf{w}||^2 \tag{2}$$

following the constraints given by the Equation (1). The cost function which will be minimized is defined by the Lagrangian multipliers, in Equation (3).

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1] \tag{3}$$

where $\boldsymbol{\alpha}$ is vector of the α_i Lagrangian Multipliers.

There are cases where the training data cannot be separated without error, as pointed out by Cortes and Vapnik (1995). In such a case, it is needed to construct a soft margin separator by inputting slack variables (ε_i). Therefore, a transformation in the Equation (2) was needed (Cortes and Vapnik, 1995), and then, it becomes

$$\min \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \varepsilon_i \tag{4}$$

where $C > 0$ is a regularization parameter. The constraints become $y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - (1 - \varepsilon_i) \geqslant 0$ and $\varepsilon_i \geqslant 0$ for $i = 1, \ldots, n$. And the cost function, which will be minimized, becomes

$$L(\mathbf{w}, b, \boldsymbol{\alpha}, \mathbf{r}) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \varepsilon_i - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 + \varepsilon_i] - \sum_{i=1}^{n} r_i \varepsilon_i \tag{5}$$

The solution considering the Lagrangian dual optimization for the soft margin problem (Fletcher, 1987) is given by

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \left(\sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_i \mathbf{x_i} \cdot \mathbf{x_i} \right) \\ \text{subject to} \quad & \sum_{i}^{n} \alpha_i y_i = 0, \\ & C - \alpha_i - r_i = 0, \\ & 0 \leqslant \alpha_i \leqslant C, \\ & r_i \geqslant 0 \end{aligned} \tag{6}$$

This approach of SVM works well to linearly classification groups and problems. In the presence of non-linearity, it may be used the kernel trick, based on Mercer's Theorem. This trick can be as the procedure of instead considering the input space, it's considered higher feature spaces, where the observations could be linearly separable through the following function $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$ that replaces the inner product in Equation (6).

The functions $K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$ are usually defined as the semidefinite kernel functions (Courant and Hilbert, 1953). Several types of kernel functions are employed in different classification tasks. The choice of distinct kernel functions provides different nonlinear mappings and, the performance of the resulting SVM often depends on the appropriate choice of the kernel (Jebara, 2004). Some works that compare the efficiency for each kernel function used in classification models (Hussain et al., 2011; Min and Lee, 2005), demonstrating that select the kernel function is an important aspect of obtaining the best model. There are some kernel functions in the general framework for SVM, which were used in this paper, that is considered the most common. They are presented in Table 1, where $\gamma \in \mathbb{R}^+$ and $d \in \mathbb{N}$. Despite several other kernel functions that also could be considered, as the hyperbolic tangent kernel and sigmoid kernel, in this paper only these four were used to keep it as a baseline for the proposed model.

The hyperparameter d, can be defined as the degree of the polynomial kernel, and generally, has a standard value of 2 and defined the complexity of the feature space transformation. On the other hand, the γ values can scale the distance measures in Gaussian and Laplacian kernels, or the dot products in linear and polynomial functions.

Nevertheless, find out which is the best kernel by grid search, or other exhaustive methods, can be an expensive and appalling computational problem (Chapelle and Vapnik, 2000). To deal with this issue many works have tried to develop a methodology that can improve the selection of the best kernel function (Jebara, 2004; Ayat et al., 2005; Wu et al., 2009; Friedrichs and Igel, 2005; Cherkassky and Ma, 2004). In this work we propose a novel approach that makes it

Table 1: Kernel functions.

Kernel	**K(x,y)**	**Parameters**
Linear Kernel	$\gamma(x \cdot y)$	γ
Polynomial Kernel	$(\gamma(x \cdot y) + \omega)^d$	γ, d
Gaussian Kernel	$e^{-\gamma\|\|x-y\|\|^2}$	γ
Laplacian Kernel	$e^{-\gamma\|\|x-y\|\|}$	γ

unnecessary to perform a grid search, or another tuning algorithm, to choose a single specific kernel function when using the trick kernel on bagging procedure.

3 Bagging Procedure

Bagging is an acronym of Bootstrapping Aggregation and was firstly proposed by Breiman (1996). Bagging is an ensemble method that can be used for different prediction tasks. In general, the Bootstrapping Aggregating generates datasets by random sampling with replacement from the training set with the same size n, also known as bootstrap samples. Then, each model $h_j(x_i)$ is trained independently for each bootstrapping sample b_j, $\forall j \in \{1, \ldots, B\}$, where B is the number of bootstrap samples. The final bagging model, for binary classification tasks, is given by the following equation,

$$H(\mathbf{x}) = sign\left(\sum_{i=1}^{B} h_i(\mathbf{x})\right) \tag{7}$$

where $h_i(\mathbf{x})$ is the model generated to each bootstrap sample from $i = 1, , \ldots, B$.

Another critical feature of Bagging classifier is the **Out of Bag** samples (Breiman, 1996). For each bootstrap sample, around 37% of observations are not included. Therefore, those observations can be used as a test sample since they were not used to train the bootstrap models and received the name of **Out of Bag** set.

3.1 Bagging SVM

In the bagging classifier, the function $h_i(\mathbf{x})$ from Equation (7) can be any model. One possibility is to use the SVM as the base classifier (Kim et al., 2002) to improve it is accuracy. The applications of the bagged SVM for predictive tasks are wide, and empirical studies (Wang et al., 2009) demonstrated that the bagged version of the support vector machine algorithm increased the accuracy and it is generalization capacity. Moreover, some of them already presented some modifications using the SVM in bagging context as (Lin and Li, 2008), and others implemented some libraries as *EnsembleSVM*, that make it short to use simple ensemble methods with SVM (Claesen et al., 2014).

Despite the numerous works using bagged SVM, none of them present a general framework to deal with the choice of the best kernel function that will be used in the base-learners, choosing it by trial evaluation or by a grid search. As this procedure is computationally expensive (Chapelle and Vapnik, 2000), this paper proposed a novel bagging approach that can overcome the difficulty to choose the best kernel function, besides showing an improvement in the accuracy of classification models by combining several different SVM models by varying the kernel functions: the random machines (RM), exposed in next section.

4 Random Machines

Considering a training set $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ with $\mathbf{x_i} \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, $\forall i = 1, \ldots, n$. The kernel bagging method initializes by training single models $h_r(\mathbf{x})$, where $r = 1, \ldots, R$, where R is the total number of different kernel functions that could be used in support vector machine models. For example, using the functions of Table 1, we could set $R = 4$, and a define h_1 as SVM with *Linear kernel*, h_2 as SVM with *Polinomial kernel*, h_3 as SVM with *Gaussian kernel* and h_4 as SVM with *Laplacian kernel*. The value of R, just depends on the number of kernel functions that will be selected in the model, also can be a hyperparameter to be settled in the method.

Each model is validated for a validation set $\{(x_k, y_k)\}_{k=1}^{L}$, and the accuracy (ACC_r) is calculated for each model, $\forall r = 1, \ldots, R$, in which R means the numbers of kernel functions that will be used. Therefore, the probabilities, λ_r, are given by the Equation (8) for each kernel function

$$\lambda_r = \frac{\log\left(\frac{ACC_r}{1-ACC_r}\right)}{\sum_{i=1}^{R}\log\left(\frac{ACC_i}{1-ACC_i}\right)} \tag{8}$$

with $\forall r = 1, \ldots, R$. Despite this probability can be sensible to the initialization of the kernels, the main function of the parameter λ_r is to avoid that kernel with very poor performance appears more during ensemble process.

Afterwards, B bootstrap samples are sampled from the training set. A support vector machine model g_b is trained for each bootstrap sample, $b = i, \ldots, B$ and the kernel function that will be used for g_k will be determined by a random choice with probability $\lambda_r, \forall r = 1, \ldots, R$. The probabilities λ_r are higher if determined kernel function used in $h_r(\mathbf{x})$ predicted correctly observations from test set. Consequently, the kernel functions with higher accuracy will appear often when the random kernel selection for each bootstrap model is made. If any kernel function applied in $h_r(\mathbf{x})$ does not do better than a random choice, then ACC_r is closer to 0.5 and the probability of select that kernel function is next to zero. The cases where $ACC_r < 0.5$, the predictions of the models $h_r(\boldsymbol{x})$ are swapped to guarantee that the Equation (8) will produce valid probabilities. Subsequently, a weight w_i is assigned to each bootstrap model calculated for g_i $\forall i = 1, \ldots, B$. The weight is given by the Equation (9).

$$w_i = \frac{1}{(1-\Omega_i)^2}, \quad i = 1, \ldots, B, \tag{9}$$

where Ω_i is the accuracy of model's prediction g_i calculated on Out of Bag Sample ($OOBG_i$) obtained from i bootstrap sample $\forall i = 1, \ldots, B$ as test sample. This step is important because it yields a robust validation for each kernel predictor since is validated over several bootstrap samples.

The final classification is held in Equation (10).

$$G(\mathbf{x_i}) = sgn\left(\sum_{j}^{B} w_j g_j(\mathbf{x_i})\right), \quad i = 1, \ldots, N \tag{10}$$

All the modeling process is summed up in the pseudo-code exposed in Algorithm 1. The entire random machines are schematically presented in Figure 1.

Algorithm 1 RM.

Input: Training Data, Test Data, B, Kernel Functions
for each Kernel Function$_r$ **do**
  Calculate the model h_r
  Calculate the accuracy α_r
Calculate the probabilities λ_r
Generate B bootstrap samples
for b in B **do**
  Model $g_b(\mathbf{x_i})$ by sampling a kernel function with probability λ_r
  Assign a weight Ω_b using $OOBG_b$ samples.
Calculate $G(\mathbf{x})$

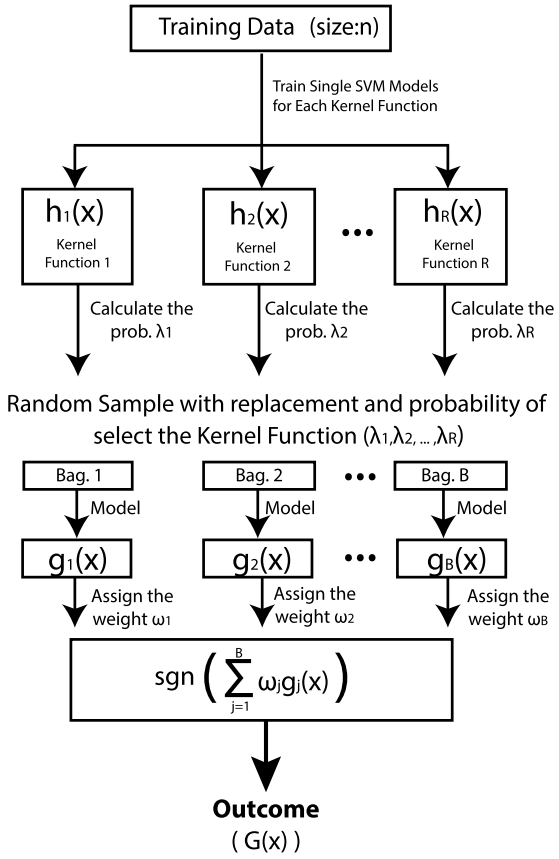

Figure 1: Workflow followed by the random machines.

5 Artificial Data Application

In this section simulation studies were conducted in order to evaluate the efficiency of the random machines applied to binary classification tasks. The other methods compared were: *Linear, Polynomial, Gaussian* and *Laplacian* SVM, beyond their bagged versions, respectively. A good variety between the simulated datasets is observed through three different scenarios of data generation. The dimensionality (p) ranges from $\{2, 10, 50\}$, the number of observations (n) ranges from $\{100, 1000, 10000\}$, and the proportion's ratio between the two classes assume three values $\{0.01, 0.1, 0.5\}$. The variation of p, n, and $ratio$ verify if the method is consistent even in different situations.

The generation from the **Dataset 1** and **Dataset 2** considers continuous explanatory variables (Breiman et al., 1998), were the observations belonging to each class follow a multivariate distribution with their respective mean vector and covariate matrix. The **Dataset 1** follows the configuration that instances from Class A are sampled from a normal multivariate which has mean vector $\boldsymbol{\mu}_A = \vec{\mathbf{0}}_p$ and covariate matrix $\boldsymbol{\Sigma}_A = 4\boldsymbol{I}_p$ and the Class B instances are sampled from a normal multivariate that has mean vector $\boldsymbol{\mu}_B = \vec{\mathbf{4}}_p$ and covariate matrix $\boldsymbol{\Sigma}_A = \boldsymbol{I}_p$. The **Dataset 2**, has the same distribution with the exception that the mean vector for Class B is given by $\boldsymbol{\mu}_B = \vec{\mathbf{2}}_p$. The difference between those two datasets relies on the difficulty to create the hyperplane that separates the two classes since the **Dataset 1** has observations from each

group that are further away when compared with **Dataset 2**, making the first situation a simpler classification case.

Dataset 3 considers a classification problem where a circle uniformly distributed is generated inside the middle of a p-dimensional cube. This dataset is fundamentally more complex to realize a classification since it has a non-linear behavior.

In general, all considered datasets generate a non-linear classification task. However, data sets 1 and 2 are simpler to classify because they are based on a separation of two normal multivariate distributions with different vectors of means and different covariance matrices. **Dataset 2** has a greater distance between the mean vectors compared to Dataset 1. In contrast, **Dataset 3** deals with a 'circle in a square problem' which is a traditional example affected by of curse of dimensionality. In this case, the mean vectors stand in the origin, and the data points are uniformly distributed on the p-dimensional cube with corners +1, so the training samples that fall outside the unit circle are more difficult to classify than samples near the center of the feature space. In this situation, it is expected that all the predictor models have a better predictive performance when p is small.

The performance of each model was estimated using values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) obtained from a confusion matrix, and therefore calculating the following metrics:

Accuracy (ACC): it measures the ratio of correctly classified observations to total observations from the sample. It is calculated by

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

Matthew's Correlation Coefficient (MCC): introduced by Matthews (1975), is usually used to evaluate the predictions made from the model (Baldi et al., 2000) and it is defined by,

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{12}$$

It can be considered an accurate coefficient, since it penalizes the FP and FN predictions, besides being considered a better evaluator if the classes are of very different sizes (Boughorbel et al., 2017). Its range varies from $[-1, 1]$, in which 1 represents a perfect prediction, 0 no better than a random choice, and -1 a complete reverse classification.

In order to compare directly with the accuracy, as the original scales between the metrics are different, we proposed a modification to MCC. The transformation is given by $uMCC = \frac{MCC+1}{2}$ and results in a new evaluation metric: Uniform MCC (uMCC). The uMCC lies in the interval $[0, 1]$, where 1 represents a perfect prediction, 0 no better than a random prediction.

The validation technique used was the repeated holdout with 30 repetitions with a split ratio of training-test of 70%–30%. The results are summarized in Tables 2 and 3 where all possible combination of kernel functions and data sets setups is presented. The empty cells are related to the smaller sample size $n = 100$, which provides in average only one observation to the target class. The setup of hyperparameters was $\gamma = 1$, $B = 100$ and $d = 2$. Analyzing the results from Table 2 and 3, it is possible to see that in most cases, the RM surpasses or equals the other methods in all setups. For instance, in Dataset 3, where the nonlinear behavior is an essential characteristic from the data, we can observe the RM overcomes the other classifiers as the dimensionality of the data increases. This result may give indications of the good efficiency of the proposed algorithm to deal with non-linearity when compared with the traditional approaches.

Table 2: Summary of the simulation's results for the Datasets 1 and 2.

Setup		$\mathbf{SVM}_{lin}$		$\mathbf{SVM}_{poly}$		$\mathbf{SVM}_{gaus}$		$\mathbf{SVM}_{lap}$		$\mathbf{BSVM}_{lin}$		$\mathbf{BSVM}_{poly}$		$\mathbf{BSVM}_{gau}$		$\mathbf{BSVM}_{lap}$		**RM**	
p	**Ratio**	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC
Dataset 1 ($n = 100$)																			
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
2	.10	1.00	0.68	6.00	0.68	0.69	0.66	0.69	0.67	0.60	0.68	0.60	0.68	0.68	0.62	0.69	0.65	0.70	0.69
	.50	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
10	.10	1.00	0.97	1.00	0.95	0.91	0.50	0.91	0.50	1.00	0.97	0.99	0.95	0.91	0.50	0.91	0.50	0.99	0.95
	.50	1.00	1.00	1.00	1.00	0.95	0.95	1.00	1.00	1.00	1.00	1.00	1.00	0.95	0.95	1.00	1.00	1.00	1.00
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
50	.10	1.00	0.97	1.00	0.97	0.91	0.50	0.91	0.50	1.00	0.97	1.00	0.97	0.91	0.50	0.91	0.50	1.00	0.97
	.50	1.00	1.00	1.00	1.00	0.49	0.56	1.00	1.00	1.00	1.00	1.00	1.00	0.54	0.54	1.00	1.00	1.00	1.00
Dataset 1 ($n = 1000$)																			
	.01	1.00	0.99	1.00	0.99	1.00	0.98	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00
2	.10	0.99	0.96	0.99	0.97	0.99	0.97	0.99	0.97	0.99	0.96	0.99	0.97	0.99	0.97	0.99	0.97	0.99	0.97
	.50	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
10	.10	1.00	1.00	1.00	1.00	0.90	0.50	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.50	1.00	1.00	1.00	1.00
	.50	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.98	1.00	1.00	1.00	1.00
	.01	1.00	1.00	1.00	1.00	0.99	0.98	1.00	0.99	1.00	0.98	1.00	1.00	1.00	0.99	0.99	0.98	1.00	1.00
50	.10	1.00	1.00	1.00	1.00	0.90	0.50	0.90	0.50	1.00	1.00	1.00	1.00	0.90	0.50	0.90	0.50	1.00	1.00
	.50	1.00	1.00	1.00	1.00	0.48	0.51	1.00	1.00	1.00	1.00	1.00	1.00	0.51	0.51	1.00	1.00	1.00	1.00
Dataset 1 ($n = 10000$)																			
	.01	1.00	1.00	1.00	1.00	0.98	0.51	0.99	0.50	1.00	0.99	1.00	0.99	0.99	0.50	0.99	0.50	1.00	1.00
2	.10	0.99	0.96	1.00	0.97	0.99	0.97	0.99	1.00	1.00	0.96	0.96	0.97	0.99	0.97	0.98	0.99	1.00	1.00
	.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	.01	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.98	1.00	0.99	1.00	1.00	0.99	0.50	0.99	0.50	1.00	1.00
10	.10	1.00	1.00	1.00	1.00	0.91	0.60	1.00	1.00	1.00	1.00	1.00	0.90	0.90	1.00	1.00	1.00	1.00	1.00
	.50	1.00	1.00	1.00	1.00	0.90	0.98	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.80	0.90	0.87	1.00	1.00
	.01	0.99	0.50	0.99	0.99	0.99	0.50	0.99	0.50	1.00	1.00	1.00	1.00	0.99	0.50	0.99	0.50	1.00	1.00
50	.10	1.00	1.00	1.00	1.00	0.90	0.60	0.90	0.60	1.00	1.00	1.00	1.00	0.52	0.60	0.57	0.55	1.00	1.00
	.50	1.00	1.00	1.00	1.00	0.61	0.58	1.00	1.00	1.00	1.00	1.00	1.00	0.62	0.61	1.00	1.00	1.00	1.00
Dataset 2 ($n = 100$)																			
2	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	.10	0.96	0.86	0.96	0.86	0.95	0.80	0.96	0.86	0.96	0.88	0.96	0.87	0.94	0.76	0.95	0.82	0.96	0.87
	.50	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.89	0.89
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
10	.10	0.99	0.93	0.95	0.78	0.91	0.50	0.91	0.50	0.98	0.90	0.96	0.79	0.91	0.50	0.91	0.50	0.96	0.83
	.50	0.93	0.94	0.94	0.94	0.85	0.87	0.98	0.98	0.94	0.94	0.94	0.95	0.81	0.85	0.98	0.98	0.96	0.96
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
50	.10	1.00	0.97	0.96	0.81	0.91	0.50	0.91	0.50	1.00	0.97	0.94	0.69	0.91	0.50	0.91	0.50	0.99	0.92
	.50	1.00	1.00	0.88	0.90	0.47	0.54	0.83	0.88	1.00	1.00	0.81	0.84	0.53	0.53	0.76	0.77	1.00	1.00
Dataset 2 ($n = 1000$)																			
	.01	0.99	0.98	1.00	0.98	1.00	0.62	1.00	0.56	0.99	0.87	0.99	0.51	0.98	0.52	0.99	0.51	1.00	0.82
2	.10	0.94	0.79	0.94	0.82	0.94	0.81	0.94	0.81	0.94	0.79	0.94	0.82	0.94	0.81	0.94	0.82	0.94	0.82
	.50	0.83	0.83	0.86	0.86	0.86	0.87	0.86	0.86	0.83	0.83	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
	.01	0.99	0.98	1.00	0.98	1.00	0.50	1.00	0.52	1.00	0.99	1.00	0.99	1.00	0.50	0.99	0.61	1.00	1.00
10	.10	1.00	0.99	0.99	0.96	0.90	0.50	1.00	0.99	1.00	0.99	0.99	0.97	0.90	0.50	0.98	0.93	1.00	0.99
	.50	0.98	0.98	0.99	0.99	0.92	0.93	1.00	1.00	0.98	0.98	0.99	0.99	0.91	0.92	1.00	1.00	1.00	1.00
	.01	1.00	0.98	1.00	0.99	1.00	0.52	1.00	0.54	1.00	0.98	1.00	0.99	0.99	0.52	0.98	0.54	1.00	1.00
50	.10	1.00	1.00	1.00	1.00	0.90	0.50	0.90	0.50	1.00	1.00	1.00	0.99	0.90	0.50	0.90	0.50	1.00	1.00
	.50	1.00	1.00	1.00	1.00	0.48	0.50	1.00	1.00	1.00	1.00	1.00	1.00	0.52	0.52	1.00	1.00	1.00	1.00
Dataset 2 ($n = 10000$)																			
	.01	1.00	1.00	1.00	1.00	0.99	0.50	0.99	0.50	1.00	0.99	1.00	1.00	0.99	0.50	0.99	0.50	1.00	1.00
2	.10	0.92	0.91	0.99	0.95	0.93	0.95	0.99	0.92	0.92	0.92	0.91	0.95	0.91	0.87	0.91	0.87	0.92	0.93
	.50	0.97	0.97	0.93	0.92	0.94	0.95	0.95	0.93	0.92	0.91	0.93	0.92	0.91	0.94	0.95	0.96	0.99	0.98
	.01	1.00	0.98	1.00	0.99	1.00	0.92	1.00	0.91	1.00	0.99	1.00	1.00	0.99	0.50	0.99	0.50	1.00	1.00
10	.10	0.95	0.89	0.99	0.90	0.90	0.67	0.92	0.61	0.92	0.90	0.95	0.64	0.90	0.81	0.92	0.78	0.99	0.95
	.50	0.95	0.94	0.95	0.93	0.95	0.91	0.95	0.90	0.95	0.92	0.95	0.91	0.99	0.98	0.95	0.95	1.00	1.00
	.01	0.99	0.50	1.00	1.00	0.99	0.50	0.99	0.50	0.99	0.50	1.00	1.00	0.99	0.50	0.99	0.50	1.00	1.00
50	.10	0.75	0.94	0.95	0.93	0.95	0.91	0.95	0.90	0.95	0.92	0.95	0.91	0.99	0.98	0.95	0.95	0.99	0.99
	.50	0.95	0.96	0.97	0.95	0.97	0.92	0.95	0.90	0.95	0.92	0.95	0.91	0.66	0.68	0.65	0.65	1.00	1.00

Table 3: Summary of the simulation's results for the Dataset 3.

Setup		$\mathbf{SVM}_{lin}$		$\mathbf{SVM}_{poly}$		$\mathbf{SVM}_{gaus}$		$\mathbf{SVM}_{lap}$		$\mathbf{BSVM}_{lin}$		$\mathbf{BSVM}_{poly}$		$\mathbf{BSVM}_{gau}$		$\mathbf{BSVM}_{lap}$		**RM**	
p	**Ratio**	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC	ACC	uMCC
Dataset 3 ($n = 100$)																			
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
2	.10	0.62	0.61	0.86	0.86	0.66	0.70	0.64	0.72	0.59	0.63	0.78	0.80	0.64	0.69	0.63	0.70	0.78	0.81
	.50	0.56	0.58	0.97	0.97	0.92	0.93	0.92	0.93	0.58	0.60	0.96	0.96	0.92	0.92	0.92	0.93	0.95	0.95
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
10	.10	0.76	0.78	0.58	0.62	0.46	0.57	0.47	0.57	0.71	0.74	0.53	0.57	0.57	0.58	0.58	0.59	0.69	0.71
	.50	0.49	0.50	0.68	0.69	0.46	0.54	0.64	0.70	0.52	0.53	0.68	0.68	0.52	0.53	0.61	0.68	0.68	0.70
	.01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
50	.10	0.52	0.55	0.46	0.50	0.41	0.53	0.53	0.65	0.53	0.58	0.46	0.50	0.47	0.51	0.58	0.63	0.57	0.62
	.50	0.58	0.59	0.58	0.61	0.45	0.52	0.54	0.59	0.63	0.63	0.55	0.58	0.52	0.51	0.57	0.57	0.69	0.70
Dataset 3 ($n = 1000$)																			
	.01	0.99	0.50	0.99	0.50	0.99	0.50	0.99	0.50	1.00	0.50	1.00	1.00	1.00	0.75	1.00	0.50	1.00	1.00
2	.10	0.48	0.51	0.99	0.99	0.94	0.94	0.95	0.95	0.52	0.53	0.99	0.99	0.95	0.95	0.95	0.95	0.98	0.98
	.50	0.49	0.51	0.99	0.99	0.98	0.98	0.98	0.98	0.52	0.53	0.99	0.99	0.98	0.98	0.98	0.98	0.99	0.99
	.01	0.99	0.50	1.00	0.99	1.00	0.50	1.00	0.50	1.00	0.50	1.00	1.00	1.00	0.51	1.00	0.51	1.00	1.00
10	.10	0.54	0.55	0.78	0.78	0.48	0.52	0.79	0.80	0.54	0.54	0.77	0.77	0.50	0.53	0.78	0.79	0.81	0.82
	.50	0.50	0.51	0.95	0.95	0.76	0.78	0.92	0.92	0.51	0.51	0.95	0.95	0.75	0.77	0.92	0.92	0.96	0.96
	.01	0.99	0.55	0.99	0.50	1.00	0.51	1.00	0.50	1.00	0.50	1.00	0.50	0.99	0.50	0.99	0.50	1.00	0.50
50	.10	0.46	0.47	0.55	0.57	0.46	0.50	0.49	0.53	0.47	0.47	0.54	0.58	0.50	0.50	0.52	0.53	0.59	0.61
	.50	0.49	0.49	0.72	0.72	0.49	0.50	0.67	0.70	0.48	0.49	0.70	0.72	0.50	0.50	0.61	0.63	0.83	0.84
Dataset 3 ($n = 10000$)																			
	.01	0.99	0.50	0.99	0.98	1.00	0.51	1.00	0.50	1.00	0.99	0.99	0.93	0.99	0.93	0.99	0.95	1.00	0.97
2	.10	0.54	0.52	0.62	0.61	0.92	0.94	0.92	0.91	0.55	0.51	0.81	0.92	0.94	0.93	0.92	0.91	0.95	0.94
	.50	0.55	0.53	0.64	0.63	0.95	0.94	0.96	0.92	0.56	0.52	0.87	0.89	0.91	0.89	0.92	0.89	0.99	0.98
	.01	0.99	0.50	0.99	0.89	0.99	0.50	0.99	0.50	0.99	0.50	1.00	0.89	0.99	0.50	0.99	0.50	1.00	0.90
10	.10	0.52	0.51	0.62	0.61	0.92	0.90	0.84	0.80	0.54	0.51	0.87	0.89	0.83	0.80	0.82	0.89	0.91	0.90
	.50	0.54	0.52	0.62	0.61	0.92	0.94	0.92	0.91	0.55	0.52	0.81	0.92	0.92	0.90	0.95	0.91	0.95	0.94
	.01	0.90	0.50	1.00	0.50	0.99	0.50	0.50	0.50	0.50	0.50	0.99	0.50	0.99	0.50	0.99	0.50	0.99	0.51
50	.10	0.53	0.50	0.62	0.61	0.82	0.84	0.82	0.81	0.55	0.51	0.81	0.82	0.84	0.83	0.80	0.79	0.89	0.87
	.50	0.81	0.51	0.63	0.62	0.85	0.83	0.84	0.83	0.52	0.50	0.81	0.82	0.81	0.83	0.81	0.82	0.92	0.90

Table 4: Description of the 27 binary data sets.

ID	Data Set	#Instance	#Features	Class Proportion	ID	Data Set	#Instance	#Feature	Class Proportion
1	haberman	306	3	81/225	15	*audit risk*	775	26	305/470
2	heart statlog	270	14	120/150	16	*adult autism*	609	20	180/429
3	*hungarian*	261	10	98/163	17	*banknote*	1372	4	610/762
4	*hepatitis*	80	19	33/47	18	*transfusion*	748	4	178/570
5	*liver disorders*	345	6	145/200	19	*caesarian*	80	4	34/46
6	*parkinsons*	195	22	48/147	20	*thoraric*	470	16	70/400
7	*sonar*	208	60	97/111	21	*circles*	100	2	50/50
8	*column 2C*	310	6	110/210	22	*spirals*	500	2	250/250
9	*ionosphere*	351	33	126/225	23	*australian*	690	14	307/383
10	*spam*	4601	57	1813/2788	24	*tic tac toe*	958	3	332/626
11	*dataR2*	116	9	52/64	25	*german*	1000	24	300/700
12	*kidney disease*	155	24	41/114	26	*sick*	2643	31	212/2431
13	*clean*	476	168	207/269	27	*vehicle*	846	18	218/628
14	*whosale*	440	7	142/298					

6 Real Data Application

Our methodology was applied on 27 benchmarking datasets from the University of California Irvine (UCI) Repository (Dua and Graff, 2017) to evaluate its performance. The datasets present a wide variety in the number of observations, dimensionality, and type of data. All of them represent a binary classification task (i.e.: two classes only), and no one of them presented missing values. Table 4 summarizes all datasets considered. The continuous features were scaled

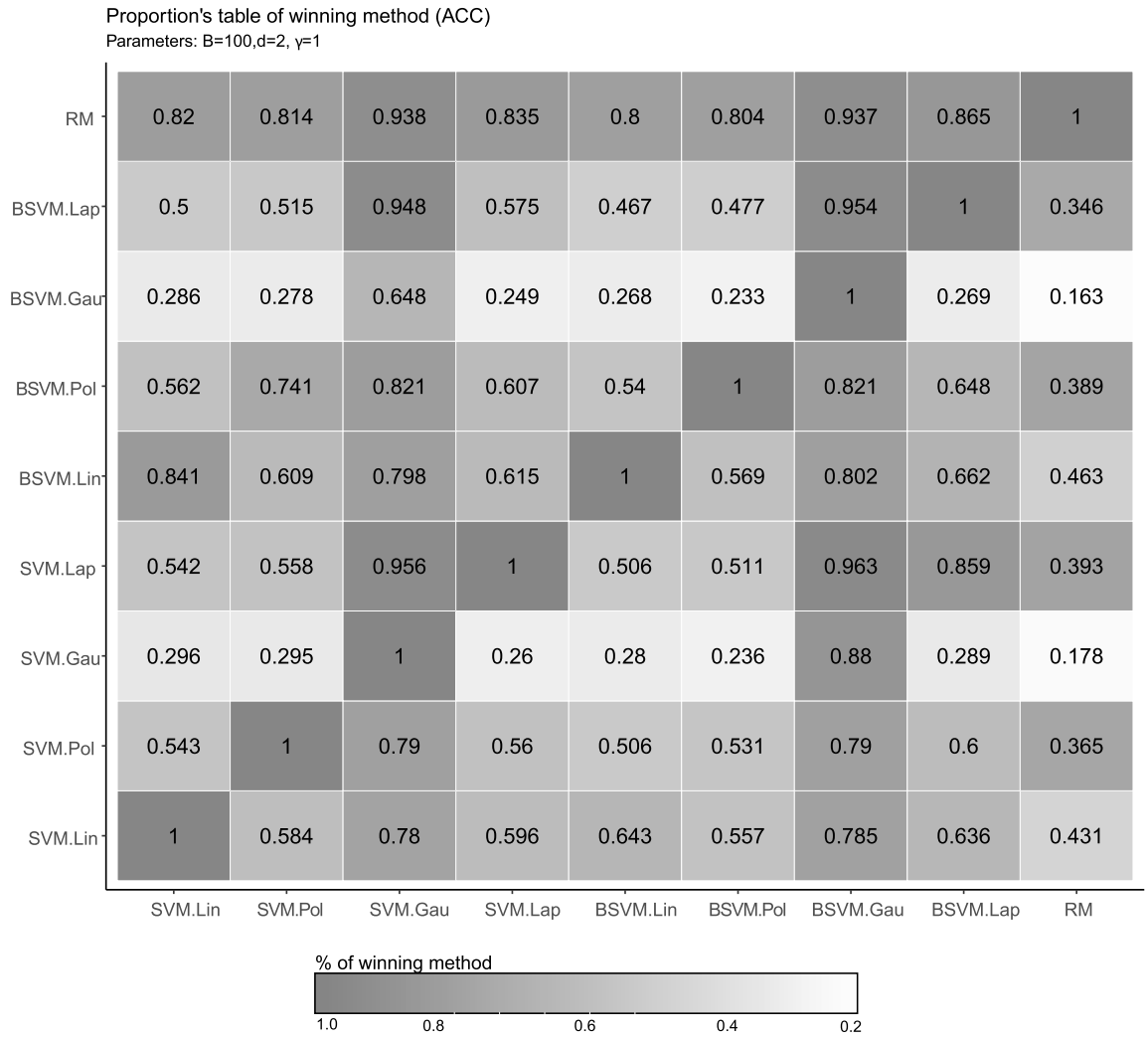

Figure 2: Proportion of the number of times which a method had greater or equal accuracy than the others. The proportion summarizes the applications over all 27 data sets and 30 holdout repetitions. Is clear the superiority of the random machines when it is compared with the other models.

to zero mean and unit variance, except the discrete features, which were preprocessed by a one-hot-encoding transformation. The validation technique is also used as the repeated holdout with 30 repetitions with a split ratio of training-test of 70%–30%.

The random machines was compared with the standard SVM and bagged SVM using each single kernel function presented in Table 1. Without losing generality, the chosen parameters were: the cost parameter $C = 1$, the number of bootstrap samples $B = 100$, the degree of polynomial kernel $d = 2$, and the hyperparameter γ from the Laplacian and Gaussian kernel $\gamma = 1$. The result is summarized in the Figure 2 considering the accuracy and in the Figure 3 considering the uMCC.

As shown in Figure 2, the RM demonstrates higher accuracy than the other bagged support vectors using unique kernel functions. Comparing the RM with the traditional bagged SVM,

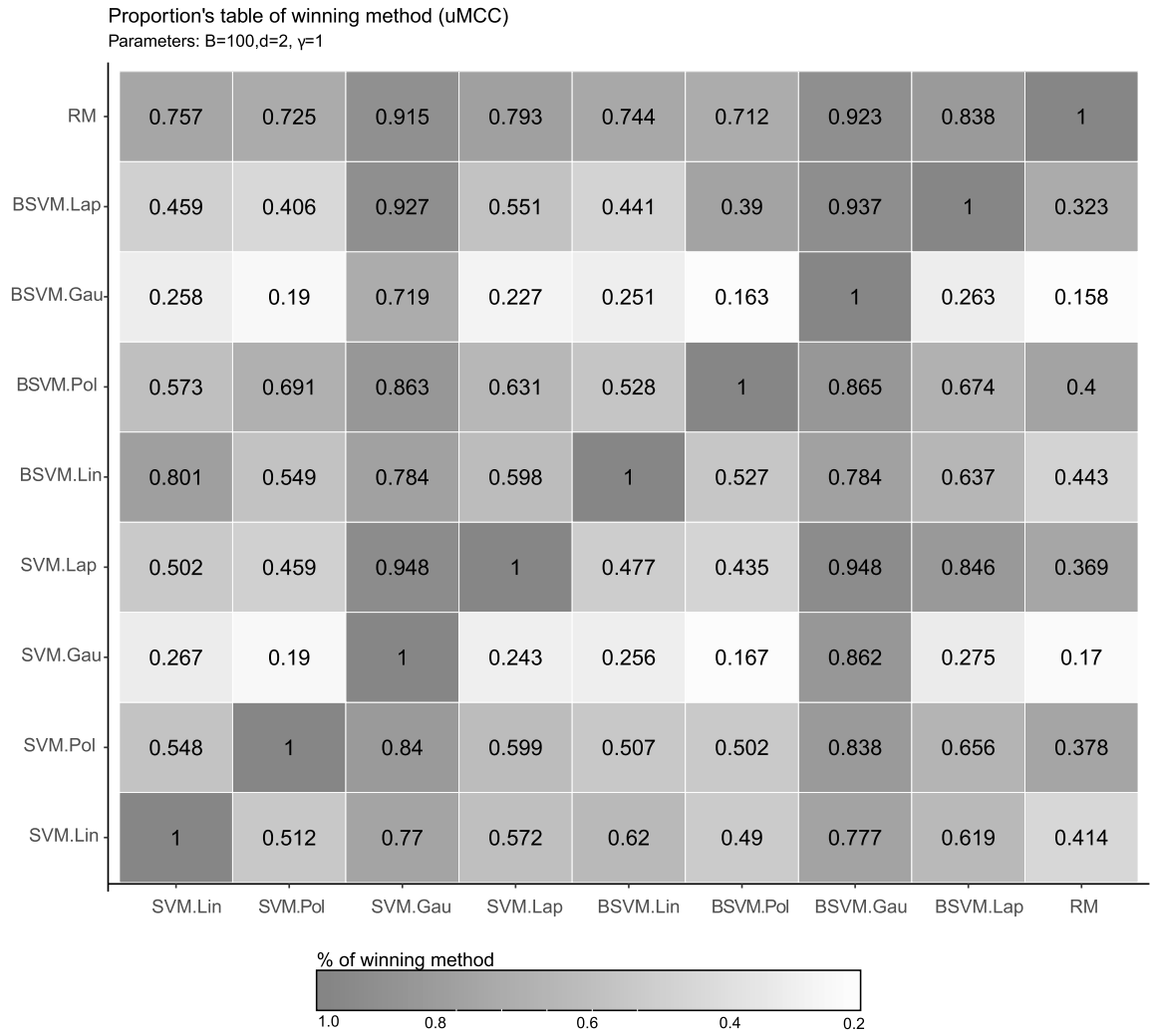

Figure 3: Proportion of the number of times which a method had greater or equal uMCC than the others. The proportion summarizes the applications over all 27 datasets and 30 holdout values. Is clear the superiority of the random machines when it is compared with the other models.

it is beaten almost 80% of times considering the Kernel Linear Bagging, 81% for the Kernel Polynomial Bagging, 94% for the Gaussian Bagging, and 87% for the Laplacian Kernel Bagging. This outcome shows off that the random weighted choice of the kernels functions improved, generally, the accuracy of the predictions from the model. The difference is even more significant when the random machines are compared with the singular SVM, where the RM is more accurate 82% of times considering the Kernel Linear, 81% for the Kernel Polynomial, 94% for the Gaussian Bagging, and 84% for the Laplacian Kernel.

The same behavior is observed when it is considered the Uniform Matthew's Correlation Coefficient, in which the RM presents a robust superiority when compared to other methods. Analyzing the RM with the traditional bagged SVM is beaten almost 74% of times considering the Kernel Linear Bagging, 71% for the Kernel Polynomial Bagging, 92% for the Gaussian Bagging,

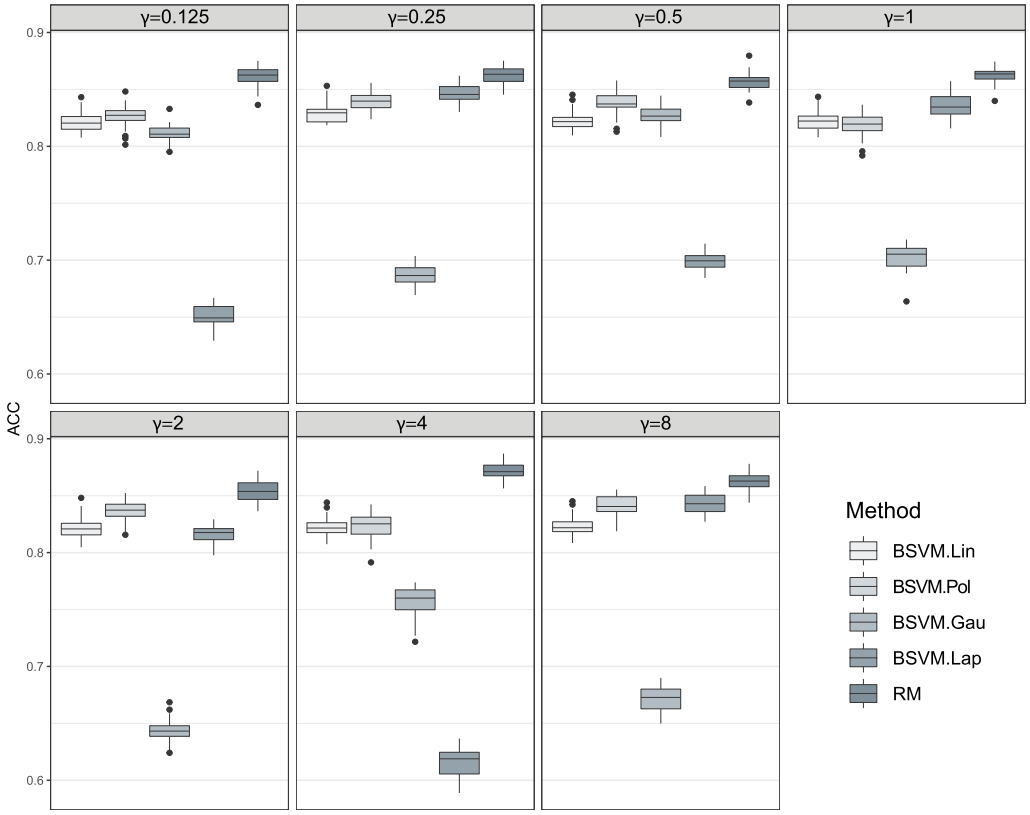

Figure 4: Summary of the ACC applied over 27 real datasets with the variation of kernel function's parameter γ.

and 84% for the Laplacian Kernel Bagging. It also happens when the RM is compared with the singular SVM, where the RM is more accurate 82% of times considering the Kernel Linear, 81% for the Kernel Polynomial, 94% for the Gaussian Bagging, and 84% for the Laplacian Kernel.

The scheme also avoids the problem of the selection of the best kernel function, since is not necessary to perform a grid-search among all the different kernel functions and define which is one has lower test error, which is the general framework adopted. Therefore, in the RM algorithm, the efficiency is increased, and computational cost is reduced.

As the hyperparameter tuning is a remarkable question in the procedure of the support vector machine vector, the value of γ, hyperparameter from kernel functions, was changed to study the variation and the behavior of random machines when this change exists. The variation experiment relies on the interval of values $\gamma = \{2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}\}$. The result is showed in Figure 4 and 5 in which it is possible to see that the RM surpassed the other bagging kernels all the times. As mentioned before the choice of these hyperparameters, as the kernel function, has a direct impact on the model performance, and the results reinforce that RM gives a good and consistent result independent for all γ values. In fact, it is possible to perform support vector machine tuning procedure using grid-search methods. The choice of the interval for the hyperparameter values is a crucial point to designate the computational cost of the traditional support vector machine modeling. In contrast, the RM model is essentially free of the tuning process and always more efficient than evaluate its independent bagging models.

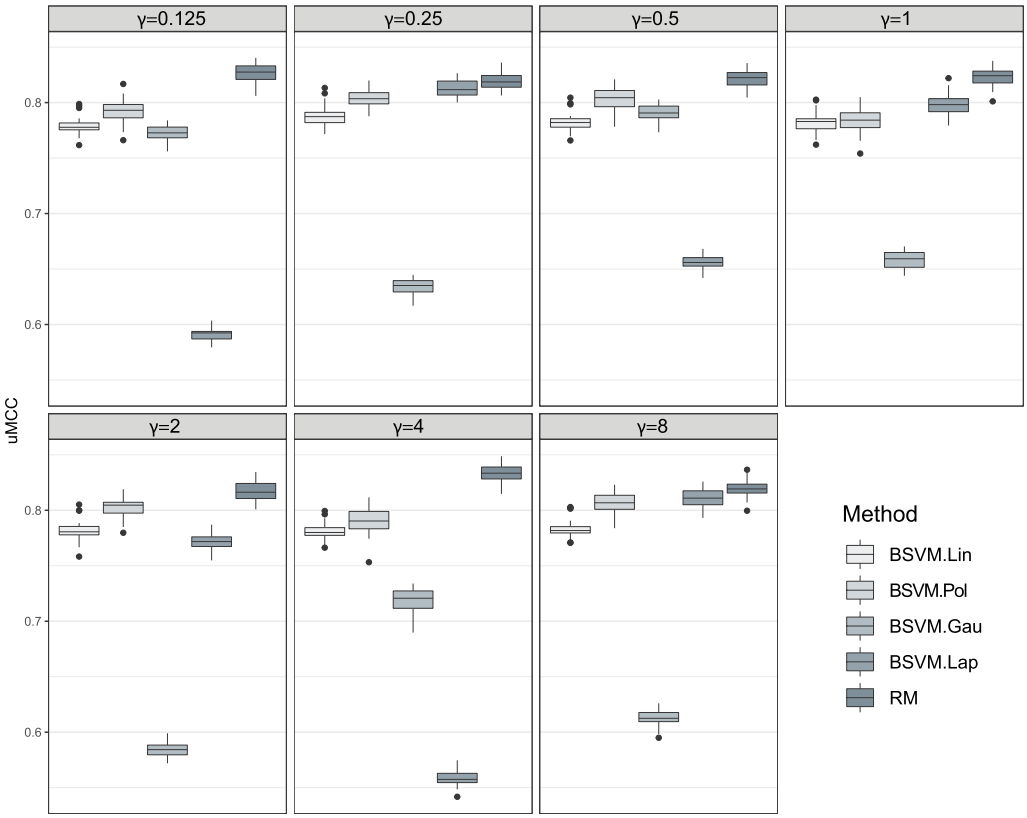

Figure 5: Summary of the uMCC applied over 27 real datasets with the variation of kernel function's parameter γ.

7 Performance and Agreement Evaluation

In this section, we justify the reason why the random machines are an ensemble method that can improve the predictive power for classification tasks. The main idea of the random selection of the kernel function is to select different functions that belong to a Reproducing Kernel Hilbert Space (RHKS). The objective of this random sample is to aim a lower correlation between classifiers that compose the RM, and a high strength of them since these characteristics improve the bagging algorithm, as is shown in Breiman (2001).

The idea of the strength of a model relies on how well a model correctly predicts a new observation, i.e.: its predictive capacity. On the other hand, the correlation between models consists of how much they are similar. A method to estimate the correlation between classification models is to calculate the area from decision boundaries that overlaps among them (Turney, 1995). Ho (1998), defines the similarity (also called as agreement) of two models as the number of observations that are equally labeled with the same class, and proposes that it can be estimated through a random sample with n observations, by the Equation (13).

$$\hat{\boldsymbol{s}}_{i,j} = \frac{1}{n}\sum_{k=1}^{n} f(\boldsymbol{x_k}) \tag{13}$$

Table 5: Summary of accuracy and agreement means to each method.

Circles Data Set	**Method**									
	BSVM.Lin		**BSVM.Pol**		**BSVM.Gau**		**BSVM.Lap**		**RM**	
p	**ACC**	**AGR**	**ACC**	**AGR**	**ACC**	**AGR**	**ACC**	**AGR**	**ACC**	**AGR**
2	0.54	0.59	0.98	0.98	0.97	0.97	0.97	0.97	0.99	0.96
10	0.49	0.64	0.95	0.92	0.74	0.72	0.91	0.91	0.96	0.84
30	0.49	0.49	0.78	0.78	0.51	0.59	0.87	0.87	0.94	0.67
50	0.55	0.67	0.71	0.71	0.49	0.61	0.57	0.62	0.79	0.62

where

$$f(\boldsymbol{x_k}) = \begin{cases} 1, & \text{if } g_i(\boldsymbol{x_k}) = g_j(\boldsymbol{x_k}) \\ 0, & \text{otherwise} \end{cases}$$

This measure can be used as a correlation metric between models.

To evaluate the correlation and strength of the RM in comparison with the traditional bagged version of SVM, the method was applied over the *Circles* database that was generated under the same configuration of **Dataset 3** presented in Section 5. The similarity of each method was estimated using the average of the similarity $\hat{s}_{i,j}$, $\forall i \neq j$ and $i, j = 1, \ldots, B$, over fixed k points generated by a Monte Carlo's simulation. The accuracy was used in order to measure the strength of the model.

The dataset was modified in three configurations, changing the dimension p in a range corresponding to $p = \{2, 10, 30, 50\}$. The average similarity was calculated using k observations, where $k = 1000 \times p$. Both accuracy and agreement were calculated using a 30 Repeated Holdout validation set with split ratio of 70–30% training-test. The parameters of the methods were: B = 100, $\gamma = 1$, $C = 1$.

One of the main results can be represented in the Figure 6 where the *circles* database with $p = 2$ was used as example. In the Figure 6 (a) is shown a plot from the observations, where each color represents a class. Panel (b) represents the final decision boundary of the RM, showing that the model captures the behavior from the observations. Panel (c) shows examples of the decision region generated by a bootstrap model g_i for each kernel. It is clear that different kernel functions used in each SVM model produce diverse decision boundaries, and that difference implies a reduction of the correlation, resulting in the decreasing of generalization error.

All the results are summarized in Table 5 where it is presented the mean accuracy and agreement for each data set for all configurations of the *circles*.

In general, it is remarkable that the higher predictive power of the RM when compared to the other methods in all cases. Moreover, beyond the great accuracy, it is possible to see that the RM shows simultaneously a lower agreement when compared with the other methods that have an excellent performance. Although sometimes the BSVM.Lin and BSVM.Gau produces a desirable low agreement, they are considered weaker, since they have a lower accuracy when compared with the others. As Ho (1998) discussed, the accuracy of the models affects the agreement and vice-versa, and optimizes both values simultaneously can be a challenging task. Generally, models with high accuracy also result in large agreement values, as we can see in the results exhibited in Table 5. On another hand, small values of accuracy produce lower agreement measures among models. However, it is clear to notice that the random machines it is capable create a better

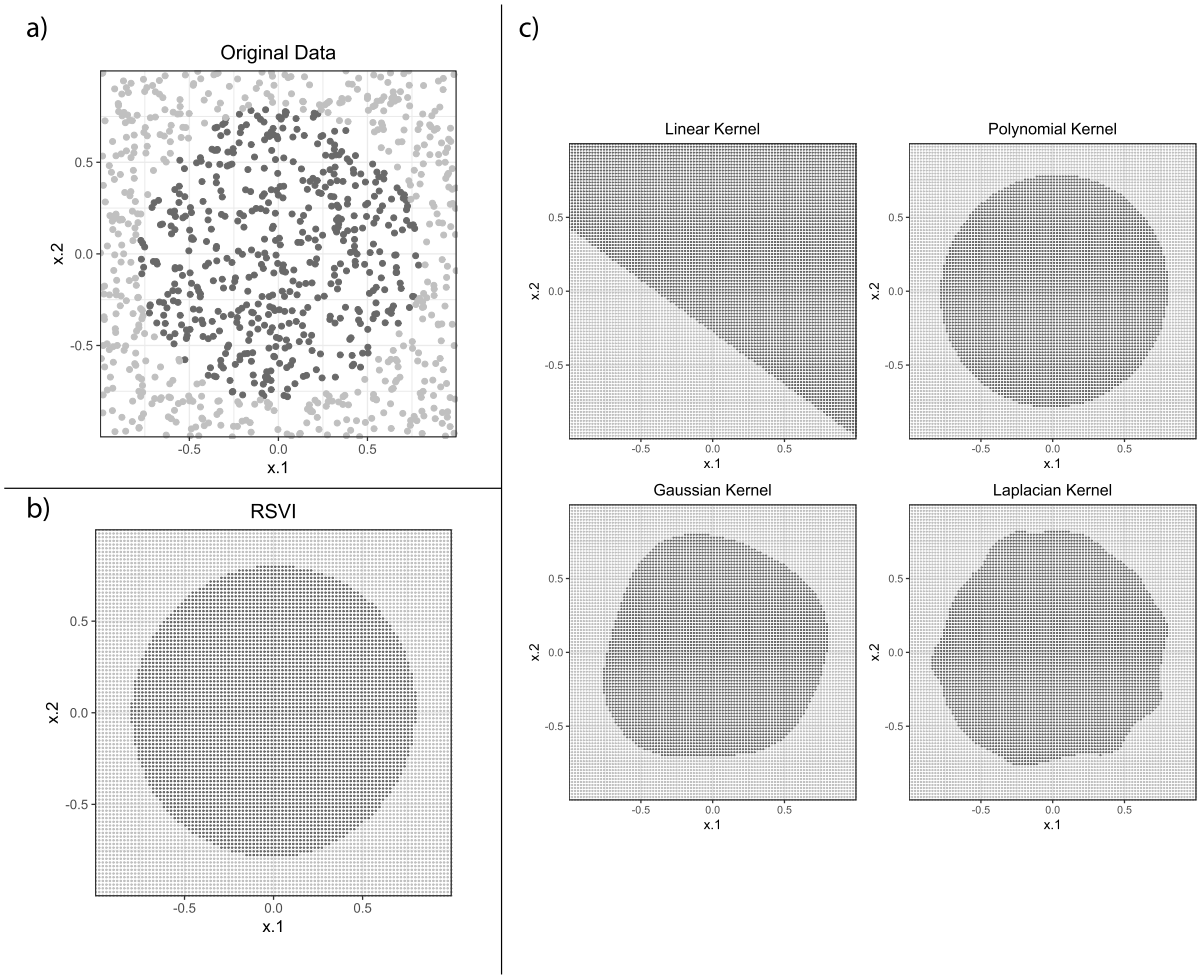

Figure 6: The *circles* database where $p = 2$. Panel (a) show all the observations with the class associated with each color. Panel (b) present the decision region given by the RM. Panel (c) reveals the diversity of decision regions produced by each kernel function of bootstrap models that composes the RM.

classification model (low generalization error) with both characteristics: low correlation and great strength (predictive power).

These proprieties become better with higher dimensions as they can be observed in Table 5. This difference is showed in Figure 7 that display the boxplots of the accuracy and agreement for each method, reinforcing even more that the RM has both better proprieties, high strength, and low agreement, than the other ones. Despite Figure 7 seems to demonstrate a large variance from RM, it is important to emphasize that those boxplots correspond to all the results in all simulation setups and summarized in Table 5. The other support vector methods were uniformly less able to predict the scenarios correctly. From Figure 4 and Figure 5 we can see that the RM was evaluated exhaustively over 27 benchmarks and shown narrow standard deviation.

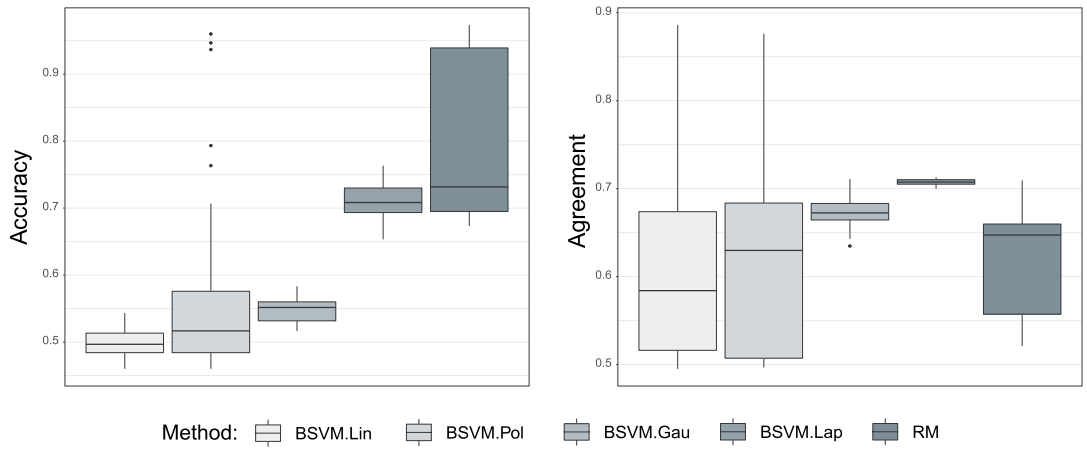

Figure 7: Boxplots of the accuracy and agreement for each method.

8 Final Considerations

The main contribution of this paper is to propose a novel learning method to do ensemble using Support Vector Machine models that can enhance the accuracy from the conventional BSVM predictions and solve the problem of choosing the best kernel function that should be used in the traditional approach. Through the random machines, the combination of different SVM using the different kernel functions states an approach that avoids the expensive computational cost of doing a grid search between the kernel functions, besides improve the accuracy.

In order to quantify the computational reduction, suppose a number of B models calculated in a traditional bagging procedure and R as the number of kernels functions that will be evaluated and used in support vector models. In a traditional bagging algorithm using SVM as base models the number of total models that will be calculated to obtain the bests results is given by $B \times K$, while using the random machines approach this number reduces to $B + K$. Using an example of $B = 100$ and $K = 400$, we have that the traditional bagging algorithm would take approximately four times the computational cost than the proposed random machines since the ratio of calculated models is $400/104$ (i.e.: four times faster).

Furthermore, results show a good behavior with different kernel hyperparameters in RM, which provides a bagged-weighted support vector model with free kernel function choice. In this way, as SVM is one of the most important and essential methods in machine learning with high-performance capacity and power of generalization, the RM method can be viewed as an extension of traditional SVM, giving an alternative solution to the hyperparameters choice problem. This methodology can be explored in many other contexts, as well as be applied to any practical machine learning problem. Future theoretical studies may be done regarding computational cost, comparison with other traditional machine learning methods, and the use of other and more kernel functions as well as other weights in the bagging phase.

Supplementary Material

The proposed model called Random Machines (RM) was also implemented in R language and it can be used through the ***rmachines*** package, available and documented at GitHub

https://github.com/MateusMaiaDS/rmachines. To a overall description of how to reproduce the results from this article just access the README at https://mateusmaiads.github.io/rmachines/.

Funding

M.M.'s work was supported by a Science Foundation Ireland Career Development Award grant 17/CDA/4695. The authors are grateful for the partial funding provided by the Brazilian agencies CNPq and CAPES.

References

Ayat NE, Cheriet M, Suen CY (2005). Automatic model selection for the optimization of SVM kernels. *Pattern Recognition*, 38(10): 1733–1745.

Baldi P, Brunak S, Chauvin Y, Andersen CA, Nielsen H (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5): 412–424.

Bergstra J, Bengio Y (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13: 281–305.

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011). Algorithms for hyper-parameter optimization. In: Dietterich T, Becker S, Ghahramani Z (Eds.). *Advances in neural information processing systems*, 2546–2554. Curran Associates, Inc..

Boser BE, Guyon IM, Vapnik VN (1992). A training algorithm for optimal margin classifiers. In: Haussler D (Chair). *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152. ACM.

Boughorbel S, Jarray F, El-Anbari M (2017). Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLoS ONE*, 12(6): e0177678.

Breiman L (1996). Bagging predictors. *Machine Learning*, 24(2): 123–140.

Breiman L (2001). Random forests. *Machine Learning*, 45(1): 5–32.

Breiman L (1998). Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3): 801–849.

Burdisso SG, Errecalde M, Montes-y Gómez M (2019). A text classification framework for simple and effective early depression detection over social media streams. *Expert Systems with Applications*, 133: 182–197.

Chapelle O, Vapnik V (2000). Model selection for support vector machines. In: Leen T, Dietterich T, Tresp V (Eds.). *Advances in neural information processing systems*, 230–236. MIT Press.

Cherkassky V, Ma Y (2004). Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1): 113–126.

Claesen M, De Smet F, Suykens JA, De Moor B (2014). Ensemblesvm: a library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15(1): 141–145.

Cortes C, Vapnik V (1995). Support-vector networks. *Machine Learning*, 20(3): 273–297.

Courant R, Hilbert D (1953). *Methods of mathematical physics, vol. i.* Interscience Publ. Inc., New York, 106.

Cueto-López N, García-Ordás MT, Dávila-Batista V, Moreno V, Aragonés N, Alaiz-Rodríguez R (2019). A comparative study on feature selection for a risk prediction model for colorectal cancer. *Computer Methods and Programs in Biomedicine*, 177: 219–229.

Dighe D, Patil S, Kokate S (2018). Detection of credit card fraud transactions using machine learning algorithms and neural networks: A comparative study. In: Landge SDP (Chair). *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–6. IEEE.

Dua D, Graff C (2017). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.

Fletcher R (1987). *Practical methods of optimization*. John Wiley & Sons, New York, 80.

Freund Y, Schapire R, Abe N (1999). A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, 14(771–780): 1612.

Friedrichs F, Igel C (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, 64: 107–117.

Gordon JJ, Towsey MW, Hogan JM, Mathews SA, Timms P (2005). Improved prediction of bacterial transcription start sites. *Bioinformatics*, 22(2): 142–148.

Ho TK (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8): 1–22.

Huang MW, Chen CW, Lin WC, Ke SW, Tsai CF (2017). SVM and SVM ensembles in breast cancer prediction. *PLoS ONE*, 12(1): e0161501.

Hussain M, Wajid SK, Elzaart A, Berbar M (2011). A comparison of SVM kernel functions for breast cancer detection. In: Banissi E, Sarfraz M (Eds.). *2011 Eighth International Conference Computer Graphics, Imaging and Visualization*, 145–150. IEEE.

Jebara T (2004). Multi-task feature and kernel selection for SVMs. In: Brodley C (Ed.). *Proceedings of the twenty-first international conference on Machine learning*, 55. ACM.

Kim H, Howland P, Park H (2005). Dimension reduction in text classification with support vector machines. *Journal of Machine Learning Research*, 6: 37–53.

Kim HC, Pang S, Je HM, Kim D, Bang SY (2002). Support vector machine ensemble with bagging. In: Lee S, Verri A (Eds.). *International Workshop on Support Vector Machines*, 397–408. Springer.

Kirkpatrick S, Gelatt CD, Vecchi MP (1983). Optimization by simulated annealing. *Science*, 220(4598): 671–680.

Lei Z, Yang Y Wu Z (2006). Ensemble of support vector machine for text-independent speaker recognition. *International Journal of Computer Science and Network Security*, 6(5): 163–167.

Li Z, Yang T, Zhang L, Jin R (2016). Fast and accurate refined nyström-based kernel svm. In: Schuurmans D, Wellman M (Eds.). *Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press.

Lin HT, Li L (2008). Support vector machinery for infinite ensemble learning. *Journal of Machine Learning Research*, 9: 285–312.

Matthews BW (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta. Protein Structure*, 405(2): 442–451.

Min JH, Lee YC (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, 28(4): 603–614.

Mokgonyane TB, Sefara TJ, Modipa TI, Mogale MM, Manamela MJ, Manamela PJ (2019). Automatic speaker recognition system based on machine learning algorithms. In: Markus ED (Ed.). *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, 141–146. IEEE.

Pang S, Kim D, Bang SY (2003). Membership authentication in the dynamic group by face

classification using svm ensemble. *Pattern Recognition Letters*, 24(1–3): 215–225.

Pham BT, Bui DT, Prakash I (2018). Bagging based support vector machines for spatial prediction of landslides. *Environmental Earth Sciences*, 77(4): 146.

Sato M, Morimoto K, Kajihara S, Tateishi R, Shiina S, Koike K, et al. (2019). Machine-learning approach for the development of a novel predictive model for the diagnosis of hepatocellular carcinoma. *Scientific Reports*, 9(1): 7704–7704.

Shah SAR, Issac B (2018). Performance comparison of intrusion detection systems and application of machine learning to snort system. *Future Generations Computer Systems*, 80: 157–170.

Smola AJ, Bartlett PJ, Schuurmans D, Schölkopf B, Jordan MI, et al. (2000). *Advances in large margin classifiers*. MIT press.

Smola AJ, Schölkopf B (2000). Sparse greedy matrix approximation for machine learning. In: Langley P (Ed.). *Proceedings of the 17th International Conference on Machine Learning*, 911–918.

Sun Y, Gilbert A, Tewari A (2018). But how does it work in theory? linear SVM with random features. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (Eds.). *Advances in Neural Information Processing Systems*, 3379–3388.

Thanh Noi P, Kappas M (2018). Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery. *Sensors*, 18(1): 18.

Tong M, Liu KH, Xu C, Ju W (2013). An ensemble of svm classifiers based on gene pairs. *Computers in Biology and Medicine*, 43(6): 729–737.

Turney P (1995). Bias and the quantification of stability. *Machine Learning*, 20(1–2): 23–33.

Van Wezel M, Potharst R (2007). Improved customer choice predictions using ensemble methods. *European Journal of Operational Research*, 181(1): 436–452.

Wang H, Zheng B, Yoon SW, Ko HS (2018). A support vector machine-based ensemble algorithm for breast cancer diagnosis. *European Journal of Operational Research*, 267(2): 687–699.

Wang Sj, Mathew A, Chen Y, Xi Lf Ma L Lee J (2009). Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications*, 36(3): 6466–6476.

Williams CK, Seeger M (2001). Using the nyström method to speed up kernel machines. In: Dietterich T, Becker S, Ghahramani Z (Eds.). *Advances in neural information processing systems*, 682–688. MIT Press.

Wu CH, Tzeng GH, Lin RH (2009). A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Systems with Applications*, 36(3): 4725–4735.

Zhou L, Lai KK, Yu L (2010). Least squares support vector machines ensemble models for credit scoring. *Expert Systems with Applications*, 37(1): 127–133.