International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-023-00472-y

REGULAR PAPER

Oblique random forests with binary and ternary decision structures and non-parallel hyperplanes classifiers

Pooja Saigal[1] · Anubhav David[2] · Reshma Rastogi[2]

Received: 7 August 2023 / Accepted: 24 October 2023

Abstract
Due to their robust nature, ensemble methods have gained a lot of popularity these days. In this work, we propose several variations of oblique decision tree ensembles called as oblique random forests, which are implemented with binary and ternary decision structures. Oblique random forests are trained using a linear classifier, where the feature axis is not invariably orthogonal to the decision hyperplanes at each internal node of the base model. For the multiclass classification problems, the training samples are partitioned at non-leaf nodes into groups of classes corresponding to the underlying geometric characteristics, with respect to a randomly chosen feature subspace. Each of the proposed models employ a different binary base classifier. The binary classifiers used for this work are twin support vector machines (TWSVM), Improvements on ν-TWSVM, multi-surface proximal support vector machine (MPSVM) and Regularized MPSVM. We also propose a novel approach to choose the final hyperplane to split the data at the non-leaf node while optimizing an impurity criterion in the decision tree. This work presents a comparative analysis of different base classifiers for implementing Oblique Random forests using binary and ternary decision structures. In addition, multiple regularization strategies like Tikhonov regularization, axis-parallel split regularization, and null space regularization are used to address limited sample size issues in the oblique random forest decision trees implemented with MPSVM and RegMPSVM. Whereas implementations for TWSVM and IνTWSVM is done with Tikhonov regularization only. All these models are compared for their generalization ability through benchmark 38 UCI classification datasets. The efficacy of these methods is also established through statistical analysis.

Keywords Ensemble methods · Oblique random forests · Twin support vector machines and Multi-surface proximal support vector machines · Classification · Decision trees

1 Introduction

Decision tree (DT) is a classification tool that focuses at extracting the underlying knowledge from the data set. DT partitions the observed data in a hierarchical manner, by employing a set of elementary decisions. Decision trees are hierarchical learners comprising of a collection of binary decision nodes. In 1984, Breiman et al. [1] proposed the tree model for classification and regression problems. Later, they became quite well known and were widely employed for innumerable learning applications. Their prosperity may be said to be benefiting from many advantages: they are quick and scalable to very large datasets; are very instinctive, and can be composed in a probabilistic manner taking uncertainty into account. Over the past decades, numerous learning models have been proposed, most popular being the C4.5 of Quinlan [2]. However, an optimal decision tree is NP-complete problem, and results into complex learners that do not generalize well, i.e. suffer from over-fitting.

In 1995, Ho formalized [3] to compute an ensemble of decision trees, known as random forest (RaF), rather than optimizing an exclusive and complex single tree. Random forest [1, 4] is an ensemble of decision trees, where DTs

Pooja Saigal, Anubhav David and Reshma Rastogi have contributed equally to this work.

✉ Pooja Saigal
pooja.saigal@vips.edu

Anubhav David
anubhavdavid@gmail.com

Reshma Rastogi
reshma.khemchandani@sau.ac.in

1 School of Information Technology, Vivekananda Institute of Professional Studies, Pitampura, Delhi 110034, India

2 Department of Computer Science, South Asian University, Chanakyapuri, Delhi 110021, India

Published online: 18 November 2023

are greedy, divide and conquer algorithms which are known for their simplicity and easy interpretation. Katuwal et al. recently proposed heterogeneous oblique random forests [5]. Decision trees have a low bias, but high variance, therefore, they are an ideal candidate for a base classifier in an ensemble framework. Ensemble classifiers stem on the idea of perturb and combine strategy [6], involving two steps. Perturb step involves training a learning algorithm on perturbed datasets whereas combine step aggregates the result of the corresponding classifiers learnt in previous step. This strategy results in significant improvement of the classifiers and is better than using a single weak classifier. DTs are sensitive towards perturbation of the training data due to its recursively partitioning strategy, yet it produces adequately accurate results. Extensive research has been done to demonstrate the advantages of ensemble paradigm over single classifier model. Ensemble classifiers, also called as multiple classifier systems, have gained their way around most of the research areas such as computer vision [7] and machine learning [8]. Bonissone et al. proposed a fuzzy variation of Random forests [9] and Yassin et al. proposed RaF for road accident prediction [10].

Random forest has been widely studied, with univariate decision tree approaches [11] employed for its implementation. Decision trees can be generated as univariate or multivariate. Univariate (or orthogonal or axis-parallel) DT singles out the best feature from several features to partition the data and generate a hyperplane parallel or orthogonal to the feature axis at each non-leaf node. However, such hyperplanes do not always learn complex decision boundaries [12], [13, 14]. By employing a substantial number of stair-like decision boundaries, DT can determine any oblique decision boundary. The multivariate (or oblique) DT [15] uses a linear combination of all or part of the features to perform a test to split the data. It generates a hyperplane which is oblique to the feature axes at each non-leaf node. Random forests are ranked as the best classifiers in a recent comparison among 179 classifiers for 121 datasets [16]. These are, however, superseded by the oblique random forest (ObRaF) in a comparison among 183 classifiers [17]. ObRaF ensemble of decision trees are grown using linear hyperplanes at each internal node to split the data in lieu of a single feature. The decision tree induction algorithm progresses by searching the split that scores the minimum value based on some impurity criterion.

Since, random forest demonstrates better generalization by aggregating results, researcher proposed to inject randomization in the learning framework to learn decorrelated decision trees. Breiman [18] proposed an approach to inject randomization in the learning phase, known as "bagging", for bootstrap aggregation. This involves learning each individual tree with a random subset of the training data samples. Decision trees in Random Forest are trained on the concepts of bagging and random subspace [19] to grow an ensemble. It consists of a collection of decision trees employing randomly selected subspaces of data features at each internal non-leaf node and optimizing some impurity criteria such as Gini index or information gain [20]. All the base classifiers perform proficiently, given the fact that they are trained on a bootstrap dataset (bagging), similar to that of the whole population. Variance is increased by employing random subspace strategy at each internal node of every base classifier. The underlying motivation behind the random subspace strategy is that by aggregating ensemble out-turn along with explicit randomization of the feature selection, variance should reduce more significantly than other exhaustive randomization schemes. After this proposition, Random Forest has been eminently used in many applications, mostly classification frameworks. Nonetheless, they are suitable to evaluate regression, clustering, density estimation, semi-supervised learning or manifold learning tasks, as exhibited in [20].

In [21], Zhang et al. used a binary classifier, the Multi-surface Proximal Support Vector Machine (MPSVM) [22] to find the discriminatory hyperplane, which understands the geometric arrangement of class distributions and appropriates the ideas from [23]. MPSVM is a non-parallel hyperplane classifier (NPHC), which generates two discriminatory hyperplanes proximal to patterns of each class. The idea is to keep the hyperplane in close proximity to its respective class and should be away from the other class. The test patterns are classified based on distance from the two classes. Angle bisector of these two discriminatory hyperplanes is chosen to optimize Gini-impurity at each non-leaf node of the decision tree. To focus on the small sample size problem, authors use Tikhonov regularization, Axis-parallel regularization and the NULL space regularization methods [24]. However, the NULL space regularization is found to be unstable [25] and is not significant [21].

Most of the support vector machine (SVM)-based classifiers are binary classifiers. So, there is a need to transform them into multi-category framework. Khemchandani et al. [26] proposed two approaches for multi-category extension of binary classifiers. These are ternary decision structure (TDS) and binary decision structure (BDS). In this paper, we propose a novel framework for oblique decision tree ensembles with binary and ternary decision structure (BDS and TDS) models where these oblique decision trees are implemented via different base classifiers. The binary classifiers used for this work are twin SVM (TWSVM), improvements on ν-TWSVM, MPSVM and regularized MPSVM. TWSVM has been widely researched in the last decade. Iν-TWSVM [27] is an improved variant of TWSVM, which is used as a base classifier in this work. Some other variations of TWSVM are TWSVM with localized kernel spectral clustering [28], ternion SVM [29], angle-based twin parametric

SVM [30], angle-based TWSVM [31], Fuzzy TWSVM [32], fuzzy least-squares TWSVM for clustering [33], projection least-squares TWSVM [34] etc. We also propose a novel approach to choose the final hyperplane to split the data at the non-leaf node while optimizing an impurity criterion, rather than just choosing one of the angle bisectors.

The remaining part of this paper is organized into following sections: decision trees, random forests, SVM based binary classifiers and multi-category classification approaches are reviewed in Sect. 2. Our proposed method is discussed in Sect. 3. The experimental results are provided in Sect. 4, and the paper concludes in Sect. 5.

2 Related work

In this section, we briefly discuss decision trees, random forests, the base (binary) classifiers used in the work and multi-category classification approaches.

2.1 Decision tree model

A decision tree (DT) [1] model is a classification algorithm that possesses a hierarchical tree structure. In a DT, the label for a data instance x is decided via a pre-structured series of conditions on the feature vector of x. Here, these conditions in the hierarchical structure lead to a single node, i.e. leaf node in a tree. This decision procedure of a series of conditions is a tree traversal method, where a single sequence of conditions leads to a path down the tree, from root to a leaf node.

Algorithm 1 Top-down Induction of Decision Tree (TDIDT) Algorithm

```
Input : = Labeled samples S
Output : node $v$
1: $v \leftarrow$ new Node()
2: if all samples in $S$ have $y$ label then
3:     $y(v) \leftarrow y$
4:     return $v$
5: end if
6: $(\phi(v), \tau(v)) \leftarrow SelectFeat(S)$
7: if $\phi(v)$ is numeric then
8:     $v_1 \leftarrow TDIDT(\{(x, y) \in S, x_{\phi(v)} < \tau(v)\})$
9:     $v_2 \leftarrow TDIDT(\{(x, y) \in S, x_{\phi(v)} \geq \tau(v)\})$
10: else
11:     $\{S_1, ....., S_{Y(\phi(v))}\} \leftarrow Split(S, \phi(v))$
12:     for i:=1 to $Y(\phi(v))$ do
13:         $v_i \leftarrow TDIDT(S_i)$
14:     end for
15: end if
16: return $v$
```

A decision tree is a divide and conquer strategy for classification tasks. They are the primary and most standard models of machine learning formalized to imitate the human-decision making process [1]. In decision tree model, each internal (non-leaf) node v is related with a feature index $\phi(v)$ and a numeric value $\tau(v)$, implementing a condition on the feature $\phi(v)$ with n possible results, resulting to one of the n child nodes, denoted as v_1, ..., v_n. Every leaf node v in the tree has a corresponding single decision value, denoted by $y(v)$. The top-down induction of the decision tree algorithm (TDIDT) is illustrated in Algorithm 1. It is a simple recursive approach to grow a decision tree model, often called as *tree induction*. The *SelectFeat()* function computes the feature index and numeric value related to the node v, whereas the "split" procedure creates $| Y(\phi(v)) |$ disjoint sets, each containing all samples from S with the identical value of $\phi(v)$, where $| Y(\phi(v)) |$ are the numerous possible values for feature $\phi(v)$. For *SelectFeat()* function, we rely on Gini index criterion, singling out the pair $\langle \phi(v), \tau(v) \rangle$ that maximizes this value, as done in the C4.5 algorithm [2], [35, 36].

When used to classify an unknown data sample x, a DT classifier operates recursively, beginning from the root, as follows: let $x_{\phi(v)}$ be the element in the feature vector related to the feature index $\phi(v)$ of a non-leaf node v. The decision at node v goes in favour of the left child if $\phi(v)$ is an index of a numeric feature, $x_{\phi(v)} \geq \tau(v)$, otherwise the right child is selected. The decision at the node v is of the child matching to $x_{\phi(v)}$ in $Y(\phi(v))$, if the $\phi(v)$ is an index of a nominal feature. Considering a data point x, for a fixed decision tree, DT, a unique path is defined from its root to a leaf node v such that we follow the edge, at each internal node u, from u to its child as discussed above. We say that x "arrives" at u, for each node u along this path. Final decision class label of DT assigned at x is $\text{DT}(x) = y(v)$ (here x arrives at the leaf v).

2.2 Random forest

Breiman [4] proposed random forests (RaF), $g(y, \lambda)$ as an ensemble model of univariate decision trees [1]. It merges the concepts of bootstrap aggregating and random subspaces to increase randomization of the individual classifiers. This improves the generalization ability while reduces the correlation among individual decision trees. In RaF, y is the input vector and λ is an independent and identically distributed $(i.i.d.)$ random vector, which is used to train the model. From the given data, Bootstrap set is obtained and is used to grow each tree for the RaF ensemble model.

From total n features at each node of the tree classifier, $mtry$ features are selected randomly. This parameter is set to be round($\sqrt{n}$) where n is the dimensionality of a given data set. Then one feature from the $mtry$ features is selected to perform a partition along the feature axis according to some impurity criterion like Gini-Index or Information Gain. Each internal node has a corresponding test function defined as:

$$g(y,\lambda) = \begin{cases} 1 & \text{if } y(\lambda_1) \ < \ \lambda_2 \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\lambda_1 \in \{1, 2, ..., c\}$ where c is the number of features in $mtry$ and threshold is represented by $\lambda_2 \in \mathbb{R}$. Data sample y is directed to the corresponding child node based on the outcome. From the $mtry$ features, one feature is selected to split the data based on an impurity criterion. The parameter λ of the test function $g(y, \lambda)$ is based on this single feature. In the ensemble, a vote is given by each tree D_i for y. Established on the majority count of votes, the final classification is made.

2.3 SVM-based binary classifiers

Pattern classification is the process of estimation of the classifier function, which discovers the underlying pattern in the data and discriminates the data belonging to two or more class distributions. It explores the patterns from the training data and possesses generalization ability, i.e. providing satisfactory accuracy while classifying unseen data. In this section, four SVM-based binary classifiers are briefly discussed, which are used as based classifiers in our proposed oblique random forest model. The classifier is learned from 'training data', it's parameters are tuned with 'validation data', and the performance is evaluated by classifying the unseen 'test data'. For instance, let the binary classification problem patterns belonging to positive and negative classes have representative matrices A and B respectively, and the patterns in these classes be represented by m_1 and m_2 with $m = m_1 + m_2$. Hence, the order of matrices A and B are $(m_1 \times n)$ and $(m_2 \times n)$ respectively, where n represents the dimension of feature space and $A_i \quad (i = 1, 2, ..., m_1)$ is the row vector in n-dimensional real space $\mathbb{R}^n$, that represents feature vector of a data sample. The labels y_i for positive and negative classes are given by $+1$ and -1, respectively.

2.3.1 Twin support vector machine (TWSVM)

TWSVM [37] is a supervised learning tool that classifies data by generating two nonparallel hyperplanes which are proximal to their respective classes and at least unit distance away from the patterns of other class. TWSVM solves a pair of quadratic programming problems (QPPs) and is based on empirical risk minimization (ERM) principle. The binary classifier TWSVM [37, 38] determines two nonparallel hyperplanes by solving two related SVM-type problems, each of which has fewer constraints than those in a conventional SVM. The hyperplanes are given by

$$x^T w_1 + b_1 = 0 \qquad \text{and} \qquad x^T w_2 + b_2 = 0, \tag{2}$$

where $w_1,\ b_1,\ w_2,\ b_2$ are the parameters of normals to the two hyperplanes, referred as positive and negative hyperplanes. The proximal hyperplanes are obtained by solving the following pair of QPPs.

TWSVM1:

$$\min_{w_1,b_1,\xi_2} \ \frac{1}{2}\|Aw_1 + e_1b_1\|_2^2 + c_1e_2^T\xi_2$$
$$\text{subject to } \ -(Bw_1 + e_2b_1) + \xi_2 \geq e_2, \quad \xi_2 \geq 0. \tag{3}$$

TWSVM2:

$$\min_{w_2,b_2,\xi_1} \ \frac{1}{2}\|Bw_2 + e_2b_2\|_2^2 + c_2e_1^T\xi_1$$
$$\text{subject to } \ (Aw_2 + e_1b_2) + \xi_1 \geq e_1, \quad \xi_1 \geq 0. \tag{4}$$

Here, $c_1\ (or\ c_2) > 0$ is a trade-off factor between error vector ξ_2 (or ξ_1) due to misclassified negative (or positive) class patterns and distance of hyperplane from positive (or negative) class; e_1, e_2 are vectors of ones of appropriate dimensions and $\|.\|$ represents L_2 norm. The first term in the objective function of (3) or (4) is the sum of squared distances of the hyperplane to the data patterns of its own class. Thus, minimizing this term tends to keep the hyperplane closer to the patterns of one class and the constraints require the hyperplane to be at least unit distance away from the patterns of other class. Since this constraint of unit distance separability cannot be always satisfied; so, TWSVM is formulated as a soft-margin classifier and a certain amount of error is allowed. If the hyperplane is less than unit distance away from data patterns of other class, then the error variables ξ_1 and ξ_2 measure the amount of violation. The objective function minimizes L_1-norm of error variables to reduce misclassification. The solution of the problems (3) and (4) can be obtained indirectly by solving their Lagrangian functions and using Karush-Kuhn-Tucker (KKT) conditions [39].

As we obtain the solutions (w_1, b_1) and (w_2, b_2) of the problems (3) and (4) respectively, a new data sample $x \in \mathbb{R}^n$ is assigned to class $r\ (r = 1, 2)$, depending on which of the two planes given by (2) it lies closer to, i.e.

$$r = arg\left(\min_{l=1,2} \frac{|x^T w_l + b_l|}{\|w_l\|_2}\right), \tag{5}$$

where $|.|$ is the perpendicular distance of point x from the plane $x^T w_l + b_l = 0,\ l = 1, 2$. The label assigned to the test data is given as $y = \begin{cases} +1\ (r = 1) \\ -1\ (r = 2) \end{cases}$.

2.4 Improvements on ν-Twin Support Vector Machine (Iν-TWSVM)

Improvements on ν-twin support vector machine solves a smaller-sized QPP or a unimodal function as the first problem and a UMP as the second problem. This is in contrast to ν-TWSVM or any other TWSVM-based classifier, which solve a pair of identical QPPs.

Iν-TWSVM (linear classifier)

The first problem of Iν-TWSVM is formulated as a QPP and is given by:

Iν-TWSVM1:

$$\begin{aligned} \min_{w_1,b_1,\rho,\xi} \quad & \frac{1}{2}\|Aw_1+e_1b_1\|_2^2+\frac{c_1}{2}(w_1^Tw_1+b_1^2)+c_2e_2^T\xi-\nu\rho \\ \text{subject to} \quad & -(Bw_1+e_2b_1) \geq \rho e_2-\xi, \\ & \xi \geq 0, \\ & \rho \geq 0, \end{aligned} \tag{6}$$

where $\|.\|_2$ is L_2-norm.

The QPP in (6) determines the hyperplane which is closer to data points of positive class (which is represented by A) and at least ρ distance away from the data points of negative class (represented by B). The first term in the objective function is similar to TWSVM and ν-TWSVM and thus, Iν-TWSVM follows the empirical risk minimization (ERM) principle. Further, Iν-TWSVM also takes into consideration the principle of SRM [40] to improve the generalization ability, by introducing a term $(w_1^Tw_1+b_1^2)$ in the objective function. This regularization term maximizes the margin between two classes with respect to the plane $w_1^Tx+b_1=0$. Here, the margin between two classes can be expressed as distance bounded by the plane proximal to the positive class $(w_1^Tx+b_1=0)$ and the bounding plane $(w_1^Tx+b_1=-\rho)$. This distance is $\rho/\|w_1\|_2$ and is the margin between two classes with respect to plane $w_1^Tx+b_1=0$. The extra term b_1^2 is motivated by proximal support vector machine [41]. Let $X=[x^T,1]^T$, $W_1=[w_1,b_1]$ then the proximal plane in $\mathbb{R}^{n+1}$ is $X^TW_1=-\rho$ and the margin is $\rho/\|W_1\|_2$, i.e. $\rho/\sqrt{(\|w_1\|_2^2+b_1^2)}$. Thus, the distance between two classes is maximized with respect to orientation (w_1) and relative location of the plane (b_1) from the origin.

The second hyperplane of Iν-TWSVM is obtained by solving a UMP and its formulation is given by

Iν-TWSVM2:

$$\begin{aligned} \min_{w_2,b_2} \frac{1}{2}\|Bw_2+e_2b_2\|_2^2+\frac{c_3}{2}(w_2^Tw_2+b_2^2) \\ -\frac{c_4}{2}\|[M_A \quad 1][w_2,b_2]^T-\rho\|_2^2. \end{aligned} \tag{7}$$

In (7), we find the hyperplane which is proximal to the data points of negative class and at the same time, the negative hyperplane should be ρ distance away from the representative (i.e. mean) of positive class. Instead of maximizing the distance of all the data points of positive class from the negative class hyperplane (as considered in TWSVM2), we are trying to maximize its distance from the mean of positive class. Here, M_A is the mean of matrix A with dimension $(1\times n)$ and is regarded as the representative of positive class. Hence, the size of the problem is reduced as we are dealing with unconstrained optimization problem. The positive constants c_3 and c_4 associate weights with the corresponding terms. For more details, please refer to [27].

2.4.1 Multi-surface proximal support vector machine (MPSVM)

MPSVM [22], learns two non-parallel hyperplanes by computing two generalized eigenvalue problems (GEPs), $Pz=\mu Qz$. Here, P and Q are symmetric positive semi-definite matrices. Each hyperplane is decided by choosing the eigenvector of the smallest eigenvalue of each GEP. Matrices A and B represent data points of two classes (referred to as positive and negative classes) respectively, with m_1 and m_2 data points. Hence, their dimensions being $(m_1\times n)$ and $(m_2\times n)$, respectively. Two non-parallel hyperplanes are computed by MPSVM, as determined by TWSVM in (2). MPSVM solves the following optimization problem:

$$\underset{w,b\neq 0}{Min} \ \frac{\|Aw-eb\|_2^2/\|[w,b]^T\|_2^2}{\|Bw-eb\|_2^2/\|[w,b]^T\|_2^2}. \tag{8}$$

Here, e is the one's vector with proper dimension and $\|.\|$ represents the L2 norm, assuming $(w,b)\neq 0$ [22]. The objective function is simplified and regularized to get a Generalized Eigenvalue Problem, which is solved to get the solution.

2.4.2 Regularized multi-surface proximal support vector machine (RegMPSVM)

The formulation of MPSVM is altered by Guarracino et al. [22] to generate both the hyperplanes by solving a single generalized eigenvalue problem (GEP). The GEP $Pz=\mu Qz$ is transformed as $P^*z=\mu Q^*z$, where

$$P^*=\mathcal{T}_1P-\delta_1Q, \quad Q^*=\mathcal{T}_2Q-\delta_2P. \tag{9}$$

Where the parameters $\mathcal{T}_1$, $\mathcal{T}_2$, δ_1, and δ_2 are selected as a singular matrix Ω,

$$\Omega=\begin{bmatrix}\mathcal{T}_2 & \delta_1 \\ \delta_2 & \mathcal{T}_1\end{bmatrix}. \tag{10}$$

The GEP $P^*z=\mu Q^*z$ would generate the same eigenvectors as that of $Pz=\mu Qz$ [42].

2.5 Multi-category extension of binary classifiers

As binary classifiers, SVM, TWSVM and MPSVM have been widely studied, and researchers have been exploring techniques to extend them as multi-category classifiers. Two approaches have been widely used to manage multi-category data samples. One approach is to establish and merge several binary classifiers that consider a part of the data and for the second, the entire data samples are considered in one single optimization problem [43]. The latter advancement is more expensive in terms of computational complexity as the single problem generally consists of a lot of variables to be considered and has applications limited to smaller datasets only.

By integrating the results of several binary classifiers, multi-category SVMs are implemented. Two such approaches are one-against-all (OAA) and one-against-one (OAO) support vector machine [43]. A series of binary classifiers are implemented by OAA-SVM where each class is separated from the remaining classes by a classifier. But, due to the huge difference in the number of pattern instances, this approach leads to an unbalanced classification. For a K-class classification task, OAA-SVM generates K binary classifiers and needs an equivalent number of binary SVM comparisons for each test data samples. In the manifestation of OAO-SVM, using a pair of classifiers, one at a time, their binary SVM classifiers are generated. So, it computes a maximum of $\frac{K\times(K-1)}{2}$ binary SVM classifiers, which leads to an increase in computational complexity.

Similarly, directed acyclic graph SVMs (DAG-SVMs) have their training phase similar to OAO-SVMs, i.e. it determines $\frac{K\times(K-1)}{2}$ binary SVM classifiers, nonetheless, it's testing phase is different. It employs a rooted binary directed acyclic graph consisting of $\frac{K\times(K-1)}{2}$ internal nodes and K leaves during its testing phase. Fuzzy linear proximal support vector machine is proposed for multi-category data classification by Jayadeva et al. [44]. Lei et al. proposed half-against-half (HAH) multi-category SVM [45]. HAH is generated by recursively partitioning the training data samples of K classes into two subsets of classes. It results in a decision tree where each node is a binary SVM classifier. Shao et al. proposed a decision tree twin support vector machine (DTTSVM) for multi-category classification [46], by generating binary-based classifier on the best splitting principle. These multi-category classification approaches, which were initially proposed for SVM, can also be extended for TWSVM and MPSVM framework. For example, Xie et al. extended TWSVM for multi-category classification [47] employing OAA approach.

3 Proposed work

Ensemble classifiers, also called as multiple classifier systems, have gained popularity around most of the research areas in pattern recognition, machine learning [8] and computer vision [7]. Ensemble classifiers are based on the perturb and combine strategy [6] that results in significant improvement of the classifiers than using a single classifier. This greatly reduces the variance of classifiers [6, 48], [49]. The classification error can be broken down into two components—bias and variance, as per bias and variance decomposition theory [6], [50]. Bias refers to the error introduced by the difference between the learning algorithm's result than that of the target. Variance refers to the amount by which learning algorithm's estimate would change around for different training sets.

The issue with decision tree induction is that it has low bias and high variance which leads to $over\text{-}fitting$. Two common methods employed to regularize DTs are pruning and voting ensembles [51, 52]. Pruning is a method to decrease the size of a predefined decision tree by swapping internal nodes in the trees with leaf nodes. This decreases the complexity of the tree by reducing its size and expectantly discards the nodes contributing to the noise in the data. However, we prefer voting ensembles, whereby multiple trees are generated, and a forest is made by applying DT induction algorithms. The essential approach to putting together such a decision forest (DF) is to use bootstrap as aggregation, normally referred to as the bagging algorithm [18]. In Bagging, the initial learning information is sampled various times with repetition, generating "bags" of samples to be accustomed to train classification models (bagged predictors). It is a common practice in bagging-based ensembles, to use equal weights within the voting process; for a binary model, $DF(x) = sgn\left(\sum_{i=1}^{k} DT_i(x)\right)$.

In a decision tree, each hyperplane at a non-leaf node splits the data to facilitate further splitting in the child nodes. Primarily, decision tree induction algorithms splits a node, by calculating the score based on impurity criterion and selecting the node with minimum score. The essence of an impurity criterion is to estimate the skewness of the distribution of different classes in the data samples passed onto the nodes down the tree. Those distributions that are nearly uniform gets the least score, and those distributions where the number of patterns of one class is more significant in strength than the rest gets the highest score. But the drawback with all impurity criterion, reported by [23], is that they only consider the distribution of different classes on each side of the hyperplane. Thus, it does not take into consideration the geometric structure of the class regions. If one swaps the label of the data without changing the relevant features of each class on either side of a hyperplane, the impurity criterion score won't change. In lieu of just considering the label information, the geometric structure assesses the internal data structure, which estimates the distance of the data samples to the decision hyperplane. Therefore, the hyperplane will change, whenever any relevant feature changes. Algorithm 2 list down the steps for implementing a random forest.

Algorithm 2 Random Forest Algorithm

```
Learning Step :
Inputs :
$A = M \times m$, contains the training data sample points, here, $M$ is the number of training instances, and, $m$ are the features of each data sample.
$B = M \times 1$, are the labels of the training data samples.
$L$ is the ensemble size, i.e., decision trees count in the forest.
$T_i$ corresponds to each random tree; a random forest contains, $i = 1, \ldots, L$.
$mtry$ is the randomly selected number of the features to split at each non-leaf node.
$minleaf$ is the minimum number of samples in an impure node.
1: Create the bootstrapped training set with the replacement for $T_i$ from $M$ given training data samples.
2: Compute the best split considering the $mtry$ randomly selected features at each non-leaf node.
3: Continue with step 2 until $T_i$ is fully generated, resulting in reaching to a pure node or the $minleaf$ criteria being violated.
Testing Step :
A given test data sample is traversed down each tree, and then each of them cast a vote on the classified class label on the given data sample. The most voted class label is assigned to the test data sample.
```

3.1 Oblique random forest

An oblique random forest (ObRaF) $f(x, \theta)$, is an ensemble algorithm developed with multivariate or oblique decision trees [15]. The input to ObRaF an independent and identically distributed $(i.i.d.)$ random vector θ and training set, where x is an input vector. The ObRaF ensemble employs bootstrap set of training set to train each tree classifier. In multivariate decision trees, θ is based on the linear combination of all or some of the features. As the decision boundary can align in any direction to the axes, the trees possessing such *oblique* hyperplanes are also known as *oblique trees* [1]. Hence, (1) can be written as:

$$f(x, \theta) = \begin{cases} 1 & \text{if } \sum_{i=1}^{q} w_i x_i < \theta_2 \\ 0 & \text{if otherwise} \end{cases} \tag{11}$$

where, weight coefficient is represented by w_i for each feature in data.

In our work, we propose novel oblique decision tree ensembles with binary and ternary decision structure models (BDS and TDS), employing oblique decision trees via different base classifiers. The binary classifiers used for this work are Twin SVM (TWSVM), Improvements on ν-TWSVM (Iν-TWSVM), MPSVM and regularized MPSVM. We propose a different approach to choose the final hyperplane to split the data at the non-leaf node while optimizing an impurity criterion, than just choosing one of the angle bisectors.

The non-parallel hyperplanes generated by MPSVM, RegMPSVM, TWSVM or Iν-TWSVM capture the geometric data property as per their definitions, which assists greatly in discriminating the classes at non-leaf nodes. But MPSVM dwells into GEP, and thus, if the matrix is ill-conditioned, it could lead to poor classification accuracy. On the other hand, TWSVM and Iν-TWSVM formulate convex optimization problems and are based on ERM principle. So, they are more robust to noise and have better generalization ability.

In our proposed approach, four binary classifiers are separately used to generate oblique splits at each node of the Decision Trees. Each of the above mentioned classifiers is originally proposed as a binary classifier. These classifiers are employed with decision trees for both binary and multi-category classification. The conviction behind using a binary classifier is to generate two non-parallel proximal hyperplanes C1 and C2 for two groups of data D1 and D2, such that C1 (or C2) is closest to the data points of D1 (or D2), and furthest from the data points of D2 (or D1). The hyperplanes are generated by solving two generalized eigenvalue problems for MPSVM and RegMPSVM and QPPs for TWSVM and Iν-TWSVM. The angle bisector of H1 and H2 is used as the hyperplane to split the data points in each node while optimizing an impurity criterion. In multi-category classification, two hyper-classes are made by dividing all the classes at each node based on the Bhattacharyya distance [53] implementing a Binary Decision Structure Model (BDS) or Ternary Decision Structure Model (TDS).

3.2 Bhattacharyya distance

Bhattacharyya distance (B-distance) is a metric that is used to measure the similarity of two discrete or continuous distributions. It derives a measure of the distance between two populations defined in any way and have the same number of variates in which a one-to-one correspondence can be established [53].

For multivariate normal distributions $p_i = \mathbb{N}(\mu_i, \sigma_i)$, B-distance is defined as

$$B(w_1, w_2) = \frac{1}{8}(\mu_2 - \mu_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\mu_2 - \mu_1) + \frac{1}{2} ln \frac{|(\Sigma_1 + \Sigma_2)/2|}{\sqrt{|\Sigma_1||\Sigma_2|}}. \tag{12}$$

where μ_i and $\sum_i$ are the means and covariances of the distributions.

The first term here relates to the location of the two distributions, where the second term helps to account for differences in shape or direction of the two populations.

3.3 Oblique decision tree via binary classifiers implemented through TDS and BDS

The classifiers TWSVM, Iν-TWSVM, MPSVM, and RegMPSVM are used to generate hyperplanes to split the data at each non-leaf node of the Decision tree. Since all of these are binary classifiers, so there is a need to address the issue of handling the multi-category problems.

In [23], the multi-category problem is transformed into a binary problem by selecting the majority class out and the remaining classes as two hyper-classes, but this fails to capture the geometric data structures. In [21], all classes are separated into two hyper-classes based on their underlying distributions employing Bhattacharyya distance [53]. The approach for binary decision structure model (BDS) partitions the data into two groups and is explained in Algorithm 3. Our proposed ternary decision structure model (TDS) is developed on the lines of [26] and employs Bhattacharyya distance to get two focused classes (the classes whose patterns are significantly distinguishable) and one unambiguous (the remaining classes) hyper-class group. The representative class hyperplanes are further generated using via linear classifiers. TDS is explained by Algorithm 4.

Algorithm 3 Binary Decision Structure Model (BDS)

```
Given: :
$A = M \times m$, contains the training data sample points, here, $M$ is the number of training instances, and, $m$ are the features of each data sample.
$B = M \times 1$, are the labels of the training data samples.
$\{y_1, ..., y_c\}$ being set of class labels present in the data.
Result: :
$Y_{+1}$ and $Y_{-1}$ represent the two groups (hyper-classes).
1: for $i = 1, ..., c$
2: compute the Bhattacharyya distance between the pair of class $y_i$ and
   $y_j$, where, $j = i + 1, ..., c$.
3: Obtain the class pair with the largest Bhattacharyya distance, assign
   them names $y_{+1}$ and $y_{-1}$, and allot them to group $Y_{+1}$ and $Y_{-1}$
   respectively.
4: With the remaining classes, for instance $y_j$ is allotted to the group
   $Y_{+1}$ if $B(y_{+1}, y_j) < B(y_{-1}, y_j)$, else it is allotted to group $Y_{-1}$.
BDS is used at every non-leaf node to handle multi-category classification problem.
```

For class separability among two normal classes w_1 and w_2, $\mathbb{N}(\mu_1, \sum_1)$ and $\mathbb{N}(\mu_2, \sum_2)$, Bhattacharyya distance provides a good estimate. As reported in [25], we consider multivariate Gaussian distributions. Gaussian distribution has the maximum uncertainty amongst all the distributions having a given mean and variance. Nonetheless, it is mostly an appropriate model in many situations and the sum of a large number of independent random distributions obeys Gaussian distribution.

Algorithm 4 Ternary Decision Structure Model (TDS)

```
Given: :
$A = M \times m$, contains the training data sample points, here, $M$ is the number of training instances, and, $m$ are the features of each data sample.
$B = M \times 1$, are the labels of the training data samples.
$Y_{+1}$ and $Y_{-1}$ being set of class labels present in the data.
Result: :
$Y_{+1}$, $Y_{-1}$, and $Y_0$ represent the two group class, and ambiguous class group (hyper-class).
1: for $i = 1, ..., c$
2: compute the Bhattacharyya distance between the pair of class $y_i$ and
   $y_j$, where, $j = i + 1, ..., c$.
3: Obtain the class pair with the largest Bhattacharyya distance, allot
   them to group $Y_{+1}$, and $Y_{-1}$ respectively.
4: Remaining classes are allotted to the $Y_0$ group.
TDS is used at every non-leaf node to handle multi-category classification problem.
```

Using the ternary decision structure (TDS) model, oblique random forest are implemented via TWSVM, Iν-TWSVM, MPSVM, and RegMPSVM (called as TDS-TWRaF, TDS-IνTWRaF, TDS-MPRaF, TDS-RegMPRaF). For each of these models, four non-parallel hyperplanes are generated, two for the focused classes, and two for the ambiguous hyper-class, with respect to each focused class. While optimizing the impurity criterion, i.e., the Gini index,

$$\text{Gini}(t) = \frac{{n_t}^l}{n_t}\left[1 - \sum_i^c \left(\frac{{n_{w_i}}^l}{{n_t}^l}\right)^2\right] + \frac{{n_t}^m}{n_t}\left[1 - \sum_i^c \left(\frac{{n_{w_i}}^m}{{n_t}^m}\right)^2\right] + \frac{{n_t}^r}{n_t}\left[1 - \sum_i^c \left(\frac{{n_{w_i}}^r}{{n_t}^r}\right)^2\right] \tag{13}$$

we choose either for the bisector of the ambiguous hyper-class hyperplanes or one of its hyperplane, employing the BDS hyperplane selection criteria, while freezing the hyperplanes of the focused classes. Then these three hyperplanes represent the three child nodes of the TDS, and data samples are assigned to these nodes depending on their distance from these three hyperplanes, resulting in three child nodes, these hyperplanes are generated by TWSVM, MPSVM, RegMPSVM. These final three hyperplanes represent the corresponding three child nodes thresholds.

In this work, we also propose the implementation of oblique random forest via TWSVM, Iν-TWSVM, MPSVM and RegMPSVM (called as BDS-TWRaF, BDS-IνTWRaF, BDS-MPRaF and BDS-RegMPRaF respectively) using the binary decision structure (BDS) model. For each of these models, two non-parallel hyperplanes are generated proximal to the two hyper-classes (or classes) at each node of BDS. While optimizing impurity criterion, i.e. the Gini index, we choose between the angle bisector or the hyperplanes, as on some occasions the hyperplanes capture more accurate patterns in the data than its bisector, generated by the binary classifiers. If the hyperplanes are selected, then they represent the two child nodes, and data samples are assigned to these nodes depending on their distance from these hyperplanes.

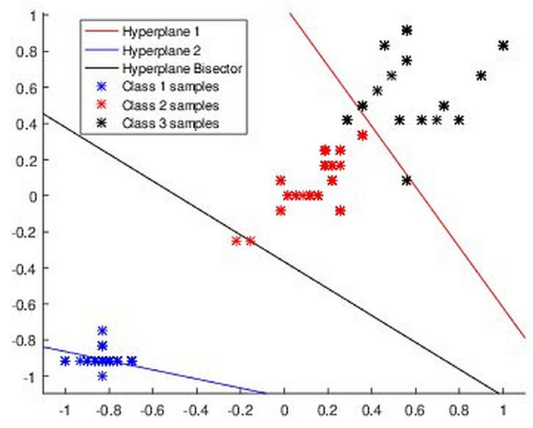
Gini index is given as

$$\text{Gini}(t) = \frac{n_t{}^l}{n_t}\left[1 - \sum_i^c \left(\frac{n_{w_i}{}^l}{n_t{}^l}\right)^2\right] + \frac{n_t{}^r}{n_t}\left[1 - \sum_i^c \left(\frac{n_{w_i}{}^r}{n_t{}^r}\right)^2\right] \quad (14)$$

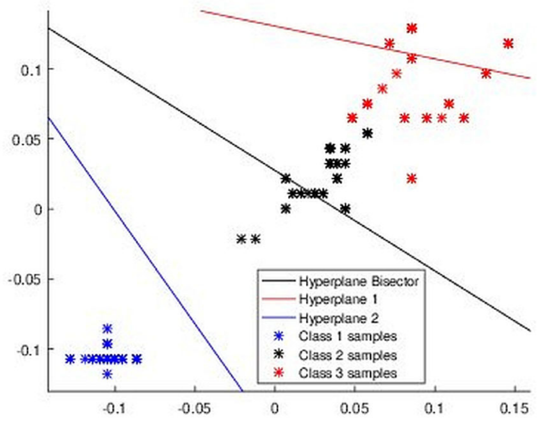
where n_t is the number of data samples reaching a particular node, $n_t{}^l$, $n_t{}^r$ is the number of data samples that reach the left and right child nodes of the current node, respectively. Also, $n_{w_i}{}^l$, $n_{w_i}{}^r$ represents the number of samples of class w_i in the left and right nodes, respectively.

The decision tree implements ternary decision structure (TDS) and binary decision structure (BDS) using TWSVM, Iν-TWSVM, MPSVM, and, RegMPSVM from root node to the leaf node to perform splits. To graphically illustrate the difference between hyperplanes generated by these classifiers, hyperplane structure is shown in Fig. 1 for a standard multi-category dataset, Iris dataset. BDS-TWSVM and BDS-MPSVM represents Binary Decision Structure multi-category approach with TWSVM and MPSVM classifiers respectively.

Similar hyperplanes are generated for these classifiers using TDS and are termed as TDS-TWSVM and TDS-MPSVM. The hyperplanes generated by TDS via TWSVM and MPSVM are illustrated in Figs. 2. (Note: The hyperplanes generated by MPSVM and RegMPSVM are almost identical. Also, the hyperplanes generated by TWSVM and IνTWSVM are quite similar. So, they are not included in the figure.)



(a) Hyperplanes generated by BDS-TWSVM



(b) Hyperplanes generated by BDS-MPSVM

Fig. 1 Hyperplanes generated by different classifiers on IRIS dataset with binary decision structure. The three classes of Iris dataset are shown with samples of different colors. The solid lines passing through the samples represent the hyperplanes

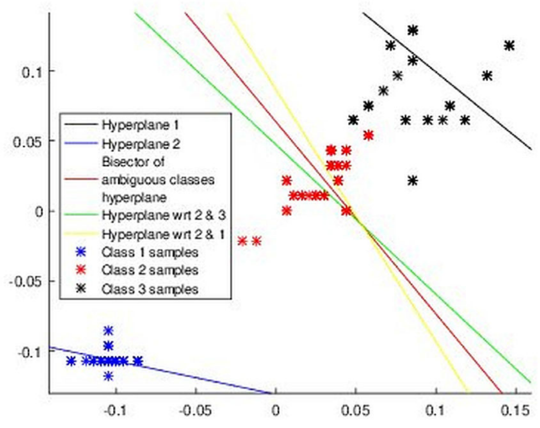
3.4 Complexity analysis

This work is an attempt to explore the strengths of both TDS and BDS for implementing oblique random forests. TDS based model performs better than BDS during testing phase. For a dataset with K classes, the height of tree generated by BDS is of the order $log_2 K$, where TDS generates the tree model of height $log_3 K$. Hence, TDS is more efficient during the testing phase.

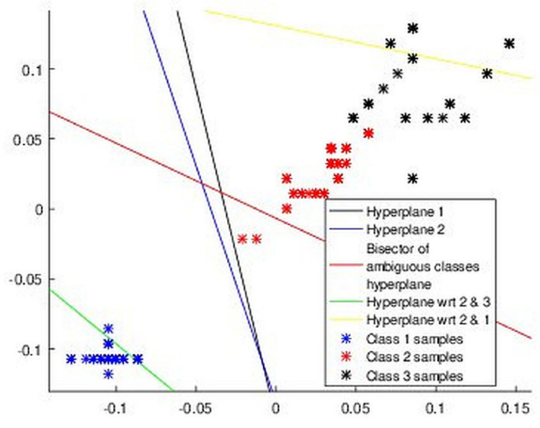
The TDS algorithm's strength is that it requires less classifier comparisons for evaluations than other cutting-edge multi-class techniques. The BDS technique divides data recursively into two halves and creates a binary tree of classifiers. For a K-class issue, BDS determines $2 * (K - 1)$ classifiers. BDS requires at most $\lceil log_2 K \rceil$ binary classifier evaluations for testing. TDS, on the other hand, divides the data at each node into at most three groups and creates a model with a height of $\lceil log_3 K \rceil$. New test samples can also be tested using $\lceil log_3 K \rceil$ comparisons, which is faster than BDS testing time. As the classes of the parent node are divided into three groups, the order of QPP drops to one-third of the parent QPP with each level of TDS for a balanced decision structure.

4 Experimental results

In order to prove the efficacy of our models, we compare the proposed frameworks on real-world benchmark classification datasets from various research fields. The datasets analysed during the current study are available in the UCI repository [54]. For the base classifiers, Decision Trees are used in the ensembles. One parameter, $minleaf$, needs to be tuned for the number of samples reaching an impure node, which is set to one by convention. The ensemble size L for



(a) Hyperplanes generated by TWSVM-TDS



(b) Hyperplanes generated by MPSVM-TDS

Fig. 2 Hyperplanes generated by different classifiers on IRIS dataset with ternary decision structure. The three classes of Iris dataset are shown with samples of different colors. The solid lines passing through the samples represent the hyperplanes

the ensemble methods can be regarded as a parameter. Generally, an increase in L does not decrease the performance of the RaF; we perform all our experiments with $L = 100$. For RaF, another parameter exists, $mtry$, representing the number of randomly selected features which controls the randomness injected in the forest. It is set to round($\sqrt{n}$), n being the dimension of the data. For all the experiments, the Tikhonov regularization term δ is assigned the value 0.01. It may get some arguments, that the term should be optimized on the validation dataset or the out-of-bag data. Nonetheless, as the number of nodes present in the ensemble is considerably large, typically when a large number of data samples are present, it becomes possibly computational expensive. In [21], optimization for parameter *'delta'* was done, but no significant improvement was observed. Ensemble classifiers characteristics make them robust in terms of this parameter. For twin support vector machine (TWSVM), parameter C is set to three values, i.e., $0.1, 0.05, 0.01$, which is the trade-off parameter between the proximity of the hyperplane to its respective class samples and the distance to the other hyperplane. The simulations of both the approaches with four algorithms on all datasets are carried out in MATLAB 2014b under Microsoft Windows environment on a machine with 3.40 GHz CPU and 16 GB RAM.

4.1 Dataset description

For all the simulations, we have used binary as well as multi-category UCI datasets. The summary of these benchmark datasets, taken from various domains, is listed in Table 1. This table include the dataset name along with the number of classes, samples and features.

4.2 Results for BDS approaches

In order to have a fair comparison of the performance of various oblique random forest approaches via MPSVM, RegMPSVM, TWSVM and Iν-TWSVM (referred as MPRaf, RegMPRaF, TWRaF and IνTWRaF respectively), we have implemented them in a uniform way. All the experiments are conducted using fivefold cross validation. Table 2 enlists the results obtained by BDS approach for all the four types of random forests, mentioned above. In the table, mean of classification accuracy for fivefold is reported with standard deviation as $mean \pm std$. For MPRaF, the results are given for all three regularization methods, i.e. Tikhonov regularization (*), Parallel axis regularization (**) and Null space regularization (***). These results are taken from [21] and are reported 'as it is' without standard deviation. For ObRaf using RegMPSVM, the implementations are done for Tikhonov regularization (*) and Parallel axis regularization (**). As reported in [21], the NULL space approach for regularization can be unstable depending on the data. This problem can be solved for a single DT by using decision tree pruning, where the most unstable node can be eliminated from by pruning. However, this approach would not work for decision tree ensemble, as this would increase the computational complexity with negligible improvement. So, we have not implemented RegMPRaF with null space regularization. TWRaF and IνTWRaF are implemented with Tikhonov regularization for this work.

The table demonstrates that all the implementations of Oblique Random forest using BDS are able to achieve satisfactory results for all the benchmark datasets. The classification accuracy obtained by averaging over 5-folds is comparable for all the methods. Although, it is observed that IνTWRaF achieves the highest average classification accuracy i.e. 87.66%, which is followed by TWRaF at 87.24%.

Table 1 Characteristics of the experimental datasets: number of classes, samples and features

Datasets	Classes, samples, features	Datasets	Classes, samples, features
Australian	2,690,14	Page block	5,5473,10
Balance scale	3,625,4	Parkinsons	2,195,22
Banknote	2,1372,4	Pima-diabetes	2,768,8
Biodeg	2,1055,41	Planning Relax	2,182,12
Blood transfusion	2,745,4	Seeds	3,210,7
Breast cancer	2,699,9	Segment	7,2310,19
Breast tissue	6,106,9	Sonar	2,208,60
Climate model	2,540,18	Spambase	2,4601,57
Ecoli	8,336,7	Teaching assistant	3,151,5
Fertility	2,100,9	Twonorm	2,7400,20
Glass	7,214,9	User Knowledge	4,258, 5
Harberman	2,306, 3	Vehicle	4,846, 18
Heart	2,270,13	Vertebral-2C	2,310,6
Hepatitis	2,155,19	Vertebral-3C	3,310,6
Ionosphere	2,351,34	Waveform1	3,5000,21
Iris	3,150,4	Waveform2	3,5000,40
Lymphography	4,296,18	Wine	3,178,13
Mam-masses	2,961,5	Wine quality(red)	6,1599,11
Ozone	2,2536,72	Yale	15,165,1024

For MPRaF, the best result is obtained with Tikhinov regularization whereas null-space gives the minimum classification accuracy. Similar trend is observed in the two implementations of RegMPRaF. In oblique decision trees implemented with classifiers like MPSVM and RegMPSVM, matrices are semi-positive definite and hence different regularization methods are required. However, no explicit regularization techniques are required for the primal problems of TWSVM and IνTWSVM as the matrices are positive definite. Still, Tikhonov regularization is used with TWSVM and IνTWSVM to avoid overfitting and this significantly improves the generalization ability of the algorithm. Since the classification accuracy for twin bounded random forest (TBRaF) [55] is not known for all the datasets, so average accuracy is not calculated for it.

4.2.1 Statistical analysis: friedman test

The Friedman test [56] is used to statistically compare the efficacy of several classifiers on different datasets. This test can compare more than two related data samples. It is a non-parametric test that makes no assumptions about the data distribution. For each dataset, the algorithms are ranked separately. The information is organized in a table with n_1 rows and n_2 columns. It scores the algorithms independently for each data set, with the top performing algorithm receiving a rank of 1, the second best receiving a rank of 2, and so on. In the event of a tie, the average ranks are assigned. The average ranks of algorithms are then compared, $R_j = \frac{1}{N}\Sigma_i r_i^j$, where r_i^j is the rank of the j^{th} algorithm on the i^{th} of N data set. The Friedman test on the classification accuracy of all the classifiers with UCI datasets is shown in Table 3. Based on average classification accuracy, BDS-IνTWRaF is at Rank 1 and is followed by BDS approaches for TWRaF*, RegMPRaF*, RegMPRaF**, MPRaF*, MPRaF** and MPRaF***. The results reiterate that the best results are obtained with Tikhonov Regularisation.

4.3 Results for TDS approaches

In addition to BDS, oblique random forest approaches via MPSVM, RegMPSVM, TWSVM and Iν-TWSVM (referred as MPRaf, RegMPRaF, TWRaF and IνTWRaF respectively) are implemented using ternary decision structure (TDS). The classification accuracy is recorded for benchmark UCI datasets and all the experiments are conducted using 5-fold cross validation. Table 4 demonstrates the results obtained by TDS approach for all the four types of Random Forests. In the table, mean of classification accuracy for 5-folds is reported with standard deviation as $mean \pm std$. For MPRaF, the results are given for all three regularization methods i.e. Tikhonov regularization (*), Parallel axis regularization (**) and Null space regularization (***). For ObRaf using RegMPSVM, the implementations are done for Tikhonov regularization (*) and Parallel axis regularization (**). TWRaF and IνTWRaF are implemented with Tikhonov regularization for this work.

Table 2 Classification accuracy (%) achieved by various obliques random forests implemented using binary decision structure (BDS) model. The proposed models are compared with three variations of MPRaF

	MPRaF implementations [21]			Proposed BDS models using				
Datasets	MPRaF^{*+}	MPRaF**	MPRaF***	TBRaF^{1}	RegMPRaF^{*+}	RegMPRaF^{**+}	TWRaF*	IνTWRaF*
Australian	86.42	86.90	86.25	67.82± 3.50	88.11 ± 0.00	88.26 ± 0.00	94.21 ± 0.03	94.84 ± 0.03
Balance scale	89.02	88.19	89.10	89.79 ± 1.80	89.20 ± 0.03	89.72 ± 0.02	91.36 ± 0.02	91.29 ± 0.02
Banknote	99.91	99.91	99.89	–	100.00 ± 0.00	99.93 ± 0.01	100.00 ± 0.00	100.00 ± 0.00
Biodeg	86.68	86.52	85.78	–	85.50 ± 0.02	86.64 ± 0.02	86.35 ± 0.02	87.87 ± 0.02
Blood transfusion	78.05	77.49	77.50	–	78.68 ± 0.04	78.14 ± 0.01	78.13 ± 0.02	79.74 ± 0.02
Breast cancer	96.72	96.72	96.92	73.36± 4.2	95.42 ± 0.00	95.42 ± 0.00	97.67 ± 0.01	97.67 ± 0.01
Breast tissue	68.87	67.55	60.94	–	72.76 ± 0.09	71.73 ± 0.06	77.46 ± 0.09	77.21 ± 0.09
Climate model	92.06	91.85	91.59	–	92.22 ± 0.01	91.67 ± 0.00	92.66 ± 0.03	92.85 ± 0.03
Ecoli	85.46	84.76	84.40	–	86.97 ± 0.02	82.55 ± 0.03	89.24 ± 0.04	89.19 ± 0.04
Fertility	87.90	88.20	88.00	88.00 ± 0.02	88.00 ± 0.02	88.00 ± 0.01	89.00 ± 0.05	88.00 ± 0.05
Glass	94.21	96.64	59.67	–	74.08 ± 0.08	74.15 ± 0.06	74.53 ± 0.05	74.54 ± 0.05
Harberman	72.39	70.29	71.34	74.05 ± 4.1	74.55 ± 0.05	70.51 ± 0.02	83.83 ± 0.05	84.05 ± 0.05
Heart	83.74	82.96	83.07	82.40±4.4	81.05 ± 0.03	82.72 ± 0.03	83.59 ± 0.06	83.40 ± 0.06
Hepatitis	83.61	83.94	79.48	81.90±3.1	87.62 ± 0.09	88.91 ± 0.05	88.91 ± 0.05	88.90 ± 0.05
Ionosphere	92.68	93.96	89.74	93.34±2.7	91.04 ± 0.01	93.48 ± 0.02	93.96 ± 0.02	93.34 ± 0.02
Iris	97.60	95.67	97.40	94.27±2.4	97.37 ± 0.02	97.97 ± 0.01	98.33 ± 0.01	98.47 ± 0.01
Lymphography	95.37	93.01	89.26	84.19±7.7	92.46 ± 0.07	87.16 ± 0.03	88.18 ± 0.08	94.19 ± 0.08
Mam-masses	82.24	81.93	82.38	82.44±2.3	83.66 ± 0.01	83.45 ± 0.02	82.54 ± 0.04	82.44 ± 0.04
Ozone	97.12	97.10	97.13	–	97.16 ± 0.00	97.16 ± 0.00	97.12 ± 0.00	97.20 ± 0.00
Page block	97.28	97.35	95.37	–	97.30 ± 0.00	97.30 ± 0.00	97.59 ± 0.01	96.47 ± 0.01
Parkinsons	91.23	90.82	86.36	91.08±4.8	89.19 ± 0.03	93.38 ± 0.02	92.82 ± 0.04	91.08 ± 0.04
Pima-diabetes	75.91	75.98	76.77	75.76±2.8	75.34 ± 0.00	75.26 ± 0.02	75.32 ± 0.02	75.98 ± 0.02
Planning Relax	70.49	70.66	71.21	71.68±6.1	73.62 ± 0.02	73.62 ± 0.02	73.52 ± 0.06	73.68 ± 0.06

Table 2 continued

Datasets	MPRaF implementations [21]			Proposed BDS models using				
	MPRaF*+	MPRaF**	MPRaF***	TBRaF^1	RegMPRaF*+	RegMPRaF**+	TWRaF*	IνTWRaF*
Seeds	94.19	93.14	92.43	94.58±3.5	94.78 ± 0.01	94.80 ± 0.03	95.23 ± 0.02	94.58 ± 0.02
Segment	94.42	95.51	89.55	–	98.39 ± 0.00	94.64 ± 0.00	98.57 ± 0.00	98.97 ± 0.00
Sonar	82.12	78.99	81.44	–	86.21 ± 0.03	84.13 ± 0.05	85.25 ± 0.01	86.06 ± 0.01
Spambase	93.68	94.68	93.77	–	94.35 ± 0.00	91.72 ± 0.00	94.95 ± 0.00	95.13 ± 0.00
Teaching assistant	55.10	55.56	53.38	–	57.57 ± 0.03	57.77 ± 0.05	58.72 ± 0.10	58.21 ± 0.10
Twonorm	97.47	97.51	97.45	–	97.77 ± 0.00	97.73 ± 0.00	97.80 ± 0.00	97.55 ± 0.00
User Knowledge	91.20	91.82	90.15	–	95.28 ± 0.00	94.80 ± 0.01	95.78 ± 0.05	96.58 ± 0.05
Vehicle	76.30	76.34	70.76	–	77.66 ± 0.02	77.89 ± 0.02	78.61 ± 0.02	78.61 ± 0.02
Vertebral-2C	84.61	84.58	84.10	85.36±3.3	86.75 ± 0.05	84.42 ± 0.00	78.38 ± 0.12	86.76 ± 0.12
Vertebral-3C	83.61	83.77	82.48	–	85.47 ± 0.03	84.81 ± 0.03	84.19 ± 0.03	85.51 ± 0.03
Waveform1	85.70	85.50	85.70	–	85.64 ± 0.00	85.46 ± 0.00	85.88 ± 0.01	85.88 ± 0.01
Waveform2	85.44	85.32	85.34	–	85.32 ± 0.01	85.34 ± 0.01	85.59 ± 0.02	85.59 ± 0.02
Wine	97.64	98.26	96.24	97.75±1.1	98.89 ± 0.01	97.43 ± 0.01	99.43 ± 0.01	98.89 ± 0.01
Wine quality (red)	67.22	67.10	59.42	–	66.67 ± 0.02	66.67 ± 0.02	67.60 ± 0.01	67.60 ± 0.01
Yale	80.00	64.56	78.22	–	82.78 ± 0.01	80.80 ± 0.03	82.72 ± 0.10	82.72 ± 0.10
Average	86.15	85.55	83.68	–	86.44 ± 0.02	85.94 ± 0.02	87.24 ± 0.03	**87.66 ± 0.03**

*: Tikhonov regularization
**: Parallel axis regularization
***: Null space regularization
+: These results are reported from [21]
1: These results are reported from [55]

Table 3 Friedman test ranks for oblique random forests implemented with binary decision structure using different classifiers

Datasets	MPRaF*	MPRaF**	MPRaF***	RegMPRaF*	RegMPRaF**	TWRaF*	IνTWRaF*
Australian	6	5	7	4	3	2	1
Balance scale	6	7	5	4	3	1	2
Banknote	5	5	7	1	4	1	1
Biodeg	2	4	6	7	3	5	1
Blood transfusion	5	7	6	2	3	4	1
Breast cancer	4	4	3	6	6	2	1
Breast tissue	5	6	7	3	4	1	2
Climate model	4	5	7	3	6	2	1
Ecoli	4	5	6	3	7	1	2
Fertility	7	2	2	2	2	1	2
Glass	2	1	7	6	5	3	3
Harberman	4	7	5	3	6	2	1
Heart	1	5	4	7	6	2	3
Hepatitis	6	5	7	4	1	1	1
Ionosphere	5	1	7	6	3	1	4
Iris	4	7	5	6	3	2	1
Lymphography	1	3	5	4	7	6	2
Mam-masses	6	7	5	1	2	3	4
Ozone	6	7	4	2	2	5	1
Page block	5	2	7	3	3	1	6
Parkinsons	3	5	7	6	1	2	4
Pima-diabetes	4	2	1	5	7	6	2
Planning Relax	7	6	5	2	2	4	1
Seeds	5	6	7	3	2	1	4
Segment	6	4	7	3	5	2	1
Sonar	5	7	6	1	4	3	2
Spambase	6	3	5	4	7	2	1
Teaching assistant	6	5	7	4	3	1	2
Twonorm	6	5	7	2	3	1	4
User Knowledge	6	5	7	3	4	2	1
Vehicle	6	5	7	4	3	1	1
Vertebral-2C	3	4	6	2	5	7	1
Vertebral-3C	6	5	7	2	3	4	1
Waveform1	3	6	4	5	7	1	1
Waveform2	3	6	4	7	4	1	1
Wine	5	4	7	2	6	1	2
Wine quality(red)	3	4	7	5	5	1	1
Yale	4	7	6	1	4	2	2
Average Rank	4.61	4.84	5.76	3.63	4.05	2.32	1.89
Final Rank	5	6	7	3	4	2	1

*Here symbols have following meaning: Tikhonov regularization, **: Parallel axis regularization, ***: Null space regularization

The table shows that all the implementations of oblique random forest using TDS are able to achieve classification results that are comparable or slightly better than BDS implementations. Similar trends are observed for BDS and TDS. IνTWRaF achieves the highest average classification accuracy, i.e. 87.75%, which is followed by TWRaF at 87.12%. For MPRaF, the best result is obtained with Tikhonov regularization, whereas Null-space gives the minimum classification accuracy.

Table 4 Classification accuracy (%) achieved by various obliques random forests implemented using ternary decision structure (TDS) model

	Proposed TDS models using						
Datasets	MPRaF*	MPRaF**	MPRaF***	RegMPRaF*	RegMPRaF**	TWRaF*	IνTWRaF*
Australian	96.26 ± 0.00	94.40 ± 0.00	95.03 ± 0.01	97.29 ± 0.00	96.11 ± 0.00	97.53 ± 0.04	97.84 ± 0.05
Balance scale	91.85 ± 0.02	85.76 ± 0.02	86.93 ± 0.03	89.71 ± 0.01	90.96 ± 0.02	89.80 ± 0.02	91.20 ± 0.02
Banknote	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Biodeg	88.64 ± 0.02	87.11 ± 0.00	68.94 ± 0.05	88.61 ± 0.00	86.50 ± 0.02	88.15 ± 0.02	87.87 ± 0.07
Blood transfusion	79.68 ± 0.05	79.14 ± 0.04	79.01 ± 0.01	79.28 ± 0.02	79.68 ± 0.04	79.13 ± 0.02	79.74 ± 0.08
Breast cancer	97.42 ± 0.00	97.28 ± 0.00	97.57 ± 0.00	97.28 ± 0.01	97.42 ± 0.00	97.80 ± 0.05	97.28 ± 0.09
Breast tissue	73.56 ± 0.07	39.42 ± 0.12	35.02 ± 0.12	94.33 ± 0.01	72.23 ± 0.02	74.50 ± 0.02	78.37 ± 0.05
Climate model	92.22 ± 0.04	92.04 ± 0.01	91.67 ± 0.02	92.04 ± 0.01	92.22 ± 0.04	92.96 ± 0.01	92.85 ± 0.07
Ecoli	73.89 ± 0.12	83.74 ± 0.02	43.73 ± 0.02	85.21 ± 0.01	83.84 ± 0.07	87.46 ± 0.03	89.24 ± 0.01
Fertility	88.00 ± 0.05	88.67 ± 0.01	88.00 ± 0.05	92.67 ± 0.00	88.00 ± 0.05	92.67 ± 0.00	88.00 ± 0.01
Glass	37.31 ± 0.05	37.69 ± 0.01	37.69 ± 0.01	43.14 ± 0.01	35.56 ± 0.05	71.44 ± 0.06	74.83 ± 0.03
Harberman	74.55 ± 0.06	88.34 ± 0.01	87.47 ± 0.01	89.00 ± 0.00	74.55 ± 0.05	89.00 ± 0.00	88.34 ± 0.01
Heart	82.72 ± 0.03	82.23 ± 0.06	54.23 ± 0.06	82.06 ± 0.05	81.72 ± 0.03	83.90 ± 0.03	83.74 ± 0.02
Hepatitis	88.91 ± 0.05	86.38 ± 0.06	80.09 ± 0.05	87.69 ± 0.07	88.91 ± 0.05	89.51 ± 0.08	88.84 ± 0.06
Ionosphere	93.48 ± 0.02	83.51 ± 0.02	90.84 ± 0.03	94.88 ± 0.01	93.48 ± 0.02	90.30 ± 0.02	92.57 ± 0.01
Iris	97.33 ± 0.00	96.66 ± 0.01	97.33 ± 0.00	98.01 ± 0.01	97.30 ± 0.01	98.00 ± 0.01	100.00 ± 0.00
Lymphography	62.84 ± 0.10	58.11 ± 0.20	58.11 ± 0.14	85.14 ± 0.01	55.41 ± 0.09	79.63 ± 0.05	87.16 ± 0.04
Mam-masses	81.68 ± 0.03	82.93 ± 0.02	82.62 ± 0.03	82.30 ± 0.02	82.72 ± 0.03	81.39 ± 0.03	82.72 ± 0.03
Ozone	97.12 ± 0.00	96.96 ± 0.00	97.12 ± 0.00	97.08 ± 0.00	97.12 ± 0.00	97.12 ± 0.00	97.20 ± 0.00
Page block	95.76 ± 0.01	89.76 ± 0.01	89.84 ± 0.01	91.97 ± 0.01	95.76 ± 0.01	96.19 ± 0.00	97.00 ± 0.00
Parkinsons	93.38 ± 0.02	90.87 ± 0.04	87.19 ± 0.03	93.32 ± 0.00	93.38 ± 0.02	93.81 ± 0.02	92.34 ± 0.04

Table 4 continued

	Proposed TDS models using						
Datasets	MPRaF*	MPRaF**	MPRaF***	RegMPRaF*	RegMPRaF**	TWRaF*	IνTWRaF*
Pima-diabetes	75.26 ± 0.02	75.26 ± 0.03	76.04 ± 0.05	75.13 ± 0.04	75.26 ± 0.04	76.04 ± 0.05	76.00 ± 0.04
Planning Relax	73.62 ± 0.02	72.74 ± 0.02	62.13 ± 0.01	72.32 ± 0.02	73.62 ± 0.02	73.62 ± 0.02	73.62 ± 0.02
Seeds	95.04 ± 0.12	91.41 ± 0.03	90.25 ± 0.11	94.66 ± 0.15	93.37 ± 0.08	95.71 ± 0.03	95.26 ± 0.02
Segment	97.54 ± 0.05	96.43 ± 0.11	84.46 ± 0.10	98.68 ± 0.00	95.97 ± 0.06	97.27 ± 0.00	99.11 ± 0.00
Sonar	84.13 ± 0.05	86.06 ± 0.03	83.17 ± 0.04	84.62 ± 0.04	85.21 ± 0.03	86.06 ± 0.03	86.06 ± 0.03
Spambase	94.58 ± 0.00	86.72 ± 0.01	93.54 ± 0.01	94.54 ± 0.00	94.61 ± 0.01	95.02 ± 0.01	95.13 ± 0.00
Teaching assistant	53.90 ± 0.06	52.40 ± 0.06	53.18 ± 0.07	52.91 ± 0.05	53.07 ± 0.06	57.01 ± 0.08	58.24 ± 0.02
Twonorm	97.47 ± 0.00	97.35 ± 0.00	97.33 ± 0.00	97.39 ± 0.00	97.54 ± 0.00	96.87 ± 0.00	97.55 ± 0.00
User Knowledge	94.27 ± 0.05	89.52 ± 0.04	89.77 ± 0.02	89.77 ± 0.02	94.02 ± 0.05	94.81 ± 0.03	95.28 ± 0.01
Vehicle	77.05 ± 0.01	77.00 ± 0.02	73.17 ± 0.01	77.17 ± 0.01	77.00 ± 0.02	77.42 ± 0.02	77.89 ± 0.02
Vertebral-2C	87.42 ± 0.00	86.76 ± 0.03	86.75 ± 0.02	86.16 ± 0.04	86.75 ± 0.05	86.75 ± 0.02	86.76 ± 0.03
Vertebral-3C	84.90 ± 0.05	84.45 ± 0.05	84.45 ± 0.05	84.45 ± 0.05	84.45 ± 0.05	84.19 ± 0.03	85.51 ± 0.04
Waveform1	83.10 ± 0.07	69.88 ± 0.03	71.20 ± 0.09	73.62 ± 0.07	82.10 ± 0.07	84.76 ± 0.01	85.88 ± 0.00
Waveform2	82.54 ± 0.12	65.64 ± 0.05	60.00 ± 0.14	69.92 ± 0.03	82.28 ± 0.12	85.04 ± 0.01	85.48 ± 0.00
Wine	97.44 ± 0.14	98.85 ± 0.14	97.10 ± 0.13	98.28 ± 0.13	97.39 ± 0.16	98.85 ± 0.01	99.46 ± 0.03
Wine quality (red)	62.59 ± 0.03	63.40 ± 0.04	60.97 ± 0.05	65.37 ± 0.09	62.59 ± 0.03	67.60 ± 0.01	67.60 ± 0.01
Yale	79.69 ± 0.03	79.56 ± 0.09	63.61 ± 0.06	80.69 ± 0.03	80.85 ± 0.02	83.34 ± 0.09	82.63 ± 0.04
Average	84.40 ± 0.04	81.96 ± 0.04	78.30 ± 0.04	85.70 ± 0.03	84.18 ± 0.04	87.12 ± 0.03	**87.75** ± 0.03

*: Tikhonov regularization
**: Parallel axis regularization
***: Null space regularization

Table 5 Friedman test ranks for oblique random forests implemented with binary decision structure using different classifiers

Datasets	MPRaF*	MPRaF**	MPRaF***	RegMPRaF*	RegMPRaF**	TWRaF*	IνTWRaF*
Australian	4	7	6	3	5	2	1
Balance scale	1	7	6	4	3	5	2
Banknote	1	1	1	1	1	1	1
Biodeg	1	5	7	2	6	3	4
Blood transfusion	2	5	7	4	2	5	1
Breast cancer	3	5	2	5	3	1	5
Breast tissue	4	6	7	1	5	3	2
Climate model	3	5	7	5	3	1	2
Ecoli	6	5	7	3	4	2	1
Fertility	3	3	3	1	3	1	3
Glass	6	4	4	3	7	2	1
Harberman	6	3	5	1	6	1	3
Heart	3	4	7	5	6	1	2
Hepatitis	2	6	7	5	2	1	4
Ionosphere	2	7	5	1	2	6	4
Iris	4	7	4	2	6	2	1
Lymphography	4	5	5	2	7	3	1
Mam-masses	6	1	4	5	2	7	2
Ozone	2	7	2	6	2	2	1
Page block	3	7	6	5	3	2	1
Parkinsons	2	6	7	4	2	1	5
Pima-diabetes	4	4	1	7	4	1	3
Planning Relax	1	5	7	6	1	1	1
Seeds	3	6	7	4	5	1	2
Segment	3	5	7	2	6	4	1
Sonar	6	1	7	5	4	1	1
Spambase	4	7	6	5	3	2	1
Teaching assistant	3	7	5	6	4	2	1
Twonorm	3	5	6	4	2	7	1
User Knowledge	3	7	5	5	4	2	1
Vehicle	4	5	7	3	5	2	1
Vertebral-2C	1	2	2	7	2	2	2
Vertebral-3C	2	3	3	3	3	7	1
Waveform1	3	7	6	5	4	2	1
Waveform2	3	6	7	5	4	2	1
Wine	5	2	7	4	6	2	1
Wine quality(red)	5	4	6	3	7	1	1
Yale	5	6	7	4	3	1	2
Average	3.32	4.95	5.39	3.84	3.87	2.42	1.82
Rank	3	6	7	4	5	2	1

Here symbols have following meaning *: Tikhonov regularization, **: Parallel axis regularization, ***: Null space regularization

4.3.1 Statistical analysis

Friedman rank test is applied on results given by different Versions of Oblique Random Forests and the ranks obtained with different classifiers is presented in Table 5. A trend similar to Table 3 for various BDS approaches, is observed here also. Rank one is assigned to IνTWRaF and the first four ranks are secured by algorithms that use Tikhonov regularization. These are followed by Parallel axis regularization algorithms and the last position is given to Random Forest with Null space regularization.

5 Conclusions

In this paper, we propose novel oblique random forest algorithms implemented using Ternary and Binary decision structures, which employ SVM-based four classifiers, i.e. multi-surface proximal SVM, regularized multi-surface proximal SVM, twin SVM and improvements on ν-twin SVM, to split the data at the internal nodes of the decision trees. The generalization ability of each of these models is experimentally verified through 38 UCI datasets. It is observed that TWSVM and IνTWSVM-based models are able to give better results than the other two classifiers, for most of the datasets.

Acknowledgements This work is an extension of our conference paper "Oblique Random Forest via Regularized Multisurface Proximal Support Vector Machine" presented at Global Conference for Advancement in Technology (GCAT), 2019.

Author Contributions All authors equally contributed to the study conception and design. Material preparation, data collection and analysis were performed by all authors. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability The datasets analysed during the current study are available in the University of California Irvine (UCI) repository [54].

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest. Authors certify that there is no actual or potential conflict of interest in relation to this article.

Financial or Non-financial interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. Wadsworth Int, Group **37**(15), 237–251 (1984)
2. Quinlan, J.R.: C4.5: Programs for Machine Learning. Elsevier, Amsterdam (2014)
3. Ho, T. K.: Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, IEEE, pp. 278–282 (1995)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. Katuwal, R., Ponnuthurai, N.S., Zhang, L.: Heterogeneous oblique random forest. Pattern Recognit. **99**, 107078 (2020)
6. Breiman, L.: Bias, Variance, and Arcing Classifiers (1996)
7. Goerss, J.S.: Tropical cyclone track forecasts using an ensemble of dynamical models. Mon. Weather Rev. **128**(4), 1187–1193 (2000)
8. Wiering, M.A., Van Hasselt, H.: Ensemble algorithms in reinforcement learning. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **38**(4), 930–936 (2008)
9. Bonissone, P., Cadenas, J.M., Garrido, M.C., Díaz-Valladares, R.A.: A fuzzy random forest. Int. J. Approx. Reason. **51**(7), 729–747 (2010)
10. Yassin, S.S.: Road accident prediction and model interpretation using a hybrid K-means and random forest algorithm approach. In: SN Applied Sciences, Vol. 2 (9), Springer, pp. 1–13 (2020)
11. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: A comparison of decision tree ensemble creation techniques. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 173–180 (2006)
12. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. J. Artif. Intell. Res. **2**, 1–32 (1994)
13. Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U., Hamprecht, F. A.: On oblique random forests. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 453–469 (2011)
14. Zhang, L., Varadarajan, J., Nagaratnam Suganthan, P., Ahuja, N., Moulin, P.: Robust visual tracking using oblique random forests. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5589–5598 (2017)
15. Murthy, K.V.S., Salzberg, S.L.: On growing better decision trees from data, Ph.D. Thesis, Citeseer (1995)
16. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014)
17. Zhang, L., Suganthan, P.N.: Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier]. IEEE Comput. Intell. Mag. **12**(4), 61–72 (2017)
18. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
19. Barandiaran, I.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1996)
20. Criminisi, A., Shotton, J., Konukoglu, E., et al.: Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Foundations and Trends® in Computer Graphics and Vision **7**(2—-3), 81–227 (2012)
21. Zhang, L., Suganthan, P.N.: Oblique decision tree ensemble via multisurface proximal support vector machine. IEEE Trans. Cybern. **45**(10), 2165–2176 (2014)
22. Mangasarian, O.L., Wild, E.W.: Multisurface proximal support vector machine classification via generalized eigenvalues. IEEE Trans. Pattern Anal. Mach. Intell. **28**(1), 69–74 (2005)
23. Manwani, N., Sastry, P.: Geometric decision tree. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **42**(1), 181–192 (2011)

24. Chen, L.-F., Liao, H.-Y.M., Ko, M.-T., Lin, J.-C., Yu, G.-J.: A new lda-based face recognition system which can solve the small sample size problem. Pattern Recognit. **33**(10), 1713–1726 (2000)
25. Jiang, X.: Linear subspace learning-based dimensionality reduction. IEEE Signal Process. Mag. **28**(2), 16–26 (2011)
26. Khemchandani, R., Saigal, P.: Color image classification and retrieval through ternary decision structure based multi-category TWSVM. Neurocomputing **165**, 444–455 (2015)
27. Khemchandani, R., Saigal, P., Chandra, S.: Improvements on ν-twin support vector machine. Neural Netw. **79**, 97–107 (2016)
28. Saigal, P., Khanna, V., Rastogi, R.: Divide and conquer approach for semi-supervised multi-category classification through localized kernel spectral clustering. Neurocomputing **238**, 296–306 (2017)
29. Saigal, P., Chandra, S., Rastogi, R.: Multi-category Ternion support vector machine. Eng. Appl. Artif. Intell. **85**, 229–242 (2019)
30. Rastogi, R., Saigal, P., Chandra, S.: Angle-based twin parametric-margin support vector machine for pattern classification. Knowl.-Based Syst. **139**, 64–77 (2018)
31. Khemchandani, R., Saigal, P., Chandra, S.: Angle-based twin support vector machine. Ann. Oper. Res. **269**(1), 387–417 (2018)
32. Gupta, D., Richhariya, B., Borah, P.: A fuzzy twin support vector machine based on information entropy for class imbalance learning. Neural Comput. Appl. **31**(11), 7153–7164 (2019)
33. Khemchandani, R., Pal, A., Chandra, S.: Fuzzy least squares twin support vector clustering. Neural Comput. Appl. **29**(2), 553–563 (2018)
34. Chen, S.-G., Wu, X.-J., Xu, J.: Locality preserving projection least squares twin support vector machine for pattern classification. Pattern Anal. Appl. **23**(2), 1–13 (2020)
35. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
36. Hunt, E.B., Marin, J., Stone, P.J.: Experiments in Induction. Academic Press, Cambridge (1966)
37. Khemchandani, R., Chandra, S., et al.: Twin support vector machines for pattern classification. IEEE Trans. Pattern Anal. Mach. Intell. **29**(5), 905–910 (2007)
38. Khemchandani, R.: Mathematical programming applications in machine learning., Ph.D. Thesis (2008)
39. Mangasarian, O.L.: Nonlinear Programming, vol. 10. SIAM, Philadelphia (1993)
40. Gunn, S. R., et al.: Support vector machines for classification and regression, ISIS Technical Report 14 (1998)
41. Mangasarian, O. L., Wild, E. W.: Proximal support vector machine classifiers. In: Proceedings KDD-2001: Knowledge discovery and data mining, Citeseer (2001)
42. Guarracino, M.R., Cifarelli, C., Seref, O., Pardalos, P.M.: A classification method based on generalized eigenvalue problems. Optim. Methods Softw. **22**(1), 73–81 (2007)
43. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. **13**(2), 415–425 (2002)
44. Khemchandani, R., Chandra, S., et al.: Fuzzy linear proximal support vector machines for multi-category data classification. Neurocomputing **67**, 426–435 (2005)
45. Lei, H., Govindaraju, V.: Half-against-half multi-class support vector machines. In: International Workshop on Multiple Classifier Systems, Springer, pp. 156–164 (2005)
46. Shao, Y.-H., Chen, W.-J., Huang, W.-B., Yang, Z.-M., Deng, N.-Y.: The best separating decision tree twin support vector machine for multi-class classification. Procedia Comput. Sci. **17**, 1032–1038 (2013)
47. Xie, J., Hone, K., Xie, W., Gao, X., Shi, Y., Liu, X.: Extending twin support vector machine classifier for multi-category classification problems. Intell. Data Anal. **17**(4), 649–664 (2013)
48. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
49. Zhang, C.-X., Zhang, J.-S.: RotBoost: a technique for combining rotation forest and AdaBoost. Pattern Recognit. Lett. **29**(10), 1524–1536 (2008)
50. Kohavi, R., Wolpert, D. H., et al.: Bias plus variance decomposition for zero-one loss functions. In: ICML, Vol. 96, pp. 275–83 (1996)
51. Mehta, M., Rissanen, J., Agrawal, R., et al.: Mdl-based decision tree pruning. In: KDD, Vol. 21, pp. 216–221 (1995)
52. Zhang, C., Ma, Y.: Ensemble Machine Learning: Methods and Applications. Springer, Berlin (2012)
53. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bull. Calcutta Math. Soc. **35**, 99–109 (1943)
54. Blake, C., Merz, C. J.: Uci repository of machine learning databases (1998). http://www.ics.uci.edu/~mlearn/MLRepository.html
55. Ganaie, M.A., Muhammad, T., Suganthan, P.M.: Oblique decision tree ensemble via twin bounded SVM. Expert Syst. Appl. **143**, 113072 (2020)
56. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)